NASA Contractor Report 3254

PAN AIR – A Computer Program for Predicting Subsonic or Supersonic Linear Potential Flows About Arbitrary Configurations Using a Higher Order Panel Method

Volume IV – Maintenance Document (Version 3.0)

David J. Purdon, Pranab K. Baruah, John E. Bussoletti, Michael A. Epton, William A. Massena, Franklin D. Nelson, and Kiyoharu Tsurusaki

(NASA-CR-3254) PAN AIR: A COMPUTER PROGRAM FOR PREDICTING SUBSONIC OR SUPERSONIC LINEAR POTENTIAL FLOWS ABOUT ARBITRARY CONFIGURATIONS USING A HIGHER ORDER PANEL METHOD. VOLUME 4:(Boeing Military Airplane

N92-22642

Unclas

H1/02 0085733

CONTRACT NAS2-12036
JANUARY 1990

Pub. Information Update – January 1992

NASA Contractor Report 3254

PAN AIR – A Computer Program for Predicting Subsonic or Supersonic Linear Potential Flows About Arbitrary Configurations Using a Higher Order Panel Method

Volume IV – Maintenance Document (Version 3.0)

David J. Purdon, Pranab K. Baruah, John E. Bussoletti, Michael A. Epton, William A. Massena, Franklin D. Nelson, and Kiyoharu Tsurusaki
Boeing Military Airplane Company
Seattle, Washington

Prepared for
Ames Research Center, and
Langley Research Center
under Contract NAS2-12036

and for the
Air Force Aeronautical Systems Division
Air Force Wright Aeronautical Laboratories
Naval Coastal Systems Center

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Concluded)

TABLE OF CONTENTS OF APPENDICES

TABLE OF CONTENTS OF APPENDICES (Continued)

TABLE OF CONTENTS OF APPENDICES (Continued)

TABLE OF CONTENTS OF APPENDICES (Concluded)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Concluded)

LIST OF FIGURES OF APPENDICES

LIST OF FIGURES OF APPENDICES (Continued)

LIST OF FIGURES OF APPENDICES (Continued)

LIST OF FIGURES OF APPENDICES (Concluded)

LIST OF TABLES

LIST OF TABLES OF APPENDICES

LIST OF TABLES OF APPENDICES (Concluded)

FOREWORD

NASA CR 3254 is published in two parts. Part 1 contains sections 1 through 5. Part 2 contains sections 6 through 17, and appendices 1 through 13. The table of contents, including a listing of figures and tables, is repeated in each part.

SUMMARY

The PAN AIR system was written in the CFT (CRAY Fortran) language except for a few CAL (CRAY Assembly Language) subprograms in the libraries. Structured programming techniques were used to provide code documentation and maintainability. The operating system is COS (CRAY Operating System).

The system is comprised of a data base management system, a program library, an execution control module and eleven separate FORTRAN technical modules. Each module calculates part of the posed PAN AIR problem. The data base manager is used to communicate between modules and within modules. The technical modules must be run in a prescribed fashion for each PAN AIR problem. In order to ease the problem of supplying the many JCL statements required to execute the modules, the JCL statements are created by procedures.

In this volume, an overview of the PAN AIR software is given in section 1.0. Sections 2.0 through 12.0 describe the individual modules and contain information describing program structure, functional decomposition, data base communication, subroutine contents, program tree structure, data base structure and details of those major algorithms used in the module which are not straightforward and not described elsewhere. Sections 13, 14 and 15 describe the PAN AIR Library Software (PALIB), the use of the Scientific Data Management System (SDMS), and the operating system dependant features of SDMS respectively. Section 16 contains the Software Glossary followed by a list of references. Each section is designed to lead the reader through the main structural code. It is not intended to be a detailed description of a module since the structured code and comments provide this information.

Most of this document has not changed from the previous version. Unlike previous versions, however, this document covers only version 3.0 of PAN AIR. The major changes are summarized below by section.

Section 1 System overview descriptions and program statistics now apply strictly to version 3.0.

Section 2 Major portions of the code in the MEC module, which generate Cyber control cards, are not executed. The routines are referenced but no longer described.

Section 3 An additional overlay (5,2) was added to the DIP module to interpret streamline and offbody directives.

Section 4 Appendix 4-M more closely describes the way the code selects boundary conditions.

Section 5 This is a complete rewrite of the description of the MAG module.

Section 8 In the sixth overlay, GPQTY, was rewritten to implement new edge splines.

Section 12 This is a new section to aid in maintaining the FDP module.

Section 13 This section completely describes only the PAN AIR Library routines used by version 3.0.

Section 14 The reprint of the Boeing Cyber SDMS User's Manual is prefaced by comments which make it applicable to the CRAY version.

Section 15 This is a new section that identifies the requirements for converting SDMS to another machine.

Section 16 Some CRAY terms were added to this glossary.

Section 17 The CRAY Operating System Manual was added as a reference.

The tree structure diagrams and master definition listings have been moved to the installation tape. They were previously printed in Appendix A and D of each of the sections. There are cross references in the Maintenance Document and the installation instructions.

The authors wish to thank Dr. Emilio J. Zeppa, Dr. John Wai and Dr. Kenneth W. Sidwell of the Boeing Company and Dr. Alfred E. Magnus formerly of the Boeing company for their efforts in reviewing and/or preparing portions of this document. The authors also wish to extend their appreciation to Bonnie J. Jones, Mary A. Kellie, Kathleen J. Christianson and Particia S. Bradley of the Boeing Company for their assistance in typing.

1.0 PAN AIR SOFTWARE SYSTEM

1.1 INTRODUCTION

This section introduces the PAN AIR software system. The major components are the 11 program modules, a database management system, a library of subprograms and the operating system. The components of the system are described in some detail and their relation to one another is explained. The use of the various charts which appear in later sections (e.g., the functional-decomposition charts, data-flow charts, etc.) is illustrated. The CPU, memory and I/O resources required by PAN AIR are detailed. A summary of the PAN AIR Modules is presented in Appendix 1-A, and an overview of the database management is presented in Appendix 1-B. Finally, software-maintenance procedures are outlined in Paragraph 1.5.

1.2 SYSTEM OVERVIEW

The PAN AIR software system consists of several parts. Figure 1.1 illustrates these parts and their overall relationship to one another. User-supplied JCL (Job Control Language) statements activate the operating system by invoking special PAN AIR JCL procedures which execute the appropriate PAN AIR modules in sequence. The MEC (Module Execution Control) module is always executed to define certain properties of any databases created by the modules. These databases are generated using the Scientific Data Management System (SDMS). The DIP (Data Input Processor) module is always executed, since it processes all input data for all other modules. In addition, all modules call a subroutine library, PALIB, to perform certain common tasks. During installation of the PAN AIR software system at a user's computer site, the special program DDP (Data Definition Processor) is used to define the structure of each database. This latter procedure is subsequently performed only when a database structure must be modified. (See paragraphs 1.3.3 and 1.5.3)

1.2.1 Program Modules and Databases

The ten program modules, the MEC module and the databases generated by the modules are illustrated in Figure 1.2. The purpose of each module is also defined. The implied execution sequence is for a typical PAN AIR problem.

1.2.2 PAN AIR System Execution Flow

The normal sequence of operation for the PAN AIR software system is displayed in Figure 1.3 and the deck arrangement for PAN AIR execution is shown in Figure 1.5. User-supplied control statements invoke special PAN AIR JCL procedures which execute the modules in the proper sequence and usually generate the user directives for MEC. The user input-data module DIP is then executed. From then on, module after module is executed in sequence. Databases are created and used for internal data-storage and for communication between modules. Printed output is always generated by the MEC and DIP modules. Other printed output is obtained from the DQG, PDP, FDP, CDP and PPP modules if requested by the user through DIP input. The PPP module also generates a plot file on disk if requested.

1.2.3 Database Manager

The Scientific Data Management System (SDMS) is a set of CFT (CRAY FORTRAN) and CAL (CRAY Assembly Language) subroutines (the SDMS Library) which are employed in the PAN AIR system to perform nearly all disk I/O (i.e., it replaces FORTRAN I/O). Unlike FORTRAN I/O, SDMS forces the user to design the database before the design of the various modules that access it. Thus, structuring the data in a logical sense early in the design cycle will support the design of a well-structured module. This section is an introduction to the concepts and structures of SDMS.

The major collection of data in SDMS is the database. Each database is described by an input file called a Master Definition file, which describes the data within the database.

A database is a collection of more basic quantities called datasets. Datasets are analogous to files, and are defined in the Master Definition by names containing up to 20 characters.

Each dataset consists of one or more element sets. Element sets are similar to records, and are distinguished from one another by the values of a set of data-items called keys. A keyset is the collection of up to ten data-items where values distinguish one element from another. An element set consists of a collection of scalars, variable-length or fixed-length vectors in any combination. Each is described in the Master Definition by a name containing up to 20 characters.

Figure 1.4 illustrates an example of a Master Definition file for a database. Each module in the PAN AIR system creates one or more databases to be used for temporary storage or for data communication between modules. The Master Definition file is discussed in paragraph 1.4.3.3.

A database can be created after creating a Master Definition file using DDP (Data Definition Processor). The database is created by a sequence of calls to routines in the SDMS Library. First, subroutine ISDMS (Initialize SDMS) is called to define areas in CM (Central Memory) which will be used to store buffer arrays required by SDMS. Then, subroutine DBOPEN (Database Open) is called to create four unblocked physical disk-files, which hold all of the database information. In PAN AIR, the routine PAOPEN (in PALIB) orchestrates the call the DBOPEN.

Communication channels between the program and the database are defined by SDMS maps. A map sets up a correspondence between the program variables (FORTRAN name for a quantity) and the elements (SDMS names for the quantities) of a dataset as defined in the Master Definition file for the database. There are two kinds of maps: static maps and dynamic maps. In a static map, program variables are put in an exact correspondence with data items (elements) in the database. In a dynamic map, the data items on the database which are to be transferred are mentioned, but the program variables are left unspecified until the I/O operation is executed. PAN AIR uses both dynamic and static maps. A map may mention any subset of the data items and must always mention all data items which are part of the keyset of the dataset. Appendix 1-B, which gives an example of how to use SDMS to access a PAN AIR

database and illustrates a static- and a dynamic-map definition. A call to subroutine DSMAP (Define Dataset Map) initiates the map-definition process. This call contains as its arguments the name associated with the map, the name of the dataset which the map refers to, and the name of the database which contains the dataset. The map is established by calling SVMAP (Static Variable Map) to define a static map and/or DVMAP (Dynamic Variable Map) to define a dynamic map. A maximum of 10 calls to SVMAP may be made, but only one call to DVMAP is permitted in a map definition. The arguments of SVMAP are first the program (FORTRAN) variable-name which will contain the data, and then (in a 20H (Hollerith) field) the SDMS element name of the corresponding data item in the dataset. The arguments of DVMAP are simply the names of the data items in the dataset (also in a 20H field). After all correspondence has been defined, the map is terminated by calling subroutine ENDMAP. A total of 32 maps may be defined for each database which is opened.

Having defined the correspondence between program variables and data items in the database, the I/O operations are executed by calling one of several other subroutines: e.g., ESGET, ESPUT, ESREP, and ESPOR. These subroutines are described as follows.

ESGET will "get" data from the database. Its calling sequence contains first the name of the map which is to be used during the transfer, and then the list of program variables which are to receive data from the dynamic part of the map (if any) are present. During its execution, data items on the disk are read into a buffer established by the SDMS routines, and those data items which were mentioned in the map are transferred to the locations of the program variables. Fixed-length vectors are always fully transferred into the same number of sequential memory locations according to their lengths. Variable-length vectors only fill the space corresponding to their length. (Note: when using variable-length vectors in an SDMS database, if the vector is mentioned in a map, the data item containing its length must also be mentioned in the same map.)

ESPUT, ESREP and ESPOR have an argument structure which is the same for ESGET, but with ESPUT, data is transferred from the program variables out to the disk. ESPUT is used to write an element set of a dataset the first time. If the data items are to be changed, one must call ESREP to replace the existing element set with a new one. If one is uncertain about whether a given element set has already been written, but one still wishes the current variables to replace what might be on the disk, a call to ESPOR ("put or replace") will perform the task. If ESPUT is called with reference to an already existing element set, an error flag is set, and no data transfer occurs. Similarly, if ESREP is called and the indicated element set does not exist, an error flag is set, and no data transfers occur.

After all required I/O has been performed, the database must be closed to guarantee the validity of all data which has been written to the database. This is accomplished by calling subroutine DBCLOS with the database name as its argument. In PAN AIR, the routine PACLOS (in PALIB) orchestrates the call to DBCLOS.

For further information regarding SDMS, the reader is advised to consult Section 14 (SDMS Reference Manual) of this document. An example of the use of

SDMS routines in a FORTRAN program is shown in Appendix 1-B. Some discussions of SDMS I/O efficiency are also presented there.

1.3 System Components

The PAN AIR software system consists of several components: the JCL statements to execute a PAN AIR run, a set of input statements for the DIP module, the Master Definition files of the databases used by the various modules, the PAN AIR modules and the actual databases generated by the modules. These components are defined in detail in other sections of this document or in the PAN AIR User's Manual (Ref. 2).

1.3.1 JCL Cards for Initiation of PAN AIR

Version 3.0 of PAN AIR is meant to be executed on the CRAY series of computers. The standard CRAY operating system (COS) JCL supports a very powerful procedure capability. This capability has been exploited to enable users to more easily run PAN AIR and manipulate PAN AIR databases. In fact, the COS operating system can automatically generate the input for MEC. To invoke this capability, the user must first access a library named PAPROCS that contains the PAN AIR procedures. The following JCL will do this:

For an operating system (such as NASA Ames) in which users are permitted to keep permanent files on the CRAY disks:

```
ACCESS(DN=TEMP,PDN=PAPROCS,ID=PANAIR)
COPYD(I=TEMP,O=$PROC)
RELEASE(DN=TEMP)
```

For an installation in which datasets must be stored on the front end computer system, the FETCH command must be used. For the Boeing EKS/VSP system it would take the form:

```
FETCH(DN=$PROC,GDN=PAPROCS,UN=PANAIR)
```

After PAPROCS has been accessed, the user may immediately begin to run PAN AIR. This is done by invoking one of the procedures FINDPF (for "FIND POTENTIAL FLOW"), FINDSU (for "FIND SOLUTION UPDTE"), FINDICU (for "FIND IC UPDATE") and FINDPPU (for "FIND POST PROCESSING UPDATE"). These four procedures can generate the input for MEC automatically.

For a description of PAPROCS, the PAN AIR procedures for the CRAY, see Section 5.2.5 of the User's Manual.

While the documentation of PAPROCS in this User's Manual (especially Section 5.2.5.1) should be sufficient for most users, others may wish for more detailed information and/or may wish to modify a copy of PAPROCS for their own purposes. The latter may be done by following the instructions in Section 5.2.5.5.

NOTE: The Version 3.0 does not generate a MECCC file (MEC control card file). The FINDPF, FINDICU, FINDPPU and FINDSU procedures perform the function that MECCC in the previous Cyber versions performed.

1.3.2 Data Input

The input data required by the PAN AIR software system consists of two sections. The module MEC, which defines the names and IDs of the PAN AIR databases and their Master Definitions, needs a set of input statements. These are typically generated by the PAN AIR procedures. Some maintenance activities may require the user to specify the input directives for MEC. For example, PAN AIR can be directed to use a different Master Definition dataset without modifying the standard. When the new Master Definition has been tested, it can be given the standard name. The module DIP processes input statements for the remaining modules. This data specifies the geometry, flow properties and output options required for the problem. The data input stream is depicted in Figure 1.5. Detailed discussion of the MEC and DIP input-data specifications are given in the PAN AIR User's Manual, (Reference 2).

1.3.3 Databases

As mentioned previously, a database manager, SDMS, is used in the PAN AIR software system. The modules communicate among themselves through the use of the databases. SDMS databases are also used to facilitate internal communication between submodules of a module. Two steps are required for generating a database; one, the creation of a Master Definition of all data to be contained in a database; and two, the creation of the databases by the respective modules by calling the appropriate SDMS subroutine (DBOPEN). An example of the usage of the database manager is given in Appendix 1-B.

The creation of a Master Definition of a database occurs during system installation or revision. The Master Definition is then used over and over again. The creation process is separate from a PAN AIR run and is initiated by use of a separate program called DDP. The resulting Master Definition is then stored as part of the PAN AIR software. The reader is referred to Appendix 1-B and the Scientific Data Management System (SDMS) User's Reference Manual, Section 14 of this document. Revision of a Master Definition is possible and the procedure to do so is described.

Access (reading and writing) to the databases is accomplished within each module using a library named SDMSLIB. Capabilities include creation of maps or pointers from program variables to Master Definition variables, and transmitting information to and from the database. The reader is referred to Appendix 1-B and the PAN AIR User's Manual (Reference 2) for more details.

The Master Definitions for each module are detailed in Appendix D of Sections 2 through 12 in this document.

1.3.4 PAN AIR Modules

The functions of each of the PAN AIR modules is illustrated in Figure 1.2. In Appendix 1-A a summary of each module is given. The reader is referred to the PAN AIR User's Manual (Reference 2), and Sections 2 through 12 of this document for more details on each module.

1.3.4.1 PAN AIR System External Interfaces

The only external data-interfaces for the PAN AIR system are user-requested plot files produced by the FDP and PPP modules. Because of the variety of plotting devices and their software, the plot files consists of labels and data in one general format. Special user-supplied processing programs are required for the user to interface with local plotting equipment.

1.3.4.2 PAN AIR System Internal Interfaces

The internal interfaces between PAN AIR modules occur only with the databases created by the modules. Some modules use non SDMS datasets for internal communication but all data transfer between modules uses an SDMS data base. Table 1.2 summarizes the data interaction during a PAN AIR run in which every module is used. The column on the left names the various modules in the order of use. The top row gives the database names. As one reads from top to bottom, each row gives the status of each database for each module. The PAN AIR system automatically releases unneeded databases (status 4 in the table) unless the user intervenes with a directive to MEC to save any or all of them (see Section 6 of the PAN AIR User's Manual for details).

1.3.4.3 Sizing and Timing Estimates

The computer CPU time required varies greatly from problem to problem. Even for a given problem, the time may vary depending on the output options selected by the user. In general, the CPU time required varies as a quadratic function of the number of panels in the configuration. Actual CPU times required in the PAN AIR validation cases are given in Table 1.3. The cases considered are described in the PAN AIR Case Manual (Reference 4). The quadratic effect becomes more evident for cases larger than case 3.

The I/O resource requirements vary greatly from problem to problem and from module to module. The MEC and DIP modules require relatively constant amounts. The modules DQG, MAG, MDG, RMS vary as a quadratic function of the number of panels. The module RHS varies linearly with the number of panels. The modules PDP, FDP, CDP, and PPP vary in proportion to the number of output options requested by the user. Table 1.4 summarizes the I/O volume requirements for the PAN AIR validation cases detailed in Reference 6, the Case Manual. Table 1.5 summarizes the I/O frequency requirements.

Version 3.0 can be run within one million words of memory on the CRAY.

The size of each module and the approximate requirements for compilation are given in Table 1.6. Note that the module DQG requires significantly more resources than the other modules.

1.3.4.4 Software Design Consideration

Structured FORTRAN coding principles were used throughout the PAN AIR software system. This approach results in a documented modular set of code, and it encourages analysts to provide comments to explain what the code is accomplishing. Structured coding does not guarantee well documented programs, but it does ensure modular and readable code.

The structured approach does aid program maintenance. Experience during the PAN AIR system validation process showed that a person familiar with the system could delve into a program, find and correct errors without the aid of the programmer who wrote the original code.

The PAN AIR software was originally designed for the Control Data 7600, 6600, Cyber 175 computer systems and then converted to the CRAY. All programs were compiled under CFT or CAL. The non-ANSI FORTRAN statements listed in Table 1.7 were used as sparingly as possible. The DECODE and ENCODE functions were used only in the MEC module. Masking operations were used in MEC, DIP, MAG and the PAN AIR Library. CAL code was restricted to the SDMS code and the PAN AIR Library. The routines in the PAN AIR library using CAL are listed in Table 1.8.

1.4 A Guide to Module Interpretation

The Maintenance document was designed to be used in conjunction with the information contained in the preface and code of each program and/or subprogram. The Maintenance document and the installation tape contain functional decomposition charts, database-communication charts, tree diagrams, subprogram definitions, and database Master Definitions. Each program or subprogram contains a decomposition level, purpose and/or method, glossary, communication-vehicle description, labeled common-blocks descriptions and design code which correspond to program statements. The structure of a program or subprogram is illustrated in Figure 1.6. The use and interpretation of these components is described as follows.

1.4.1 Functional Decomposition and Structure

The functional-decomposition chart gives a complete overview of what a particular overlay of a module accomplishes. Consider the functional-decomposition chart of the MEC module (Appendix 2-B, Section 2 of this document). One can easily see, for example, that overlay (1,0), called READUD at the B level, consists of three main portions. The B.C decomposition portion, namely PREXEC is the most complicated, but the structure and the tasks performed are clear. One should also note that if a subprogram is used, the name of the routine appears, as does the decomposition level. For example, the decomposition level B·C·B·L corresponds to subprogram DBASE. One can compare the functional decomposition to the program listing and find a direct correspondence to the code and structure of the code. In the code, the decomposition level of a particular section would typically appear at the right in column 55 and would also indicate the name of a subprogram, if one is used at that level. For example, if one looks at level B.C.C.A, the subprogram LODREC is used. The code decomposition would read (A=PALIB=LODREC) with the upper level at MEC.B.C.C. This would indicate that the routine LODREC is in the PAN AIR Library and is used in MEC module at level B.C.C.A. If a routine is used at more than one level, then the symbol .LIB is attached to the end of the unique portion of the decomposition level of that routine. Hence, B.C.C.A.LIB indicates that the routine with this decomposition is used at different levels below the level B.C.C.A.

The tree diagrams in Appendix A of each section give another complete overview of a module and its subprograms and are very useful for tracing the

path of a formal parameter of a subprogram back to its calling programs. Also, if one modifies a subprogram, one can determine what other subprograms may be affected. Finally, if COMMON is used for data communication, the calling program will almost always include the common blocks used in its subprogram.

The alphabetical list of subprograms and the associated abbreviated functional description in Appendix B of each section can be used in conjunction with the tree diagram (Appendix A of each section) to gain another view of the structure and purpose of a module.

1.4.2 Preface of Modules and Subprograms

The preface of each program and subprogram (see Figure 1.6) contains the upper decomposition-level, the purpose (if the title of the routine is not self-explanatory), a method (algorithm) if appropriate, and communication-vehicle descriptions which give an overview of the input/output. Any labeled common-blocks of data used for input or output are listed. Formal parameters of subprograms are also indicated.

1.4.3 Data Flow

Labeled common, database input/output and formal parameters of subprograms are major vehicles used for data communication within modules and between modules. Only the modules FDP and PPP write a disk file without the use of the database manager. They can produce plot information file for post-processing.

Internal communication refers to data flow within a module. The various modules use labeled common and formal subprogram parameters for internal communication. Sometimes temporary databases are also used for intermediate data storage if the volume of data exceeds central-memory limits.

External communication refers to data flow between modules. The database manager is used for this purpose.

Methods to analyze data communications will now be described in some detail. There are three kinds of data flow within the typical PAN AIR module: (1) data flow from a database to the program, (2) data flow from one part of the program to another and (3) data flow from the program to a database.

The first and third kinds of data flow intimately involve the use of SDMS maps (Paragraph 1.2.3 and Appendix 1-B). To aid in the process of tracing data flow, each module-maintenance section includes three related database-communication-charts. The first form of the chart lists in alphabetical order, for each overlay of the module, the databases and datasets which are accessed in that overlay. Corresponding to each dataset there is listed the name of the map which sets up the correspondence between data items in the dataset and program variables. Also listed is the common block(s) in which the mapped program variables lie, if applicable, and the name of the subroutine in which the map is defined. The second form of the chart contains the same information but it is arranged with the map names in alphabetical

order. The third form of the charts repeats the same information but has it ordered alphabetically by common-block name. The use of these charts is illustrated in Paragraph 1.4.3.3.

To allow speedy tracing of the second kind of data flow, several documentation devices have been incorporated in the coding of the modules. Chief among these is the glossary.

The glossary of each program or subprogram lists all those FORTRAN variables which are used in the program for input, output or as auxilary parameters. Each variable is flagged with an I for input or an O for output. All formal parameters (arguments of a subprogram) are so indicated by an "F.P." flag. If the variable appears in a labeled common block, the name of the block is listed. Finally, a short definition of the variables is also given when appropriate.

1.4.3.1 Formal Parameters

The analysis of formal parameters for internal communication is straightforward. The glossary identifies and defines these parameters. The tree diagram can be used to relate the parameter to other programs and subprograms.

1.4.3.2 Labeled Common

Labeled common is used for internal communication between subprograms and calling program/subprograms. The glossary defines the input/output variables and indicates in which labeled common-block the variables reside. If a labeled common-block is mentioned, one can look at the data-group section of the code (See Figure 1.6) and find a definition of the variables contained in the block.

The section of code in the Preface of each program called COMMUNICATION VEHICLES can also be used to find common blocks which are used for input/output.

1.4.3.3 Database Communication

The various modules use databases to pass data to other modules and, sometimes, for temporary scratch storage. Usually, labeled common is used to store data obtained from a database. Unlabeled common, usually called blank common, is also used to hold data until it is transferred to a database. The source code calls subroutines from the SDMS library to accomplish these data transfers.

A means is available to analyze or trace data from labeled common to a database, if such a correspondence exists. The data communication charts in conjunction with the glossaries and the database map-definitions are the available analysis tools. An example of the analysis is presented using part of the DQG code.

Suppose that we wish to find out what happens to the user-defined abutment data (PAN AIR User's Manual, Reference 2) as it is transferred from the DIP

database, through the DQG module, and then to the DQG database. Figures 1.7 and 1.8 show relevant excerpts from the DIP module Master Definition (Section 3, Appendix 3-D) and the DQG module Master Definition (Section 4, Appendix 4-D). Both databases contain dataset USER-ABUT which contains information about user-defined abutments. Examination of the first form of the database communication-chart for module DQG (Fig. 1.9) shows that the DIP database is used in the first overlay (1,0) of DQG. Further, within that overlay, dataset USER-ABUT on the DIP database is connected with program variables in the /ABUT/ common block of DQG by means of a map named USABIN. This map is defined in the subroutine DIPDAT of the DQG module.

Examination of the map USABIN in subroutine DIPDAT (See the following paragraphs which discuss Figure 1.13a) shows that the keyset data-item "ABUT-INDEX" is mapped dynamically and the other data items are mapped statically to program variables NBRNAB, POSABT, SMOOAB and the array USABUT. In the glossary of the subroutine DIPDAT (Fig. 1.10) we find that these variables are all located in common block /ABUT/. At the beginning of the subroutine DIPDAT, we find the common-block contents described (Fig. 1.11). Here NBRNAB is the number of networks in the abutment and that the array USABUT contains information which identifies the network, the edge, and the corner points marking the start and end of the abutments. Figure 4.5 (Structure and Data Flow of DQG Overlay (1,0)) in Section 4 of this document shows that the dataset is read from the DIP database by the DQG module in subroutine DIPDAT.

Thus far, we have traced the data from the DIP database into the DQG program. We can now examine how the data gets to the DQG database. Figure 4.5 (Section 4) shows that the DQG module generated dataset USER-ABUT is written to the DQG database within the same overlay. We already know that the program variables which contains the data resides in the common block /ABUT/. If we look at the third form of the database communication-chart (Fig. 1.12) we find that there is a map called USABIN which maps data from common block /ABUT/ to the dataset USER-ABUT on the DQG database. As mentioned above, this map is defined in subroutine DIPDAT of the DQG module. Examination of the maps in subroutine DIPDAT (Fig. 1.13) shows that the program variables NBRNAB, SMOOAB and the array USABUT are mapped onto data items (elements) in the two USER-ABUT datasets and we see that some items do not seem to appear on the DQG database (i.e., "POS-FLAG") while some items are "longer" than they were (the array USABUT (I,J) is filled only for J=1,4 by the map to the DIP database, while the map to the DQG database connects to the full array USABUT (I,J), J = 1,6). Yet other items have one-to-one correspondence in both databases, (e.g., NO-NET-ABUT in the DIP database and NMBR-NETWK-IN-ABUT in the DQG database). This is not a surprising result if we examine the short description of the subroutine DIPDAT in Paragraph 4.4.2 of Section 4.

Subroutine DIPDAT reads data from the DIP database and copies it to the DQG database, sometimes changing its form to better suit DQG's requirements. Clearly, the array USABUT has undergone some transformation and a detailed examination of the code and comments in DIPDAT clarifies what has occurred. Looking through the code in subroutine DIPDAT, we find a call to ESGET with the map name "USABIN". This is where the data enters the DQG program. Now, after the data is available, if the plane-of-symmetry flag is set, the number of networks in the abutment is increased by one and the new network edge is

labeled by defining the entry of USABUT(NBRNAB,1) = -POSABT. This explains why the plane of symmetry flag is not present in the DQG dataset. Further, the USABUT(I,3) and USABUT(I,4) entries are stored as ISTART and IEND in the program. The glossary of DIPDAT describes these as the start and end corner-points of an abutment in the counter-clockwise sequential-index system (Appendix 4-F). These are passed to a subroutine EDGLAT. The call is commented by "COMPUTE COARSE LATTICE INDICES FOR START AND END POINT" and the preface of EDGLAT indicates its function is to transform the counter-clockwise indexing scheme along a network edge to the coarse grid lattice-indexing scheme. Immediately after the call to EDGLAT, a call is made to ESPUT with the USABUT map name. This writes the abutment data to the DQG database.

From this analysis we can see that the user-defined abutment is read from the DIP database with the data stored in a particular fashion; the data is then transcribed into a form which DQG finds more useful and is written to the DQG database. Thus, we have traced the data and have observed its transformations. Figure 1.14 summarizes the analysis of the data flow.

If more detail is required concerning the transformations, then we must make use of the glossary of subroutine EDGLAT of the DQG module to find the correspondence between the local variables as they appear in the subroutine EDGLAT and the variables which appear in the call to EDGLAT in subroutine DIPDAT. The glossary of EDGLAT identifies which of the formal arguments are input and which are output. Examination of the code defines precisely the form of the transformation.

The database Master Definitions also can be used to relate program variables to database element names. Usually, the correspondence between variable names and element names is placed after a $ appearing at the right-hand side of the Master Definition. Therefore, once a variable name is attached to a database name using the glossary and the data communication chart, the correspondence between program-variable names and element names can usually be found using the Master Definition (sometimes, as in Figure 1.8, the FORTRAN variable names are not given). For example, in Figure 1.7 the variable names IABUT is mapped into the element name ABUT-INDEX of the DIP database dataset named USER-ABUT.

The Master Definitions can be obtained by executing the UPDATE tool with the Master Definition Program-Library provided on the installation tape.

1.5 Maintenance of PAN AIR Software

The continued maintenance of both source code and documentation is absolutely necessary to improve and insure the integrity of a large software-system such as PAN AIR. Several tools are available to aid the maintenance process.

1.5.1 UPDATE Feature

The PAN AIR software was developed using the UPDATE software-management program. Using this tool, a program-library file is created; then corrections, additions, deletions, etc. are easily made to the library. A running history of changes is an output of the UPDATE program. Details of

using this feature are to be found in the UPDATE Reference Manual. Each module in the PAN AIR system is maintained as a separate Program Library (PL). In addition there is a separate PL for the routines in the PAN AIR Library and the SDMS Library. The PL for SDMS includes both the routines for SDMSLIB and the program DDP.

The creation of an absolute program for each PAN AIR module is straightforward. The UPDATE program can be used to generate a COMPILE file, which is then compiled. This generates a file of relocatable binaries. Then the LDR feature of the operating system can be used to link the relocatable binaries with the PAN AIR libraries to generate the absolute file.

It is strongly recommended that the UPDATE program (or one similar in function) be used in the future to maintain the PAN AIR software system. Configuration control of a large software system mandates that changes to a dataset be reproducible. An UPDATE modification set can first be tested and then later applied to the controlled version of the code. The effects of the modification set may be undone at a later time. By saving modification sets, there is a precise definition of the changes from version to version.

1.5.2 Common Data Blocks

The PAN AIR software system relies heavily on the use of labeled common-blocks. This condition was the result of using the SDMS database manager which is executed most effectively using labeled common.

Each labeled common-block is used many times with the various subprograms and modules. If a modification was made to one block of one subprogram without making the same changes in the same block used elsewhere, the PAN AIR software would no longer function correctly.

Fortunately, the UPDATE feature can also be used to maintain each common block. Each block is placed in a COMDECK and becomes part of the program library (PL) of a module. If a change is made to a common block, the UPDATE feature automatically makes the change in all subprograms and programs using the modified labeled common block.

1.5.3 Master Definition Modification and Maintenance

Modification of a module may require a change to the Master Definition of a database. For example, a new element or collections of elements, called datasets, may be added or deleted. The modification process is quite straightforward.

Each Master Definition is stored as an UPDATE deck on a Master Definition Program Library (MDFPL). The deck is changed and the new version is written to the COMPILE file. The old Master Definition is then purged and the program DDP (stored with the PAN AIR software system) is run using the COMPILE file produced by the UPDATE execution as input to the program. This process will result in a new Master Definition containing the changes made previously.

1.5.4 Document Maintenance

Program modifications may require revisions to be made to the supporting documents, of References 1, 2 and 3. In particular, the functional-decomposition charts, tree diagrams, data-communication charts, Master Definitions and text of each section of this document may have to be modified. At each major computer-installation, a utility program to produce tree diagrams is usually available. This tool could be used to produce a new tree-diagram if the subprogram linkage is modified. If the Master Definition of a database is changed, a new listing is automatically produced by the DDP utility program of the PAN AIR software.

The functional-decomposition chart of each section of this document must be modified if a subprogram is changed. Most computer installations have a software utility-program which extracts the structure of pseudocode of a program. The extraction process is keyed upon finding a "C" in column 1 of a program listing and/or other key words or symbols. The PAN AIR code was developed using "C", "C.", "CP", "CPE", "GLOSSARY", "DATA GROUPS" as key words to separate sections of comments and code. If such a software tool is available, it could be used to an advantage to modify the functional-decomposition charts of this document.

Table 1.1- PAN AIR Installation Considerations

Location	Operating System	Computer Hardware	Front End Computer
NASA AMES	COS 1.14	CRAY X-MP	Cyber and Vax
AEDC	COS 1.14	CRAY X-MP	Amdahl
NCSC	COS 1.12	CRAY 1	Cyber

Table 1.2 - Module and Database Interactions

Using Module	Database Name							
	DIP	DQG	MAK	RMS	RHS	MDG	PDP	CDP
DIP	1	0	0	0	0	0	0	0
DQG	2	1	0	0	0	0	0	0
MAG	2	2	1	0	0	0	0	0
RMS	0	0	2	1	0	0	0	0
RHS	2	2	2	4	1	0	0	0
MDG	2	3	4	0	4	1	0	0
PDP	2	0	0	0	0	2	1	0
FDP	2	0	0	0	0	2	0	0
CDP	2	0	0	0	0	4	0	1
PPP	4	4	0	0	0	0	4	4

Codes for Databases

0 - Not used or created

1 - Created

2 - Used

3 - Not needed thereafter unless PPP was requested or a save directive has been issued.

4 - Not needed thereafter unless requested for a save

Table 1.3 - Validation Case CPU Time Requirement
(NASA AMES CRAY X-MP)

Cumulative CPU Execution Requirements X-MP (Sec)

After Module	Case 1	Case 2	Case 3	Case 6
MEC	0.2411	0.2472	0.3097	0.2646
DIP	0.4760	0.5301	1.5773	0.7674
DQG	1.7998	3.3792	61.5878	13.4035
MAG	2.4605	6.9902	176.3091	40.3996
RMS	2.6413	7.2486	202.3822	42.3715
RHS	3.1715	10.0269	212.1791	45.5931
MDG	4.3120	14.5731	239.4363	58.3829
PDP	5.1179	16.3864	243.3971	62.8354
CDP	5.7290	17.4047	251.9595	67.0223
FDP	5.8903	17.5910	252.5155	67.3297
PPP	6.5467	18.2493	254.9603	68.3105

Table 1.4 - Validation Case I/O Volume Requirements

Cumulative Disk Sectors Moved

After Module	Case 1	Case 2	Case 3	Case 6
MEC	683	684	702	686
DIP	1274	1281	1540	1301
DQG	2663	2989	23573	5551
MAG	3497	4411	47519	11833
RMS	3929	5023	62297	13956
RHS	4712	6116	69575	16500
MDG	6442	8672	82315	22407
PDP	7154	9608	84390	23637
CDP	7784	10406	85993	24702
FDP	8330	11030	87307	25580
PPP	9021	11766	88409	26406

Table 1.5 - Validation Case I/O Frequency Requirements

After Module	Cumulative I/O Requests			
	Case 1	Case 2	Case 3	Case 6
MEC	112	113	114	114
DIP	245	247	345	250
DQG	648	801	18558	2905
MAG	847	1185	23407	4394
RMS	947	1293	24727	4593
RHS	1162	1528	27625	5002
MDG	1560	2036	32674	6604
PDP	1727	2218	33445	6796
CDP	1874	2379	34156	7160
FDP	1966	2476	34315	7279
PPP	2110	2630	34527	7440

Table 1.6 - Module Size and Compilation Time

Module	Lines	Statements	Compilation Time (Seconds)
MEC	8,630	1,982	.7293
DIP	42,555	7,734	3.1778
DQG	79,253	15,532	8.4553
MAG	24,887	4,442	2.8676
RMS	2,659	449	.2139
RHS	11,161	2,074	.9896
MDG	28,635	5,648	2.7902
PDP	23,006	3,967	2.1710
FDP	20,820	5,009	3.0672
CDP	19,540	3,638	2.0957
PPP	17,492	2,522	1.5384

Table 1.7 - Non-ANSI FORTRAN CODE USAGE

OVERLAY	
PROGRAM (INPUT,OUTPUT,...)	
DATA arrays	
variable = 3H XXX	(Hollerith constants)
J.AND.K,J.OR.K	(Masking)
Array referenced with fewer subscripts than in DIMENSION	
FORMAT(3HXXXA10)	(No field separator)
7B	(Octal constant)
DECODE function	
ENCODE function	
Mixed mode arithmetic and comparisons	
nLf	(Left justified Hollerith)
Equivalencing of arrays	

Table 1.8 - CAL CODE USAGE

Subroutine
BIT$LGN
BIT$LOC
BIT$MSK
CAB
GETT
MXMACA
PAC
PUTT
REDUCR
STRMOV
UNPAC
ZERO

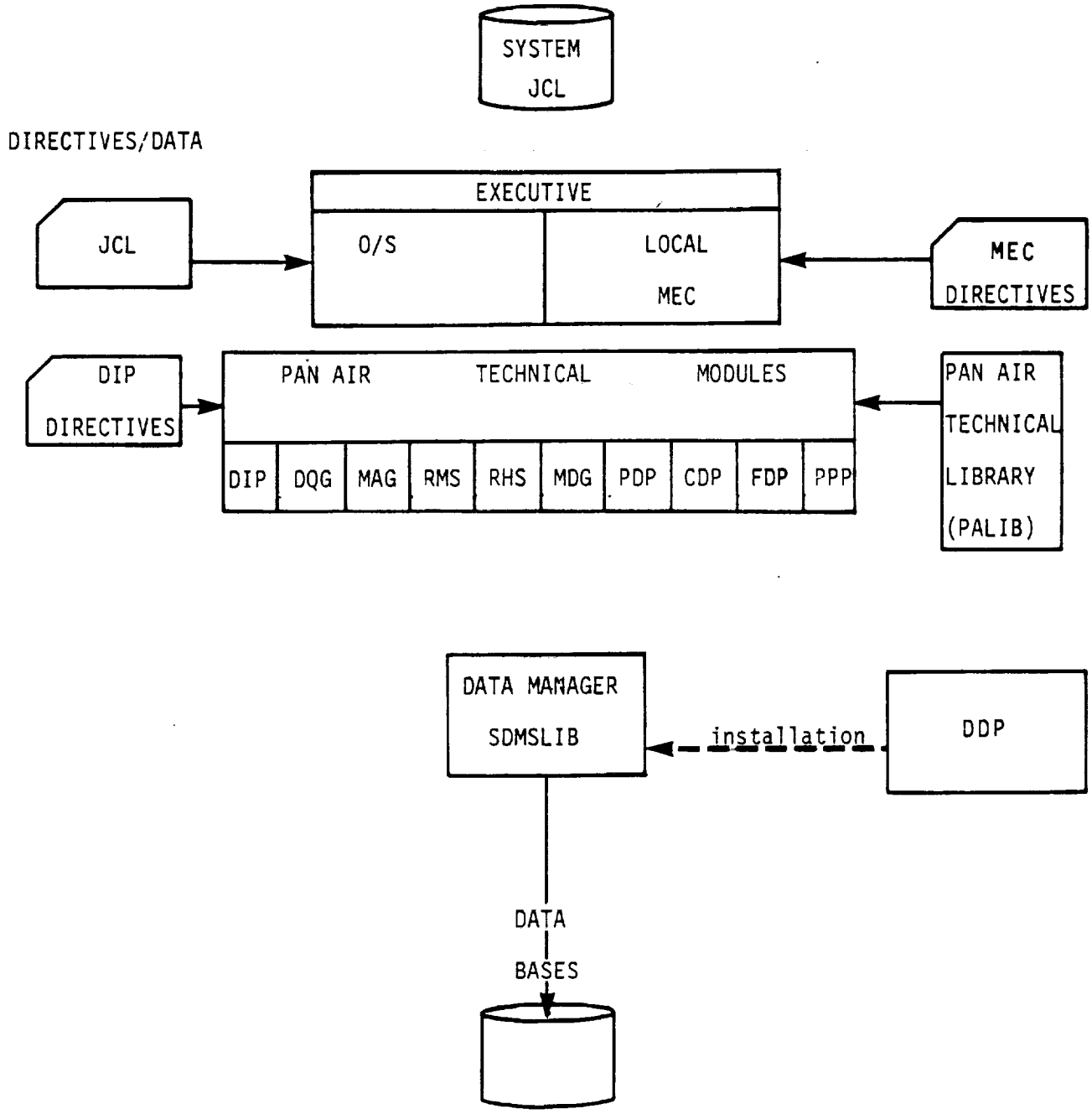
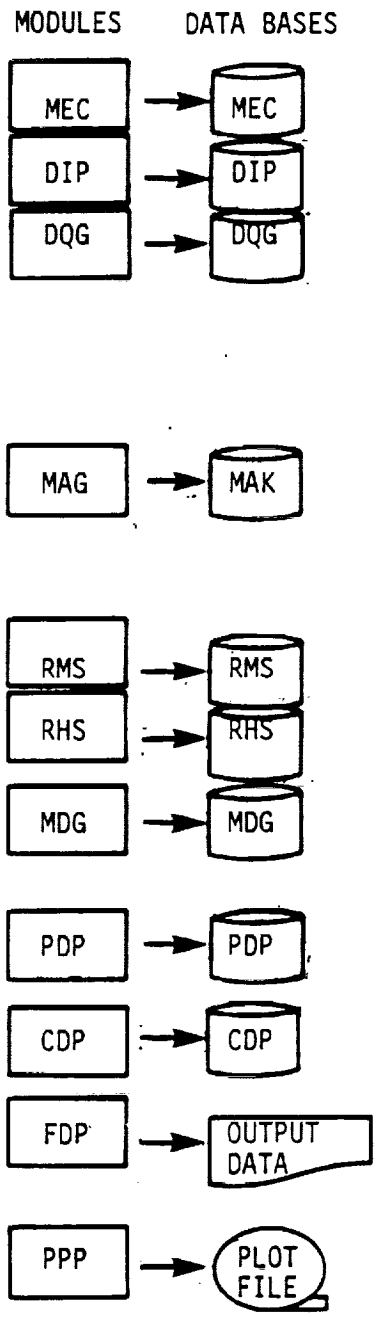


Figure 1.1 - PAN AIR Software System



MODULES AND THEIR PURPOSE

MEC generates control cards for problem

DIP interprets user input

DQG generates panel defining quantities plus data for control points, boundary conditions and singularities

AIC — MAG creates Aerodynamic Influence Coefficients Unknown Singularity Portion

AIC — MAG creates Aerodynamic Influence Coefficients Known Singularity Portion

IC — MAG computes Influence Coefficients

RMS Decomposes AIC unknown

RHS processes singularities and boundary condition data

MDG finds average potential , velocity and normal mass flux at control and grid points plus DQG geometry

PDP computes potential, velocity, mass flux, and pressures for selected surfaces

CDP computes forces and moments accumulated over portions of congifuration

FDP computes potential, velocity, mass flux and and pressures at locations off configuration and along streamlines

PPP selects data formatted for external display processing

Figure 1.2 - Program Modules and Data Bases

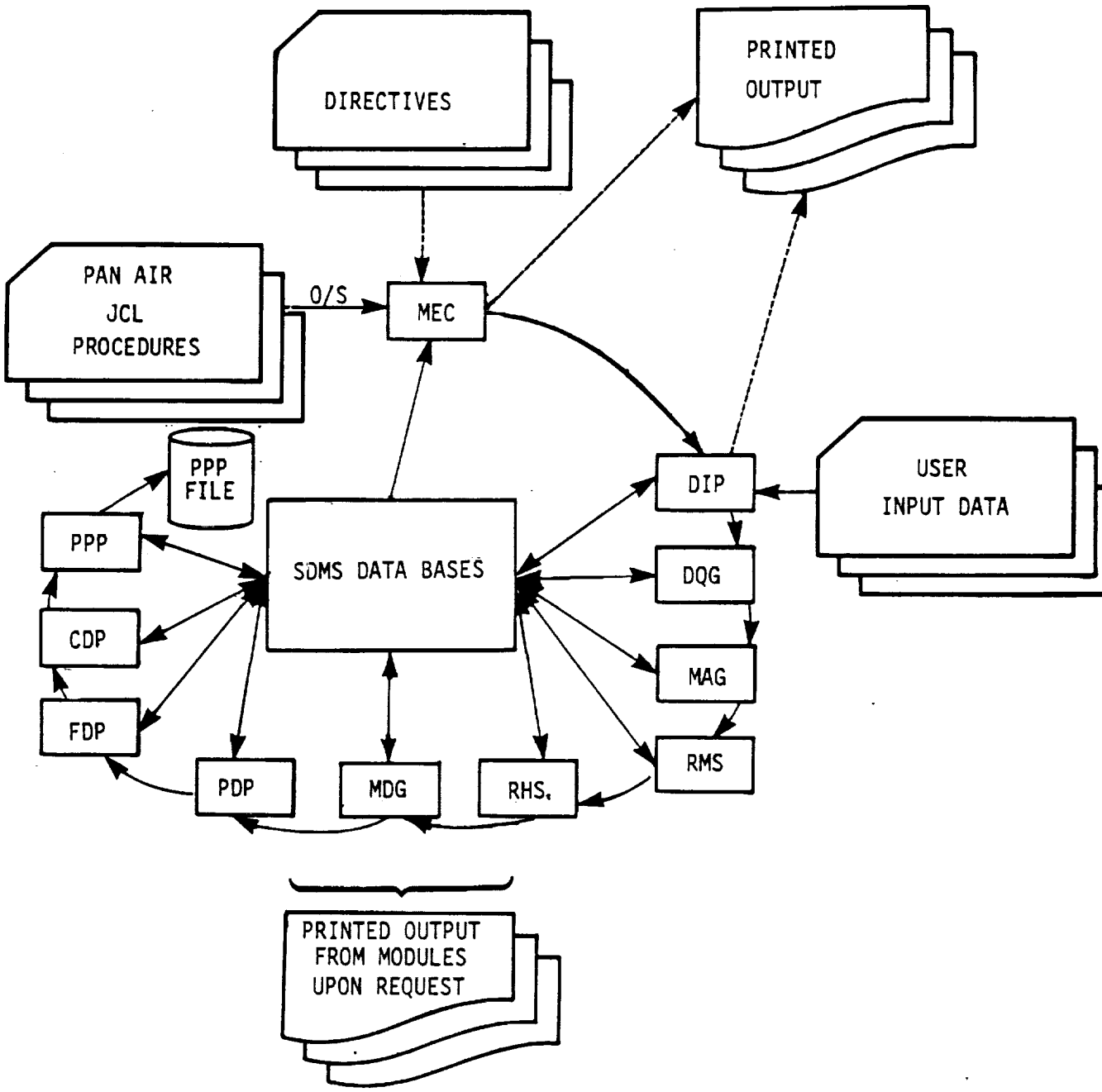


Figure 1.3 - PAN AIR Data Flow

```
MASTER DEFINITION MDFILE                    $
                                            $
                                            $
   DATASET FILE-NUMBER-1                    $
                                            $
                                            $
     KEY SET                                $
                                            $
                                            $
        REC-IND-1                           $
        REC-IND-2                           $
                                            $
                                            $
    END                                     $
                                            $
                                            $
    ELEMENT SET                             $
                                            $
                                            $
      INTEGER-VAR-1                   I     $
      SCALAR-VAR-1                    R     $
      VECTOR-VAR-1        3           R     $
      VAR-LENG-VECTOR-1   INTEGER-1   R     $
                                            $
                                            $
    END                                     $
                                            $
  END DATASET                               $
                                            $
                                            $
END MASTER DEFINITION                       $
```

Figure 1.4 - Example of Master Definition Structure in SDMS

User Supplied JCL	(Limited set of control statements to invoke special PAN AIR procedures)
———————	(CRAY end of file)
Dip Input Data	(User Input Data Defining Problem)
———————	(CRAY end of dataset)

Figure 1.5 - Deck arrangement for PAN AIR Execution

NAME: name - short title		
DECOMP: program decomposition		
DESIGNER: name, date, company		
PROGRAMMER: name, date, company		
[PURPOSE: test]		PREFACE
[METHOD: test]		
INPUT DATA:		
OUTPUT DATA:		
[ARGUMENT LIST:]		
[GLOSSARY]		
DATA GROUPS i.e., common blks		
TYPE DECLARATIONS		SPECIFICATION STATEMENTS
DATA STATEMENTS		
INITIALIZATION		
TASK 1	(A)	
SUB-TASK 1	(AA)	
SUB-TASK 2	(AB)	BODY
TASK 2	(B)	
SUB-TASK 1	(BA)	
SUb-TASK 2	(BB)	
ERROR PROCESSING		
5000 series - input		
6000 series - output		FORMATS
8000 series - errors		

Figure 1.6 - Program/Subprogram Structure

```
$ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
$
 DATASET      USER-ABUT                        $USER DEFINED ABUTMENT
                                               $
      KEY SET                                  $
                                               $ IABUT  =
   ABUT-INDEX                            I     $ ABUTMENT INDEX NUMBER.
                                               $ RANGE IS 1 TO NO-USER-ABUT
                                               $ VALUE IN DATASET NAMED  NETWORK-DATA
                                               $ON THIS (DIP) DATABASE.
                                               $
                                               $
  END                                          $
                                               $
                                               $
      ELEMENT SET                              $
                                               $
                                               $ NONEAB  =
       NO-NET-ABUT                         I   $ NUMBER OF NETWORKS IN ABUTMENT
                                               $
                                               $ NETIND  =
       NETWK-LIST  NO-NET-ABUT             I   $ LIST OF NETWORKS IN ABUTMENT.
                                               $ ENTRIES REFLECT INPUT ORDER.
                                               $
                                               $ EDGNO  =
       EDGE-NMBR   NO-NET-ABUT             I   $ EDGE OF NETWORK IN ABUTMENT
                                               $
                                               $ STRTPT =
       STRT-PT          NO-NET-ABUT         I   $ STARTING EDGE POINT NUMBER FOR
                                               $ NETWORK EDGE.
                                               $
                                               $ ENDPT =
       END-PT      NO-NET-ABUT             I   $ ENDING EDGE POINT NUMBER FOR
                                               $ NETWORK EDGE.
                                               $
                                               $ ABUPOS =
       POS-FLAG                                I   $ PLANE OF SYMMETRY FLAG.
                                               $ ALLOWABLE OPTIONS ARE:
                                               $   0 = NO PLANES OF SYMMETRY
                                               $   1 = FIRST PLANE OF SYMMETRY
                                               $   2 = SECOND PLANE OF SYMMETRY
                                               $   3 = BOTH PLANES OF SYMMETRY
                                               $
                                               $ EDGTRE =
       EDGE-TREAT                          I   $ EDGE TREATMENT FLAG.
                                               $ ALLOWABLE OPTIONS ARE:
                                               $   0 = STANDARD ABUTMENT
                                               $   1 = SMOOTH ABUTMENT
                                               $
      END                                      $
                                               $
      END DATASET                              $
```

Figure 1.7 - Excerpt from DIP Master Definition

```
DATASET        USER-ABUT                  $ USER DEFINED ABUTMENT
                                          $
      KEY SET                             $
                                          $
    INDEX                           I     $                        I
                                          $
  END                                     $
                                          $
  ELEMENT SET                             $
                                          $
    NMBR-NETWK-IN-ABUT              I     $                        I
                                          $
                                          $ NUMBER OF NETWORKS WHOSE EDGES
                                          $ MAKE UP THE ABUTMENT
                                          $
    NETWK-ID                 NMBR-NETWK-IN-ABUT        I   $
    EDGE-NMBR                NMBR-NETWK-IN-ABUT        I   $
    STRT-CRNR-PT-NMBR-I      NMBR-NETWK-IN-ABUT        I   $
    STRT-CRNR-PT-J           NMBR-NETWK-IN-ABUT        I   $
    END-CRNR-PT-NMBR-I       NMBR-NETWK-IN-ABUT        I   $
    END-CRNR-PT-J            NMBR-NETWK-IN-ABUT        I   $
                                          $
                                          $ THESE ARE THE NETWORK EDGES WHICH MAKE
                                          $ UP THE ABUTMENT.            $
                                                                        $
    SMOOTH-ABUT-FLAG                T     $
                                          $
                                          $
  END                                     $
                                          $
END DATASET                               $
```

Figure 1.8 - Excerpt from DQG Master Definition

DATABASE	DATASET NAME	MAP NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
MEC	DATA-BASE-HEADER	IDS	/RUNIDS/	OPENER
DIP	CLOS-COND	DIPCLOSDAT	/GENBCD/	BNDYIN
DIP	COEF-GEN-BC	CGBCMP	/GENBCD/	BNDYIN
DIP	GLOBAL	GLOBAL-EN	/GLOBAL/	DIPDAT
DIP	GLOBAL-PRINTS	PRINT-OPT	Dynamic	DIPDAT
DIP	NETWN-BDC	NETBDC	/NETBDC/	BNDYIN
DIP	NETWN-SPEC	NETMAP	/NETWN/	DIPDAT
DIP	PANEL-COORDS	PAN-COR-PI	/COORDS/	DIPDAT
DIP	TANG-VEC	TVECTCOEFF	/GENBCD/	BNDYIN
* DIP	USER-ABUT	USABIN	/ABUT/	DIPDAT
DQG	CLASS-5-BC-DATA	CLASS 5	/NBCDIN/	BNDYIN
DQG	CLOSURE-DATA-IN	CLOSDIN	/CLOSUR/	BNDYIN
DQG	GLOBAL	GLOB-DYN	/GLOBAL/ Dynamic	DIPDAY
DQG	NETWK-BNDRY-CONDN-IN	BCDATIN	/NSCDIN/	BNDYIN
DQG	NETWK-SPEC	NETMAP	/MESWN/	DIPDAT
DQG	PANEL-CORNER-COORDS	COORDS-GEN	Dynamic	DIPDAT
DQG	USER-ABUT	USABUT	/ABUT/	DIPDAT

Figure 1.9 - DQG Database Communication Chart,
First Form for (1,0) OVERLAY

NAME	TYPE	ORIGIN	USAGE	DESCRIPTION
C.LCLASS	I	/NETWK/	I/O	
C.LENGTH	I			LENGTH OF ARRAY OF COLUMN OF CORNER POINTS
C.MACH	R	/GLOBAL/	I/O	
C.MESH	I	/NETWK/	I/O	
C.MVCOMP	I	/NETWK/	I/O	
C.NABUT	I	/GLOBAL/	I/O	
C.NBRCP	I	/GLOBAL/	I/O	
C.NBRHS	I		I/O	NUMBER OF SOLUTIONS
* C.NBRNAB	I	/ABUT/		I/O
C.NBRNET	I	/GLOBAL/	I/O	
C.NBRPOS	I	/GLOBAL/	I/O	
C.NBRSNG	I	/GLOBAL/	I/O	
C.NCOL	I			UPPER LIMIT ON CORNER POINT COLUMNS
C.NETCTR	R	/NETWK/	I/O	
C.NETID	I	/GLOBAL/	I	
C.NETORD	I	/GLOBAL/	I	
C.NGAPNL	I	/GLOBAL/	I	
C.NIAB	I			UPPER LIMIT OF USER SPECIFIED ABUTMENTS
C.NLRCLS	I		I/O	NUMBER OF LEFT/RIGHT CLASS 4 BC
C.NMBROW	I	/COORDS/	I/O	
C.NNETOT	I	/GLOBAL/	I/O	
C.NPT	I			NUMBER OF POINTS IN NETWORK EDGE SEGMENT
C.OMMINF	R	/GLOBAL/	I/O	
* C.POSABT	I	/ABUT/		I
C.POSFLG	I	/NETWK/	I/O	
C.POSLOC	R	/GLOBAL/	I/O	
C.POSNRM	R	/GLOBAL/	I/O	
C.PRTOPT	I	/PRNTOP/	I	
C.RCLASS	I	/NETWK/	I/O	
C.RORG	R		I/O	ORIGIN FOR ROTATIONAL ONSET FLOW
C.RUNTYP	I	/GLOBAL/	I/O	
C.RVEC	R		I/O	AXIS FOR ROTATIONAL ONSET FLOW
C.SECMET	R	/GLOBAL/	I/O	
C.SLDF	I	/NETWK/	I/O	
* C.SMOOAB	I	/ABUT/		I/O
C.SNGTYP	I	/NETWK/	I/O	
C.SOLID	R		I/O	SOLUTION IDENTIFIER
C.SPFLG	I	/NETWK/	I/O	
C.SUPSUB	R	/GLOBAL/	I	
C.TRDMET	R	/GLOBAL/	I/O	
C.TRNGTL	R	/NETWK/	I/O	
C.UNIF	R		I/O	UNIFORM ONSET FLOW
C.UPDATN	I	/NETWK/	I/O	
* C.USABUT	I	/ABUT/		I/O
C.WAKSOL	I	/NETWK		I/O
C.WEABUT	I	/ABUT		O

Figure 1.10 - Portion of Glossary of Subroutine DIPDAT of the DQG Module

```
C.        WEABUT(J,I)              CONSISTENT ABUTMENT DESCRIPTION
C.                                 J FROM 1 TO 15 NETWORKS IN ABUTMENT
C.                                 (J,1) - NETWORK INDEX
C.                                 (J,2) - EDGE INDEX
C.                                 (J,3) - START CORNER POINT INDEX-I
C.                                 (J,4) - START CORNER POINT INDEX-J
C.                                 (J,5) - END CORNER POINT INDEX-I
C.                                 (J,6) - END CORNER POINT INDEX-J
C.        NBRNAB                   NUMBER OF NETWORKS IN ABUTMENT
C.
C.        POSABT                   Plane of symmetry (POS) flag indicating
C.                                 that the POS is part of an abutment
C.                                 Value = 0          if no POS
C.                                       = 1          if 1st POS
C.                                       = 2          if 2nd POS
C.
C.        USABUT(I,J)              USER ABURMENT DESCRIPTION
C.                                 SEE DESCRIPTION OF WEABUT ARRAY
C.        SMOOAB                   SMOOTH ABUTMENT FLAG
C.
C.        GAPSET                   FLAG INDICATING GAP-PANELS ADDED
C.                                 TO ABUTMENT.
C.
      COMMON /ABUT/IABUT(20,8),WEABUT(5,6),NBRNAB
     1,IESABT(6),NUMBER(5),ASSINF(5,3,3),QTRCRD(5,3,3),
     2TWEABUT(10,6),EDGPOS(4),USABUT(5,6),DSMTCH(3,2),
     3SMOOAB,POSABT,GAPSET
      INTEGER POSABT,GAPSET
      INTEGER WEABUR,TWEBUT,EDGPOS,USABUT,SMOOAB,DSMTCH
```

Figure 1.11 - Excerpt from Common Block /ABUT/in Subroutine DIPDAT of the DQG Module

COMMON BLOCK	DATABASE	MAP NAME	DATASET NAME	PROGRAM/ SUBROUTINE
/RUINDS/	MEC	IDS	DATA-BASE-HEADER	OPENER
* /ABUT/	DIP	USABIN	USER-ABUT	DIPDAT
/COORDS/	DIP	PAN-COR-PT	PANEL-COORDS	DIPDAT
Dynamic	DIP	PRINT-OPT	GLOBAL-PRINTS	DIPDAT
/GENBCD/	DIP	DIPCLOSDAT	CLOS-COND	BNDYIN
/GENBCD	DIP	CGBCMP	COEF-GEN-BC	BNDYIN
/GENBCD/	DIP	TVECTCOEFF	TANG-VEC	BNDYIN
/GLOBAL/	DIP	GLOBAL-IN	GLOBAL	DIPDAT
/NETBDC/	DIP	NETBDC	NETWK-BDC	BNDYIN
/NETWK/	DIP	NETMAP	NETWK-SPEC	DIPDAT
/ABUT/	DQG	DIPDAT	USABUT	USER-ABUT
/CLOSUR/	DQG	CLOSDIN	CLOSURE-DATA-IN	BNDYIN
Dynamic	DQG	COORDS-GEN	PANEL-CORNER-COORDS	DIPDAT
/GLOBAL/	DQG	GLOB-DYN	GLOBAL	DIPDAT
/NBCDIN/	DQG	CLASS5	CLASS-5-BC-DATA	BNDYIN
/NBCDIN/	DQG	BCDATIN	NETWK-BNDRY-CONDN-IN	BNDYIN
/NETWK/	DQG	NETMAP	NETWK-SPEC	DIPDAT

Figure 1.12 - DQG Database Communication Chart, Third Form for (1,0) Overlay

(a) MAP FROM DIP DATABASE TO DQG MODULE

```
C   BEGIN MAP USABIN
    CALL DSMAP(10HUSABIN          ,20HUSER-ABUT                    ,DIPDBD)
C
C   DEFINE STATIC MAP
    CALL SVMAP(NBRNAB                      ,20HNO-NET-ABUT
   1           USABUT(1,1)                 ,20HNETWK-LIST
   2           USABUT(1,2)                 ,20HEDGE-NMBR                  ,
   3           USABUT(1,3)                 ,20HSTRT-PT                    ,
   4           USABUT(1,4)                 ,20HEND-PT                     ,
   5           POSABT                      ,20HPOS-FLAG                   ,
   6           SMOOAB                      ,20HEDGE-TREAT                 ,
C

C   DEFINE DYNAMIC MAP
    CALL DVMAP(20HABUT-INDEX                )
C
C   END OF MAP
    CALL ENDMAP
C
```

(b) MAP FROM DQG DATA BASE TO DQG MODULE

```
C   BEGIN MAP USABUT
    CALL DSMAP(10HUSABUT          ,20HUSER-ABUT                    .DQGDBD)
C
C   DEFINE STATIC MAP
    CALL DVMAP(20HINDEX             )
    CALL SVMAP(NBRNAB                      ,20HNMBR-NETWK-IN-ABUT         ,
   2           USABUT(1,1)                 ,20HNETWK-ID                   ,
   3           USABUT(1,2)                 ,20HEDGE-NMBR                  ,
   5           USABUT(1,3)                 ,20HSTRT-CRNR-PT-NMBR-I        ,
   6           USABUT(1,4)                 ,20HSTRT-CRNR-PT-J             ,
   7           USABUT(1,5)                 ,20HEND-CRNR-PT-NMBR-I         ,
   8           USABUT(1,6)                 ,20HEND-CRNR-PT-J
   9           SMOOAB                      ,20HSMOOTH-ABUT-FLAG           )
C
C   END OF MAP
    CALL ENDMAP
C   BEGIN MAP GLOB-DYN
```

Figure 1.13 - Maps of dataset USER-ABUT from DIP and DQG Data Bases to DQG Module

DOCUMENTATION PROCESS
TO FOLLOW DATA FLOW

Fig 1.7 DIP MASTER DEFINITION

↓

Fig 1.9 DQG DATA BASE COMMUNICATION CHART FIRST FORM

↓

Fig 1.13(a) SDMS MAP FROM DIP DATA BASE, SUBROUTINE DIPDAT OF DQG MODULE

↓

Fig 1.10 GLOSSARY OF SUBROUTINE DIPDAT OF DQG MODULE

↓

Fig 1.11 COMMON BLOCK/ABUT/ DESCRIPTION

↓

Fig 1.12 DQG DATA BASE COMMUNICATION CHART THIRD FORM

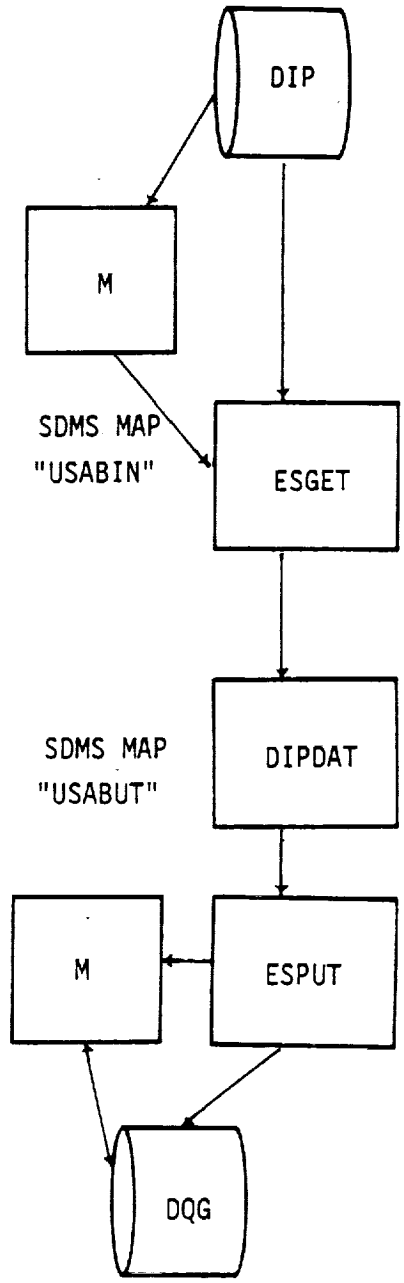
↓

Fig 1.13(b) SDMS MAP FROM DQG DATABASE TO SUBROUTINE DIPDAT OF DQG MODULE

↓

Fig 1.8 DQG MASTER DEFINITION

DATA FLOW
WITHIN PROGRAM DQG



DIP DATA BASE

ESGET OPERATION

PROCESS ABUTMENT INFORMATION

ESPUT OPERATION

DQG OPERATION

Figure 1.14 - Summary of Example Data Flow Analysis

APPENDIX 1-A SUMMARY OF PAN AIR MODULES

1-A.1 MEC - Module Execution Control

MEC creates a temporary database named, MEC for use by other PAN AIR modules. It contains database information on databases used or created by the other modules. Run identification is also processed and stored in the database. Codes are set to indicate whether databases are used, in existence or saved.

User directives for modifying the database information table are processed by MEC and appropriate modifications to the MEC database are made.

1-A.2 DIP - Data Input Processor

1-A.2.1 Purpose

The DIP module reads user input data which describes the PAN AIR problem and stores the data on the DIP database.

1-A.2.2 Tasks Performed

Following the execution of the MEC module, the DIP module accesses the MEC database to read the type of PAN AIR problem to be run. From this dataset, DIP can determine whether a new or updated database is to be created from the inputs. The possible options, described in detail in Section 4.3.2 of Reference 2, are as follows:

1. Creation run - no preexisting database.

2. Post processing run - use existing database and update only directives to it.

3. Right-hand-side update-run - use existing database and update only "solution data."

4. IC update run - use existing database and update geometric data.

The input data is read in free field format from card images. Each card image is read, printed and processed. The data is organized and stored on the DIP database. The initial input data for DIP should contain global data to described the boundary value problem and global defaults, network data to describe the surface definition and boundary conditions, and the geometric edge matching data to describe network edge matching. The above data (original or updated) is required for solving a potential flow solution.

The post processing input data for DIP may contain post-solution calculation cases and database output directives. Both of these types of data require a preexisting DIP database plus the results of a potential flow solution on the database produced by the MDG module.

1-A.3 DQG - Defining Quantities Generator

1-A.3.1 Purpose

The Defining Quantities Generator computes and defines a large number of intermediate quantities required for solution of the potential flow problem. These quantities fall into three classes: control data, geometrical data and boundary condition data.

The control data consists of indices of all singularity parameters and control points in the configuration as well as an indication of those singularity parameters that are "known" and those singularity parameters and control points that are "null" (not used to solve the problem).

The geometrical data includes descriptions of network abutments and abutment intersections, the coefficients of the source and doublet splines that define the singularity strengths over the surfaces of the networks and those geometrical properties of panels which are required to compute the AIC matrix in module MAG.

The boundary condition data processing includes assignment of user specified boundary conditions as well as automatic imposition of doublet matching conditions at network boundaries.

All of the data are stored on the DQG database. A small amount of printed data is available to the user through selection of certain print options in the input to DIP.

DQG also analyzes the configuration for many types of errors which may lead to an erroneous or singular solution and produces diagnostic information that the user might use to correct his input to DIP.

1-A.3.2 Tasks Performed

The basic tasks of DQG are performed in the six primary overlays of DQG. (A seventh primary overlay performs some useful but perfunctory communication to the user.) In the first overlay, data from the DIP database is read, copied and (in some cases) transcribed onto the DQG database. In the second, the data associated with individual networks are defined. Also included are error checks on network size and indexing of singularity parameters and control points. The third overlay of DQG deals with the inter-relationship of networks with each other: abutments and abutment intersections. User defined abutments are imposed and a search is made for any additional abutments in the configuration. A determination is made of network edges and corner points where doublet matching boundary conditions will be imposed. If additional paneling is required to fill in gaps between network edges, gap filling panels are generated. Also network overlaps are found, if any, and diagnostics are given as printed output. The fourth overlay assigns the appropriate number and type of boundary conditions at each control point in the configuration. The fifth overlay constructs source and doublet spline vectors for networks. The sixth overlay computes panel geometrical data, assembles spline matrices describing source and doublet strength over the surface of the panel and computes the moments of source and doublet strength over the surface of the

panel. The seventh overlay produces printed output of control point data and boundary condition data.

1-A.4 MAG - Matrix Generator

1-A.4.1 Purpose

The Matrix Generator module uses output from the DQG database to generate influence coefficients, incorporate symmetry constraints, assemble the influence coefficient (IC) matrix, and perform operations related to the transformation of the boundary value problem into systems of simultaneous linear equations.

1-A.4.2 Tasks Performed

The singularity and control point data from DQG are grouped into categories of updatable and non-updatable. In addition, the singularity data is further divided into known and unknown partitions. The new grouping of data is put into two directories relating DQG data and MAG data. The directories are stored in the MAK database. A number of matrices are formed from the DQG data. First, the panel geometry specifications and the reformatted control point data are obtained from the DQG and MAK databases respectively. The panel influence coefficients (PIC) are then formed from complex computations defined in Section 4.2.2 of the PAN AIR Theory Document (Reference 1). These PIC matrices are symmetrized to form the entries of the IC matrices. These IC matrices are stored temporarily. Next, the IC matrices in required row form (up to 5000 words long) are produced. The aerodynamic influence coefficients (AIC) are then constructed from the boundary conditions specified by DQG and the IC matrices. The AIC matrices which correspond to the known and unknown singularities are stored in the MAK database. Finally, the influence coefficients (IC's) needed by the MDG module are transferred from the temporarty database to the MAK database.

1-A.5 RMS - Real Matrix Solver

1-A.5.1 Purpose

The Real Matrix Solver (RMS) module decomposes the partition of the AIC matrix associated with the unknown singularity parameters.

1-A.5.2 Tasks Performed

The RMS matrix solution subroutines operate on the matrices in "blocked partitioned format." The major tasks of RMS are to block and decompose the AIC matrices into upper and lower triangular matrices and pivot terms for use in the solution process in the RHS module.

1-A.6 RHS Right-Hand-Side Generator

1-A.6.1 Purpose

The RHS creates the right-hand-side equality constraints for the linear system of equations defining the aerodynamic problem. The constraints are

formed from the boundry conditions and other known quantities. The module also obtains the solutions to the linear system for each control point by forward and backward substitution with the decomposed AIC matric obtained from the RMS module.

1-A.6.2 Tasks Performed

The constraint data for the right-hand-side is obtained from the DIP database and transformed into a usable form by RHS. The transformed constraint data is then stored in a temporary database.

The RHS module also generates the symmetrized right-hand-side matrix consisting of two partitions; those for the known AIC elements and those for the unknown. Using matrix partition algebra and backward substitution on the decomposed AIC matrix, all singularity parameters from all solutions are found.

1-A.7 MDG - Minimal Data Generator

1-A.7.1 Purpose

The Minimal Data Generator module is the primary interface of the upstream PAN AIR modules, DIP, DQG, MAG and RHS, with the post processing PAN AIR Modules, PDP and CDP. It reads geometry, influence coefficient, and singlarity data to generate a minimal database of information at control point and panel grid point locations. This data, used by PDP and CDP, consists of geometric information and basic flow quantities: source and doublet singularities, average potential, average mass flux, and in specific instances, average velocity in three components. All basic flow quantities are stored on the MDG database for all solutions and (if planes of symmetry are present) for all distinct images. (See PAN AIR Theory Document, Sections 5.7.2 and K.1 (Reference 1)).

The minimal database generated by MDG enables PDP and CDP to process data without accessing the DQG, MAK, and RHS databases and have that data available in a convenient format at either control points or panel grid points for a given image and solution.

A-7.1.2 Tasks Performed

MDG opens and checks the condition of the databases from DQG, MAG, and RHS to assure that other upstream modules have executed without errors. It forms the MDG permanent database for the global, network-spec, and solution data sets. For each network, the control points are determined for each panel. The control point and grid point geommtry is output to the MDG database.

The IC-matrices from MAK and the singularities from RHS are postmultiplied to form control point values of average potential, mass flux and velocity in three components if specified by the user. Singularities are reformatted uniformly and unsymmetrized.

Using spline vectors created by DQG, singularity values are obtained at nine defining grid points and five defining grid points for doublet and source

singularities respectively on each panel. Subpanel splines are used to calculate singularity values at control points.

At control point locations where IC values were not calculated, values are calculated from the boundary conditions. If IC's were calculated, the mass flux is calculated from the inner product of these velocities and the control point conormal. The values of average potential, mass flux, velocity, if specified, and singularities at control points are placed on the MDG database.

Potential splines, similar to DQG doublet analysis splines, are calculated to produce values of flow quantities at grid points from values at control points. The same quantities output at control points are output at grid points on each network.

1-A.8 PDP - Point Data Processor

1-A.8.1 Purpose

The Point Data Processor module is designed to compute flow quantities on configuration body and wake surfaces. These surface flow quantities consist of perturbation and total potential, perturbation and total velocities, perturbation, total and normal mass flux, pressure coefficients and local Mach numbers for isentropic, linear, second-order, reduced second order and slender body approximations.

Each of these computed data items is printed out and/or stored on a permanent database for later retrieval as selected by the user. The PDP database is generated only if database storage is requested by the user.

The user options are available to PDP in the DIP database. These consists of computation options for potential, velocity, velocity correction and computation schemes, pressure coefficient and local Mach numbers.

The user has the option of requesting a printed output of the computed quantities for each case.

1-A.8.2 Tasks Performed

The configuration geometry and a minimal set of velocity data (perturbation velocities at points computed from the AIC matrices and the local incremental onset flow velocities etc.) are available to PDP in the MDG database. PDP computes the average and difference velocities at user selected point types for each selected network, image and solutions and uses these data to compute the perturbation and total velocities on each selected surface. The velocities are corrected by PDP by the user selected correction schemes and are then used to compute pressure coefficients and local Mach numbers for the selected rules (isentropic, linear, second order, reduced second order and slender body). Details of the computation of surface flow properties can be found in Section N of the PAN AIR Theory Document (Reference 1).

These flow quantites are written to the output file and/or to the PDP database for later retrieval by the PPP module.

1-A.9 FDP - Field Data Processor

1-A.8.1 Purpose

The Field Data Processor module is designed to compute flow quantities at designated points off the configuration body and along streamlines in the flow field. These flow quantities consist of perturnation and total potential, perturbation and total velocity, perturbation and total mass flux, and pressure coefficients and local Mach numbers for isentropic, linear, second order, reduced second order and slender body approximations. Arc length and time of traversal are to additional flow quantities associated with streamlines.

Each of these computed data items is printed out and/or written to a plot file for later retrieval as selected by the user. The FDP plot file is generated only if requested by the user.

The user options are available to FDP in the DIP database. These consists of computation options for potential, velocity, velocity correction and computation schemes, pressure coefficient and local Mach numbers.

The user has the option of requesting a printed output of the computed quantities for each case.

1-A.9.2 Tasks Performed

The panel defining quantities and the singularity solutions are available to FDP in the MDG data base. For a point off the configuration surface, FDP uses that data to compute the perturbation and total velocity for selected solutions. The velocity is corrected by FDP according to user selected correction schemes and is then used to compute pressure coefficients and local Mach numbers for the selected rules (isentropic, linear, second order, reduced second order and slender body). To compute the points along a velocity or mass flux streamline, FDP uses a predictor-corrector method of integration. A more detailed explanation can be found in appendix P of the PAN AIR Theory Document (Reference 1).

These flow quantites are written to the output file and/or to the FDP plot file.

1-A.10 CDP - Configuration Data Processor

1-A.10.1 Purpose

The Configuration Data Processor is designed to compute forces and moments on configuration body and wake surfaces. The computed forces and moments are printed out and/or stored in a permanent database for later retrieval as selected by the user. The CDP permanent database is generated only is it is requested by the user.

The user options for CDP are obtained from the DIP database and the configuration geomtry and other minimal data are obtained from the MDG database.

1-A.10.2 Tasks Performed

The Configuration Data Processor obtains the processed user input from the DIP database. These consists of lists of user selected networks, solutions, axis systems and configuration options for forces and moments.

The user has the option of requesting printed output and/or storage in the CDP database of the computed force and moment data for each case of options.

The configuration geometry and a minimal set of velocity data are available from the MDG database. The CDP module computes the average and difference velocities on the points of each panel, corrects these velocities according to the user selected correction schemes, and computes the selected pressure coefficients from the velocity in a user-selected preferred direction. These pressure coefficieints are used to compute forces and moments on each panel. The edge forces and the corresponding moments are also computed on user selected network edges.

The computed forces and moments are transformed to user selected axis systems (a maximum of 4) and printed out and/or stored in the CDP database for later retrieval by the user with the PPP module.

The CDP module allows the user to sum forces and moments for all panels in a column, for all columns in a network and for all networks in a configuration. A configuration consists of all selected networks for a particular case. In addition the user may request to sum or accumulate forces and moments for selected configurations of a PAN AIR run.

1-A.11 PPP - Print/Plot Processor

1-A.11.1 Purpose

The Print Plot Processor module extracts user selected information from selected PAN AIR databases and prepares the data in a format suitable for processing by plot programs external to PAN AIR.

1-A.11.2 Tasks Performed

The PPP module extracts user selected data from the DQG, PDP and CDP databases and reformats the information for use in preparing plot files. The data are selected from a menu consisting of geometry data from DQG, point data from PDP, and configuration data from CDP.

APPENDIX 1-B Example of How to Use SDMS

1-B.1 SDMS Example Program

The PAN AIR user, with specific needs not satisfied by the standard PAN AIR output options, may obtain additional information from the permanent databases created during a PAN AIR run. A simple FORTRAN program prepared by the user performs this task. This example illustrates the correct procedure to use to generate such a program.

In this example, the data identified by SDMS names (elements) is loaded into Fortran variables as indicated below:

SDMS Name	Fortran	Map	Dataset Name
NMBER-ACT-NETWK	NBRNET	GLOBAL-MAP	GLOBAL
NETWK-ORDER	NETORD	GLOBAL-MAP	GLOBAL
TOTAL-EDGE-LENGTH	EDGLEN	EDGE-LENG	NETWK-SPEC

The FORTRAN program performs the I/O transfers from the database to central memory by calling the same SDMSLIB subroutines which PAN AIR uses to perform similar operations. The SDMS routines needed to read data from the database are listed below. All of these may be loaded by following the control procedures outlined below. A more detailed discussion of SDMS routines may be found in Section 14 of this document.

Subroutine Table

Name	Action
DBCLOS	Closes the Database
DBOPEN	Opens the Database
DSMAP	Initiates Map definition
DVMAP	Defines Dynamic Map
ENDMAP	Terminates Map definition
ESGET	Gets a specified element set at a specified dataset from the database
ISDMS	Initiates SDMS tables
SVMAP	Defines static map

```
      PROGRAM EXAMPL(INPUT,OUTPUT)
C
C
C   PURPOSE
C   THIS PROGRAM IS AN EXAMPLE OF THE USE OF SDMS ROUTINES
C   TO TRANSFER DATA FROM A DATABASE TO CENTRAL MEMORY
C
C   THIS PROGRAM READS THE DQG DATABASE.
C
C
C   DATA GROUP LOCAL   DIMENSIONED DATA
    DIMENSION NETORD(100),EDGLEN(4)
    DIMENSION DBN(3)
    DIMENSION IWSA(2000)
    DATA INFIL/5LINPUT/,IUTFIL/6LOUTPUT/
    DATA IWSA(1) / 1 /
```

```
C
C   INITIALIZE SDMS TABLES
    CALL ISDMS(IWSA(1),IWSA(2000))
C
C   READ DATABASE DESCRIPTION FROM INPUT FILE
    READ (INFIL,5000)(DBN(I),I=1,3)
    READ (INFIL,5000) DBPW
C
C   IF BLANK NAMES, THEN SET DEFAULT VALUES
       IF (DBN(2).EQ.1H ) DBN(2)=0
       IF (DBN(3).EQ.1H ) DBN(3)=0
       IF (DBPW.EQ.1H ) DBPW=0
C
C   ENDIFC
C   OPEN DATABASE
    CALL DBOPEN(DBN(1),9HPERMANENT,DBPQ,3HOLD)
C
C   DEFINE MAP TO GLOBAL DATASET OF DQG DATABASE
C   TO FIND NUMBER AND ORDER OF ACTIVE NETWORKS
    CALL DSMAP(10HGLOBAL-MAP,20HGLOBAL                        ,DBN(1))
C
C   DEFINE STATIC MAP
    CALL SVMAP(NBRNET,20HNMBER-ACT-NETWK     ,
   1           NETORD(1),20HNETWK-ORDER          )
C
C   TERMINATE MAP
    CALL ENDMAP
C
C   DEFINE MAP TO NETWK-SPEC DATASET OF DQG DATABASE
C   TO FIND EDGE LENGTHS OF NETWORKS
    CALL DSMAP(10HEDGE-LENG ,20HNETWK-SPEC                    ,DBN(1))
C
C   DEFINE STATIC MAP FOR EDGE LENGTH
    CALL SVMAP(EDGLEN(1),20HTOTAL-EDGE-LENGTH    )
C
C   TERMINATE MAP
    CALL ENDMAP
C
C   NOTE IN THE ABOVE MAP A STATIC MAP WAS USED FOR THE
C   DATA ITEMS WHILE A DYNAMIC MAP WAS USED FOR THE KEY
C   SET DATA.  THIS IS NOT REQUIRED.  EITHER METHOD OF
C   MAPPING MAY BE USED FOR EITHER DATA OR KEY SET
C   INFORMATION.  HOWEVER, WE RECOMMEND THE APPROACH
C   AS ABOVE.
C
C   GET NUMBER OF NETWORKS AND ORDER
    CALL ESGET(10HNBRNET     )
C
C   WRITE LABELS AT TOP OF PAGE
    WRITE (IUTFIL,6000)
```

```
C
C   FOR EACH ACTIVE NETWORK DO
C   DO 100  IN=1,NBRNET
       INET=NETORD(IN)
C
C      GET EDGE LENGTH DATA
       CALL ESGET(10HEDGE-LENG ,INET)
C
C      WRITE NETWORK INDEX AND EDGE LENGTHS
       WRITE (IUTFIL,6001) IN,INET, EDGLEN(I), I=1,4)
C
C   ENDDO ON NETWORKS
  100 CONTINUE
C
C   CLOSE DATABASE
    CALL DBCLOS(DBD(1))
C
C   EXIT
C
 5000 FORMAT (8A10)
C
 6000 FORMAT(1H1,7X,3HIN,6X,4HINET,2X,12HEDGE LENGTHS)
 6001 FORMAT (5X,I5,5X,I5,4(2X,IPE10.3))
C   END
```

1-B.2 Efficiency Considerations and SDMS

The Scientific Data Management System (SDMS) used in PAN AIR provides a powerful mechanism for storage and classification of scientific data. Through its use of English descriptions of data elements it allows data classifications which are easily understood. The dataset construction allows grouping of data items in ways which reflect actual use or application instead of ways which force an artifical grouping (e.g., all scalars stored in one fashion, vectors in another).

When SDMS is used to solve complex problems which involve many I/O operations, as in the PAN AIR system, some special consideration needs to be given to SDMS usage to avoid undue I/O cost. To understand these considerations it is necessary to know a little more about the internal operations of SDMS.

An SDMS database consists of four files. The first file is a copy of the Master Definition file and is used to generate maps. The second file contains indexing information (pointers) which describe where on the third and fourth files a particular element set is to be found. The third file contains all of the random access data. The fourth file contains all of the sequential access data.

A two level pointer system is used in SDMS. The top level pointer array indicates which of several second level pointer arrays contains the disk address of the required data. Each SDMS buffer can store a pointer array. One buffer will hold the top level pointer array. Second level pointer arrays can reside in the remaining buffers. When an SDMS operation is performed (e.g. an ESGET), if all of the pointer information is already in core, only one disk access is made to obtain the data. If the second level pointer array is not in core, SDMS reads the second file to obtain the second level pointer required and then reads the third file to obtain the data. The second level pointer array fills any buffer that is empty. If none are available and no write operations have occurred, the second level pointer array overwrites the oldest full buffer. If there have been some write operations, the oldest pointer data in the buffer is written to the database before the new second level pointer data is read in. A similar process occurs if the top level pointer array required is not in core.

Several disk accesses may be required for each SDMS operation. If a small number of available SDMS buffers forces the top and second level pointer arrays to be swapped in from the disk, then up to five accesses may be required for a single SDMS operation. Increasing the number of SDMS buffers can decrease the number of disk accesses per SDMS operation.

If four buffers are available to hold indexing data, there can be at most two SDMS operations to two different datasets within a loop unless multiple disk accesses per SDMS operation can be tolerated.

Fig. 1-B.1 shows an example of inefficient use of SDMS(assuming four available buffers). Within the inner loop there are SDMS requests to five different datasets. The first two "GET" operations fill the buffers. The REPLACE of the Singularity-Map dataset occurs efficiently (one disk access for

the operation) but the next REPLACE overwrites the indexing information for the CONTROL-POINTS data, even though the next operation is the replacement of that same dataset. Overall this loop structure yields four data accesses per SDMS operation.

Fig. 1-B.2 shows a more efficient approach. The CONTROL-POINT DATASET, the BOUNDARY-CONDITION DATASET and the RIGHT-HAND SIDE dataset have been combined into one dataset, the BOUNDARY CONDITION/CONTROL POINT dataset. The replacement of the SINGULARITY MAP dataset has been removed to a separate loop of its own. There are only two different datasets which are accessed in the inner loop. Thus on the average there will be between 1 and 2 disk accesses per SDMS operation. This restructuring of the data and control logic will save a factor of 3 in (I/O) cost over the approach in Fig. 1-B.1.

By far the most efficient I/O operations with random access files occur when the random access data is transferred in a sequential fashion. SMDS has a key set system which indexes elements sets within a dataset. If the element sets are stored in an order in which the last key set index changes most rapidly, this will minimize accesses to the pointer file. If the data are read in a similar order this will also reduce I/O cost. In Fig. 1-B.3 the key set structure is shown for the BOUNDARY CONDITION/CONTROL POINT dataset. Since the panel row index (number of panels in a column) is the inner loop in Fig. 1-B.2, this random access data is being read and written in a sequential manner.

```
FOR EACH NETWORK DO
    GET NETWORK DATASET
    FOR EACH PANEL COLUMN DO
        GET COLUMN OF COORDINATES DATASET
        GET COLUMN OF COORDINATES DATASET
        FOR EACH PANEL IN COLUMN DO
            GET CONTROL-POINT DATASET
            (COMPUTE CONTROL POINT DATA)
            GET SINGULARITY-MAP DATASET
            (MODIFY DATA)
            REPLACE SINGULARITY-MAP DATASET
            REPLACE SINGULARITY-SPEC DATASET
            (ASSIGN BOUNDARY CONDITIONS)
            REPLACE CONTROL-POINT DATASET
            PUT BOUNDARY CONDITION DATASET
            PUT RIGHT-HAND-SIDE DATASET
        ENDDO ON PANELS ON COLUMN
    ENDDO ON COLUMNS OF PANELS
ENDDO ON NETWORKS
EXIT
```

Figure 1-B.1 - Inefficient use of SDMS.
Average of four disk accesses per SDMS operation.

```
FOR EACH NETWORK DO
    GET NETWORK DATASET
    FOR EACH COLUMN OF PANELS DO
        GET A COLUMN OF COORDINATES DATASET
        GET A COLUMN OE COORDINATES DATASET
        FOR EACH PANEL IN THE COLUMN DO
            GET BOUNDARY CONDITION/CONTROL POINT DATASET
            (COMPUTE CONTROL POINT DATA)
            GET SINGULARITY-MAP DATASETS
                (MODIFY DATA)
            REPLACE SINGULARITY-MAP DATASET
            (ASSIGN BOUNDARY CONDITIONS)
            REPLACE BOUNDARY-CONDITION/CONTROL POINT DATASET
        ENDDO OF PANELS IN COLUMN
    ENDDO ON COLUMNS OF PANELS
ENDDO ON NETWORKS

FOR EACH SINGULARITY PARAMETER DO
    GET SINGULARITY-MAP DATASET
    REPLACE SINGULARITY-SPEC DATASET
ENDDO ON SINGULAIRTY PARAMETERS
EXIT
```

Figure 1-B.2 - A More Efficient Approach to the Problem of Figure 1
A maximum of two disk accesses per SDMS operation

```
DATASET BOUND-COND/CONT-PT
    KEY SET
        NETWORK-INDEX
        PANEL-COLUMN
        PANEL-ROW
    END
    ELEMENT SET

    END
```

Fig. 1-B.3 Key Set for Example in Figure 1-B.2

2.0 MODULE EXECUTION CONTROL(MEC) MODULE

2.1 INTRODUCTION

A temporary data base is created by the MEC module for use by the other PAN AIR modules. The data base contains the names, accounts, disk set names, passwords and status of all permanent and temporary data bases used by the other PAN AIR modules. These data base parameters can be modified by means of user supplied data base directives described in Paragraph 6.4 of Reference 2.

The MEC module consists of a top level program with main overlays. The first overlay reads all the user directives. The data base directives are processed first and the data base information table is updated as directed. The executive directives used for problem identification are then processed and stored for future use by the third overlay. The second overlay displays the data base information table and actually stores this data in the MEC data base on disk. The third overlay, which previously generated control cards for Cyber computers, is not invoked.

2.2 MEC OVERVIEW

2.2.1 Purpose of MEC

Originally MEC was intended to generate control cards for the Cyber versions. This function is not used by version 3.0. It has been replaced by CRAY JCL procedures. The remaining task for MEC is the accumulation of information about PAN AIR data bases.

2.2.2 MEC Input/Output Data

The MEC input data includes the data base directives discussed in Paragraph 2.1. The MEC directives are described in Paragraph 6.5 of Reference 2.

The printed output from MEC consists of the data base information table. This table contains the same information which is stored in the data base. An example of the MEC output can be found in Paragraph 8.5 of Reference 2.

2.2.3 Data Base Interface

The MEC module creates a temporary data base which is used by the other PAN AIR modules. The data base contains run identification information, data base information for the other PAN AIR modules and status of the other data bases. The data base information consists of data base default names, actual data base names, user numbers, set names, user id's, Master Definition names, user names for the Master Definition, set names for the Master Definition, user ID names for the Master Definition and passwords. The data base status information includes items such as permanent or temporary, existing or not existing and used or not used during a PAN AIR run.

The DIP module is the only other PAN AIR module which writes on the MEC data base. It passes back to MEC the problem identification information and the user identification information.

2.3 MODULE DESCRIPTION

The high level aspects of the MEC module design is described in this section. The lower level functions are described in Paragraph 2.4. The functional decomposition of MEC is illustrated in Appendix 2-B.

2.3.1 Overall Structure

The overall structure of MEC is depicted in Figure 2.1.

2.3.2 Overlay Descriptions

2.3.2.1 MEC Overlay (0,0)

The top level overlay initializes the data base and other parameters. The module then calls upon the second level overlays READUD and GENDB. Overlay READUD processes the user input directives. The data base information table and the data base are created in the overlay GENDB.

2.3.2.2 READUD Overlay (1,0)

The second level overlay READUD reads all the user directives. The data base directives are processed by the third level overlay (1,1) in PRDATA. The executive directives for solving a PAN AIR problem are processed in the other third level overlay (1,2) PREXEC.

2.3.2.3 PRDATA Overlay (1,1)

The third level overlay PRDATA processes the data base user directives. The available user directives are defined in Paragraph 6.4 of the PAN AIR User's Manual (Ref. 1). Such things as names, user accounts, disk set names, user identification and data base status may be changed. An in-core data base information table is used to store the old and new information.

2.3.2.4 PREXEC Overlay (1,2)

The third level overlay PREXEC processes the executive user directives defined in Paragraph 6.5 of Reference 2. The executive directives refer to standard and non-standard PAN AIR Problem definitions. The commands are stored in-core in a table.

2.3.2.5 GENDB Overlay (2,0)

The second level overlay GENDB displays the data base information in printed form and also stores the table data in the MEC temporary data base for use by the other modules.

2.3.2.6 GENCC Overlay (3,0)

The overlay generated control cards for Cyber versions of PAN AIR. It resides in MEC but it is not invoked.

2.3.3 MEC Data Base

A temporary data base named MEC is created by MEC module. The Master Definition for this data base is described in Appendix 2-D.

2.3.4 MEC Interfaces

2.3.4.1 System Interfaces

The MEC module is accessed by JCL procedures.

2.3.4.2 External Interfaces

The MEC data base is used by all other PAN AIR modules. MEC and DIP are the only modules which write on the MEC data base. The problem identification (PID), and user identification (UID), are the only variables affected.

2.3.4.3 Internal Interfaces

The interfaces between the overlays and the subprograms is defined by a tree structure diagram in Appendix 2-A.

2.3.5 Data Flow

The flow of execution is depicted in Figure 2.2. During execution data flows between overlays, subprograms, data bases and disk files. Figure 2.2 depicts this activity in a general way. Detailed data flow information can be found by consulting the glossaries of those programs/subprograms which are of interest. Also, Appendix 2-C has been included to aid analysis of data flow between MEC and its temporary data base. Section 1, Paragraph 1.4 of this document can be consulted for more detailed information of the use of the tools available for analysis of data flow.

2.4. LOWER LEVEL FUNCTIONS

The following paragraphs present the functional decompositions (hierarchial structure) of the overlays and their subprograms and gives the purpose of each subroutine.

2.4.1 Functional Decomposition

See Appendix 2-B for a description of the MEC functional decomposition. Section 1, Paragraph 1.4.1 of this document can be consulted for more detailed information of the use of the functional decompositions.

2.4.2 Subroutine Descriptions

The subroutines used in the MEC module are described below. Also refer to the tree structure in Appendix 2-A. The subroutines called by GENCC are included with PAN AIR but are not invoked by version 3.0. They are not described below.

APPEND

Appends a three character or less suffix to the default name of one or more data bases.

DBASE

Updates the in-core data base information table as prescribed by the user directives for data base modification.

DISBIT

Displays the data base information table in printed form.

KEEP

Processes user directives indicating which data bases should be kept for later PAN AIR problems.

KEYCHK

Selects a portion of an input character string to determine the key portion of the string. The length of the extraction is either three or four leading characters for the PAN AIR software.

PLIMIT

Processes PL directive for setting print limits for PAN AIR modules

PRCHEC

Processes the user directive for a data check. The output is used by CHECK to create control cards (CDC computers only).

STATDB

Alters data base type from PAN AIR default to user specified PERM/TEMP status.

STRWRD

Stores a designated entry into the data base information table.

WRITDB

Transfers the entries from the in-core data base information table to the MEC data base stored on disk.

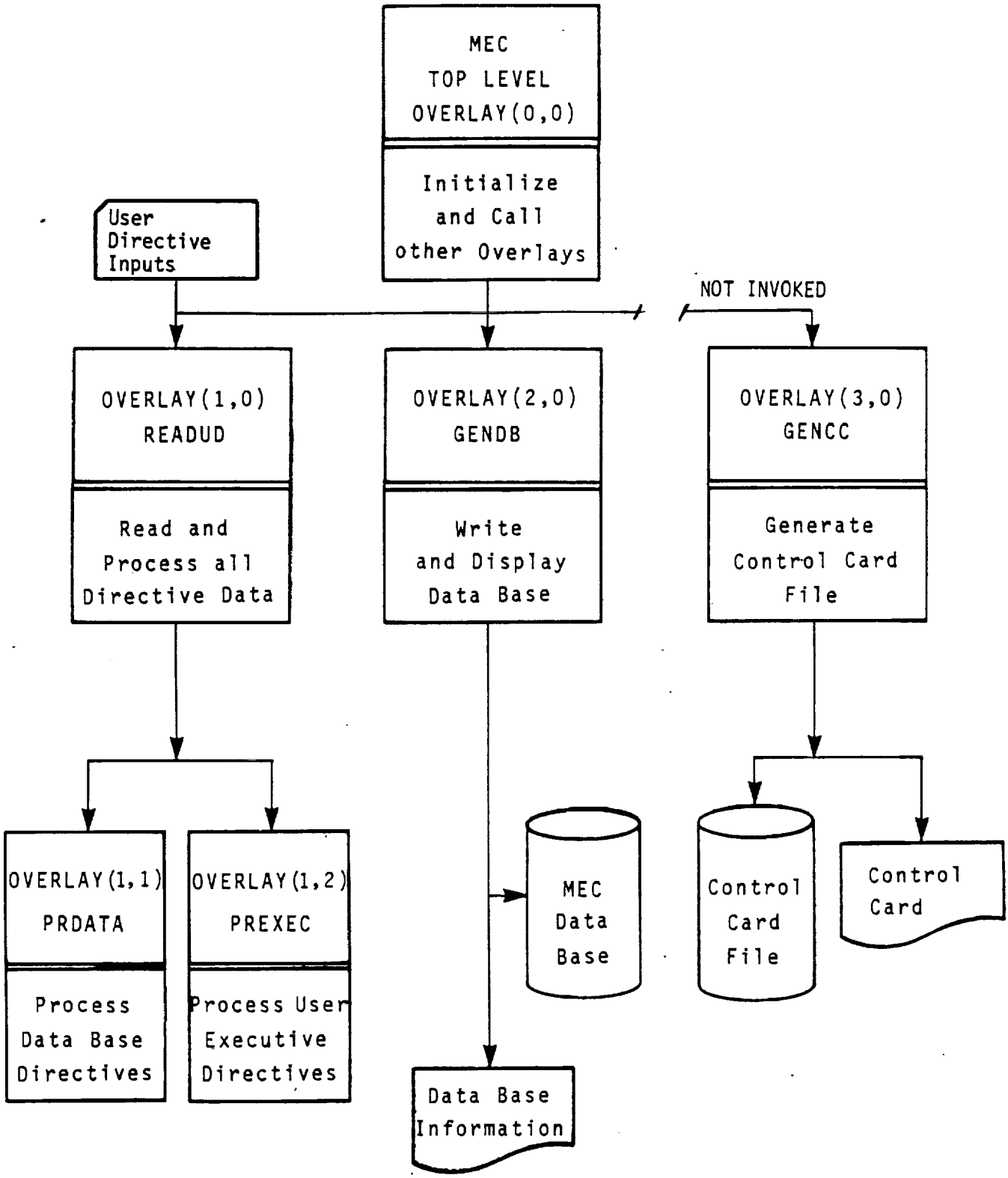


Figure 2.1 - MEC Structure

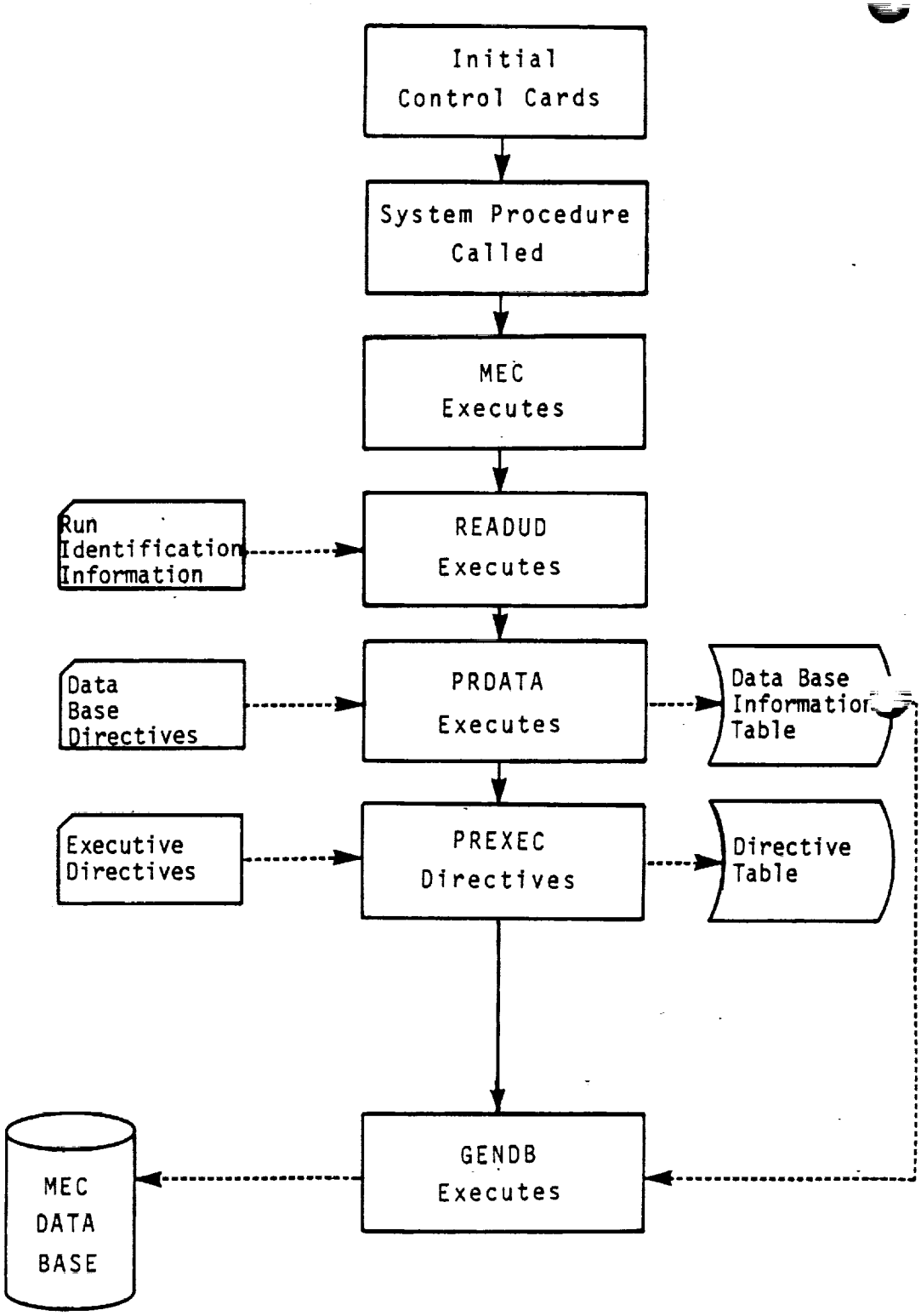


Figure 2.2 - Data Execution Flow

APPENDIX 2-A TREE STRUCTURE

The tree structure diagram of the MEC module has been deleted from this document. It is, however, available on the installation tape.

APPENDIX 2-B MEC FUNCTIONAL DECOMPOSITION

MEC - Module Execution Control

A MEC (0,0) Overlay Initialize MEC Execution
- A PRGEG - Initialize Program Printout
- B ISDMS - Initialize SDMD Execution

B READUD-OVERLAY (1,0) - Read User Directives and Store
- A Initialize READUD
- B LOADREC - Read a record from Input Card File
  - A - If End of File, Set Error Flag and Abort
  - B - STRMOV - Extract Keyword
- C Process Input Record
  - A - Store Run Identification if Present
  - B - PRDATA OVERLAY (1,1) - Process Data Base Directives if "DATA" is keyword
    - A LOADREC - Read Input Record Determine Input Errors and Extract Keyword if Present
    - B KEEP - Record Data Bases to be Saved if "KEEP" is Present Keyword
    - C STRWRD - Record Data Bases to be Dropped if "RELEASE" is Present Keyword
    - D APPEND - Add Suffix to Data Base Name if "APPEND" is Present Keyword
    - E STRWRD - Record Data Base User's Account if "UN" is Present Keyword
    - F STRWRD - Record Data Base ID if "UID" is Present Keyword
    - G STRWRD - Record Data Base Set Names of "SET" is Present Keyword
    - H STRWRD - Record Master Definition User Account if "MUN" is Present Keyword
    - I STRWRD - Record Master Definition ID if "MUID" is Present Keyword
    - J STRWRD - Record Master Defintion Set Name if "MEET" is Present Keyword
    - K STRWRD - Record Password for Data Bases if "PW" is Current Keyword
    - L DBASE - Determine Name and Location of Single Data Base for "DBASE" Keyword
    - M STATDB - Alter status of selected data bases to permanent
    - N STATDB - Alter status of selected data bases to temporary
    - O - Indicate End of Data Base Directives if "END" is Keyword
    - P - Diagnose Unrecognizable Directive and Abort "RUN"
    - Q - Diagnose and take error exit, if number of errors in input exceeds program limit
  - C PREXEC OVERLAY (1,2) - Process all EXEC Directives if "EXEC" is Keyword
    - A LOADREC - Read Input Record
    - B KEYCHK - Determine Keyword

C If Keyword is FIND, Determine which Type
- A If "POTENTIAL" is Present, Store Executive Type as POTENTIAL
- B If IC "UPDATE" is Present, Store Executive Type as IC
- C If "SOLUTION" is Present, Store Executive Type as SOLUTION
- D If 'POST' is present, store execution type as post processing
- E Diagnose Unrecognizable Directive if Detected

D If "FIELD" or "PLOT" are Present, Store in Executive Parameters

E If Keyword is "RUN", Process and Store Module Information

F IF Keyword is "DROP", Process and Store Purge Data Base Information

G If Keyword is "MOUNT", Store Dismount Disk Command Information

H If Keyword is "DISMOUNT", Store Dismount Disk Command Information

I If Keyword is "CC=", Store Control Card Image Information

J If Keyword is "ERROR", Store EXIT Parameters if Present

O If Keywpord is "INPUT", Store File Name

K If Keyword is "END", Record End of EXEC Directives

L Diagnose Unrecongizable Directives if Present

M Diagnose too many Directives

N Diagnose too many Errors on Input

D PRCHEC - Process Data Check Directive if Present
- A - Initialize Data Check Options
- B - If "DQG" is Requested, Record Via Switch
- C - If "PLOTS" are requested, Record Via Switch

E Store SYSTEM Card Parameters if Present for Boeing, Ames, Langley or WPAFB computer installations

F If Keyword is "END", Indicate No More PAN AIR Directives Exist

G If Too Many Input Errors Were Recorded, Print Diagnostic and Abort Run

C GENCC OVERLAY (3,0) - Generate JCL Control Cards for Requested PAN AIR Problem (This code was previously used for Cyber computers and is not invoked)

D GENDB OVERLAY (2,0) - Define MEC Data Base Table
- A DIST - Display MEC Data Base Table
- B WRITDB - Write the Data Base Table on the MEC Data Base

E PRGEND - Terminate the Execution of the MEC Module

APPENDIX 2-C DATA BASE COMMUNICATIONS CHART

The Data Base Communications Chart is presented in three forms. The first form has a column order of Data Base, Dataset Name, Map Name, Common Block, and Program/Subroutine. The second form has a column order of Data Base, Map Name, Dataset Name, Common Block, and Program/Subroutine. The third form has a column order of Common Block, Data Base, Map Name, Dataset Name, and Program/Subroutine. Thus a person can get a cross reference on a data element by knowing either the Dataset Name, Map Name or Common Block.

FIRST FORM

DATA BASE	DATASET-NAME	MAP NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
MEC	DATA-BASE-HEADER	DBHED	/MECDB/	WRITDB
MEC	DATA-BASE-LOCATION	DBLOC	/MECDB/	WRITDB

SECOND FORM

DATA BASE	MAP NAME	DATASET-NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
MEC	DBHED	DATA-BASE-HEADER	/MECDB/	WRITDB
MEC	DBLOC	DATA-BASE-LOCATION	/MECDB/	WRITDB

THIRD FORM

COMMON BLOCK	DATA BASE	DATASET-NAME	MAP NAME	PROGRAM/ SUBROUTINE
/MECDB/	MEC	DATA-BASE-HEADER	DBHED	WRITDB
/MECDB/	MEC	DATA-BASE-LOCATION	DBLOC	WRITDB

APPENDIX 2-D MASTER DEFINITION

The data base master definition listing of the MEC module has been deleted from this document. It is produced from the PAN AIR tape during installation.

3.0 DATA INPUT PROCESSOR (DIP) MODULE

3.1 INTRODUCTION

The DIP module is the input processor for the PAN AIR system. It reads user supplied PAN AIR directives, collects them into related groups of data and stores them on the DIP database for use by other modules in the PAN AIR system. DIP also provides some diagnostic information to the user on the output file. Most of the contents of DIP deals with the recognition of alphanumeric data and the consequent storage of the data. DIP has two modes of operation. The first mode of operation is invoked to define a new problem. The second mode of operation involves an "update" or change in the parameters describing a previously executed problem. User supplied directives to module MEC (see Paragraph 6.3 of Reference 2) determine the mode of DIP operation.

3.2 DIP OVERVIEW

3.2.1 Purpose of DIP

The DIP module reads the user's description of the problem and stores it for use by other modules. The DIP module consists of a top level program which calls from two to seven primary overlays. The first primary overlay performs the module initialization function. The second primary overlay reads and loads the global data. The third primary overlay reads and loads the network data. The fourth primary overlay reads and loads the geometric edge matching data. The fifth primary overlay reads and loads the flow properties calculation data. The sixth primary overlay reads and loads the data printout directives. The seventh primary overlay performs the module termination function. All PAN AIR input data, except execution control directives, are read by the DIP module. The data is checked for accuracy and loaded into the DIP data base for use by subsequent modules.

3.2.2 DIP Input/Output Data

The DIP module receives input from three sources. The first is the MEC data base which provides DIP with problem identification, user identification and run mode. The run mode will indicate that DIP is either to generate a new data base or use an old data base. The second source of input is the old DIP data base if this is an update or follow-on run. The final source of input is the user supplied input data for DIP.

The DIP module produces a printout of each input record (card) read, followed by any diagnostics associated with the record. The printed output also contains a summary of the global data, a list of the solutions and a summary of the networks.

3.2.3 Data Base Interface

The DIP module creates/updates a DIP data base which is used by the other PAN AIR modules. This data base contains the flow regime data, configuration data, a list of networks plus individual network data, and a list of solutions plus individual solution data. It also contains the DIP global defaults, flow properties data, PAN AIR module print flags, and data printout directives.

The DIP module also writes the problem identification and user identification on the MEC data base.

3.3 MODULE DESCRIPTION

3.3.1 Overall Structure

The main overlays of DIP are briefly summarized in this paragraph. Lower level subroutines are described in Paragraph 3.4. The DIP functional decomposition and a chart of the subroutine tree diagram are presented in Appendices 3-B and 3-A, respectively. The overall structure of DIP is depicted in Figure 3.1.

3.3.2 Overlay Descriptions

3.3.2.1 DIP Overlay (0,0)

The top level overlay initializes the data base and default parameters by calling Overlay (1,0) (Program INITIL). The module then responds to input data, calling overlays GLOBDP, NETWDP, GEOMDP, FLOWDP, and PPPDIR. The global data is processed by GLOBDP, the network data is processed by NETWDP, the geometric edge matching data is processed by GEOMDP, the flow properties data is processed by FLOWDP and the data printout directives are processed by PPPDIR. At the completion of input data, the module calls overlay FINIS.

3.3.2.2 INITIL Overlay (1,0)

The second level overlay INITIL (Figure 3.2) opens the data base and reads the data base header and the run options. The DIP data base is then checked and opened. If the MEC run options indicate an update run, INITIL reads the global level data sets from the DIP data base into core.

3.3.2.3 GLOBDP Overlay (2,0)

The second level overlay GLOBDP (Figure 3.3) is called in response to a "BEGIN GLOBAL DATA" input record. It processes all of the global data input by the user. Data transmitted to the DIP data base consists of header data, global defaults, and global prints. Data transmitted to the MEC data base is header data.

3.3.2.4 NETWDP Overlay (3,0)

The second level overlay NETWDP (Figure 3.4) is called in response to a "BEGIN NETWORK DATA" input record. It processes all of the network data input by the user. Data transmitted to the data base consists of individual network data for panel coordinates and constraints.

3.3.2.5 GEOMDP Overlay (4,0)

The second level overlay GEOMDP (Figure 3.5) is called in response to a "BEGIN GEOMETRIC EDGE MATCHING" input record. It processes the edge matching (abutment) data. Data transmitted to the data base consists of the user - defined abutments.

3.3.2.6 FLOWDP Overlay (5,0)

The second level overlay FLOWDP (Figure 3.6) is called in response to a "BEGIN FLOW PROPERTIES CALCULATIONS DATA" input record. Surface flow properties are processed by the third level overlay SURFLO. Forces and moments are processed by the third level overlay FORMOM.

3.3.2.7 SURFLO Overlay (5,1)

The third level overlay SURFLO (Figure 3.6) is called in response to a "SURFACE FLOW PROPERTIES" input record.

3.3.2.8 FFDATA Overlay (5,2)

The third level overlay FFDATA (Figure 3.6) is called in response to a "FIELD FLOW PROPERTIES" input record.

3.3.2.9 FORMOM Overlay (5,3)

The third level overlay FORMOM (Figure 3.6) is called in response to a "FORCES AND MOMENTS" input record.

3.3.2.10 PPPDIR Overlay (6,0)

The second level overlay PPPDIR (Figure 3.7) is called in response to a "BEGIN PRINT PLOT" input record. The data group processed by this overlay specifies point options for the Print/Plot Processor (PPP) module. Geometry print options are processed by the third level overlay PPGEOM. Flow properties at points print options are processed by the third level overlay PPPOIN. Force and moment data for surface configurations is processed by the third level overlay PPCONF.

3.3.2.11 PPGEOM Overlay (6,1)

The third level overlay PPGEOM (Figure 3.7) is called in response to a "GEOMETRY DATA" input record. The input record set processed by this overlay specifies the print files that PPP will create from DQG data.

3.3.2.12 PPPOIN Overlay (6,2)

The third overlay PPPOIN (Figure 3.7) is called in response to a "POINT DATA" input record. The input record set processed by this overlay specifies the print files that PPP will create from PDP data.

3.3.2.13 PPCONF Overlay (6,3)

The third level overlay PPCONF (Figure 3.7) is called in response to a "CONFIGURATION DATA" input record. The input record set processed by this overlay specifies the print files that PPP will create from CDP data.

3.3.2.14 FINIS Overlay (7,0)

The second level overlay FINIS (Figure 3.8) is called in response to a "END PROBLEM" input record or an END-OF-FILE mark on the input file. This overlay writes the global and global flow data sets to the DIP data base and closes the DIP data base. It then closes the MEC data base.

3.3.3 DIP Data Base

DIP creates one permanent data base. The Master Definition is described in Appendix 3-D.

3.3.4 DIP Interfaces

3.3.4.1 System Interfaces

The DIP module is accessed through MEC by user control cards and a system procedure. This interface is described in Sections 1.0 and 2.0 of this document.

3.3.4.2 External Interfaces

The DIP data base is used by all other modules. DIP is the only module which can write on the DIP data base. DIP also writes on the MEC data base. The problem identification (PID), and user identification (UID), are the only variables affected.

3.3.4.3 Internal Interfaces

The interfaces between the overlays and the subprograms are defined by a tree structure diagram in Appendix 3-A.

3.3.5 Data Flow

The flow of execution is depicted in Figure 3.9. During execution, data flows between overlays, subprograms and data bases via labeled common blocks. Figure 3.9 illustrates this activity in a general way. Detailed data flow information can be found by consulting Figures 3.2 through 3.8, Appendix 3-C (Data Base Communications Chart) , and the glossaries of the programs/subroutines which are of interest.

3.4 LOWER LEVEL FUNCTIONS

The following paragraphs describe the general structure and purpose of the overlays and their subprograms.

3.4.1 Functional Decomposition

See Appendix 3-B for a description of the DIP decomposition.

3.4.2 Subroutine Descriptions

3.4.2.1 Subroutines from GLOBDP - Overlay (2,0)

ADDE

Processes the "ADDED MASS COEFFICIENTS" input record. This record may contain a moment reference point, but CDP apparently does not use it.

The input data is loaded into DIP data set GLOBAL-FLOW-PROP.

AMCGLR

Generates the 6 solutions required for added mass coefficient calculations by CDP, checks Mach number and checks for symmetric planes of symmetry. The generated data is loaded into DIP data set GLOBAL.

CHEC

Processes the "CHECKOUT PRINTS" input record. This record contains a parameter list of one or more abbreviated module names, each followed by its own list of integer print options. The input options are loaded into DIP data set GLOBAL-PRINTS.

CONF

Processes the "CONFIGURATION" input record. This record contains the configuration and flow symmetry data. The input options are loaded into DIP data set GLOBAL.

GLDAPR

Transforms the WM (magnitude of rotational flow) and WDC (direction cosines of axis of rotation) into the rotational flow vector for all new solutions.
Prints the global data, including new solutions, if the DIP global data print flag is set true. This flag is set in CHEC in response to DIP option "3."

GLOOPT

Processes the solution update parameter on the "BEGIN GLOBAL DATA" input record. The options are:

NEW (DEFAULT)	-	no updates.
REPLACE	-	purge solution data from previous run(s).
UPDATE	-	old solution data can be selectively updated. No new solutions may be defined. Solution idents remain fixed.

GLOSOL

Processes the option 2 input records for global solution data. This option introduces data by columns. An example is:

```
ALPHA = .2 ,      .3 ,      .5
```

The input data is loaded into DIP data set GLOBAL.

IDCNCV

Checks for missing input for Problem ID, User ID, configuration symmetry and flow regime data. Load default values for all missing items just listed. This routine is only called during a creation run. None of the above items are updatable. The defaults are written into the DIP data set GLOBAL.

MACH

Processes the flow regime definition record. This record defines the freestream Mach number and the direction of compressibility effects. Angles are input in degrees. Examples:

```
MACH = 1.2,  CALPHA = 2.0,  CBETA = .05
MACH = 1.2,  CALPHA = 2.0
```

Parameter defaults:

```
MACH = 0.,  CALPHA = 0.,  CBETA = 0.
```

The input data is loaded into DIP data set GLOBAL.

OP1DAT

Processes the option 1 data input records for global solutions. This option introduces data by rows. Examples:

```
.2    SOL-1
.3    SOL-2
.5    SOL-3
```

See OP1HED for headers.

The input data is loaded into DIP data set GLOBAL.

OP1HED

Processes the option 1 header input record(s) for global solutions. This option introduces data by rows. Example:

```
ALPHA SID
```

See OP1DAT for data.

The input data is loaded into DIP data set GLOBAL.

PIDUID

Processes the "PID" and "UID" record types. Examples:

```
PID = THIS IS A SAMPLE PROBLEM ID
UID = THIS IS A SAMPLE USER ID
```

The input data is loaded into DIP and MEC data sets DATA-BASE-HEADER.

RVPFIL

Provides reference velocity for pressure defaults and ratio of specific heats defaults for solutions, when necessary.

The input data is loaded into DIP data set GLOBAL-DEFAULTS.

SOLFIL

Provides default solution data as required. If no solutions were defined (creation run only), generate a single solution with the values indicated below. Parameter defaults for solution data:

ALPHA	BETA	UINF	WM	WDC	WCP	SID
0.	0.	1.	0.	0.,1.,0.	0.,0.,0.	(2 blank words)

SOLTRN

Transforms the alpha, beta and unif (magnitude of uniform onset flow) into the uniform onset flow vector for all new solutions. The results are loaded into DIP data set GLOBAL.

3.4.2.2 Subroutines from NETWDP - Overlay (3,0)

BCSTEC

Checks user inputs and load defaults when required for the following types of network data:

- Supplement record duplication checks;
- Boundary condition class and subclass input;
- Method of velocity computation;
- Singularity types;
- Edge control point data;
- Closure input for edges;
- No doublet edge matching; and
- Adjacent edge check for control point edges.

BOUN

Processes the Boundary Condition Specification record for networks. Examples:

```
BOUNDARY CONDITION = OVERALL , 1, 3
BOUNDARY CONDITION = LOCAL , 1, 4
```

The class and subclass data is loaded into DIP data set NETWK-SPEC.

CBC123

Defines defaulted general boundary condition coefficients for classes 1, 2, and 3. Check user inputs of specified flows for classes 2 and 3. Check user inputs of tangent vectors for class 3.

CHKBC4

Checks user inputs of constraint data for a boundary condition class 4 problem.

CHKBC5

Checks user inputs to determine if the boundary condition coefficient terms for RHS tangential (term indices 15 and 30) have been specified. If user did not input these terms, define default term with a value of -1 for first and second equations.

CLDATA

Processes the values for the closure edge boundary condition data set for a network. The closure values may appear as a floating point value, an array of values, or as indexed input. Indexed input starts with a left paren as follows:

```
(row, column) = value
```

CLOS

Processes the closure edge boundary condition data set for a network. It recognizes the following record types:

```
TERM =

     SOLUTIONS =

          values
```

This routine is responsibile for loading data into DIP data sets CLOS-COND and NETWK-BDC.

COEF

Processes the coefficients of general boundary condition equation data set for a network. It recognizes the following record types:

```
TERM =

     SOLUTIONS =

          POINTS =

               values
```

This routine is responsible for loading data into DIP data sets COEF-GEN-BC and NETWK-BDC.

GRID

Processes the network grid point data which follows the network ID record. Each point is defined in triplet form (X, Y, Z) and must not spill across record boundaries. Each data set contains one complete grid column.

The data is loaded into DIP data set PANEL-COORDS.

LOCA

Processes the local incremental onset flow data set for a network. It recognizes the following record types:

```
TERM =
     INPUT-IMAGES =
          SOLUTIONS =
               POINTS =
                    values
```

This routine is responsible for loading data into DIP data sets LOCAL-FLOW and NETWK-BDC.

METH

Processes the "METHOD OF VELOCITY COMPUTATION" record for network data. Examples:

METHOD OF VELOCITY COMPUTATION = LOWER-SURFACE-STAGNATION

The data is loaded into DIP data set NETWK-SPEC.

NDELDR

Loads general network data defaults in response to the network identifier record. All defaults may be over-written by user inputs.

NECDWR

Writes following network control data sets to DIP data base:

NETWK-SPEC

NETWK-BDC

NETWORK-UPDATE-CODES

NEDAPP

Prints network data for all known networks, including input order number, user label, status (NEW, REPLACED, UPDATED, DELETED, or OLD), boundary condition class and subclass, singularity types, and grid point row and column counts.

NETIDS

Isolates the network ID (if any) found in the parameter list of the network identifier record.

NETOPT

Process the option parameter at the end of the network record. The options are:

DELETE
SOLUTION (IC)
REPLACE
NEW (Default)

NETWID

Recognizes network data records. Examples:

STORE VIC MATRIX
STORE LOCAL INCREMENTAL ONSET FLOW
DELETE REFLECTION IN PLANE OF SYMMETRY
WAKE FLOW PROPERTIES TAG
TRIANGULAR PANEL TOLERANCE =
UPDATE TAG =
BOUNDARY CONDITION =
METHOD OF VELOCITY COMPUTATION =
SINGULARITY TYPES = SA DA
EDGE CONTROL POINT LOCATIONS =
NO DOUBLET EDGE MATCHING =
CLOSURE EDGE CONDITION
COEFFICIENTS OF GENERAL BOUNDARY CONDITION EQUATION
TANGENT VECTORS FOR DESIGN
SPECIFIED FLOW
LOCAL INCREMENTAL ONSET FLOW
NETWORK

NODOUB

Processes the "NO DOUBLET EDGE MATCHING" record for network data. Examples:

NO DOUBLET EDGE MATCHING = 2, 4

NO DOUBLET EDGE MATCHING = 1

The data is loaded into DIP data set NETWK-SPEC.

NOPCHK

Processes the network option for update runs. This option is specified or defaulted in the parameters list of the network identifier record. The option was decoded by routine NETOPT. Also loads network ID for new networks. The ID is loaded into DIP data set GLOBAL.

SING

Processes the "SINGULARITY TYPES" record for network data. Examples:

```
SINGULARITY TYPES   =   NOS, NOD
SING                =   SA,  DA
SING                =   SD1, DD1
SING                =        DW1
SING                =        DW2
```

The data is loaded into DIP data set NETWK-SPEC.

SPEC

Processes the specified flow data set for a network. The following record types are recognized:

```
TERM =
     INPUT-IMAGES =
          SOLUTIONS =
               POINTS =
                    values
```

This routine is responsible for loading data into DIP data sets NETWK-BDC and SPEC-FLOW.

TANG

Processes tangent vectors for design data set for a network. It recognizes the following record types:

```
TERM =
     UNALTERED
          SOLUTIONS =
               POINTS =
                    values
```

This routine is responsible for loading data into DIP data sets NETWK-BDC and TANG-VEC.

UPDA

Processes the "UPDATE TAG" record for network data. Examples:

```
UPDATE TAG = 1, 2, 3, 4
UPDATE TAG
UPDATE TAG = 1
```

The data is loaded into DIP data set NETWK-SPEC.

3.4.2.3 Subroutines from GEOMDP - Overlay (4,0)

ABNEID

Processes the abutment definition records parameter list. The list included the network ID's and their whole or partial edges which form the abutment. Each network ID must be preceded by an equal sign (=). Examples:

```
ABUTMENT  = NETWORK-NO-2, 2, 1, 5 +
          = NETWORK-NO-5, 4, ENTIRE-EDGE +
          = 7           , 4
ABUT = 2, 2, 1, 5 = 5, 4, = 7, 4
```

The data is loaded into DIP data set USER-ABUT.

ABUT

Recognizes the abutment definition record. It also processes the supplement records for planes of symmetry and smooth edge treatment. Examples:

```
ABUTMENT  = 7  , 4

PLANE     = SECOND

SMOOTH EDGE TREATMENT
```

The data is loaded into DIP data set USER-ABUT.

3.4.2.4 Subroutine from FLOWDP - Overlay (5,0)

FLWOPT

Processes the post solution update option on the "BEGIN FLOW PROPERTIES CALCULATION = option" input record. The options are:

```
NEW (Default)  /   All new cases
REPLACE        /   Purge old, all new cases
UPDATE         /   Update old, add new cases
```

3.4.2.5 Subroutines from SURFLO - Overlay (5,1)

FPPOIN

Processes the calculation point locations record for surface flow properties calculations. Examples:

```
POINTS = CENTER-CONTROL-POINTS
POINTS = EDGE-CONTROL-POINTS
POINTS = ADDITIONAL-CONTROL-POINTS
POINTS = ALL-CONTROL-POINTS
POINTS = GRID
```

The data is loaded into DIP data set SURF-FLOW.

SFDELO

Checks surface flow case record type counts to determine if there is any duplication of same, or a missing surface flow properties record. Also load default values for any missing record types.

3.4.2.6 Subroutines from FFDATA - Overlay (5,2)

CVRTVC

Translates velocity correction requests from an input record form to an output database specification form.

CVTPDR

Translates print and database requests from their input form to their output form.

FFDEFA

Uses the defaults for any record in a field flow properties case that has not been specified. If the record cannot be defaulted then the user is warned and the case is dropped.

FFINIT

Initializes the labeled common blocks which describe the allowable record syntax and parameter values. It also maps the record type to the posiition in a labeled common block where its parameter values are stored.

FFOREQ

Processes the parameters specified in the PRINTOUT or DTA BASE records. It reads the compressed list of option selections (by number of keyword) and produces a full option list whose set entries correspond to selected options.

FFSOL

Processes the SOLUTION record by transfering the solution number directly to a local array and by interpreting the solution number from the solution name and transfering it to a local array.

LISPAC

Converts a list of options selected and not selected to a packed list of only those options selected. It can extract two types of packed lists. It does not do word packing.

LOOKUP

Finds the occurance(s) of an item in a list of items.

MARKRC

Records the occurance of a record type and warns if that record type was previously specified.

OBCASE

Controls the handling of off body case records.

OBCLOS

Replaces unspecified records in an off body case with their defaults, converts the input specifications to a form suitable for the Field Data Processor to user, and writes the data to the DIP data base.

OBOPEN

Initializes the defaults for an off body case. For a standard run, thses defaults will include global defaults. For an update run, these defaults will be the values for the previous case.

OFLOAD

Transfers a numerical (integer or real) list of parameters in an input record to a local array.

PDRCVT

Translates print and data base requests from their output form to their input form.

REPARS

Reads and parses the next valid record in the input stream.

SLCASE

Controls the handling of streamline case records.

SLCLOS

Replaces unspecified records in a streamline case with their defaults, converts the input specifications to a form suitable for the Field Data Processor to use, and writes the data to the DIP data base.

SLOPEN

Initializes the defaults for a streamline case. For a standard run, these defaults will include global defaults. For an update run, these defaults will be the values for the previous case.

VCHECK

Checks the validity of the current input record by comparing it with the allowable forms of syntax and values defined by several labeled

common areas initialize by FFINIT.

3.4.2.7 Subroutines from FORMOM - Overlay (5,3)

FMACCU

Processes the "ACCUMULATE" record from the forces and moments data subgroup of the flow properties data group.

FMACDE

Processes the parameter defaults for the forces and moments "ACCUMULATE" record. The data is loaded into DIP data set SURF-FAM.

FMACPL

Processes the parameter list for the forces and moments "ACCUMULATE" record. The data is loaded into the DIP data set SURF-FAM.

FMASDL

Loads the defaults for user selected axis systems. The data is loaded into DIP data set SURF-FAM.

FMASPS

Processes the parameter list on the AXIS SYSTEM record. The data is loaded into DIP data set SURF-FAM.

FMAXSY

Processes the AXIS SYSTEMS record from the forces and moments data subgroup of the flow properties data group.

FMCASE

Processes forces and moments "CASE" records plus 14 supplement record types. The supplement record types are:

NETWORKS-IMAGES
EDGE FORCE CALCULATION
MOMENT AXIS
LOCAL REFERENCE PARAMETERS
SURFACE SELECTION
SELECTION OF VELOCITY COMPUTATION
COMPUTATION OPTION FOR PRESSURES
VELOCITY CORRECTIONS
PRESSURE COEFFICIENTS RULES
RATIO OF SPECIFIC HEATS
REFERENCE VELOCITY FOR PRESSURE
LOCAL PRINTOUT
LOCAL DATA BASE
ACCUMULATE

FMEDFO

Processes the EDGE FORCE CALCULATION record from the forces and moments data subgroup of the flow properties data group. The data is loaded into DIP data set SURF-FAM.

FMGLDE

Initializes supplement (global) record type counts and load global defaults for the forces and moments data subgroup. The default data is loaded into the DIP data set SURF-FAM.

FMLODE

Checks inputs and load defaults as required for case level data in the forces and moments subgroup. The data is loaded into DIP data set SURF-FAM.

FMLOIN

Initializes case level defaults and parameter values. The default data is loaded into DIP data set SURF-FAM.

FMMDAX

Processes the MOMENT AXIS record from the forces and moments subgroup of the flow properties data group. The default data is loaded into DIP data set SURF-FAM.

FMSURF

Processes the SURFACE SELECTION record from the forces and moments data subgroup of the flow properties data group. The data is loaded into the DIP data set SURF-FAM.

3.4.2.8 Subroutine from PPGEOM - Overlay (6,1)

NETDQG

Processes the network ID list on the NETWORKS record for the PPP "GEOMETRY DATA" group. The data is loaded into DIP data set GEOM-PRINT-PLOT.

3.4.2.9 Subroutines from PPPOIN - Overlay (6,2)

NETPDP

Processes the network ID list and corresponding images on the NETWORK-IMAGES record for the PPP "POINT DATA" group. The data is loaded into DIP data set POINT-PRINT-PLOT.

PPARAY

Processes the "ARRAY" record for PPP "POINT DATA". This record indicates grid direction (rows or columns) and point type (control or grid). The data is loaded into DIP data set POINT-PRINT-PLOT.

3.4.2.10 Subroutine from PPCONF - Overlay (6,3)

NETCDP

Processes the network ID list and corresponding images plus panel and/or column-sum options on the NETWORK-IMAGES record for the PPP CONFIGURATION DATA group. The data is loaded into DIP data set CONFIG-PRINT-PLOT.

3.4.2.11 Subroutine from FINIS - Overlay (7,0)

AMCFLR

Responds to an "ADDED MASS COEFFICIENTS" input record at time of DIP termination. Wake networks are eliminated. CDP cases are updated to reflect the 6 new Added Mass Coefficients onset flows (SOLUTIONS).

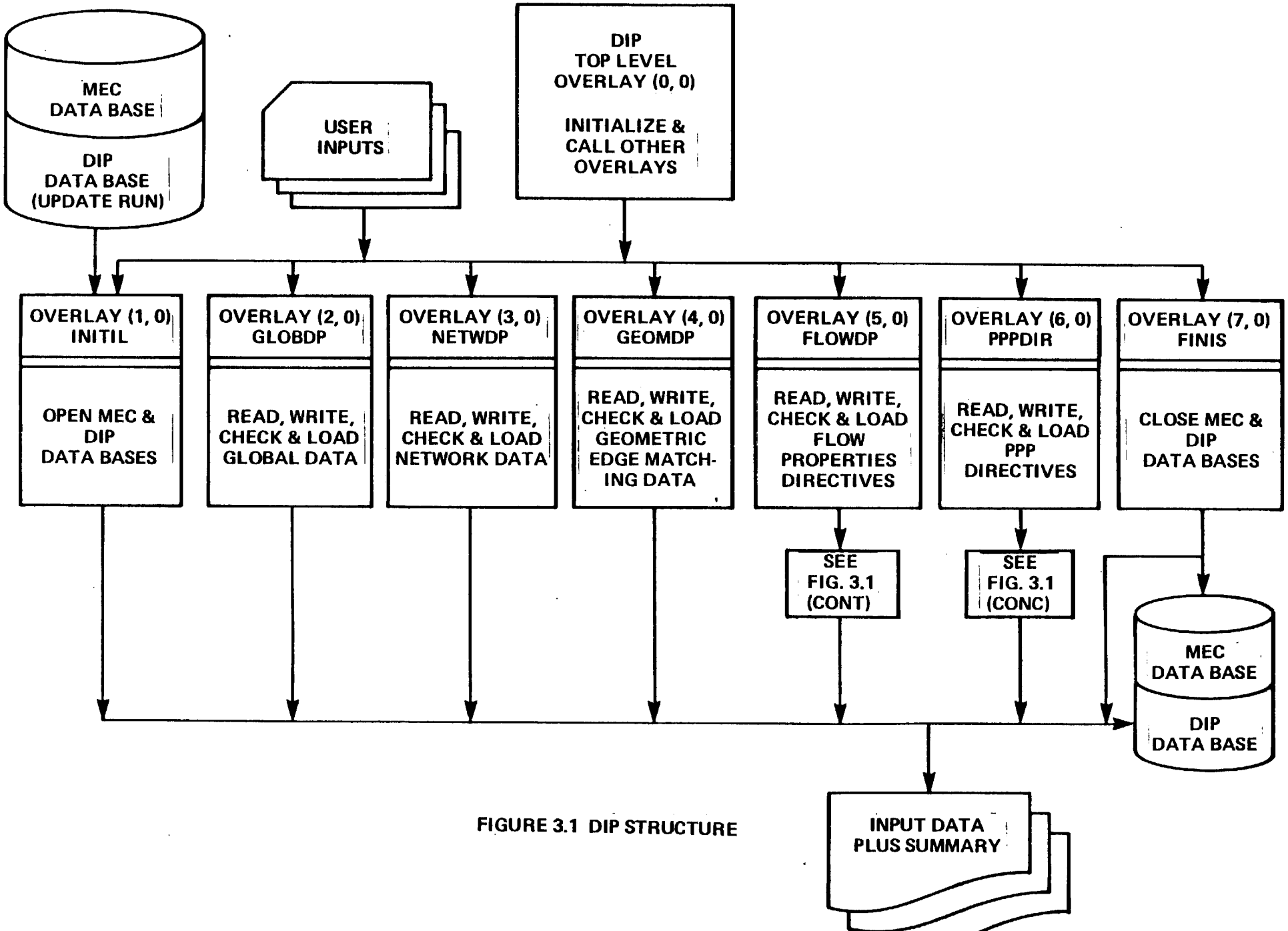


FIGURE 3.1 DIP STRUCTURE

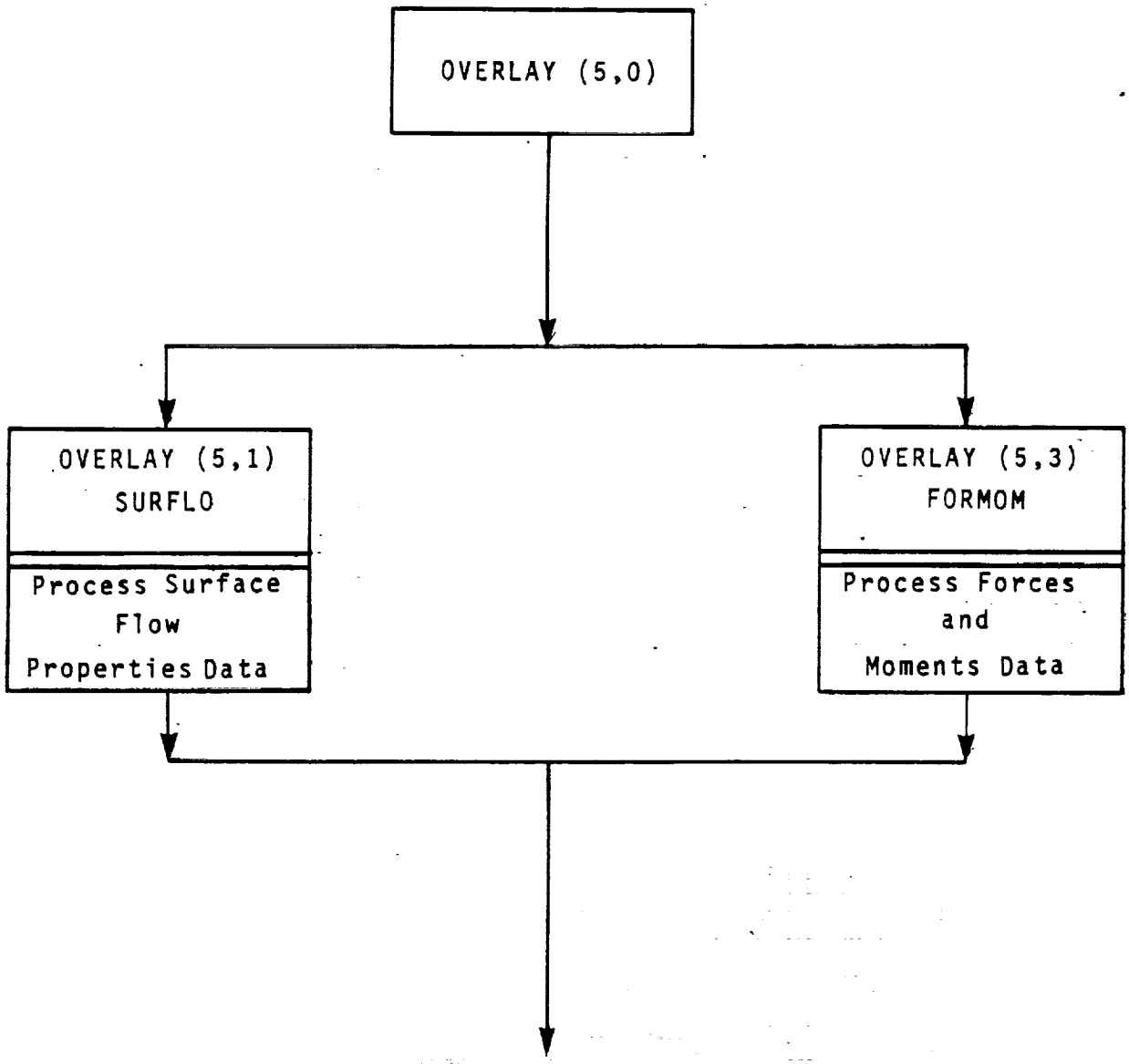


Figure 3.1 - Continued

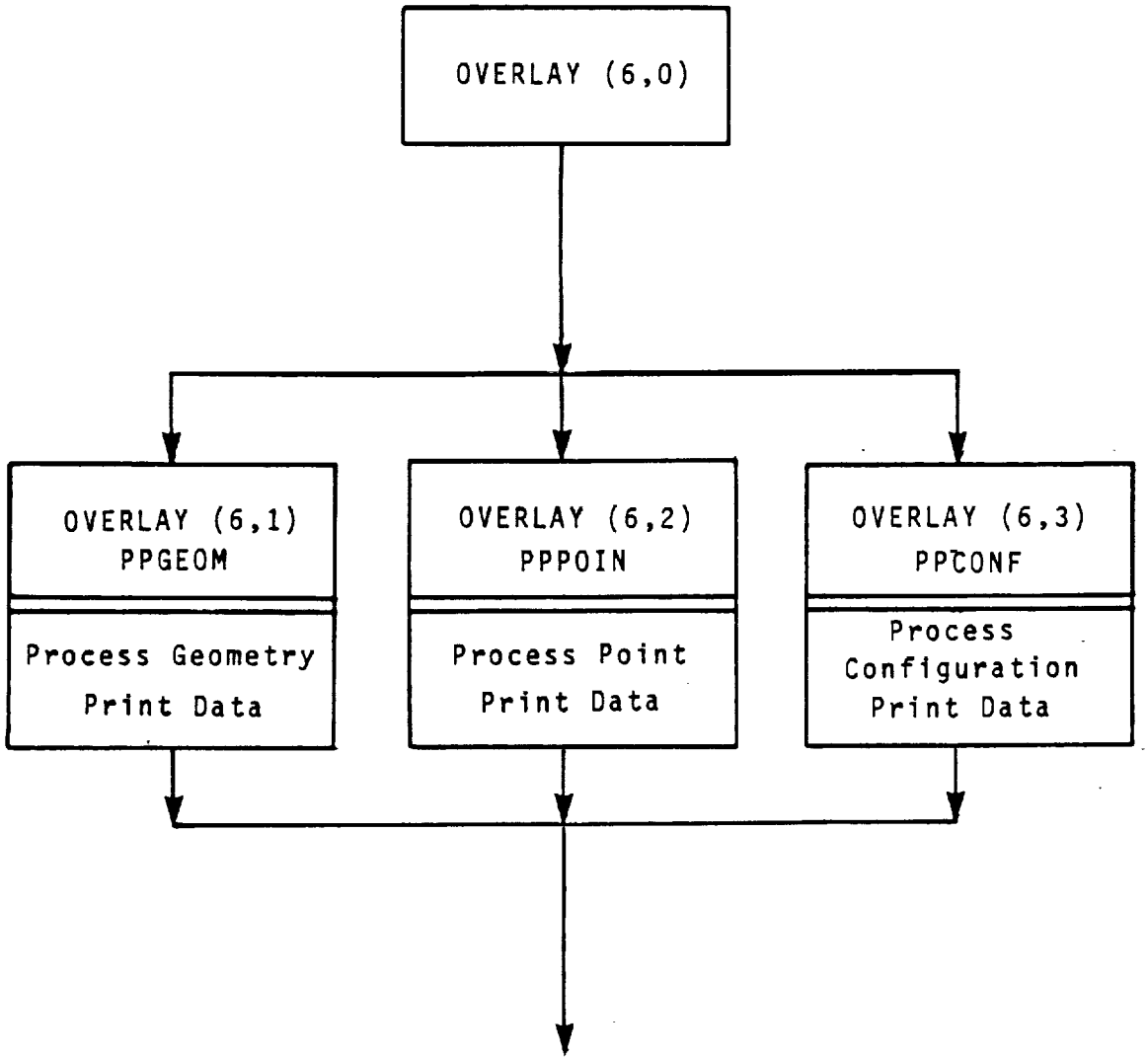
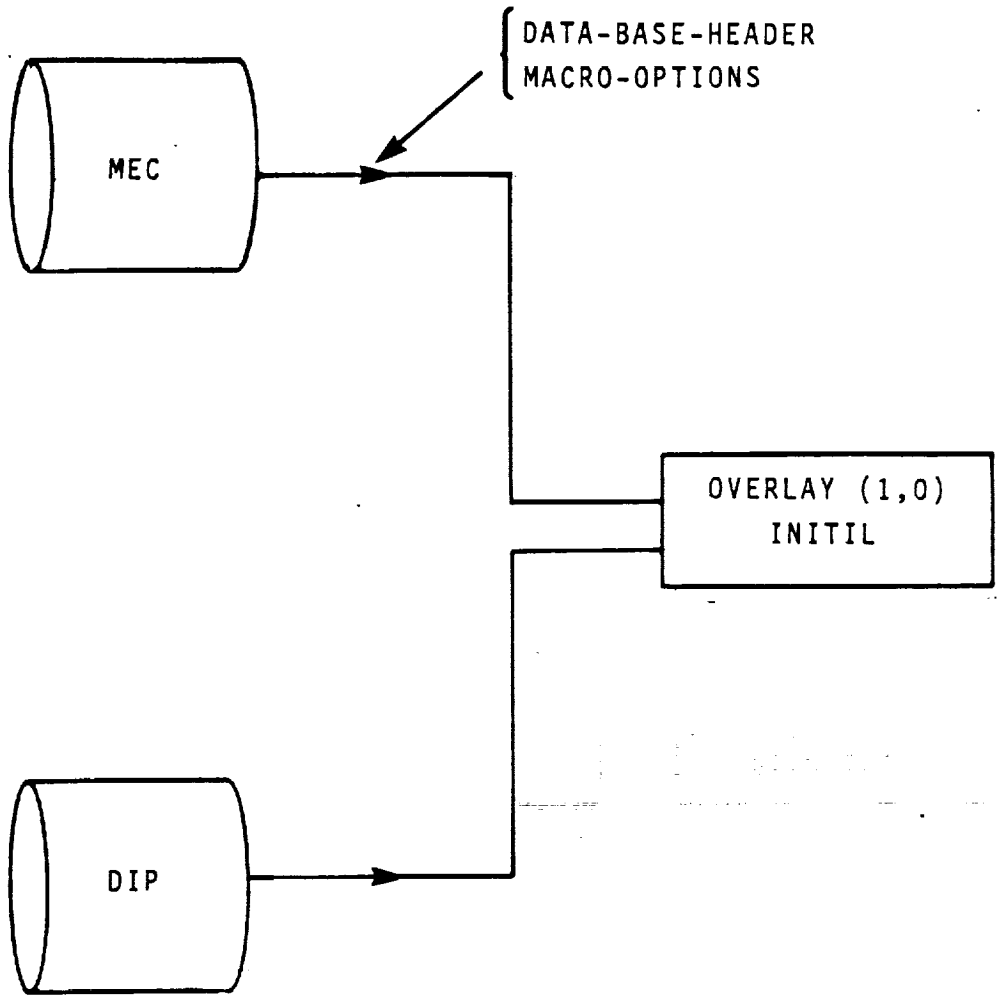


Figure 3.1 - Concluded



(Update Runs Only)

DIP:

DATA-BASE-HEADER
GLOBAL
GLOBAL-FLOW-PROP
GLOBAL-DB-OUTPUT
GLOBAL-DEFAULTS
GLOBAL-PRINTS
NETWORK-UPDATE-CODES

Figure 3.2 - Structure and Data Flow of OVERLAY (1,0)

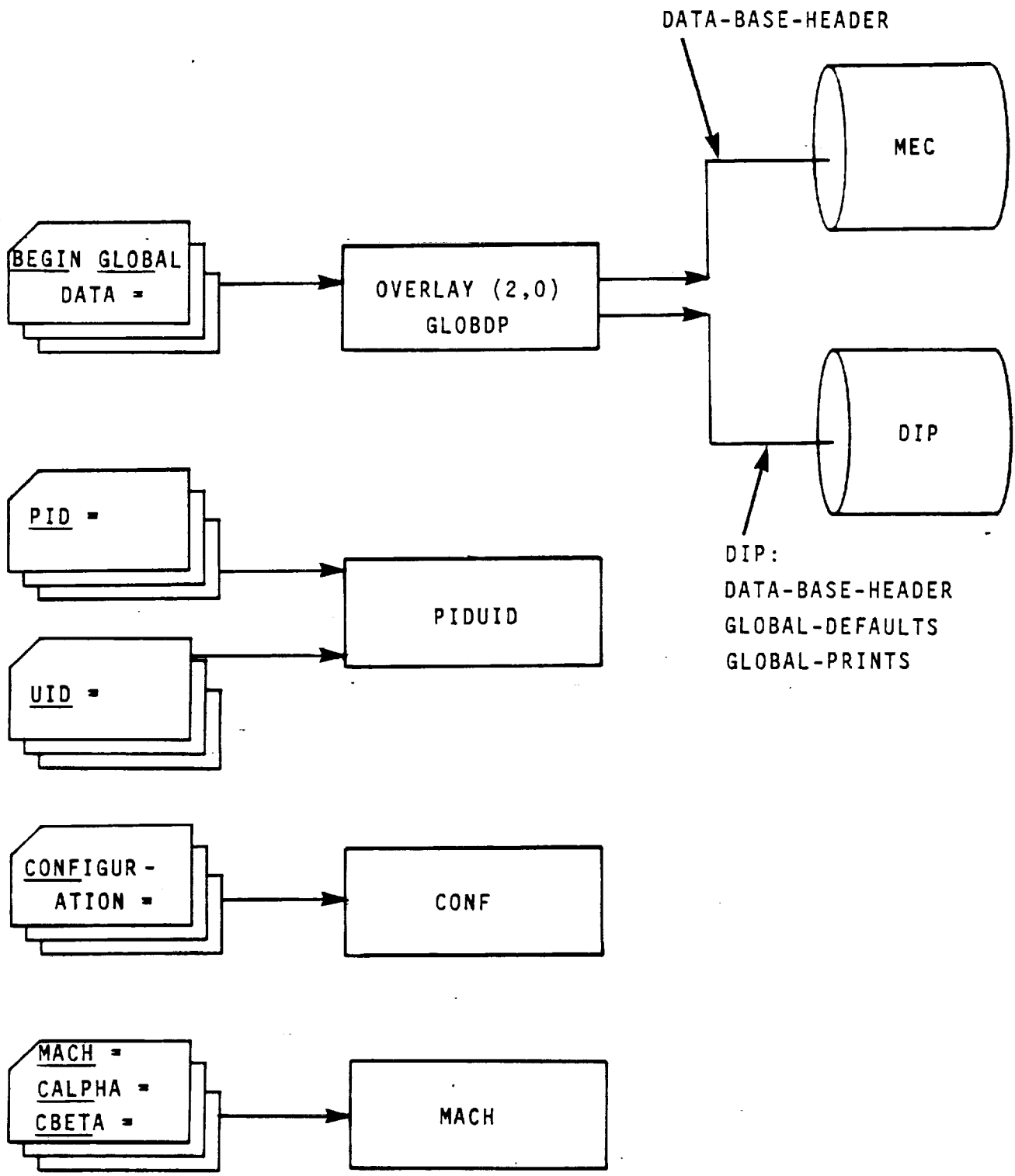


Figure 3.3 - Structure and Data Flow of OVERLAY (2,0)

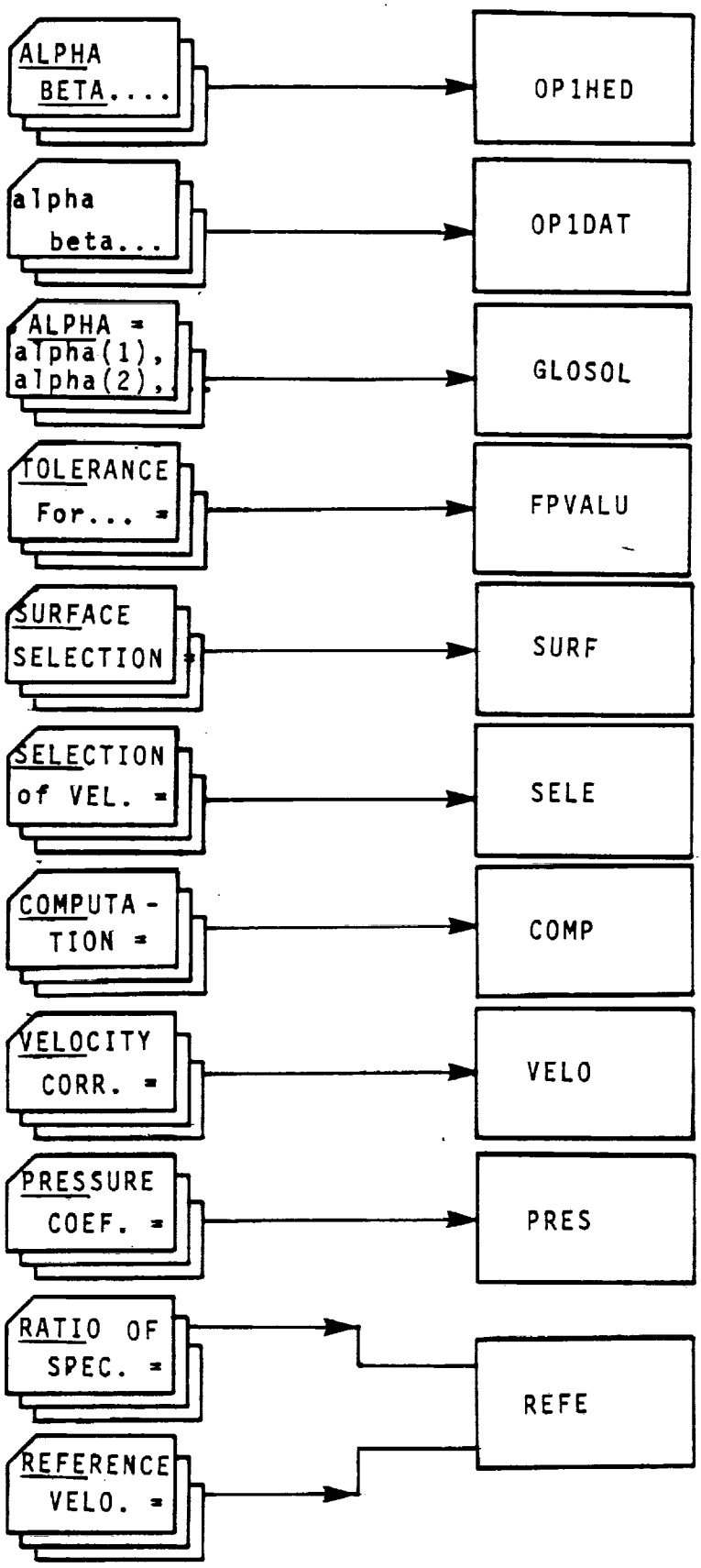


Figure 3.3 - Continued

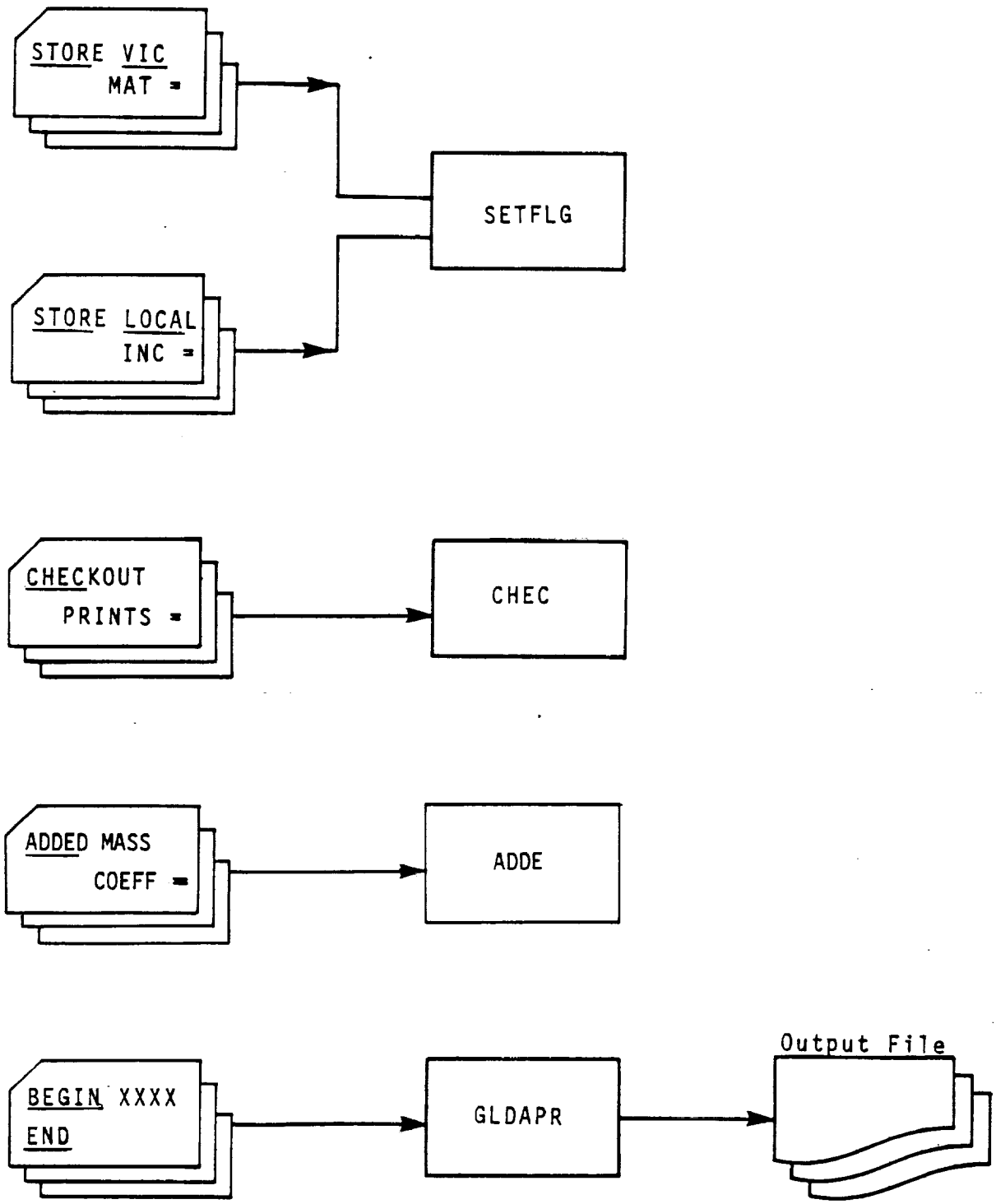


Figure 3.3 - (Concluded)

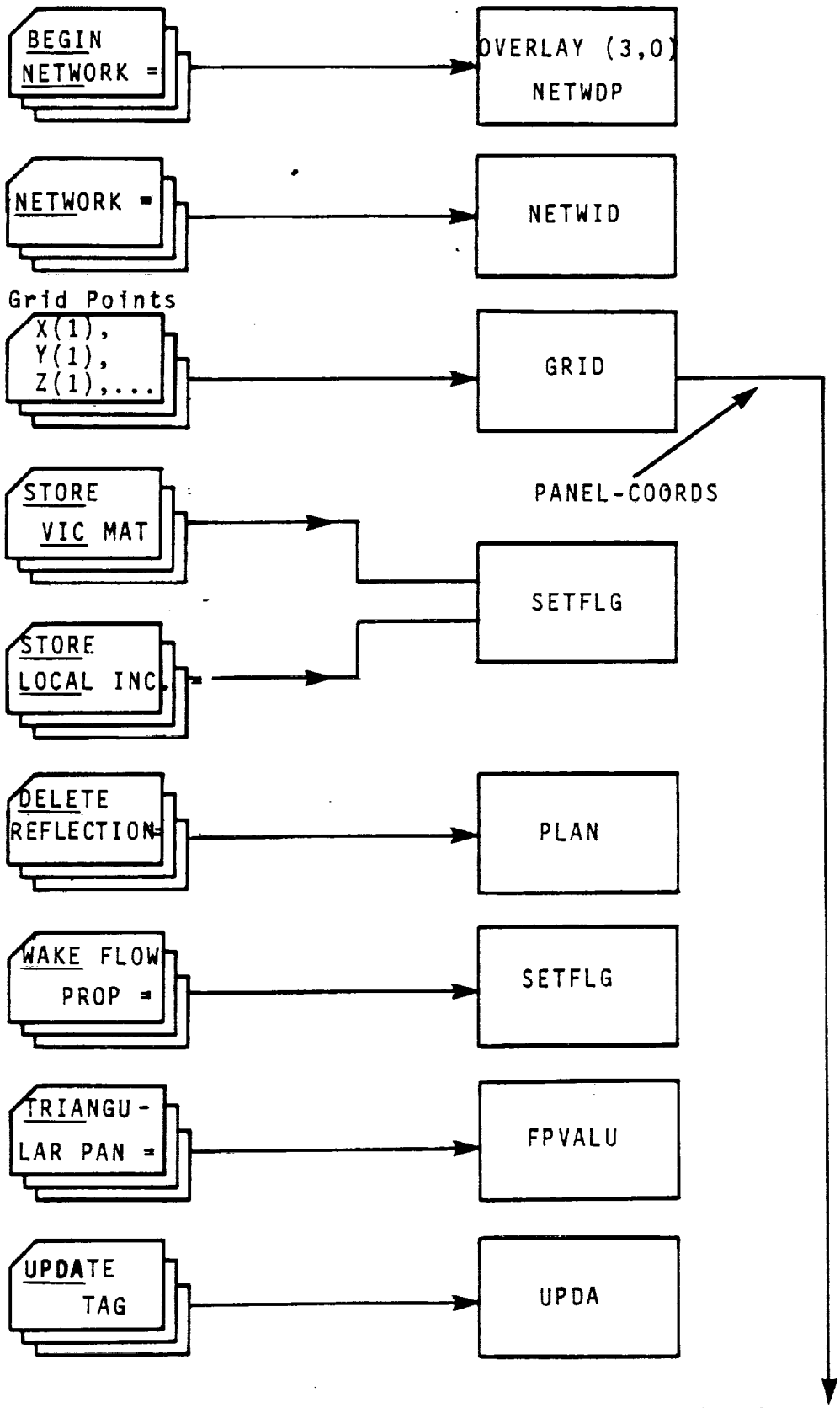


Figure 3.4 - Structure and Data Flow of OVERLAY (3,0)

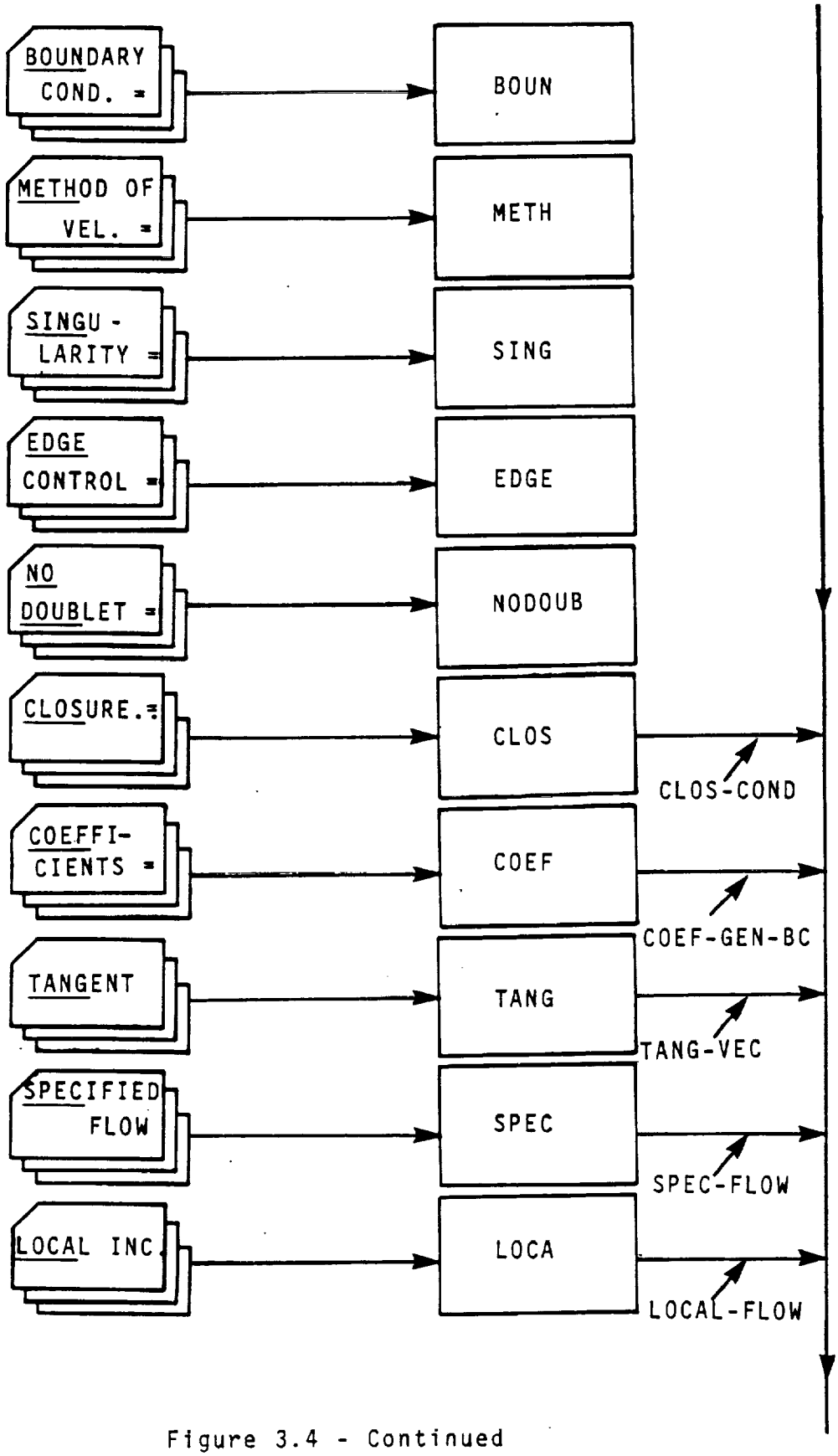


Figure 3.4 - Continued

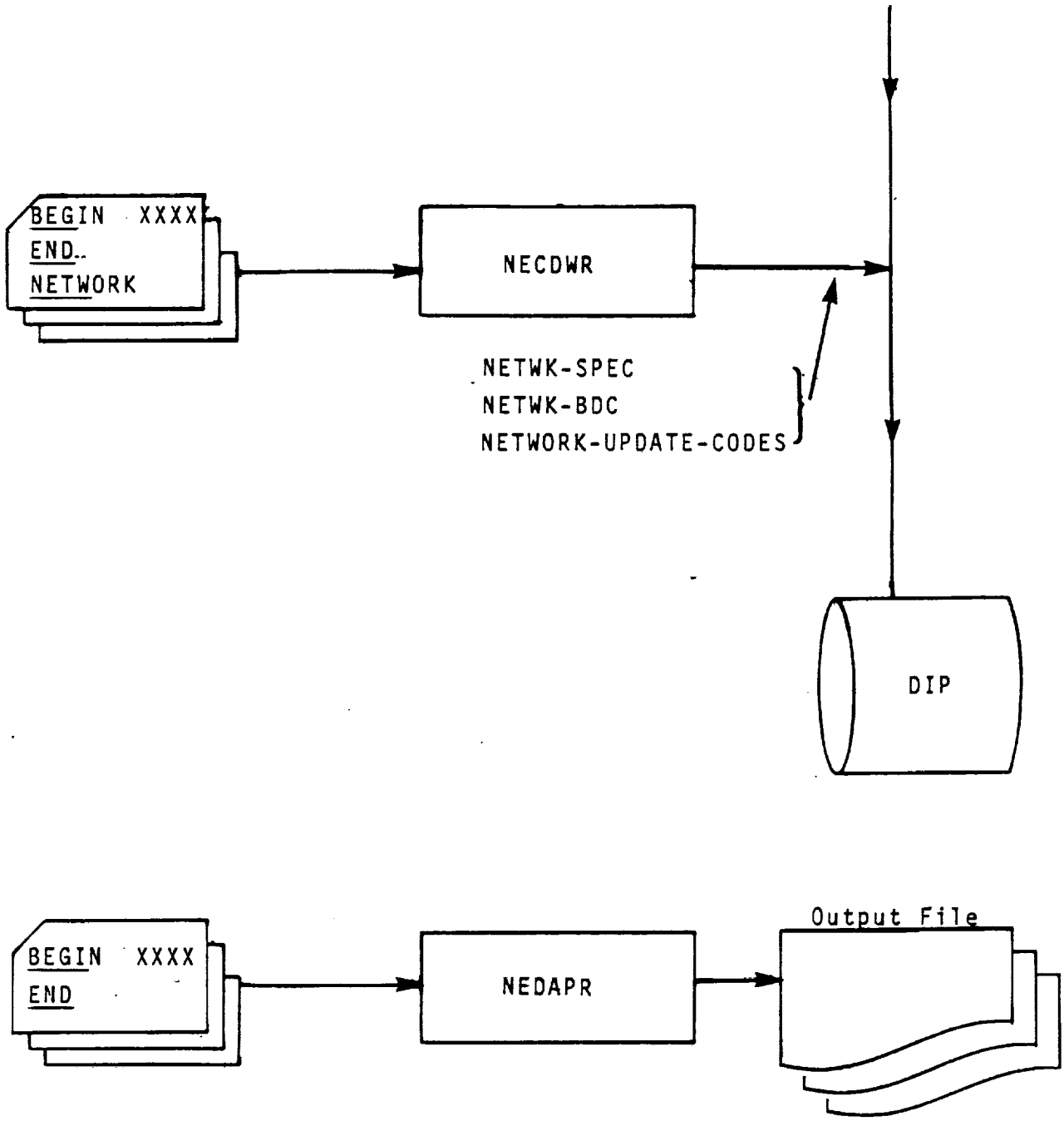


Figure 3.4 - Concluded

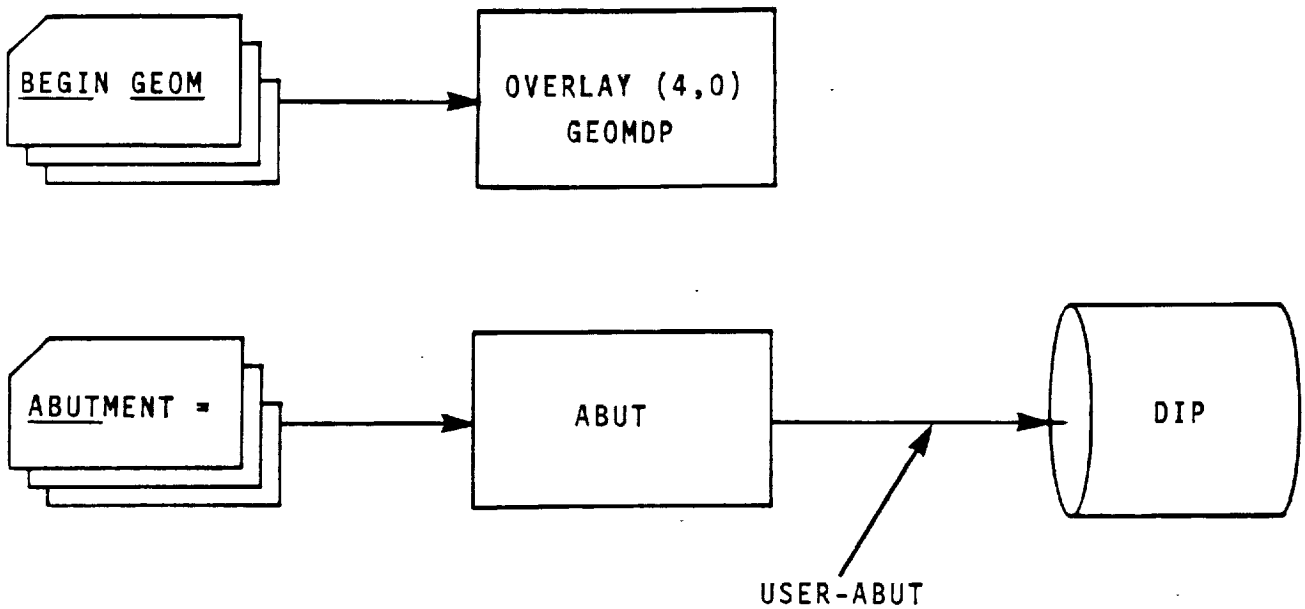


Figure 3.5 - Structure and Data Flow of OVERLAY (4,0)

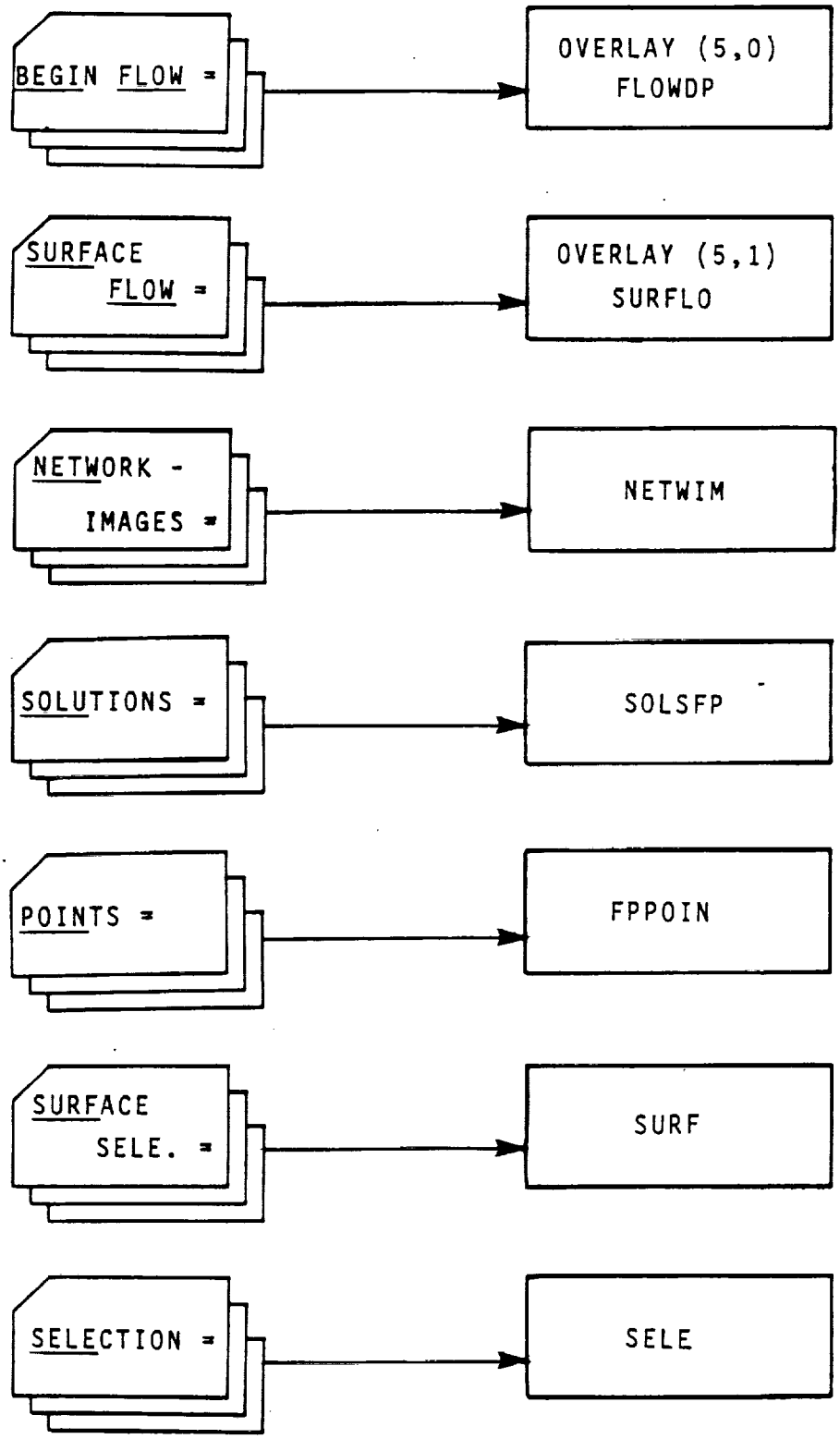


 Figure 3.6 - Structure and Data Flow of OVERLAY (5,0)

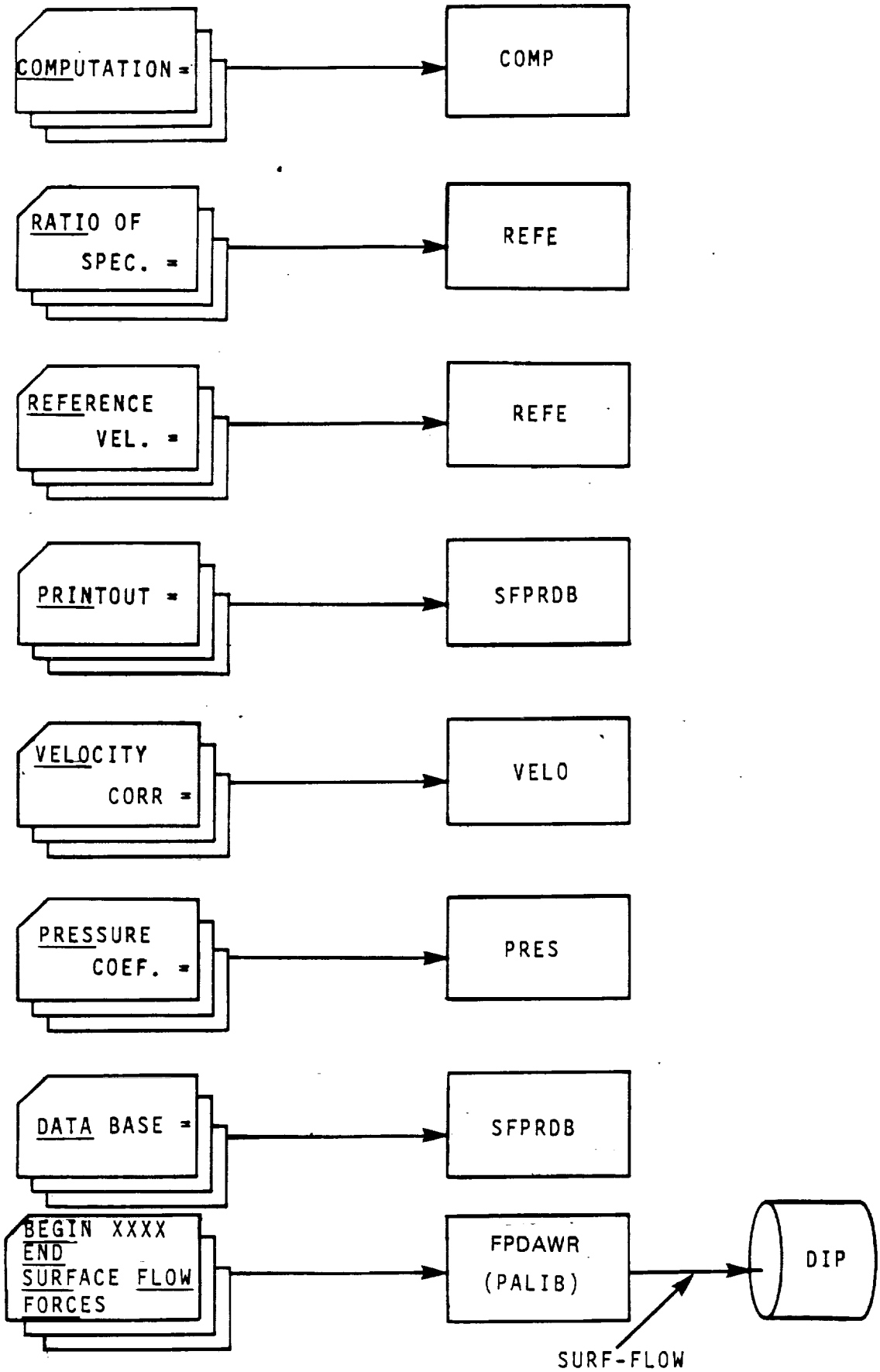


Figure 3.6 - Continued

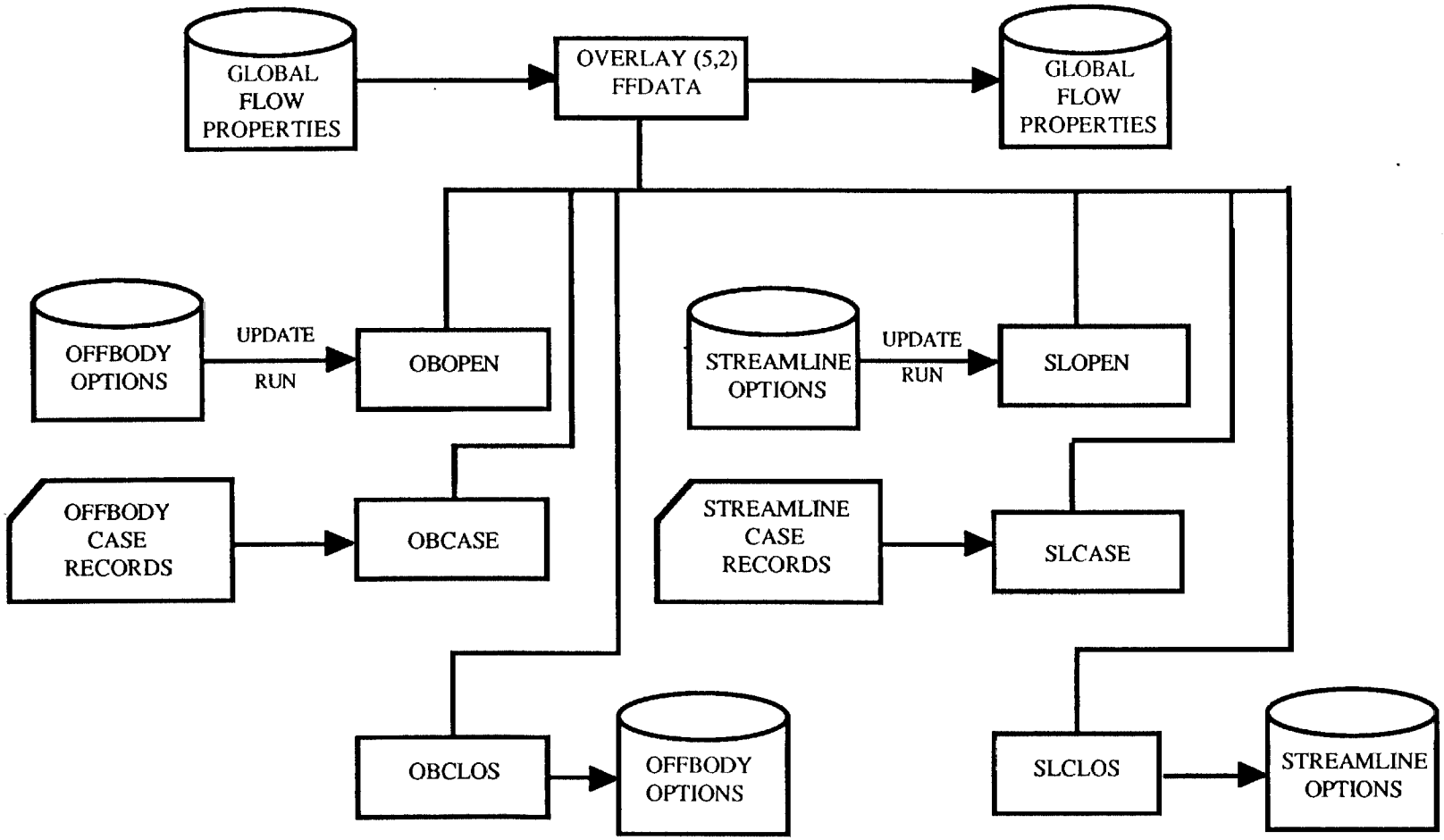


FIGURE 3.6 - CONTINUED

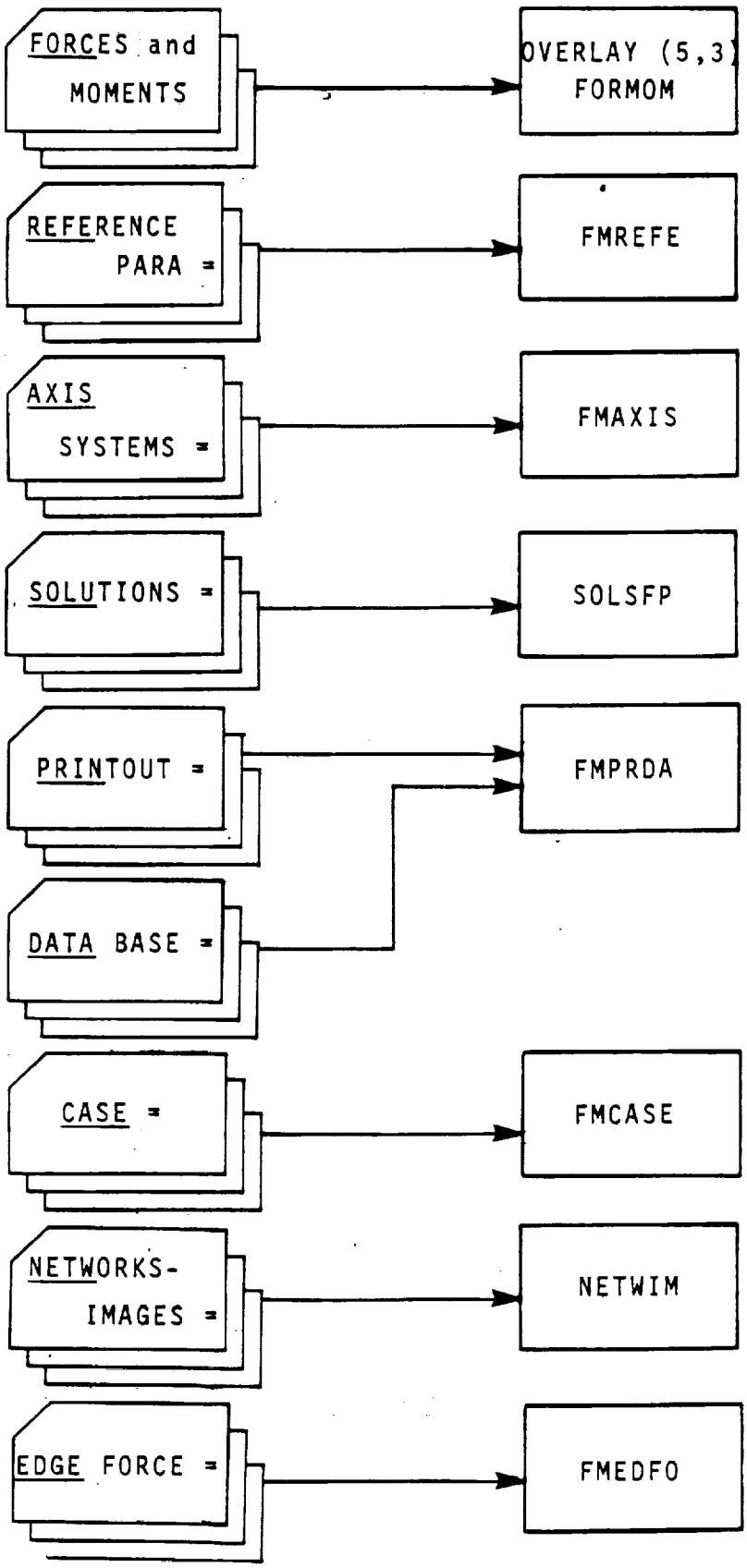


Figure 3.6 - Continued

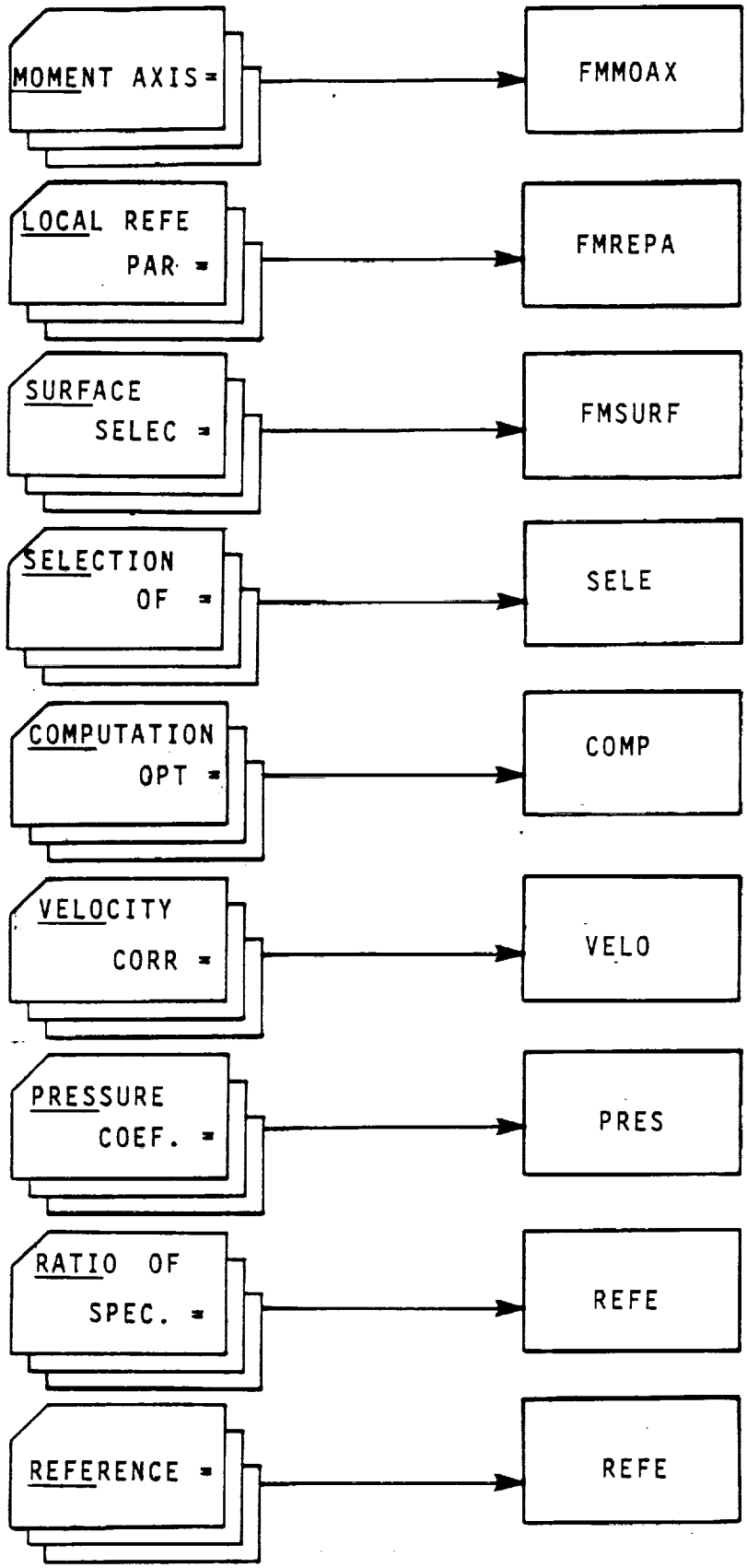


Figure 3.6 - Continued

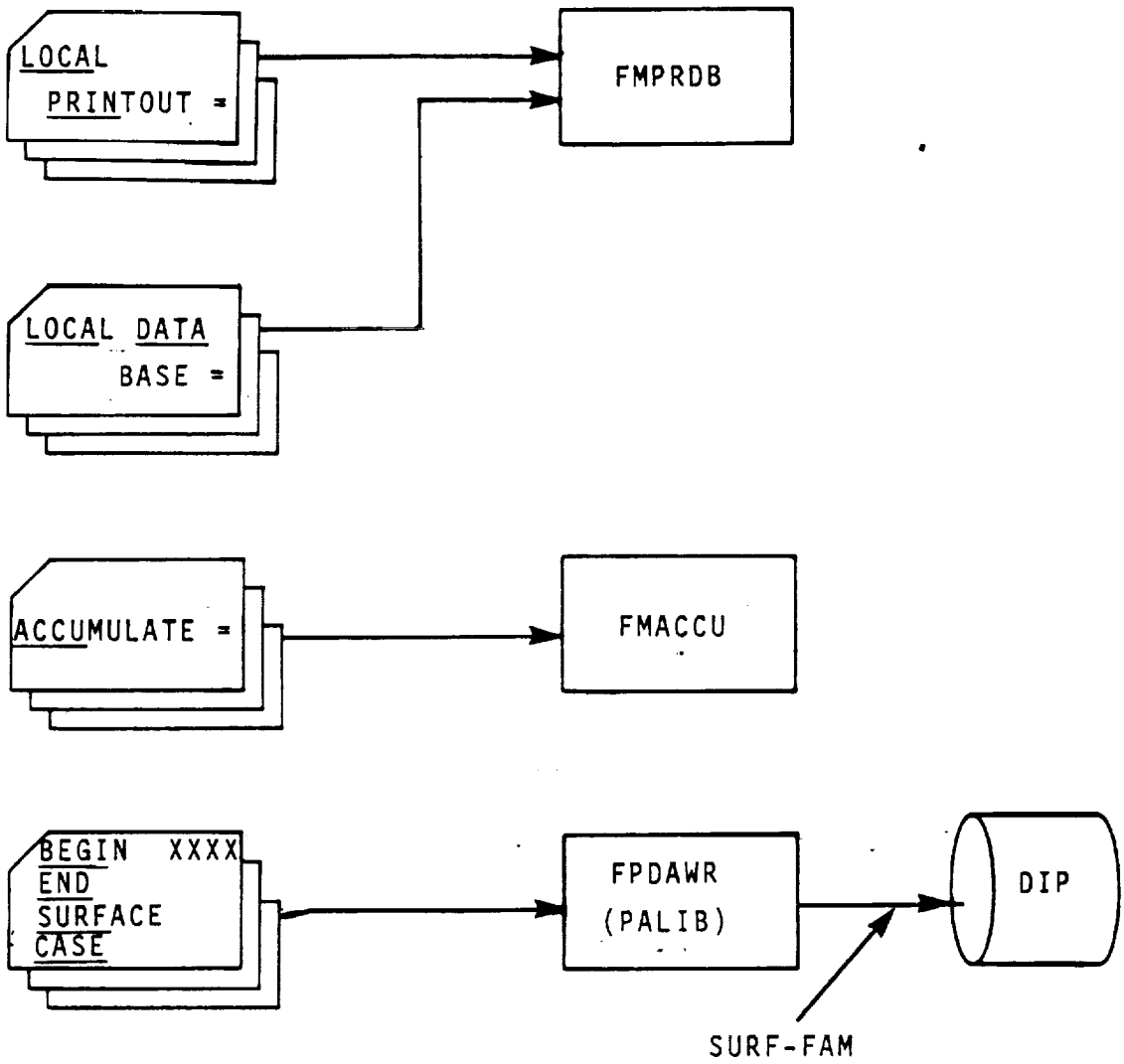
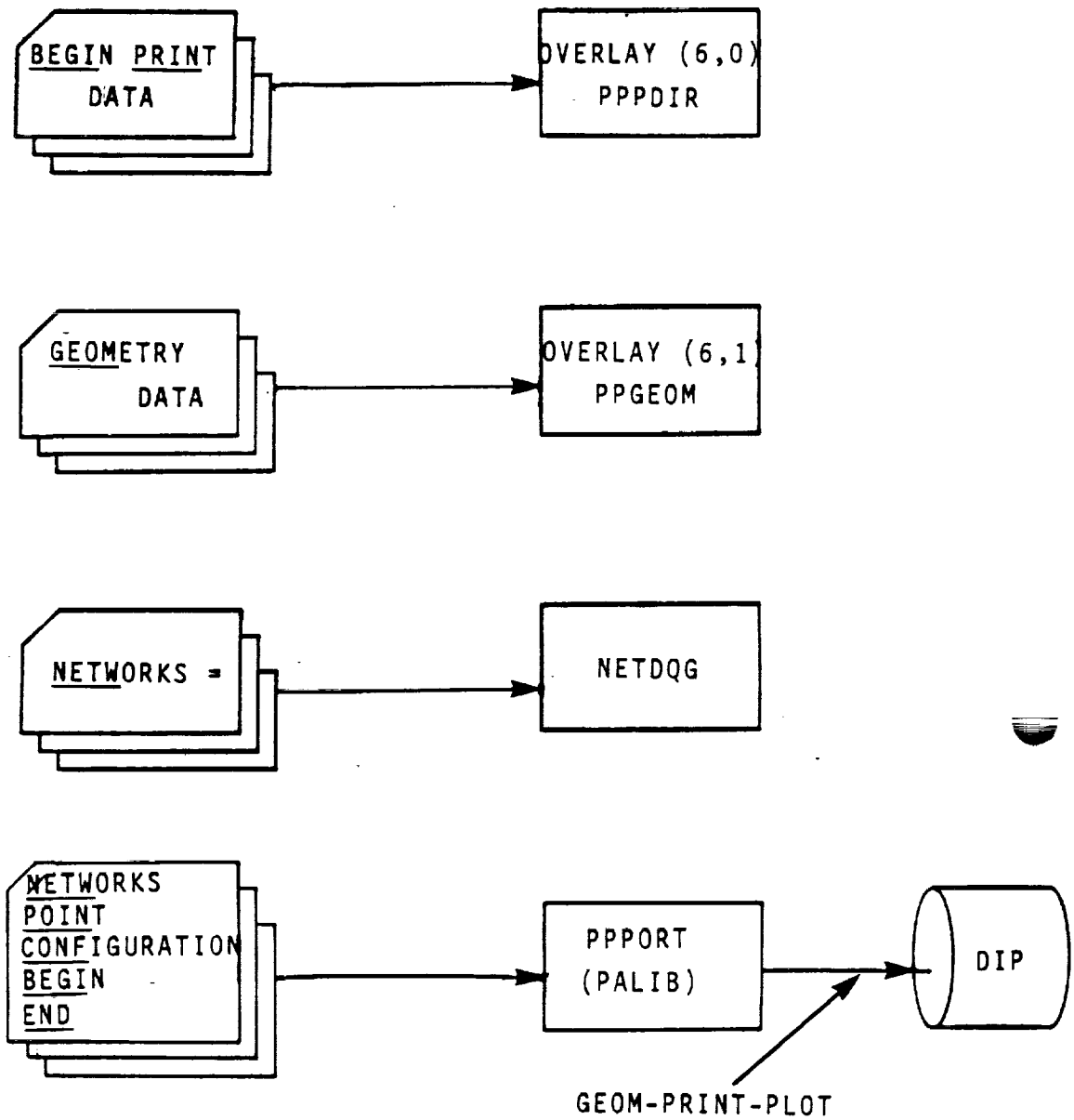


Figure 3.6 - Concluded



3.36 Figure 3.7 - Structure and Data Flow of OVERLAY (6,0)

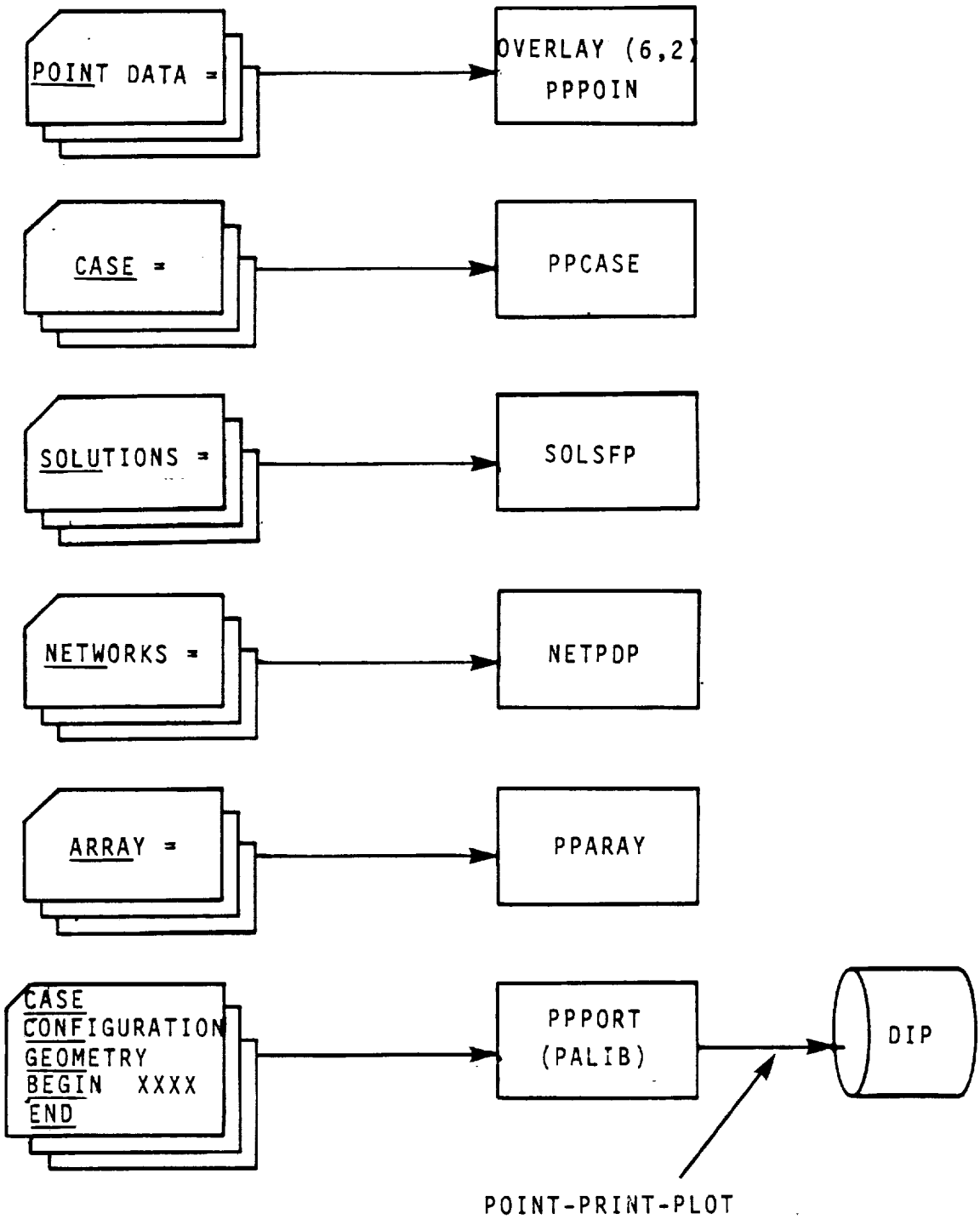


Figure 3.7 - Continued

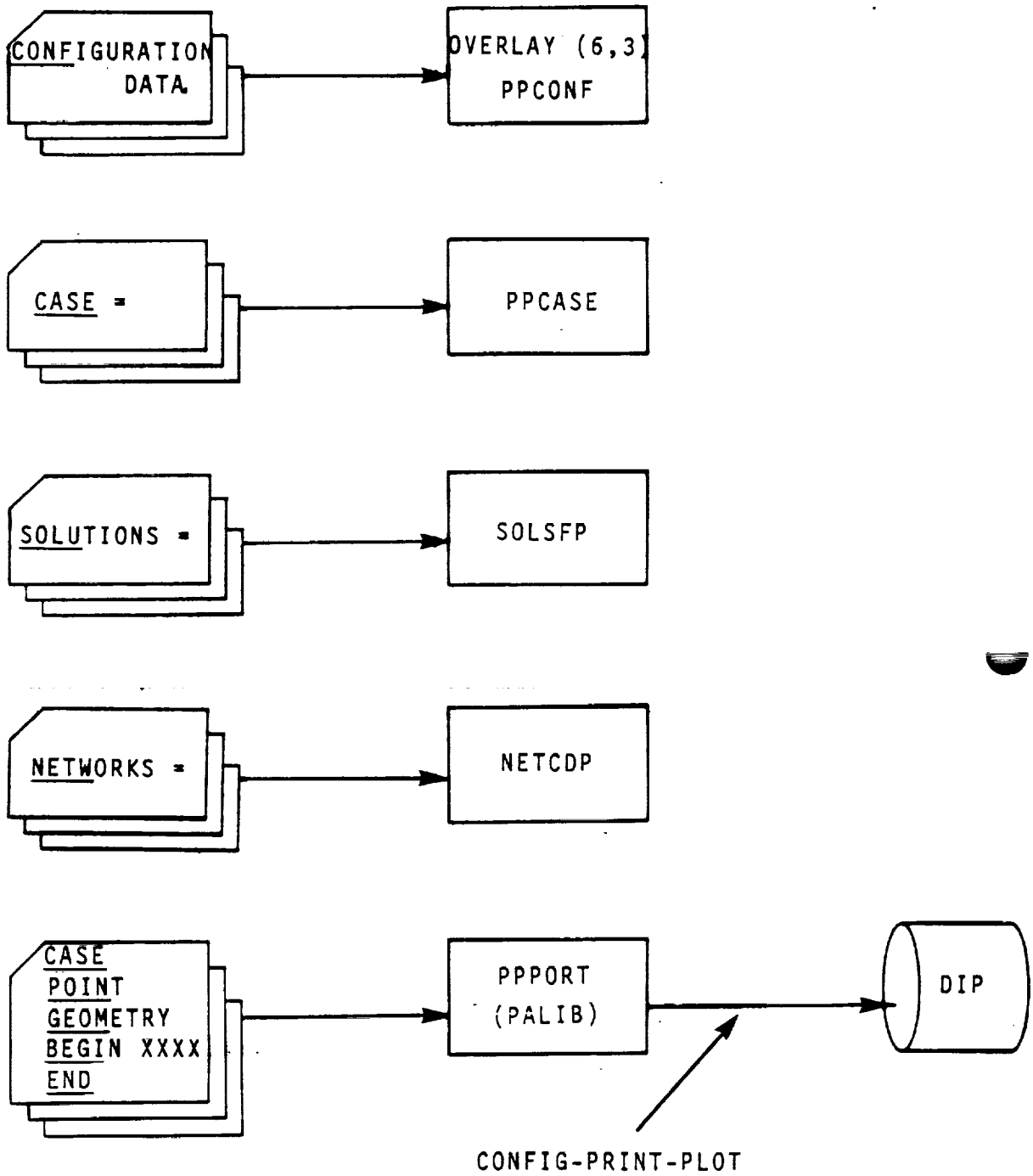


Figure 3.7 - Concluded

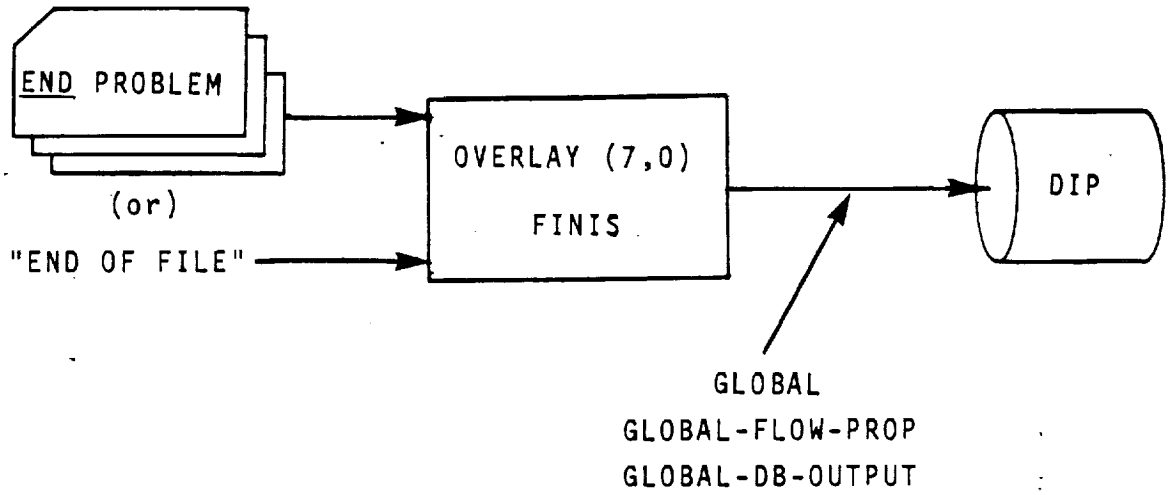


Figure 3.8 - Structure and Data Flow of OVERLAY (7,0)

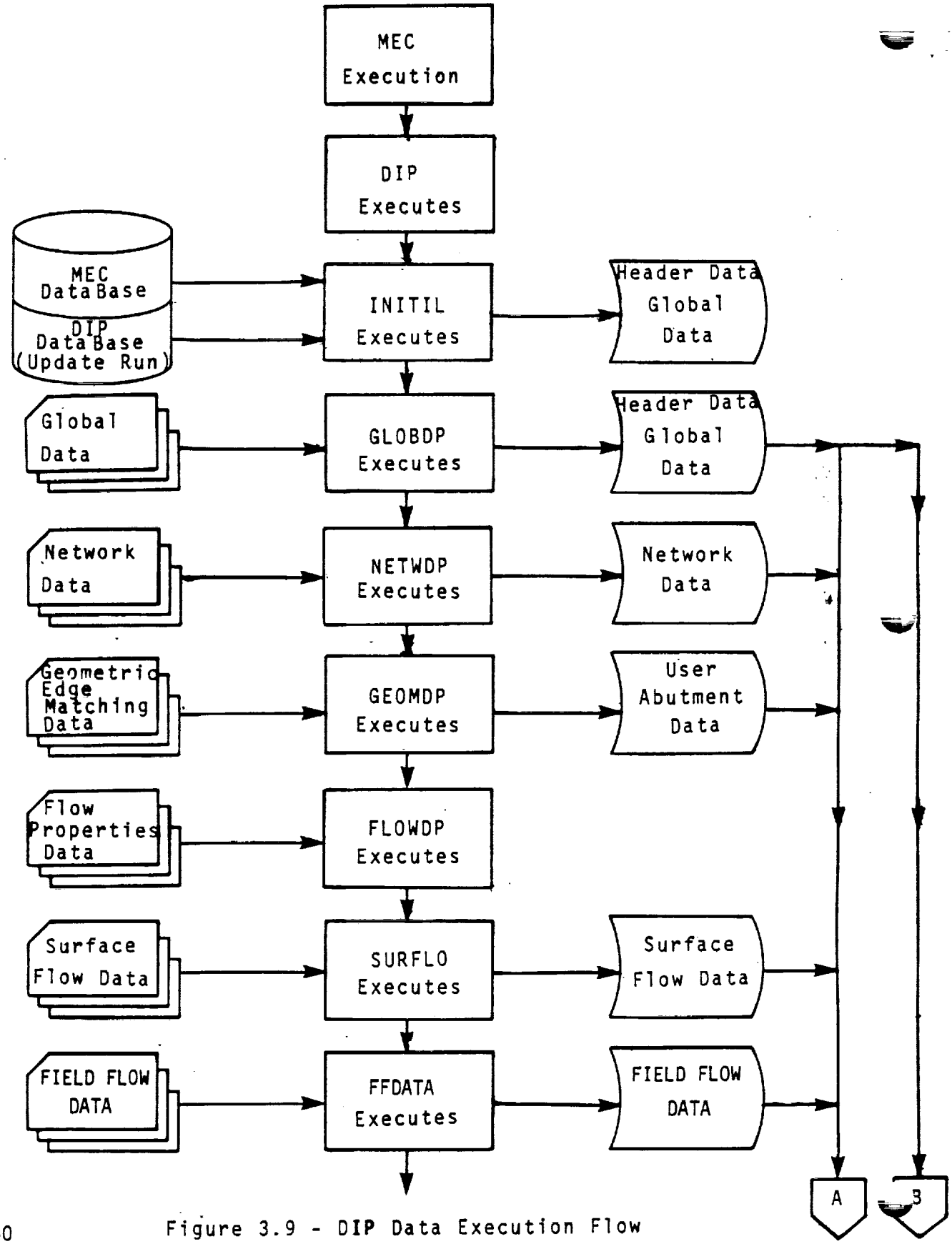


Figure 3.9 - DIP Data Execution Flow

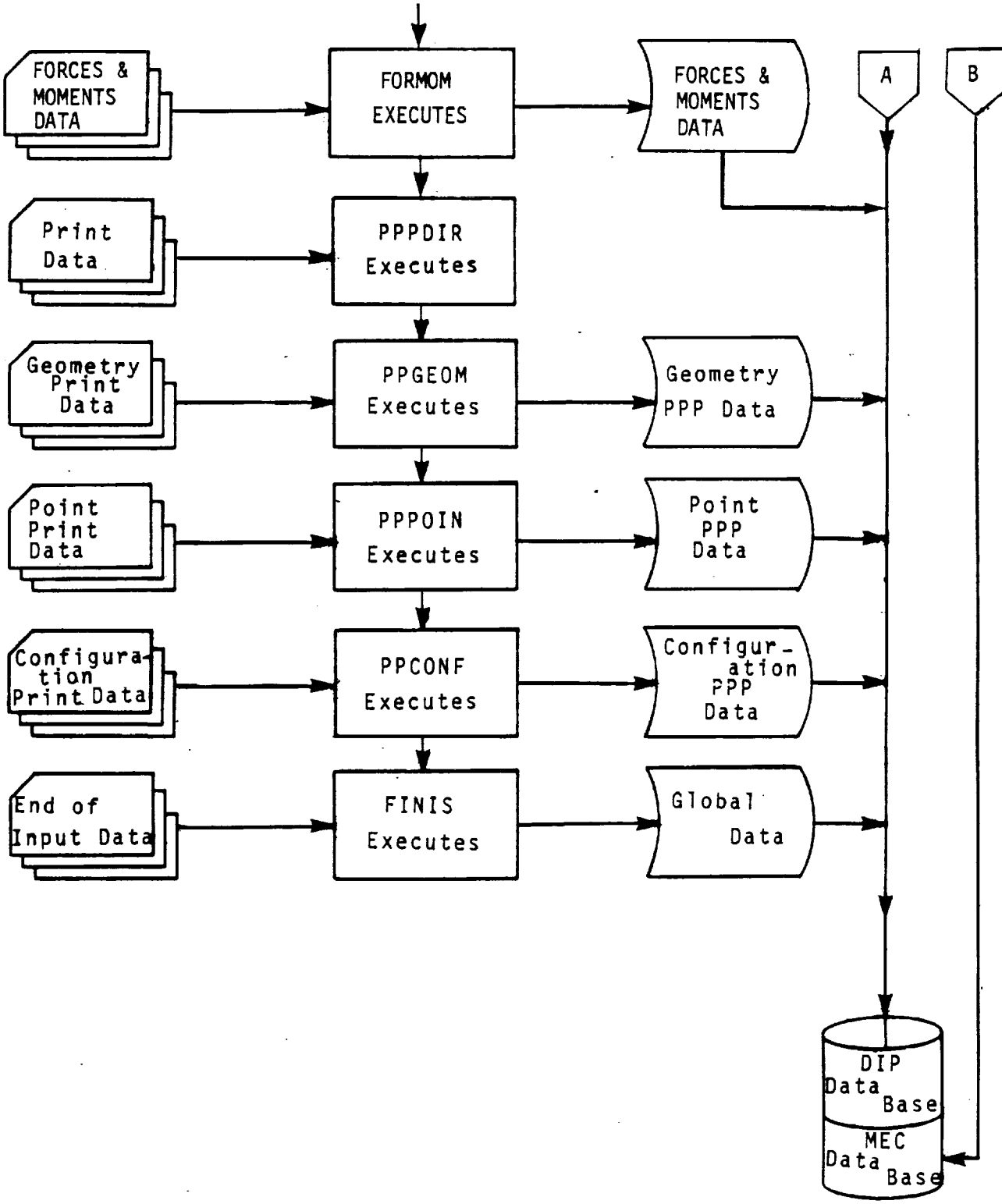


Figure 3.9 - Concluded

APPENDIX 3-A TREE STRUCTURE

The tree structure diagram of the DIP module has been deleted from this document. It is, however, available on the installation tape.

APPENDIX 3-B DIP FUNCTIONAL DECOMPOSITION

The functional decomposition of the DIP module is presented here. The decomposition labels are given in the order of their execution and therefore may not be alphabetic.

DIP - Data Input Processor

DIP (0,0) Overlay -

I - Program initialization

H PRGBEG - Begin DIP execution

ISDMS - Initialize SDMS

J LODREC - Read input record

JB - Load first two key words, if any.

JC - If input file empty, terminate run.

A INITIL (1,0) Overlay - Initialize for input data processing

A DBOPEN - Open MEC data base.

B DSMAP, ESGET - Define MEC maps and read MEC header data and MEC run options (update flags).

C CHPADB - Check DIP data base - if bad, terminate run.

D PAOPEN - Open DIP data base.

E DSMAP - Define DIP maps for DIP global level datasets

F ESGET - If run options indicate an update run, read DIP global level datasets.

G - Initialize key global parameters

AA - If errors occurred in INITIL, terminate DIP.

B GLOBDP (2,0) Overlay - Read, write, check and load global input data, if first two key words are B E G I N G L O B A L.

W GLOOPT - Process global option flag on BEGIN GLOBAL DATA record

B LODREC - Read next input record.

- Load first two key words, if any.

V OP1DAT - Load option 1 solution data, if present.

C PIDUID - Process problem ID or user ID record if first key word is P I D or U I D.

D CONF - Process configuration record, if first key word is C O N F I G U R A T I O N.

E MACH - Process flow regime data record, if first key word is M A C H or C A L P H A or C B E T A.

F OP1HED - Process option 1 solution header record if first key word is A L P H A or B E T A or U I N F or W M or W D C or W C P or S I D and no parameter list delimeter exists.

X GLOSOL - Process option 2 solution data record if first key word is A L P H A or B E T A or U I N F or W M or W D C or W C P or S I D and is followed by a parameter list.

G FPVALU - Process geometric network edge matching tolerance record, if first key word is G E O M E T R I C.

H SURF - Process surface selection record, if first key word is S U R F A C E.

I SETFLG - Process store vic matrix record when first two key words are S T O R E V I C.

J SETFLG - Process store local onset flow for computation of pressure record, if first two key words are S T O R E L O C A L.

A SELE - Process selection of velocity computation record, if first key word is S E L E C T I O N.

O	COMP	-	Process computational option for pressure record, if first key word is C O M P U T A T I O N.
K	VELO	-	Process velocity corrections record, if first key word is V E L O C I T Y.
T	PRES	-	Process pressure coefficient rule record, if first word is P R E S S U R E.
L	REFE	-	Process reference velocity for pressure record, if first key word is R E F E R E N C E.
M	REFE	-	Process ratio of specific heats record, if first key word is R A T I O.
N	CHEC	-	Process checkout prints record, if first key word is C H E C K O U T.
U	ADDE	-	Process added mass coefficients record, if first key word is A D D E D.
P	IDCNCV	-	Check need of defaults for problem ID, user ID, configuration and field flow data.
Q	SOLFIL	-	Check need of default solution data.
E	AMCGLR	-	Generate 6 solutions required for added mass coefficient calculations.
Y	RVPFIL	-	Check need to fill reference velocity for pressures list.
Z	CLDAPR	-	Check need to print new solutions.
R	SOLTRN	-	Transform input solution data to vectors.
T		-	Compute first, second and third handy matrices.
S		-	Write global level data to data base.
	SA REP	-	Write header data to MEC data base.
	SB ESPUT thru SD	-	Write header data, global defaults and global prints to DIP data base.
C	NETWDP (3,0) Overlay	-	Read, write, check and load network input data, if first two key words are B E G I N N E T W O R K.
A	DSMAP	-	Define maps for individual network datasets.
B	LODREC	-	Read input record.
		-	Load first two key words, if any.
C	NETWID	-	Process network ID record if first key word is N E T W O R K.
T	NETOPT	-	Process network record update parameter option, if any.
U	NETIDS	-	Process network record ID, if any.
V	NOPCHK	-	Check network update option.
Z	NDELDR	-	Load network defaults.
M	GRID	-	Read network grid data.
B	LODREC A	-	Read input record. Load first two key words.
L	SETFLG	-	Process store vic matrix records, if first two key words are S T O R E V I C.
D	SETFLG	-	Process store local onset flow for computation of pressure record, if first two key words are S T O R E L O C A L.
E	PLAN	-	Process delete reflection in plane of symmetry record, if first key word is D E L E T E.
F	SETFLG	-	Process wake flow properties tag record, if first key word is W A K E.
G	FPVALU	-	Process triangular panel tolerance record, if first key word is T R I A N G U L A R.

H UPDA - Process update tag record, if first key word is U P D A T E.

I BOUN - Process boundary conditions record, if first key word is B O U N D A R Y.

C METH - Process method of velocity computation record, if first key word is M E T H O D.

J SING - Process singularity types record, if first key word is S I N G U L A R I T Y.

K EDGE - Process edge control point locations record, if first key word is E D G E.

O NODOUB - Process no doublet edge matching record, if first key word is N O.

P CLOS - Process closure edge condition data, if first key word is C L O S U R E.

- A EDGE - Process the identifier and locator parameters on the closure edge condition record.
- B NBDORT - Read next record, load key word, identify, and check order of input.
- D BDTERM - Process term, ID, if key word is T E R M.
- E SOLSFP - Process solutions list, if key word is S O L U.
- F CLDATA - Process closure data values, if first key word is a numeric.

N COEF - Process coefficients of general boundary condition equation data, if first key word is C O E F F I C I E N T S.

- A NBDORT - Read next record, load key word, identify, and check order of input.
- C BDTERM - Process term ID, if key word is T E R M.
- D SOLSFP - Process solutions list, if key word is S O L U.
- E NEPOIN - Process control point locations record, if key word is P O I N T.
- D NEDATA - Process coefficient data values, after a control point locations record.

Q TANG - Process tangent vectors for design data, if first key word is T A N G E N T.

- A NBDORT - Read next record, load key word, identify, and check order of input.
- C BDTERM - Process term ID, if key word is T E R M.
- D - Set flag to suppress scaling of vectors to unit length, if key word is U N A L T E R E D.
- E SOLSFP - Process solutions list, if key word is S O L U T I O N S.
- F NEPOIN - Process control point locations record, if key word is P O I N T.
- G NEDATA - Process tangent data values or a method of computation flag record after a control point locations record.

R SPEC - Process specified flow data, if first key word is S P E C I F I E D.

A NBDORT - Read next record, load key word, identify, and check order of input.
C BDTERM - Process term ID, if key word is T E R M.
D INPIUM - Process input or images options, if key word is I N P U T.
E SOLSFP - Process solutions list, if key word is S O L U T I O N S.
F NEPOIN - Process control point locations record, if key word is P O I N T.
G NEDATA - Process specified flow data values after a control point locations record.
H ESGET - Read existing boundary condition coefficients data from dataset NETWK-BDC

S LOCA - Process local incremental onset flow data, if first key word is L O C A L.
A NBDORT - Read next record, load key word, dentify, and check order of input.
C BDTERM - Process term ID, if key word is T E R M.
D INPUIM - Process input or image options, if key word is I N P U T.
E SOLSFP - Process solutions list, if key word is S O L U T I O N S.
F NEPOIN - Process control point locations record, if key word is P O I N T.
G NEDATA - Process local incremental onset flow values after a control point locations record.

W - If key word is B E G I N or E N D, set network complete flag.

A - Perform network checks.
AC BCSTEC - If not a solution update, check user inputs for current network.
AD CBC123 - Check data for boundary condition 1, 2 or 3.
AE CHKBC4 - Check data for boundary condition 4.
AF CHKBC5 - Check data for boundary condition 5.
AG NECDWR - Write network control data to data base.
AH NEDAPR - Print summary of networks data, if networks data complete flag set.

D GEOMDP (4,0) Overlay - Read, write check and load geometric edge matching data, if first two key words are B E G I N G E O M E T R I C.

A DSMAP - Define network dimensions and use abutment data maps.

B LODREC - Read a record from input.
BB - Load first two key words, if any.

C ABUT - Process abutment record, if first key word is A B U T M E N T.
A - Initialize and update parameters.
B ABENID - Process parameter list of abutment definition record.

C LODREC - Read a record from input.
E - Load first two key words, if any.
H PLAN - Process planes of symmetry record, if first key word is P L A N E S.
I SETFLG - Process smooth edge treatment record, if first key word is S M O O T H.
J thru M - When first key word is A B U T M E N T or B E G I N or E N D, test current abutment data.
N ESPOR - Write user abutment on data base.

E FLOWDP (5,0) Overlay - Read, write, check and load flow properties calculations data, if first two key words are B E G I N F L O W.

A FLOWPT - Process the post solution update option on BEGIN FLOW record.

B LODREC - Read a record from input and load first two key words.

C SURFLO (5,1) Overlay - Process surface flow properties, if first two key words are S U R F A C E F L O W.

A DSMAP - Define surface flow properties data maps
B FPCASE - Process case name on "SURFACE FLOW" record.
C - Initialize output array and supplement record counts.
D LODREC - Read a record from input.
F - Load first two key words, if any.
I NETWIM - Process network images list record, if first key word is N E T W O R K - I M A G E S.
J SOLSFP - Process solutions list records, if first key word is S O L U T I O N S.
K FPPOIN - Process control points record, if first key word is P O I N T.
M SURF - Process surface selection record, if first key word is S U R F A C E and second key word is not F L O W.
N SELE - Process selection of velocity computation record, if first key word is S E L E C T I O N.
O COMP - Process computation option for pressures record, if first key word is C O M P U T A T I O N.
R REFE - Process ratio of specific heats record, if first key word is R A T I O.
S REFE - Process reference velocity for pressure record, if first key word is R E F E R E N C E.
T SFPRDB - Process printout/database record, if first key word is P R I N T O U T or D A T A.
U SFPRDB

B SFOULD - Process parameters list of printout/data base record.

C LODREC - Read a record from input.
E - Load first two key words, if any.
F PRES - Process pressure coefficient rules record, if first key word is P R E S S U R E.
G VELO - Process velocity corrections record, if first key word is V E L O C I T Y.
I SFOUCL - Load output array with pressure and velocity data.

V SFDELO - Load defaults for current surface flow properties calculation case.

W FPDAWR - Write current surface flow properties calculation case to data base.

E FFDATA (5,2) Overlay - Process field flow properties if first two key words are F I E L D F L O W

A Perform field flow initialization.
A Define maps
B FFINIT - Initialize labeled common areas
C Process field flow subgroup identifier

B REPARS - Read and parse a valid input record
A LODREC - Read and parse the next input record
B VCHECK - Check the validity of that input record
A Check for valid record type
B Check for valid record sequence
C Check for valid record syntax

C Interpret off body case specifications
A OBOPEN - Initialize processing for an off body case
A Initialize default data with global defaults
B Retrieve case identifier
C Add a new case on an update run
D Use an old case on an update run
E Add a new case on a standard run
B OBCASE - Process off body case records
A REPARS - Read and parse a valid input record (see EEB)
B FFSOL - Process solution record
C Process point list header record
D Process point list record
E Process grid definition record
F Process grid limits record
G Process grid plane density record
H Process pressure computation record
I Process ratio of specific heats record
J Process reference velocity record
K FFOREQ - Process print request record
L Process velocity correction print record
M Process pressure rule print record
N FFOREQ - Process data base request record
O Process velocity correction data base record
P Process pressure rule data base record

C OBCLOS - Complete processing for an off body case
- A FFDEFA - Use available defaults on unspecified records
- B Convert data in input form to output form to match dataset specifications
- C Write data to database

D Interpret streamline case specifications
- A SLOPEN - Initialize processing for a streamline case
  - A Initialize default data with global defaults
  - B Retrieve case identifier
  - C Add a new case on an update run
  - D Use an old case on an update run
  - E Add a new case on a standard run
- B SLCASE - Process streamline case records
  - A REPARS - Read an parse a valid input record (see EEB)
  - B FFSOL - Process solution record
  - C Process step size record
  - D Process number of integrations record
  - E Process integration error record
  - F Process streamline direction record
  - G Process streamline type record
  - H Process streamline limits record
  - I Process print frequency record
  - J Process starting points header record
  - K Process starting points list record
  - L Process pressure computation record
  - M Process ratio of specific heats record
  - O FFOREQ - Process print request record
  - P Process velocity correction print record
  - Q Process pressure rule print record
  - R FFOREQ - Process database request record
  - S Process velocity correction data base record
  - T Process pressure rule database record
- C SLCLOS - Complete processing for a streamline case
  - A FFDEFA - Use available defaults on unspecified records
  - B Convert data in input form to output form to match dataset specifications
  - C Write data to database

E Replace global flow properties dataset

F FORMOM (5,3) Overlay - Process forces and moments data, if first key word is F O R C E S.
- A DSMAP - Define surface forces and moments data map.
- B FMGLDE - Initialize forces and moments global values.
- C LODREC - Read a record from input and load first two key words.
- F FMREPA - Process references parameters record, if first key word is R E F E R E N C E.
- G FMAXSY - Process axis systems record, if first key word is A X I S.

H	SOLSFP		-	Process solutions list record, if first key word is S O L U T I O N S.
I	FMPRDA		-	Process printout/data base record, if first key word is P R I N T O U T or D A T A.
J	FMPRDA			
K	FMPDCK		-	Check/load global printout/data base options.
L	FMCASE		-	Process case record, if first key word is C A S E.
	B	FPCASE	-	Process case record parameter list.
	C	FMLOIN	-	Initialize local variables.
	D	LODREC	-	Read a record from input.
		D	-	Load first two key words.
	E	NETWIM	-	Process networks images list record, if first key word is N E T W O R K S - I M A G E S.
	F	FMEDFO	-	Process edge force calculations record, if first key word is E D G E.
	G	FMMOAX	-	Process moment axis record, if first key word is M O M E N T.
	H	FMREPA	-	Process local reference parameters record, if first two key words are L O C A L R E F E R E N C E.
	I	FMSURF	-	Process surface selection record, if first key word is S U R F A C E.
	J	SELE	-	Process selection of velocity computation record, if first key word is S E L E C T I O N.
	K	COMP	-	Process computation option for pressures record, if first key word is C O M P U T A T I O N.
	L	VELO	-	Process velocity corrections record, if first key word is V E L O C I T Y.
	M	PRES	-	Process pressure coefficient rules record, if first key word is P R E S S U R E.
	N	REFE	-	Process ratio of specific heats record, if first key word if R A T I O.
	O	REFE	-	Process reference velocity for pressure record, if first key word is R E F E R E N C E.
	P	FMPRDA	-	Process local printout/local data base record, if first two key words are L O C A L P R I N T O U T or L O C A L D A T A.
	Q	FMPRDA		
	R	FMACCU	-	Process accumulate record, if first key word is A C C U M U L A T E.

S		-	End of case processing.
SA	FMPDCK	-	Check/load local printout/data base options, if first key word is S U R F A C E, or B E G I N, or E N D or C A S E.
SB	FMLODE	-	Load defaults.
T	FPDAWR	-	Write dataset SURF-FAM to data base.
U		-	Set forces and moments done flag, if first key word is S U R F A C E, or B E G I N or E N D.
W		-	Generate a pure accumulation case,
X			if any accumulations in previous data.
XA	FPDAWR	-	Write dataset SURF-FAM to data base.
		-	Set flow complete flag, if first key word is B E G I N or E N D.

F PPPDIR (6,0) Overlay - Read, write, check and load data base directives.

C LODREC - Read a record from input.

D - Load first two key words, if any.

E PPGEOM (6,1) Overlay - Process geometry data record set, if first key word is G E O M E T R Y.

A	DSMAP	-	Define geometry PPP data maps.
B		-	Initialize.
C		-	Load default network count.
D	PPPORT	-	Read next record, load key word, identify, and check order of input.
F	NETDQG	-	Process networks record, if key word is N E T W O R K S.

F PPPOIN (6,2) Overlay - Process point data record set, if first key word is P O I N T.

A	DSMAP	-	Define point PPP data map.
B		-	Initialize.
C		-	Load defaults.
D	PPPORT	-	Read next record, load key word, identify and check order on input.
F	PPCASE	-	Process case record and load lower level defaults, if key word is C A S E.
G	SOLSFP	-	Process solutions list record and load lower level defaults, if key word is S O L U T I O N S.
H	NETPDP	-	Process surface record and load lower level defaults, if key word is S U R F A C E.
I	PPARAY	-	Process array record, if key word is A R R A Y.

G PPCONF (6,3) Overlay - Process configuration data record set, if first key word is C O N F I G U R A T I O N.

A	DSMAP	-	Define configuration PPP data maps.
B		-	Initialize.
C		-	Load defaults.

D	PPPORT	-	Read next record, load key word, identify and check order of input.
F	PPCASE	-	Process case record and load lower level defaults, if key word is C A S E.
G	SOLSFP	-	Process solutions list record and load lower level defaults, if key word is S O L U T I O N S.
H	NETCDP	-	Process surface record, if key word is S U R F A C E.

H - Set PPP complete flag, if key word is B E G I N or E N D.

G	FINIS (7,0) Overlay	-	Conclude input data processing
B	ESPOR	-	Write global data to data base.
F	PACLOS	-	Close DIP and MEC data bases.
H	PREGEND	-	End DIP execution.
J	AMCFLR	-	Delete wake networks and update solution lists for CDP cases (response to added mass record).

APPENDIX 3-C DATA BASE COMMUNICATIONS CHART

The Data Base Communications Chart is presented in three forms. Each form is alphabetized by columns, from left to right. The first form has a column order of Data Base, Dataset Name, Map Name, Common Block, and Program/Subroutine. The second form has a column order of Data Base, Map Name, Dataset Name, Common Block, and Program/ Subroutine. The third form has a column order of Common Block, Data Base, Map Name, Dataset Name, and Program/Subroutine. Thus a person can get a cross reference on a data element by knowing either the Dataset Name, Map Name or Common Block name.

FIRST FORM

DATA BASE	DATASET NAME	MAP NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
DIP	CLOS-COND	DIP-CLOS	/NETABC/	NETWDP
DIP	COEF-GEN-BC	DIP-COEF	/NETABC/	NETWDP
DIP	CONFIG-PRINT-PLOT	DIP-CONF	/CONFDA/	PPCONF
DIP	DATA-BASE-HEADER	DIP-HEADER	/HEADER/	INITIL
DIP	GEOM-PRINT-PLOT	DIP-GEOM	/GEOMDA/	PPGEOM
DIP	GLOBAL	DIP-GLOGLO	/GLOBAL/ /DQGPAR/ /NETDAT/ /SOLDAT/	INITIL
DIP	GLOBAL-DB-OUTPUT	DIP-GLODBO	/GLOPPP/	PPPDIR
DIP	GLOBAL-DEFAULTS	DIP-GLODEF	/GLODEF/	INITIL
DIP	GLOBAL-FLOW-PROP	DIP-GLOFLO	/GLOFLO/	INITIL
DIP	GLOBAL-PRINTS	DIP-GLOPRI	/GLOPRI/	INITIL
DIP	LOCAL-FLOW	DIP-LOCF	/NETABC/	NETWDP
DIP	NETWK-BDC	DIP-NETBDC	/NETBDC/	INITIL
DIP	NETWK-SPEC	DIP-NETSPC	/NETSPC/	INITIL NETWDP
DIP	NETWORK-UPDATE-CODES	DIP-NETWUD	/NETWUD/	INITIL
DIP	OFFBODY-OPTIONS	DIP-OBCOUT	/OBCOUT/	FFDATA
DIP	PANEL-COORDS	DIP-NETDIM	/NETDIM/	GEOMDP
DIP	PANEL-COORDS	DIP-PANCRD	/PANCRD/ /NETABC/	GRID
DIP	POINT-PRINT-PLOT	DIP-POIN	/POINDA/	PPPOIN
DIP	SPEC-FLOW	DIP-SPEC	/NETABC/	NETWDP
DIP	STREAMLINE-OPTIONS	DIP-SLCOUT	/SLCOUT/	FFDATA
DIP	SURF-FAM	DIP-SURFAM	/SURFAM/	FORMOM
DIP	SURF-FLOW	DIP-SURDAT	/SURDAT/	SURFLO
DIP	TANG-VEC	DIP-TANV	/NETABC/	NETWDP
DIP	USER-ABUT	DIP-USEABU	/USEABU/	GEOMDP
MEC	DATA-BASE-HEADER	MEC-HEADER	/HEADER/	INITIAL
MEC	MACRO-OPTIONS	MEC-RUNOPT	/RUNOPT/	INITIL

SECOND FORM

DATA BASE	MAP NAME	DATASET NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
DIP	DIP-CLOS	CLOS-COND	/NETABC/	NETWDP
DIP	DIP-COEF	COEF-GEN-BC	/NETABC/	NETWDP
DIP	DIP-CONF	CONFIG-PRINT-PLOT	/CONFDA/	PPCONF
DIP	DIP-GEOM	GEOM-PRINT-PLOT	/GEOMDA/	PPGEOM
DIP	DIP-GLOGLO	GLOBAL	/GLOBAL/ /DQGPAR/ /NETDAT/ /SOLDAT/	INITIL
DIP	DIP-GLODBO	GLOBAL-DB-OUTPUT	/GLOPPP/	PPPDIR
DIP	DIP-GLODEF	GLOBAL-DEFAULTS	/GLODEF/	INITIL
DIP	DIP-GLOFLO	GLOBAL-FLOW-PROP	/GLOFLO/	INITIL
DIP	DIP-GLOPRI	GLOBAL-PRINTS	/GLOPRI/	INITIL
DIP	DIP-HEADER	DATA-BASE-HEADER	/HEADER/	INITIL
DIP	DIP-LOCF	LOCAL-FLOW	/NETABC/	NETWDP
DIP	DIP-NETBDC	NETWK-BDC	/NETBDC/	INITIL
DIP	DIP-NETDIM	PANEL-COORDS	/NETDIM/	GEOMDP
DIP	DIP-NETSPC	NETWK-SPEC	/NETSPC/	INITIL NETWDP
DIP	DIP-NETWUD	NETWORK-UPDATE-CODES	/NETWUD/	INITIL
DIP	DIP-OBCOUT	OFFBODY-OPTIONS	/OBCOUT/	FFDATA
DIP	DIP-PANCRD	PANEL-COORDS	/PANCRD/ /NETABC/	GRID
DIP	DIP-POIN	POINT-PRINT-PLOT	/POINDA/	PPPOIN
DIP	DIP-SLCOUT	STREAMLINE-OPTIONS	/SLCOUT/	FFDATA
DIP	DIP-SPEC	SPEC-FLOW	/NETABC/	NETWDP
DIP	DIP-SURFAM	SURF-FAM	/SURFAM/	FORMOM
DIP	DIP-SURDAT	SURF-FLOW	/SURDAT/	SURFLO
DIP	DIP-TANV	TANG-VEC	/NETABC/	NETWDP
DIP	DIP-USEABU	USER-ABUT	/USEABU/	GEOMDP
MEC	MEC-HEADER	DATA-BASE-HEADER	/HEADER/	INITIL
MEC	MEC-RUNOPT	MACRO-OPTIONS	/RUNOPT/	INITIL

THIRD FORM

COMMON BLOCK	DATA BASE	MAP NAME	DATASET NAME	PROGRAM/ SUBROUTINE
/CONFDA/	DIP	DIP-CONF	CONFIG-PRINT-PLOT	PPPCONF
/DAGPAR/		See /GLOBAL/		
/GEOMDA/	DIP	DIP-GEOM	GEOM-PRINT-PLOT	PPPGEOM
/GLOBAL/	DIP	DIP-GLOGLO	GLOBAL	INITIL
/GLODEF/	DIP	DIP-GLODEF	GLOBAL-DEFAULTS	INITIL
/GLOFLO/	DIP	DIP-GLOFLO	GLOBAL-FLOW-PROP	INITIL
/GLOPPP/	DIP	DIP-GLODBO	GLOBAL-DB-OUTPUT	PPPDIR
/GLOPRI/	DIP	DIP-GLOPRI	GLOBAL-PRINTS	INITIL
/HEADER/	DIP	DIP-HEADER	DATA-BASE-HEADER	INITIL
/NETABC/	DIP	DIP-CLOS	CLOS-COND	NETWDP VALUE
/NETABC/	DIP	DIP-COEF	COEF-GEN-BC	NETWDP VALUE
/NETABC/	DIP	DIP-LOCF	LOCAL-FLOW	NETWDP VALUE
/NETABC/	DIP	DIP-PANCRD	PANEL-COORDS	GRID
/NETABC/	DIP	DIP-SPEC	SPEC-FLOW	NETWDP VALUE
/NETABC/	DIP	DIP-TANV	TANG-VEC	NETWDP NEDATA VALUE
/NETBDC/	DIP	DIP-NETBDC	NETWK-BDC	INITIL GLOOPT NETWDP NECDWR
/NETDAT/		See /GLOBAL/		
/NETDIM/	DIP	DIP-NETDIM	PANEL-COORDS	GEOMDP ABNEID
/NETSPC/	DIP	DIP-NETSPC	NETWK-SPEC	NETWDP NECDWR NEDAPR NOPCHK
/NETWUD/	DIP	DIP-NETWUD	NETWORK-UPDATE-CODES	INITIL NECDWR
/OBCOUT/	DIP	DIP-OBCOUT	OFFBODY-OPTIONS	FFDATA
/PANCRD/	DIP	DIP-PANCRD	PANEL-COORDS	GRID
/POINDA/	DIP	DIP-POIN	POINT-PRINT-PLOT	PPPOIN PPPORT
/RUNOPT/	DIP	MEC-RUNOPT	MACRO-OPTIONS	INITIL
/SLCOUT/	DIP	DIP-SLCOUT	STREAMLINE-OPTIONS	FFDATA
/SOLDAT/		See /GLOBAL/		
/SURDAT/	DIP	DIP-SURFLO	SURF-FLOW	SURFLO FPDAWR
/SURFAM/	DIP	DIP-SURFAM	SURF-FAM	FORMOM FPDAWR
/USEABU/	DIP	DIP-USEABU	USER-ABUT	GEOMDP ABUT
/HEADER/	MEC	MEC-HEADER	DATA-BASE-HEADER	INITIL

APPENDIX 3-D MASTER DEFINITION

The data base master definition listing of the DIP module has been deleted from this document. It is produced from the PAN AIR tape during installation.

4.0 DEFINING QUANTITIES GENERATOR (DQG) MODULE

4.1 Introduction

The Defining Quantities Generator (DQG) is a stand alone program which is a module of the PAN AIR system. DQG performs many tasks which, from a general point of view, translate the definition of the configuration and flow properties in terms which are convenient to a user into a definition which is more mathematically tractable. DQG also performs a number of convenience operations (such as automatically indexing control points and singularity parameters and automatically defining abutments) and performs a comprehensive analysis of the problem for errors in the configuration which might lead to a singular or invalid solution.

4.2 DQG OVERVIEW

4.2.1 PURPOSE OF DQG

The problem of finding the flow around a body of arbitrary shape is reduced by PANAIR to the problem of solving a large system of linear equations viz., $[AIC]\ [\lambda] = [b]$. This is done by approximating the surface of the body by flat rectanguar and/or triangular panels. For each panel two unknown singularity parameters are introduced. Once these parameters $[\lambda]$ are found, the solution to the original problem can be constructed.

DQG performs a variety of calculations to provide data necessary for the construction of the AIC matrix. These calculations are associated with four classes of data: network data, abutment data, control point data and panel data. With regard to network data, DQG indexes control points and singularity parameters in the network and assures that panels in the network are large enough to allow accurate calculation and that panels do not have excessive aspect ratios (less than 10,000). For more details see PAN AIR User's Document, Section B.1.3 (Reference 2). The relationships between networks are defined by the abutment data. This data defines where alternate boundary conditions must be imposed to assure doublet continuity across network boundaries. The control point data defines which user-defined boundary condition or alternate boundary condition is imposed at control points and provides geometrical data (tangent and normal to surface) required for the evaluation of the boundary condition. The panel data includes geometrical properties of the panel and a description of how the source and doublet distribution on the panel surface depends on surrounding singularity parameters. Also included in the panel data are certain integrated moments of source and doublet strength evaluated over the surface of the panel. These are employed by MAG to more efficiently determine the panel influence on control points which are not too near the panel.

While computing the required quantities, DQG constantly evaluates the results for conditions that might produce a singular or incorrect solution. More than seventy irregular conditions are noted by fifty-nine error messages and sixteen warning messages. In addition the user may require DQG to produce printed output which can be evaluated by the knowledgeable user to assure that not only will the data produced by DQG produce a solution, but the solution will be the solution to the problem the user thinks he has defined. (See the PAN AIR User's Manual, Section 7, record G.17 (Reference 2) and Section 8 of the same document.)

4.2.2 DQG Input/Output Data

Input to DQG occurs only through the SDMS database system (see Section 1 and Section 13 of this document). DQG reads databases created by the MEC and DIP modules. The MEC database contains information concerning the names of files which contain the databases which DQG requires or will generate. The DIP database contains the user's description of the problem. The information which DQG reads from the DIP database may be classified as global configuration data (such as Mach number, direction of flow, symmetry properties of the problem), network data (singularity types of networks, boundary conditions which the user wishes to impose, coordinates of points in the network) and abutment data, if any is supplied (a description of how the networks connect together to form the configuration). DQG also reads information on the DIP database which defines what types of printed output the user has requested from DQG.

DQG offers a number of output options which may be selected by the user through data provided to DIP.

The default output consists of a general description of the status of DIP and DQG data bases, timing statements at the end of each overlay, and a description of some global and network properties of the problem at the end of execution. In addition fatal errors encountered during execution produce diagnostic messages. A small number of mandatory warning messages are also produced when a questionable situation arises. No more than ten fatal error messages are permitted to accumulate before execution stops. There is no limit to the number of warning messages. A complete list of error and warning messages is provided in Appendix 4-E. The PAN AIR User's Manual Section 8 (Reference 2) discusses the interpretation of the error and warning messages.

Additional warning messages are printed as situations arise if the user has specified that warning messages are desired. There is no restriction on the number of warning messages that are produced.

Either the coarse grid coordinate of the networks (corner point coordinates) or the fine grid coordinates of the network (corner, edge midpoint and center point coordinates) or both are printed as the user requests. Those network edges which are collapsed have the corner point coordinates flagged to indicate the DQG modified the collapsed edge points to assure that they all had identical coordinates.

DQG may print descriptions of gap filling panels which have been added. Included in this printout are the corner points of the gap filling panels, the edges of the networks to which they are attached, and whether they are triangular gap filling panels.

A description of empty space abutments or of all abutments may be produced. Besides indicating how the networks are joined together, this output also describes which network edges and corner points will be assigned doublet or source matching boundary conditions to replace those which are specified by the user.

Control point and boundary condition data may be printed at the end of execution if the user requests. The control point data include global index and network and fine grid lattice indices of the control point as well as it's hypothetical location, normal vector and boundary condition characterization. The boundary condition data includes all of the indexing information of the control point data and lists all non-vanishing cofficients for the left hand side of the boundary condition equation.

In addition to the above, DQG may be compiled from its program library with diagnostic print statements inserted automatically. A description of how to accomplish this is given in Appendix 4-F.

Appendix 4-G contains an example of output obtained from DQG execution. Section 8.1 of the PAN AIR User's Manual (Reference 2) discusses the interpretation of the DQG output.

4.2.3 Database Interface

Module DQG reads input data from databases created by MEC and DIP. The MEC database provides database names, account numbers, database status, date of execution and other similar information. The DIP database contains the user's description of the problem.

DQG creates a single database during its execution. The database provides a description of the user's problem in a form that the other PAN AIR modules can easily process. The information is used by the MAG, RHS, MDG and PPP modules.

The DQG database master definition is described in Appendix 4-D. (See Section 1 of this document for an introduction to SDMS).

4.3 MODULE DESCRIPTION

The main overlays and subroutines of DQG are briefly summarized in this section. Estimates of the core requirements and execution time requirements of the overlays of DQG are also provided. Lower level subroutines are described in secton 4.4. A tree diagram of the calling relationships of the subroutines in DQG may be found in Appendix 4-A. The DQG functional decomposition is contained in Appendix 4-B.

Figure 4.1 contains a simplified configuration which illustrates the concepts of panel, network and abutment. Singularity parameters are defined to be located on networks. These parameters are related to perturbations in the flow field. The values of the singularity parameters are determined by imposing boundary conditions at selected points on the network called control points. DQG translates fairly simple geometric data into mathematical descriptions of the boundary conditions and singularity parameters.

4.3.1 Overall Structure

The overall structure of DQG is described in Figures 4.2, 4.3 and 4.4. The figures also provide some indication of data flow during DQG execution. The data flow aspects of the figures is discussed in paragraph 4.3.5.

The seven primary overlays of DQG are indicated as rectangular blocks in Figure 4.2. Two primary overlays (the (3,0) and (5,0) overlays) are divided into six and two secondary overlays respectively as is indicated in the figure. The dotted line connecting the main (0,0) overlay with the seven primary overlays indicates that the (0,0) overlay causes each one of the primary overlays to be loaded and executed. Besides the overlay index (e.g. (1,0)) the figure also gives the name of the main program in the overlay (for the (1,0) overlay it is OPENER). Below this there is a short summary of the operations which the overlay performs. The solid lines in the figure indicate the flow of data from the program to the disk files that make up the DQG database and from the MEC, DIP and DQG database files into the program. Note for example that the MEC and DIP databases are read only in the (1,0) overlay and that DQG never writes on either one of them. All other input and output for DQG occurs from or to the DQG database or to the printed output file. Note that the output to the printed output file is not shown in the figure. All overlays of DQG produce some printed outputs.

Figures 4.3 and 4.4 provide a similar overview of the structure and data flow for the secondary overlays of the (3,0) and (5,0) overlays respectively.

4.3.2 Overlay Descriptions

This paragraph describes the major functions which are performed in each primary and secondary overlay of DQG. Paragraph 4.3.5 discusses data flow in the program.

4.3.2.1 OPENER Overlay (1,0)

This overlay obtains input data from the MEC and DIP databases and copies data required to solve the problem onto the DQG databases. Certain data are transformed into a form consistent with efficient processing and some useful data is derived from the basic parameters describing the problem. Figure 4.5 illustrates the main subroutine structure for the (1,0) overlay of DQG. The main program OPENER opens the DIP and MEC databases and creates an empty DQG database. There is only one major subroutine in the overlay. It is called DIPDAT and copies data from the DIP database onto the DQG database.

4.3.2.2 NETDEF Overlay (2,0)

The second overlay checks that the networks satisfy certain required properties and provides a global index for all control points and singularity parameters in the problem. Indexing schemes used in DQG are described more fully in Appendix 4-F. Figure 4.6 illustrates the main subroutines for the (2,0) overlay of DQG. The main program in the overlay is NETDEF. It calls a sequence of subroutines which perform the varied tasks of the overlay. Three main tasks are performed. They may be roughly characterized by the terms geometrical tasks, indexing tasks and output operations. The geometrical tasks are discussed first.

Subroutine DFEDGE defines the coordinates of the corner points on the perimeter of the network. EDGCHK computes the length of the edges to check for collapsed edges (see PAN AIR Theory Manual, Section 1.4 of Appendix D (Reference 1) and the PAN AIR User's Manual Section B.1(Reference 2)). INDCTR computes the coordinates of a point that is at the indicial center of the network. Subroutine TRICHK checks each panel in the network for both aspect

ratio and triangularity. (No interior panel of a network is permitted to be triangular.) Subroutine FINGRD defines the fine grid coordinates of the network from the panel corner point coordinates and writes them in the FINE-GRID-COORDS dataset.

The indexing tasks are performed by SINGDF (and the subroutines it calls) and by CONTPT. SINGDF defines a unique index for every source and doublet parameter in the configuration. The exact indexing schemes are discussed in Appendix 4-H of this manual. This subroutine also labels singularity parameters that lie on a collapsed network edge as "null", i.e., they do not contribute any effects to the flow. Control point indexing is performed by the subroutine CONTPT. Note that control points are always defined for a network at the same locations (all panel center points, the four network corner points and the edge midpoints on the perimeter of the network) even though, for example, on a wake network, all control points located at panel centers do not have any boundary conditions (see PAN AIR User's Manual, Section B.3.4 (Reference 2) and the PAN AIR Theory Document, Appendix G (Reference 1)). This is done to allow consistent processing of flow data in post processing. The final function performed by the (2,0) overlay is to print the coordinates of all of the corner points and/or fine grid points in the network if requested by the user. Subroutine PRTNET performs this task.

4.3.2.3 EDGDEF Overlay (3,0)

The (3,0) overlay calls the secondary overlays (3,1) through (3,6). These programs perform an analysis of abutments in the configuration. At the end of the analysis a complete description of the abutments is printed at the user's request.

4.3.2.4 PRABUT Overlay (3,1)

This program lists the abutments defined by the user in a more complete form than the user provides to DIP. It initiates the automatic abutment search by describing all pairwise abutments which have not been described already by the user. A detailed description of the automatic abutment search is given in Appendix 4-I. Figure 4.7 illustrates the main subroutines in the overlay. First USEABT is called. This subroutine reads the user defined abutments data and fills in any missing information. For example, the user can specify only the network and edge index for all the networks in the abutment. In this case USEABT defines the coarse grid lattice indices which correspond to the start and end points of each network edge in the abutment and adds these coarse grid lattice indices to the abutment data. After all user abutment data is processed, a lower level subroutine (not illustrated) prepares a list of network edge segments which the user has not defined to take part in an abutment. Then in NETABT the automatic abutment search begins. Each edge segment which has not been described by the user is examined to see if any of the other such segments lie near it (see PAN AIR Theory Document, Section 3 of Appendix F (Reference 1)). Subroutine EDGLST prepares a preliminary list of all network edges which lie somewhat close to the edge in question. The remainder of PRABUT defines all pairwise abutment descriptions in which each segment takes part. This process is discussed more fully in Appendix 4-I.

4.3.2.5 ABTMNT Overlay (3,2)

Overlay (3,2) completes the automatic abutment search. This procedure is described in Appendix 4-I. After the search is over, all abutments are checked to assure that they satisfy certain rules. Warning and error messages are produced as questionable or erroneous situations arise. Figure 4.8 illustrates the major subroutines in the (3,2) overlay. Subroutines ABXPND, CONABT and SEARCH complete the automatic abutment search. This process is discussed fully in Appendix 4-I. Subroutine EDGPRP defines some additional data that is required to characterize abutments (labelling of matching edges, etc.). Subroutine CHECK examines all network abutments to see that they conform to the appropriate set of rules concerning abutments (see PAN AIR User's Manual, Section B.3.6 (Reference 2)).

4.3.2.6 GAPSIZ Overlay (3,3)

This overlay computes gap sizes for all of the network abutments. The gap size for a panel and a network edge is the greatest of the distances from the panel to the closer point on all other network edges which take part in the abutment.

4.3.2.7 MATCH Overlay (3,4)

Program MATCH determines which edges and corner points will be used to impose doublet and source matching boundary conditions. Figure 4.9 illustrates the main subroutine structure of the (3,4) overlay. Subroutine EMATCH determines which network edge among those that form an abutment will be used to impose doublet matching boundary conditions at the abutment (see PAN AIR Theory Document, Section 5.3 (Reference 1)). In an abutment where a wake has been assigned voritcity matching, EMATCH will find a doublet analysis edge on which to place the actual boundary condition. EMATCH also defines those edges and corner points where source matching boundary conditions are required. Subroutine INTRSC analyzes the configuration for abutment intersections using a technique from graph theory. This is discussed more fully in Appendix 4-J. Subroutine ASSIGN examines each abutment intersection and assigns an appropriate number of corner points to insure doublet matching at the abutment intersection.

4.3.2.8 GAPPNL Overlay (3,5)

This program adds gap filling panels between network edges which have been declared to form an abutment by the user but which lie further apart from one another than the global tolerance distance. A description of how gap filling panels are constructed is given in the PAN AIR Theory Document, section 6 of Appendix F (Reference 1) and Appendix 4-L of this document. Figure 4.10 illustrates the subroutines in this overlay. The main program GAPPNL searches the abutment related data for abutments where the gap size exceeds the tolerance distance. The edges of network which make up such an abutment are parameterized by subroutine PRMEDG (see the PAN AIR Theory Document, Section 5 of Appendix F (Reference 1)). Subroutine DEFPNL defines the data required to describe the gap filling panel (see Appendix 4-L). Subroutine POSPNL defines gap filling panels for abutments with planes of symmetry and which have gaps larger that the tolerance distance.

4.3.2.9 ADCPSG Overlay (3,6)

ADCPSG adds additional control points and doublet singularity parameters at panel corner points where an abutment begins or ends, if the corner point is not one of the four network corner points. This program also sets up a description of matching edges and corner points in a form which is usable by the fourth overlay (subroutine MTCHPT) and defines the extra hypothetical locations for the matching points (subroutine XHLOC)(see PAN AIR Theory Document, Appendix G (Reference 1)). Figure 4.11 illustrates the subroutine structure of the overlay.

4.3.2.10 BNDYDF Overlay (4,0)

This overlay defines geometrical data required at each control point and selects the appropriate number and type of boundary condition to impose at the point from those supplied by the user and those supplied by DQG. A detailed description of what this overlay produces is described in the PAN AIR Theory Document, Appendices G and H (Reference 1). Figure 4.12 illustrates the major subroutines in the overlay. GETBC obtains the boundary conditions for the control points in the network. Subroutine CENTCP defines geometric data and boundary condition coefficients for the generalized boundary condition equation (see PAN AIR Theory Document, Section 5.4 (Reference 1)) and subroutine EDGECP defines similar data for corner and edge midpoint control points.

4.3.2.11 TOPSPL Overlay (5,0)

This overlay calls the (5,1) and (5,2) overlays in sequence.

4.3.2.12 SAEDGS Overlay (5,1)

This overlay computes doublet spline vectors at points along the edges of networks which form a smooth abutment. A detailed discussion of this process occurs in Appendix 4-K of this document. Figure 4.13 shows the major subroutines in the (5,1) overlay. Subroutine PTSFIL obtains coordinates of the corner points in the vicinity of the smooth abutments (See Appendix 4-K). SNGFIL obtains the singularity parameter indices for doublet singularities located at center points near the smooth abutment. Subroutine PARMSA parameterizes the smooth abutment (see PAN AIR Theory Document, Section 1.2 of Appendix I and Appendix F (Reference 1)). Subroutine COARSP defines the outer spline vectors at each corner point and edge midpoint along the smooth edge (see Appendix 4-K and the PAN AIR Theory Document, Section 1 of Appendix I (Reference 1)). Subroutine FINESP defines the outer spline vector for points on the finer edge in terms of the splines along the coarse edge and the parameterization of the abutment. The details of this process are discussed in Appendix 4-K of this document and in the PAN AIR Theory Document, Appendix I (Reference 1).

4.3.2.13 SPLINR Overlay (5,2)

This program computes source and doublet spline vectors for all points which do not fall on the edge of a smooth abutment. The details of this procedure are provided in the PAN AIR Theory Document, Section 1 of Appendix I (Reference 1) and in Appendix 4-I of this document . Figure 4.14 illustrates the main subroutines in the (5,2) overlay. ANALS computes the outer spline

vectors for source analysis networks. ANALD computes the outer spline vectors for doublet analysis networks. DSGN1S and DSGN1D compute the outer spline vectors for source design and doublet design networks. Subroutine WAKGAP defines the outer spline vectors for wake networks and for gap filling panels. Appendix 4-K discusses the spline operations in greater detail. See also the PAN AIR Theory Document, Sections 1 and 2 of Appendix I (Reference 1).

4.3.2.14 PANDEF Overlay (6,0)

The sixth overlay computes and assembles the panel defining quantities required by MAG for the construction of the AIC matrix. These include geometrical properties (areas, normal vectors and tangents, computed in GEOMQU), the outer spline matrices, $[B^S]$ and $[B^D]$, and subpanel spline matrices, $[SPSPL^S]$ and $[SPSPL^D]$, computed in SDSPLM, (see PAN AIR Theory Document, Sections 1, 2, and 3 of Appendix I (Reference 1)) and the far field moments, computed in FFMOM, (PAN AIR Theory Document, Section 4 of Appendix I (Reference 1)) for each network panel and each gap filling panel. Subroutine PANDEF collects all the information, computes discontinuous source spline vectors, and writes out the data as the PANEL-SPEC dataset. Figure 4.15 illustrates the main subroutines of this overlay.

4.3.2.15 SUMMRY Overlay (7,0)

The (7,0) overlay program SUMMRY transcribes some information on the GLOBAL and NETWK-SPEC datasets and re-writes the datasets to the DQG database. The other routines in this overlay read either the NETWK-SPEC dataset or the BNDRY-CONDN-SPEC dataset in order to produce the requested printed summary of network, control point and boundary condition properties.

4.3.3 Module Database

The master definition of the DQG database are given in Appendix 4-D. The dataset names and contents are described in detail. The database communication charts (See Section 1) may be found in Appendix 4-C.

4.3.4 Data Interfaces

4.3.4.1 System Interfaces

Figure 4.2 through 4.4 illustrate the internal and external interfaces between the module and the MEC, DIP and DQG databases. The DQG database is used by modules MAG, RHS and MDG.

4.3.4.2 Subprogram Interfaces

A tree diagram of all routines in DQG is given in Appendix 4-A. This shows the interrelationships among the subroutines which make up DQG. Each subroutine is briefly described in Section 4.4.2.

4.3.5 Data Flow in DQG

Figures 4.5 to 4.16 illustrate the data flow for the major sections of DQG. They will aid the discussion in the following paragraphs.

After opening the DIP database and creating the DQG database the first overlay calls DIPDAT. Global, network data and abutment data are copied onto the DQG database. Then the boundary conditions are transcribed and written to the DQG database. The transcription is the major task of this overlay. DIP provides information on how many boundary conditions have been specified by the user in the NETWK-BDC dataset. Each coefficient is obtained from the COEF-GEN-BC dataset and the TANG-VEC dataset. The coefficients might be for the whole network, for only center, edge-mid-point or corner control points in the network or for only one control point in the network. DQG requires that all coefficients for one control point are grouped together. Data for network wise NETWK-BNDY-CONDN-IN dataset and is keyed by point type (center, edge mid-point or corner point). Point-by-point specification of boundary conditions is stored by CLASS-5-BC-DATA and is keyed by fine-grid lattice index of the point (See Appendix 4-H). The transcription operation consists of reading a non-zero coefficient from the DIP database, reading the current DQG dataset for the control point, copying the additional coefficient to the output array and writing a modified version of the DQG dataset. This is a somewhat costly I/O operation if point-by-point boundary conditions are specified, i.e., if the user chooses Class 5 boundary condition specifications.

In the second overlay (Figure 4.6) networks are checked for short edges and triangular panels, an EDGE-POINT-COORDS dataset is created which contains the corner points on the network edges sequenced in a counter clockwise direction when looking at the "upper" surface around the network perimeter, and singularity parameters and control points are indexed. (See Appendix 4-H for details concerning indexing schems used in DQG.)

Two complimentary datasets are created by subroutine SINGDF. One is called SINGULARITY-MAP and the other is called SINGULARITY-SPEC. The SINGULARITY-MAP dataset allows one to find the index of a singularity parameter given information about its location in the network. The SINGULARITY-SPEC dataset gives information about where a singularity parameter is located in the network given its index. The CONTROL-PT-SPEC dataset is created by the CONTPT subroutine.

In the third overlay (Figure 4.7) user defined abutments are created by USEABT. The subroutine reads the user description from the dataset USER-ABUT and writes the data to the ABUTMENT-SPEC dataset. The SEARCH-LIST dataset specifies those networks which the user has not defined to form abutments. The SPECIAL-POINTS dataset defines which corner points on a network form start or end points of the abutment. Subroutine EDGLST reads the NETWK-SPEC and the EDGE-PT-COORDS datasets in order to decide which edges lie sufficiently close to another edge that they might form an abutment. The main program PRABUT reads the EDGE-POINTS-COORDS and the NETWK-SPEC datasets. Subroutine NETABT reads SEARCH-LIST to define the pairwise abutment data. This data is written to the database as the IABUT dataset. In the (3,2) overlay (Figure 4.8), subroutine ABXPND reads the IABUT dataset and expands the pairwise abutment description to form the expanded abutment description (see Appendix 4-I). This data is stored on the database in the ABUT-KEYS and EXPANDED-ABUTMENT datasets.

Subroutine CONABT reads the expanded abutment descriptions, contracts them to form the abutment description (see Appendix 4-I) and writes the abutment data to the ABUTMENT-SPEC dataset. After all abutments are processed, subroutine SEARCH reads the abutment descriptions and writes out the SEARCH-LIST dataset again. This dataset is used later by subroutine MTABUT (not illustrated) to define the empty space abutments. Subroutine CHECK reads the ABUTMENT-SPEC dataset and checks that the abutments satisfy certain rules. Occasionally the subroutine modifies some of the abutment data (when the rules have been violated) and must re-write the ABUTMENT-SPEC dataset. CHECK also labels singularity parameters which lie on a smooth abutment as null. Thus it reads the SINGULARITY-MAP dataset and re-writes the information contained in it as both the SINGULARITY-MAP and SINGULARITY-SPEC datasets after it sets a flag indicating that the singularity parameter is null. It also reads, sets a flag and re-writes a SPECIAL-POINTS dataset to indicate to the fourth overlay that those control points on the smooth edge are also null, i.e., that they do not have any boundary conditions to impose at them.

In the (3,4) overlay, (Figure 4-9) the main program MATCH reads the ABUTMENT-SPEC and EMPTY-SPACE-ABUT DATASETS and re-writes them after the edge matching data has been defined by EMATCH. Subroutine EMATCH reads the NETWK-SPEC dataset to determine the edge types of networks which might make up and abutment.

Subroutine INTRSC reads the ABUTMENT-SPEC and EMPTY-SPACE-ABUT datasets and generates the INTERSECTION dataset. This describes the abutment intersections in the problem. The INTERSECTION dataset is read by subroutine ASSIGN. A decision is made as to which corner point is assigned to impose doublet matching and the appropriate abutment data is read from the database. The data is re-written after setting the proper matching corner point flag.

In the (3,5) overlay GAPPNL reads the abutment data and checks the gap sizes stored in th GAP-SIZE dataset. The GAP-SIZE dataset had been defined in the (3,3) overlay. If gap filling panels are defined for the abutment, a flag is set and the abutment data is re-written to the database. Also a GAP-PANEL dataset is created which describes the gap filling panel.

In the (3,6) overlay (Figure 4.11) subroutine ADCPSG reads the SPECIAL-POINTS dataset and, by noting where abutments start or end at places other than at the corner of a network, adds extra singularity parameters and control points. It writes CONTROL-PT-SPEC datasets, SINGULARITY-MAP datasets and SINGULARITY-SPEC datasets. Subroutine MTCHPT reads the abutment data (ABUTMENT-SPEC) and the SPECIAL-POINTS datasets, transfers the matching information from the abutment data to the SPECIAL-POINTS dataset and re-writes the SPECIAL-POINTS dataset. Subroutine XHLOC reads the abutment data, edge point coordinates and the network data and determines the coordinates of the extra hypothetical locations of control points along an abutment. See PAN AIR Theory Document Section 5.4.1 (Reference 1). This data is written as the EXTRA-HYPO-LOC dataset.

The data flow in the (4,0) overlay (Figure 4.12) is fairly simple. The NETWK-SPEC dataset is read by BNDYDF as is the coordinates of the corner points (PANEL-CORNER-COORDS). Boundary conditions are read from either NETWK-BNDY-IN or CLASS5-BC-IN datasets by subroutine GETBC. The CONTROL-PT-SPEC dataset is read by CENTCP and EDGECP to obtain the control

point index. After all the required data is assembled, the CONTROL-PT-SPEC, BNDRY-CONDN-SPEC and B-POINTER datasets are written. These summarize all of the boundary condition information for the control point. Note from Figure 4.12 that the subroutine EDGECP additionally reads the SPECIAL-POINTS dataset to obtain information about where to impose matching boundary conditions. Note also that some additional I/O occurs if there are any known singularity parameters. If some boundary conditions lead to known singularity parameters, a lower level subroutine in the fourth overlay reads the SINGULARITY-MAP dataset, sets the appropriate known singularity flag and re-writes the data as both the SINGULARITY-MAP and SINGULARITY-SPEC datasets.

In the (5,1) overlay (Figure 4.13), the main program reads the ABUTMENT-SPEC dataset to find a smooth abutment. Subroutine PTSFIL obtains the required corner point coordinates from the PANEL-CORNER-COORDS dataset and subroutine SNGFIL obtains the required singularity parameter indices from the SINGULARITY-MAP dataset. The edges are parameterized by subroutine PARMSA, which reads the EDGE-POINT-COORDS dataset. Subroutine COARSP computes the outer spline vectors and writes them as the B-SPLINE-DOUBLET dataset. This process requires reading some previously computed outer spline data as well as the NETWK-SPEC dataset. Also an INTERIOR-SPLINE dataset is written. It is used in the (5,2) overlay to prevent a dependence of doublet strength on too many doublet parameters (see Appendix 4-K and the PAN AIR Theory Document, Section 1 of Appendix I (Reference 1)). Subroutine FINESP reads the doublet spline data for the coarse edge, computes the doublet spline for the fine edge and writes it as an additional element set of the B-SPLINE-DOUBLET dataset.

The I/O in the (5,2) overlay (Figure 4.14) for each major subroutine is very similar to the others. Coordinate data (PANEL-CORNER-COORDS and EDGE-POINT-COORDS), singularity parameter indices (SINGULARITY-MAP dataset) and surrounding spline vectors (B-SPLINE-DOUBLET or B-SPLINE-SOURCE datasets) are read, the new spline vector is computed and the vector is written to the B-SPLINE-DOUBELT or B-SPLINE-SOURCE dataset.

In the (6,0) overlay (Figure 4.15) network data, panel corner coordinates and gap panel data is read by the main program PANDEF. Subroutine GEOMQU reads the GAP-SIZE dataset to compute the gap size to panel size ratio. SPLINM reads the source and doublet spline vectors for the nine panel defining points and assembles them into the outer spline matrix. After all panel data are computed, PANDEF computes discontinuous source splines and writes the data as the MAG-PANEL-SPEC, MDG-PANEL-SPEC and PANEL-SING datasets.

The (7,0) overlay program SUMMRY transcribes some information on the GLOBAL and NETWK-SPEC datasets and re-writes the datasets to the DQG database. The other routines in this overlay read either the NETWK-SPEC dataset or the BNDRY-CONDN-SPEC dataset in order to produce the requested printed summary of network, control point and boundary condition properties.

This completes the execution of DQG.

4.4 LOWER LEVEL FUNCTIONS

The following paragraphs present the functional decompositions of the overlays and their subprograms and gives the purpose of each subroutine.

4.4.1 Functional Decomposition

DQG functional decomposition is given in Appendix 4-B.

4.4.2 Subroutine Descriptions

The subroutines used in DQG are described below.

ABASGN

Sets matching doublet flag in ABUTMENT-SPEC or EMPTY-SPACE-ABUT dataset for imposition of matching boundary condition at corner point of network. In this fashion it assigns a matching corner point to the abutment (See Appendix 4-J).

ABXPND

Constructs expanded abutment descriptions from the pairwise abutment descriptions (See Appendix 4-I).

AINV

Constructs the inverse of the reference coordinate to local subpanel coordinate transformation (i.e., it computes the subpanel local to reference coordinate transformation.) See PAN AIR Theory Document, Appendix E, Section E.3 (Reference 1).

ANALD

Top level routine for the computation of doublet spline vectors for doublet analysis networks.

ANALS

Top level routine for the computation of source spline vectors for source analysis networks.

ANDFW

Top level routine for the computation of forward weighted doublet analysis network splines. This subroutine is a copy of ANALD.

ASGNBC

Assigns boundary conditions to control points from user-specified boundary conditions and DQG generated conditions.

ASGNM

Defines boundary condition coefficients and arrays to impose source, doublet or vorticity matching boundary conditions; if closure conditions are specified, calls routine to define closure data.

ASGNU

Defines boundary condition coefficients and arrays from user-specified boundary conditions.

ASSIGN

Sets up the data needed for the selection and assignment of matching corner points to abutments which form an abutment intersection (See Appendix 4-J).

BNDYIN

Creates boundary condition dataset from class and subclass data or term-by-term data provided by DIP.

CBLFFM

Computes doublet cross product far field moments. See PAN AIR Theory Document, Section 4 of Appendix I (Reference 1).

CCPFN

Selects index of corner point on finer network edge which is closest to specified point on coarser edge. Used to generate smooth abutment splines (see Appendix 4-K of this document).

CCPGEO

Computes geometric properties of corner control points. See PAN AIR Theory Document, Appendix G (Reference 1).

CENTCP

Creates CONTROL-PT-SPEC and BNDRY-CONDN-SPEC datasets for center control points by computing geometric properties and assigning boundary conditions to the point. See PAN AIR Theory Document, Appendix G and Appendix H (Reference 1).

CENTER

Computes coordinates of center point of panels when performing smooth abutment spline calculations.

CHECK

Checks all network abutments to assure that they do not violate certain rules (See PAN AIR User's Document Section 3.5 of Appendix B (Reference 2)).

CHOOSE

Chooses boundary conditions from user specified and DQG specified boundary conditions to assign to control points.

CHKPOS

Checks that networks which lie on a plane of symmetry totally lie on a plane of symmetry. See PAN AIR Theory Document, Section 1.4 of Appendix K, and PAN AIR User's Document Sec. B.1.3 (Reference 2).

CLOSTR

Creates a DQG database CLOSURE-IN dataset containing all of the values of the closure coefficient required from the DIP input which is in the form of one coefficient value per dataset.

CNCPBC

Assigns center control point boundary conditions.

COARSP

Top level routine for computation of smooth abutment spline vectors along coarser edge of smooth abutment.

COLAPS

Collapses coordinates of a short network edge to a single point.

COLCPT

Sets up an array required to assure that the assignment of corner points to abutments to assure doublet matching at abutment intersections is performed correctly when the intersection includes the collapsed edge of a network (See Appendix 4-J).

CONABT

Contracts the expanded abutment description to form an abutment description (See Appendix 4-I).

CONTPT

Indexes control points in a network (See Appendix 4-H).

COPYBC

Copies value of single boundary condition coefficient from data supplied by DIP to array of boundary condition coefficients used by DQG.

CPANAL

Computes spline vector for corner point on the edge of a doublet analysis network.

CPCSEL

Selects indices of adjacent corner points on coarser edge of a smooth abutment which spans specified point on finer edge (See Appendix 4-K of this document).

CPDSGN

Computes spline vector for corner points on the edge of a doublet design I network.

C13QTR

Computes the coordinates of a point one-quarter and three-quarters of the way along an edge segment of an abutment (See appendix 4-I).

DATANL

Selects surrounding points at which doublet singularity parameters are located to use in computing a doublet spline vector for a specified point. See Appendix 4-K of this document and the PAN AIR Theory Document, Section 1 of Appendix I (Reference 1).

DATS13

Selects surrounding points at which source singularity parameters are located to use in computing a source spline vector for a specified point. See Appendix 4-K of this document and the PAN AIR Theory Document, Section 1 of Appendix I (Reference 1).

DBLFFM

Computes the doublet for field moment integrals for a panel.

DCSASP

Computes discontinuous outer splines for source analysis networks.

DEFLSQ

Defines coordinate and spline vector at specified point for use in computing spline vectors at points on a smooth abutment.

DEFPNL

Defines geometrical data required to create a gap filling panel (See Appendix 4-L).

DEGOUT

Copies degenerate boundary condition to output array for case of network which lies on a plane of symmetry.

DEGPRP

Defines boundary condition data for degenerate case of a network which lies wholly on a plane of symmetry.

DFEDGE

Creates EDGE-POINT-COORDS dataset in which corner point coordinates of points on a network edge and adjacent to a network edge are listed in a sequence which corresponds to traversing the network edge in a counter clockwise direction.

DIPDAT

Reads data from the DIP database and writes datasets on the DQG database, sometimes changing or combining the data into a form which DQG requires.

DISTQT

Computes the distances from the one-quarter and three-quarter points of a network edge segment to the one-quarter and three quarter points of a reference network edge segment. Used in the automatic abutment search procedure (See Appendix 4-I).

DQGOUT

Copies DQG boundary condition to output array.

DSCT

Determines how many DIP boundary condition related datasets must be read by DQG to define the complete boundary condition arrays.

DSGN1D

Top level routine for the computation of doublet spline vectors for doublet design I networks.

DSGN1S

Top level routine for the computation of source spline vectors for source design I networks.

DSGN2D

Dummy routine in case design II doublet capabilities are added to PAN AIR.

DSGN2S

Top level routine for the computation of source design II spline vectors.

DTENSR

Computes D tensor for computation of far field moments. (See PAN AIR Theory Document, Section 1 of Appendix I (Reference 1)).

ECPGEO

Computes geometric properties of edge control points. (See PAN AIR Theory Document, Appendix G (Reference 1)).

EDGCAL

Computes average panel length and minimum panel length along edge of network.

EDGCHK

Decides whether a network edge satisfies the conditions which require it to collapse. (See PAN AIR User's Document, Section 3.2.2 (Reference 2), and PAN AIR Theory Document, Section 1.3 of Appendix D (Reference 1)).

EDGCLS

Reads closure boundary conditions, adds required geometrical information and writes a closure dataset.

EDGECP

Controls processing of corner point and edge midpoint control points. Causes geometric properties to be computed and boundary conditions to be defined for control points on network edges.

EDGLAT

Transforms counter-clockwise-sense- sequential index of corner point on a network edge to coarse lattice indices of point (See Appendix 4-H).

EDGLS

Computes quadratic one-dimensional fit to four points. Used for computation of doublet edge spline for non-matching edges of design networks.

EDGLST

Prepares a list of edge candidates for a pairwise abutment (See Appendix 4-I).

EDGPRP

Computes and defines properties of network edge segments which make up an abutment, such as upstream factor, matching/non-matching flag and supersonic factor. See Appendix 4-I and 4-J and the PAN AIR Theory Document, Section 4 of Appendix F (Reference 1).

EDGSGQ

Defines edge segments of a network edge which will be fit by a quadratic one dimensional spline. See Appendix 4-K and the PAN AIR Theory Document, Section 1.2.5 of Appendix I (Reference 1).

EMATCH

Sets matching source, doublet and vorticity flags in Abutment-Spec or Empty-Space-Abut datasets for imposition of matching boundary conditions at edge midpoints along edge segment in abutment. See Appendix 4-J and the PAN AIR Theory Document, Section 4 of Appendix F (Reference 1).

EMDSGN

Computes spline vectors at edge midpoints along non-matching edges of doublet design I networks.

FFMOM

Computes basic far field moments and calls routines which compute source, doublet and doublet cross product moments. See PAN AIR Theory Document, Section 4 of Appendix I (Reference 1).

FINESP

Computes spline vectors on the finer edge of a smooth abutment. See Appendix 4-K.

FINGRD

Computes fine grid coordinates for a network.

FLIND

Computes fine grid lattice indices for points on network edges which form part of a smooth abutment.

GAMVEC

Computes the gamma vector (described in PAN AIR Theory Document, Section 1.5 of Appendix I (Reference 1) which is used to construct the edge spline for non-matching edges of doublet design networks.

GAPSPL

Defines spline vectors for gap filling panels. See Appendix 4-L.

GEOMQU

Controls computation of geometric data which are written to the Panel-Spec dataset.

GETBC

Obtains user's boundary condition coefficients for control points from the DQG data base.

GISTYP

Determines the symmetry type for a particular boundary condition.

INDCTR

Defines the coordinate of the indicial center of a network.

INTERN

Writes a simplified spline vector for use in splining calculations for points on the interior of a network which lie close to an edge which is part of a smooth abutment. This assures that the spline is carried over to the adjacent network across the smooth boundary without producing a panel doublet spline matrix which depends on too many singularity parameters. See Appendix 4-K.

INTRSC

Defines connections between corner points at abutment intersections and then finds all abutment intersections in the entire configuration. See Appendix 4-J.

KAPVEC

Defines the Kappa vectors used to compute subpanel doublet spline matrices. See PAN AIR Theory Document, Section 2.2.2 of Appendix I (Reference 1).

KNOWSP

Defines known singularity parameters for control points which have known source or known doublet characteristics. See PAN AIR Theory Document, Appendix H (Reference 1).

LATBC

Defines fine grid lattice indices for control points when the user has defined point-by-point boundary condition specifications for a network. See PAN AIR User's Manual Sections 7.4 and B.3.1 (Reference 2).

LATEDG

Transforms coarse lattice indices of corner point at edge of network to counter-clockwise-sense-sequential index around the edge. See Appendix 4-H and the inverse routine EDGLAT.

LATIND

Computes fine grid lattice index from coarse lattice indices and point type (corner, center edge mid-point row and edge mid-point column). See Appendix 4-H.

LOC2D

Computes a local two dimensional coordinate system for use in computing doublet splines on the edge of a network which forms part of a smooth abutment.

LSQDAT

Defines coordinate of a point and index of singularity parameter located there for use in computing doublet and source splines at points on a network interior.

MAPB

Defines SDMS maps used in the (2,0) overlay of DQG.

MDCP

Finds the most distant center point adjacent to the corner point on the finer edge of a smooth abutment which is closest to a specified edge mid-point on the coarser edge. See Appendix 4-K.

MDPLSG

Computes minimum distance from a point to a line segment.

MODBC

Modifies boundary conditions on superinclined panels and subpanels. See PAN AIR Theory Document, Section 2.1 of Appendix H (Reference 1).

MPPARM

Sets flags indicating to MAG which of the VIC, VIC $\cdot$ N_C or (VIC_x, VIC_y, VIC_z) need to be computed and/or saved for use by MDG.

MTABUT

Defines empty space abutments at all network edge segments which do not take part in network abutments.

MTCHPT

Reads matching flags from abutment data and sets flags in SPECIAL-POINTS dataset which indicates the 4th overlay which control points will receive matching boundary conditions.

NBCLAS

Determines number of boundary conditions user has imposed at control point and the boundary condition class of the user input.

NETABT

Searches the network edges, finds edges which abut and defines the pairwise abutment data.

NTEDGA

Controls computation of doublet edge spline on analysis-type edges of networks.

NTEDGD

Controls computation of doublet edge spline on non-matching edges of design networks.

NTRLST

Defines abutment intersections. See Appendix 4-J.

ONDFCT

Computes a one-dimensional source spline for one column or one row networks.

PARMSA

Controls parametrization of network edge segments which take part in a smooth abutment. See Appendix 4-K.

PANGEO

Computes geometrical data associated with a panel.

PANPRJ

Project reference coordinates to local panel coordinates.

PANSIZ

Computes panel size for panels on network edges and compares them with the gap size.

PANSUB

Computes source and doublet panel sub-splines for use in intermediate field PIC computations in MAG. See PAN AIR Theory Document, Section 3 of Appendix I (Reference 1).

PBCDAT

Prints boundary condition data at all control points in the configuration. See Appendix 4-H.

PCPDAT

Points control point data at all control points in the configuration. See Appendix 4-H.

PGNDAT

Points global and network properties of the problem. See Appendix 4-H.

POINT

Computes the coordinates of the specified center point, edge midpoint or corner point from the columns of corner point coordinates which are available in core. Used in the computation of spline vectors at network interiors.

POSPNL

Constructs gap filling panels at network edges which abut a plane of symmetry, see Appendix 4-L and the PAN AIR Theory Document, Appendix F (Reference 1).

PRMEDG

Parameterizes a network edge segment. See PAN AIR Theory Document, Section 6 of Appendix F (Reference 1).

PRTNET

Prints network corner point coordinates and fine grid coordinates as requested by the user. See PAN AIR User's Document, Section 7.3 (Reference 2).

PTSFIL

Fills arrays of coordinates that are required to compute smooth abutment splines. See Appendix 4-K.

Q1DFIT

Computes a quadratic one dimensional fit to three one-dimensional coordinates.

SALSQC

Controls computation of doublet splines at corner points on a network edge which takes part in a smooth abutment. See Appendix 4-K.

SALSQE

Controls computation of doublet splines at edge midpoints on a network edge which takes part in a smooth abutment. See Appendix 4-K.

SDSPLM

Assembles source or doublet spline matrix from spline vectors obtained from the DQG database.

SEARCH

Defines segments of network edges which have not already been included as part of a network abutment. See Appendix 4-I.

SEDGCL

Redefines source spline vectors on the collapsed edge of a network. See Appendix 4-K.

SELECT

Chooses a corner point for assignment to an abutment in order to impose doublet matching at an abutment intersection. This subroutine is not used in PAN AIR Version I. See Appendix 4-J.

SINGDF

Controls the process of assigning indices to singularity parameters of networks. See Appendix 4-H.

SIP

Computes the trace of the inner product of the D tensor with a specified segment of the far field moment matrix, that is, the shifted inner product of the tensors. See PAN AIR Theory Document, Section 4 of Appendix I (Reference 1).

SNGDA

Indexes singularity parameters on analysis edges of networks. See PAN AIR Theory Document, Section 1 of Appendix D (Reference 1) and Appendix 4-H of this document.

SNGDD2

Indexes singularity parameters on edges of source design II networks. See Appendix 4-H.

SNGDW1

Indexes singularity parameters on matching edge of doublet wake I network. See PAN AIR Theory Document, Section 1 of Appendix D (Reference 1) and Appendix 4-H of this document.

SNGDW2

Indexes singularity parameter on matching edge of doublet wake II network. See Appendix 4-H and the PAN AIR Theory Document, Section 1 of Appendix D (Reference 1).

SNGDEX

Computes or obtains from the database the singularity index of a singularity parameter located at the specified point. Used for spline computations.

SNGDFW

Generates the singularity parameters for the edges of a forward weighted doublet analysis network.

SNGFIL

Fills array with indices of singularity parameters which are required to compute doublet splines at a smooth abutment. See Appendix 4-K.

SNGNUL

Defines null singularity parameters on collapsed edges of networks. See PAN AIR Theory Document, Section 1.4 of Appendix D (Reference 1).

SNGPAN

Indexes singularity parameters on network interiors. See Appendix 4-H.

SNGSD1

Indexes singularity parameters on edges of doublet design I networks. See Appendix 4-H.

SNGSD2

Indexes singularity parameters on edges of source design II networks. See Appendix 4-H.

SPLA

Obtains data, computes doublet spline and assembles spline vector for specified point on the interior of a network. See Appendix 4-K.

SPLINM

Controls computation and assembly of source and doublet spline matrices, subpanel spline matrices and panel-wide subspline matrices.

SPLTRN
Computes coordinate of specified point in local two dimensional coordinate system for use in computation of spline.

SRCFFM
Computes source far field moments. See PAN AIR Theory Document, Section 4 of Appendix I (Reference 1).

SSIP
Computes scaled shifted inner product of D-tensor with basic far field moment. See paragraph 4.4.2 SIP and PAN AIR Theory Document, Section 4 of Appendix I (Reference 1).

SSP13
Obtains data computes source spline and assembles spline vector for specified point. See Appendix 4-K.

SUBGEO
Computes geometric data for subpanels.

SUBSPL
Computes subpanel spline matrix. See PAN AIR Theory Document, Section 4.2.1, Section 5.5 and Section 2 of Appendix I (Reference 1).

SYMFFM
Symmetrizes far field moments. See PAN AIR Theory Document, Section 4 of Appendix I (Reference 1).

TANGOP
Computes tangent vector at control point according to user option. See PAN AIR User's Manual, Section 7.4 (Reference 2).

TRICHK
Checks panel in networks for short edges which indicate panel is triangular.

UNISPL
Defines unit spline vector. See Appendix 4-K.

UPDOWN
Calculates upstream downstream parameters. See Appendix 4-J.3

USEABT
Defines abutments according to user specification. See Appendix 4-I, and the PAN AIR User's Document, Section B.3.5 (Reference 2).

USROUT
Copies user boundary condition data to output array.

VECUNM
Assembles spline matrix from spline vectors.

VECUNV
Assembles spline vectors from coefficients of constrained least squares fit and spline vectors of points used in fit.

XHLOC

Computes extra hypothetical locations of control points which are used to match source, doublet or vorticity strength. See PAN AIR Theory Document, Appendix G (Reference 1).

XIETAV

Computes a local two dimensional coordinate system for use in computing splines at points on the interior of a network. See Appendix 4-K and the PAN AIR Theory Document, Section 1.2.3 of Appendix I (Reference 1).

XSCRIB

Transcribes DIP boundary condition coefficients (stored one per dataset) into a form required by DQG (all coefficients for one control point in one dataset).

XTEND

Computes the first four rows of the extension matrix for source design II networks. See PAN AIR Theory Document, Section 2.1.3 of Appendix I.

WAKGAP

Computes spline vectors for wake networks and calls routine which defines spline vectors for gap filling panels.

WTLSQ

Defines upstream weighting coefficient for computing source and doublet splines. See Appendix 4-K and PAN AIR Theory Document, Section 1.2.4 of Appendix I (Reference 1).

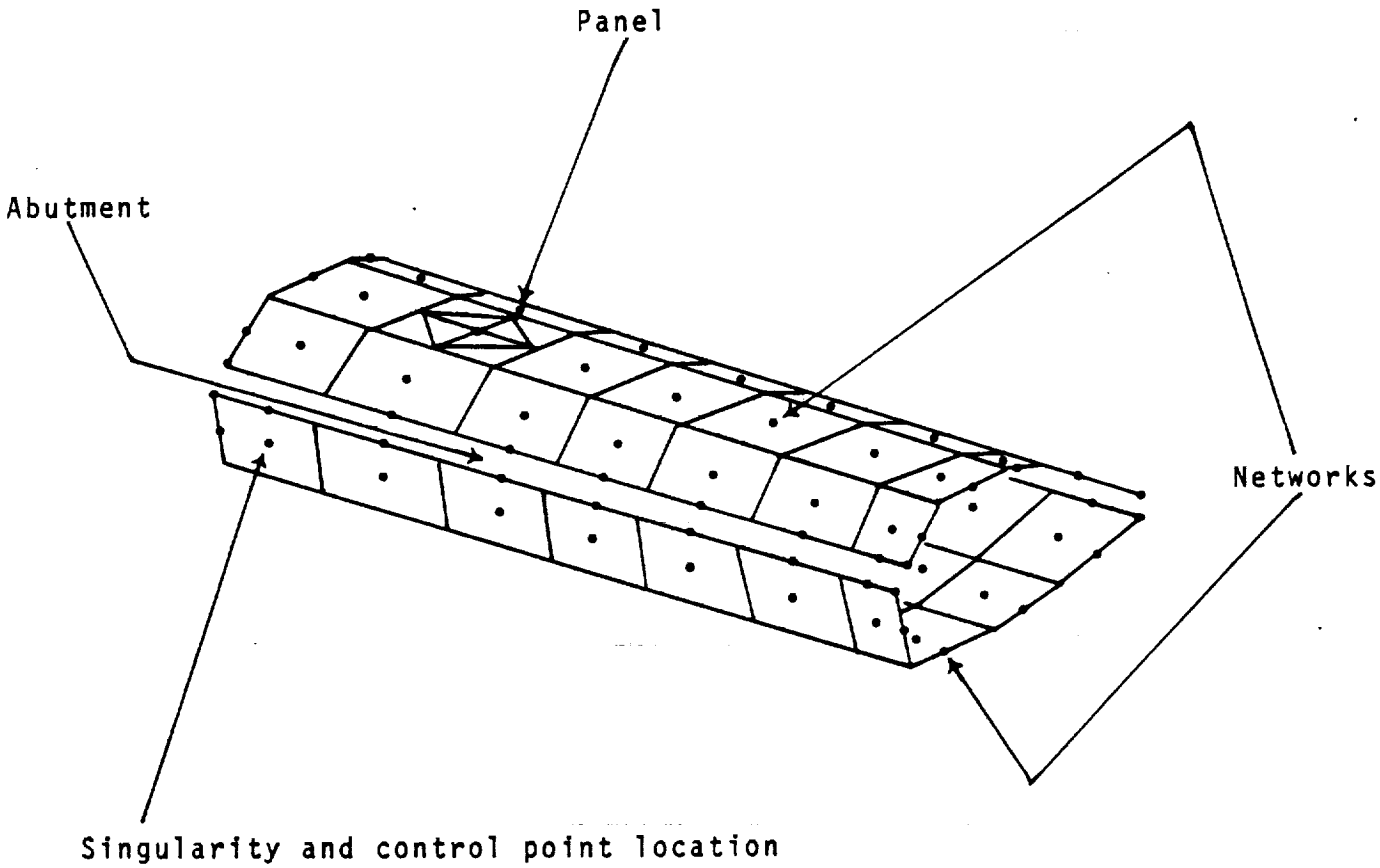


Figure 4.1 - Illustration of Network, Abutment and Panel

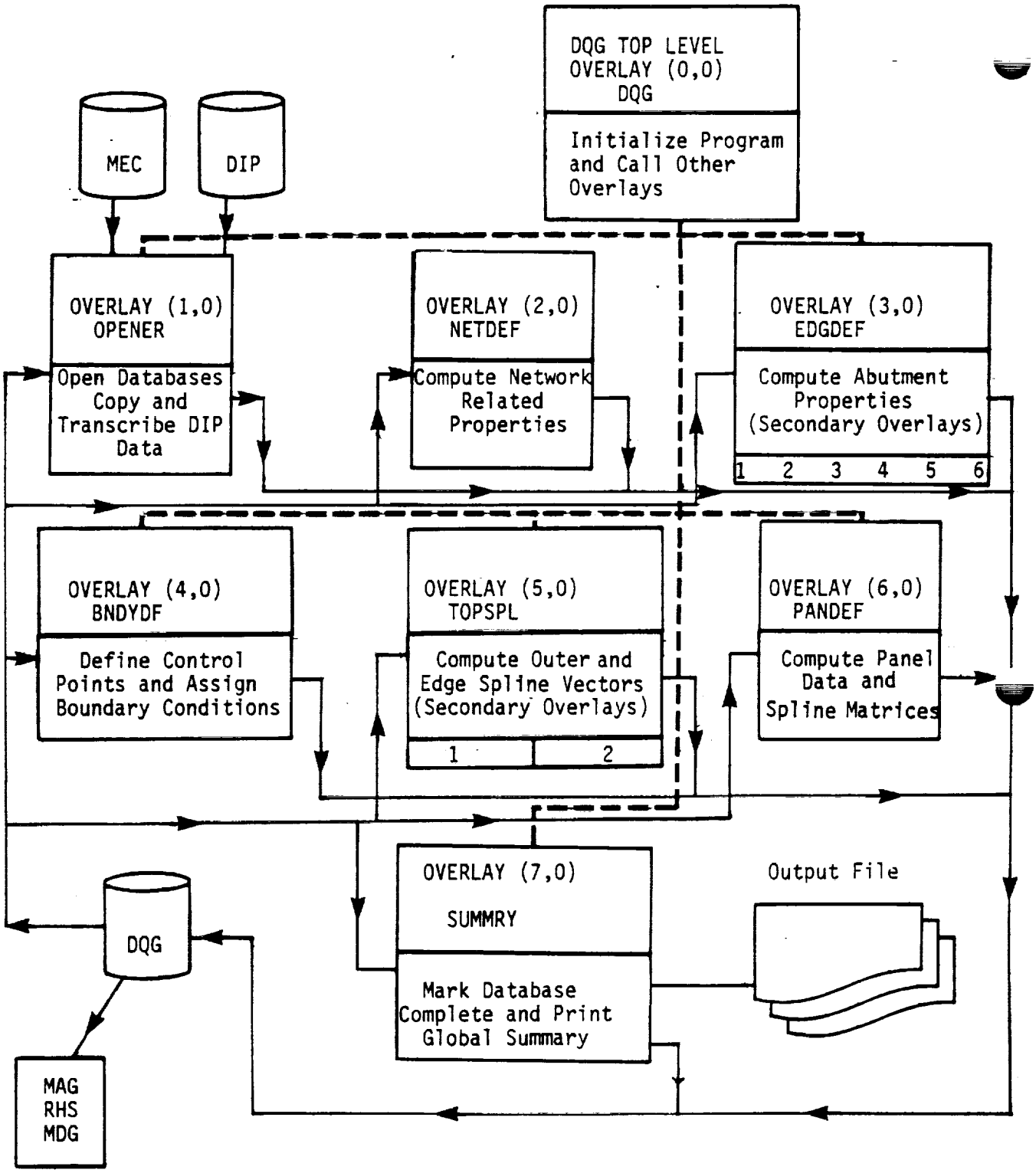


Figure 4.2 - Top Level Structure of DQG

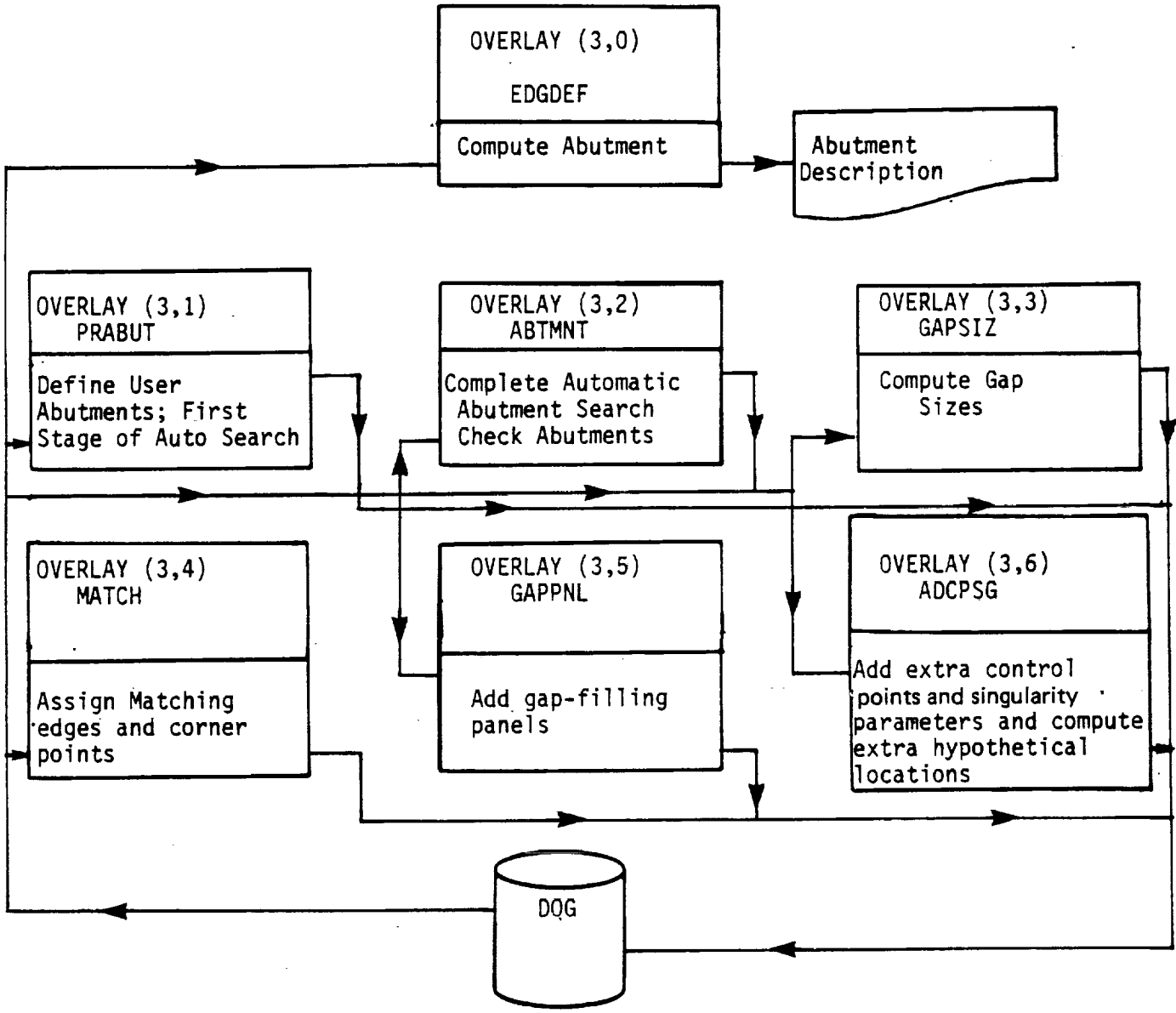


Figure 4.3 - Top Level Structure of Overlay (3,0)

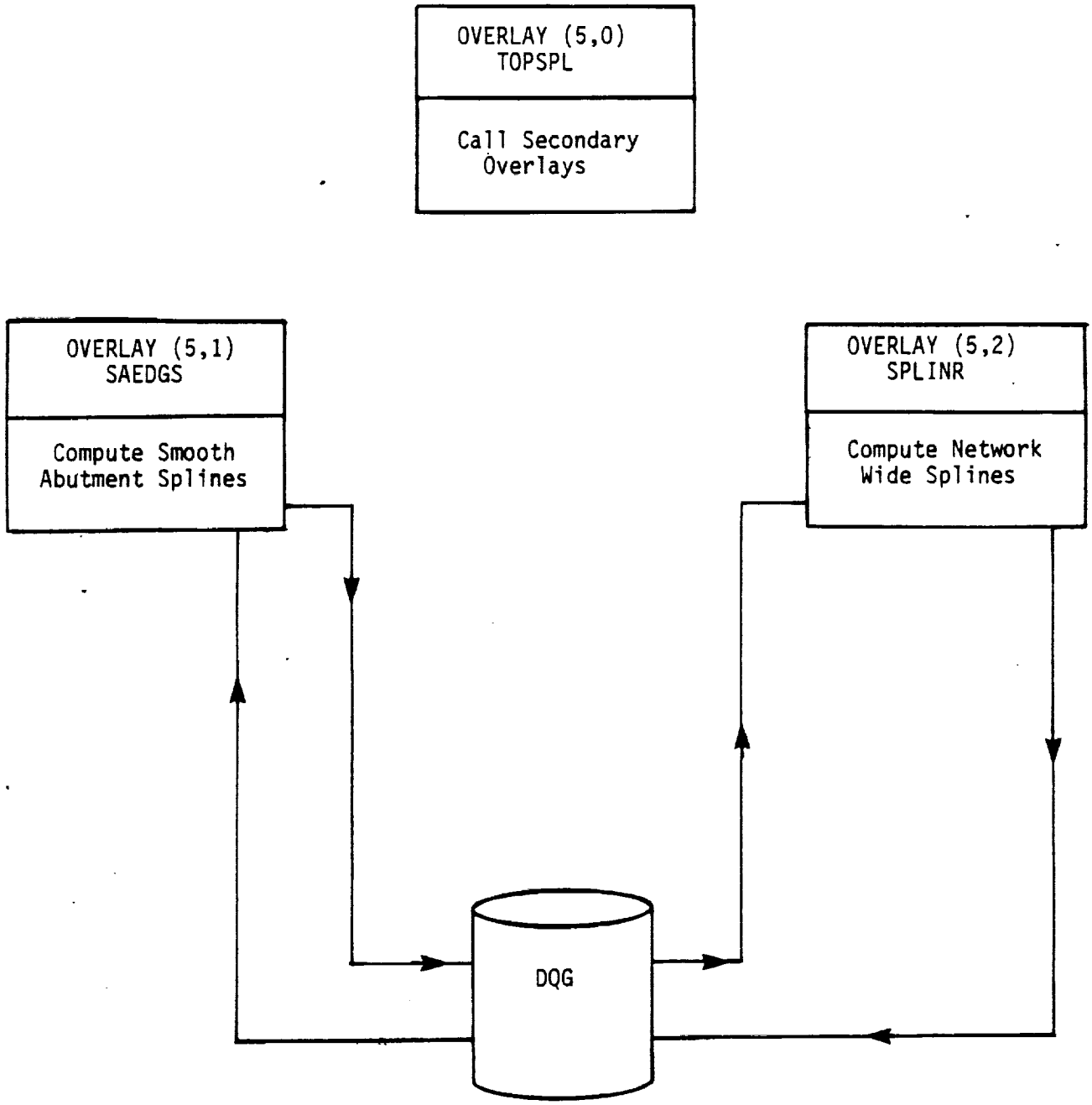


Figure 4.4 - Top Level Structure of Overlay (5,0)

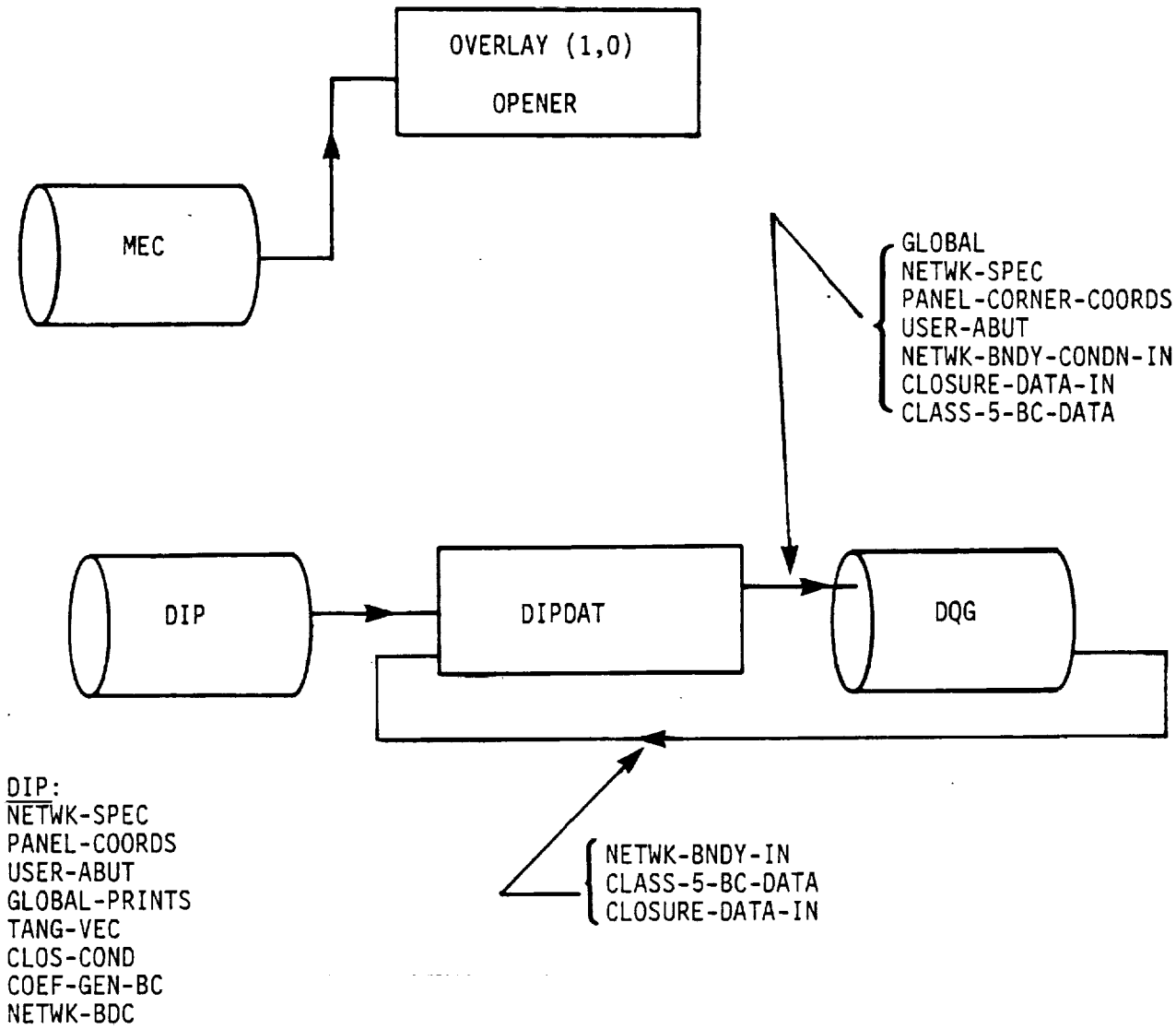


Figure 4.5 - Structure and Data Flow of Overlay (1,0)

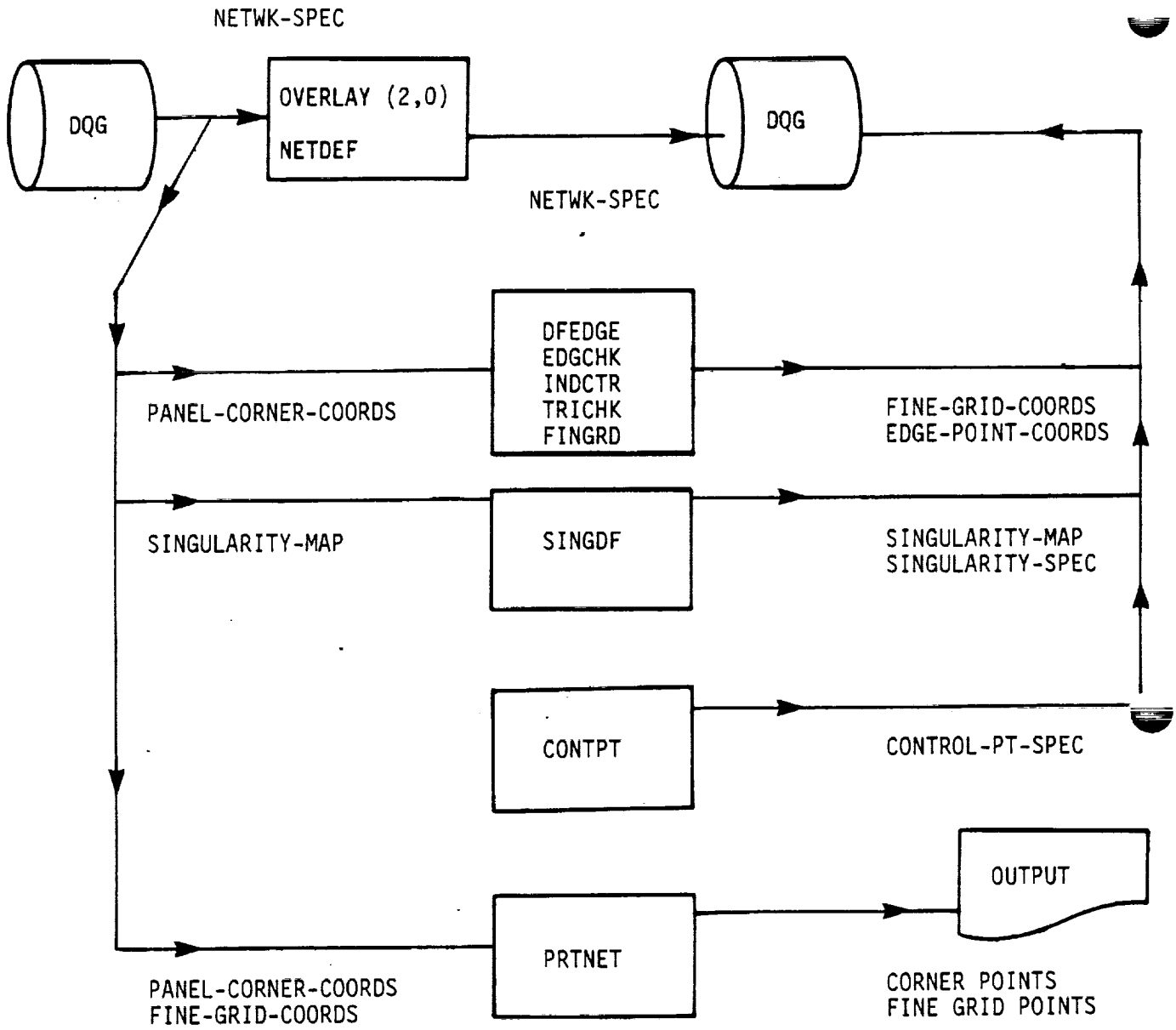


Figure 4.6 - **Structure** and Data Flow for Overlay (2,0)

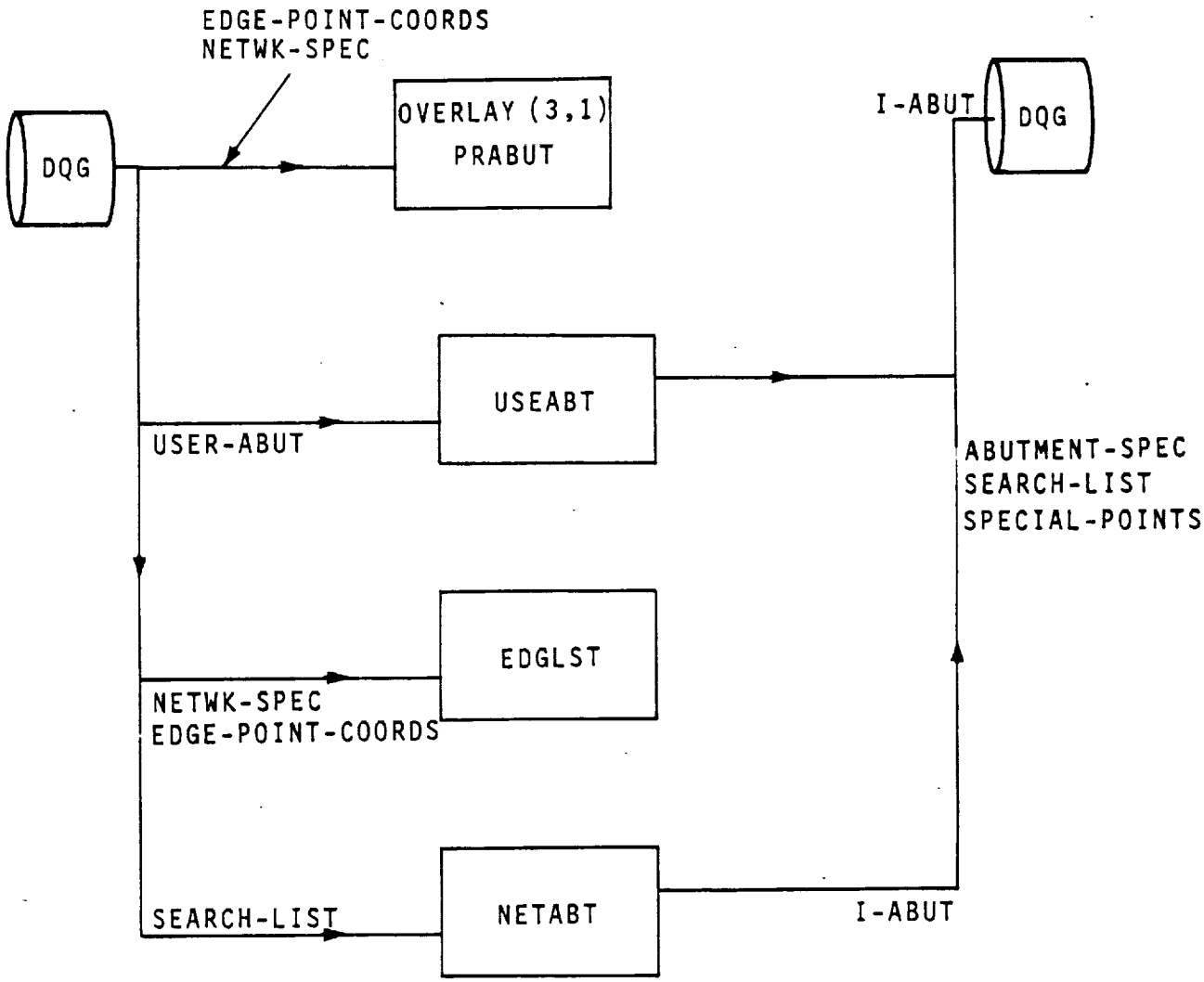


Figure 4.7 - Structure and Data Flow for Overlay (3,1)

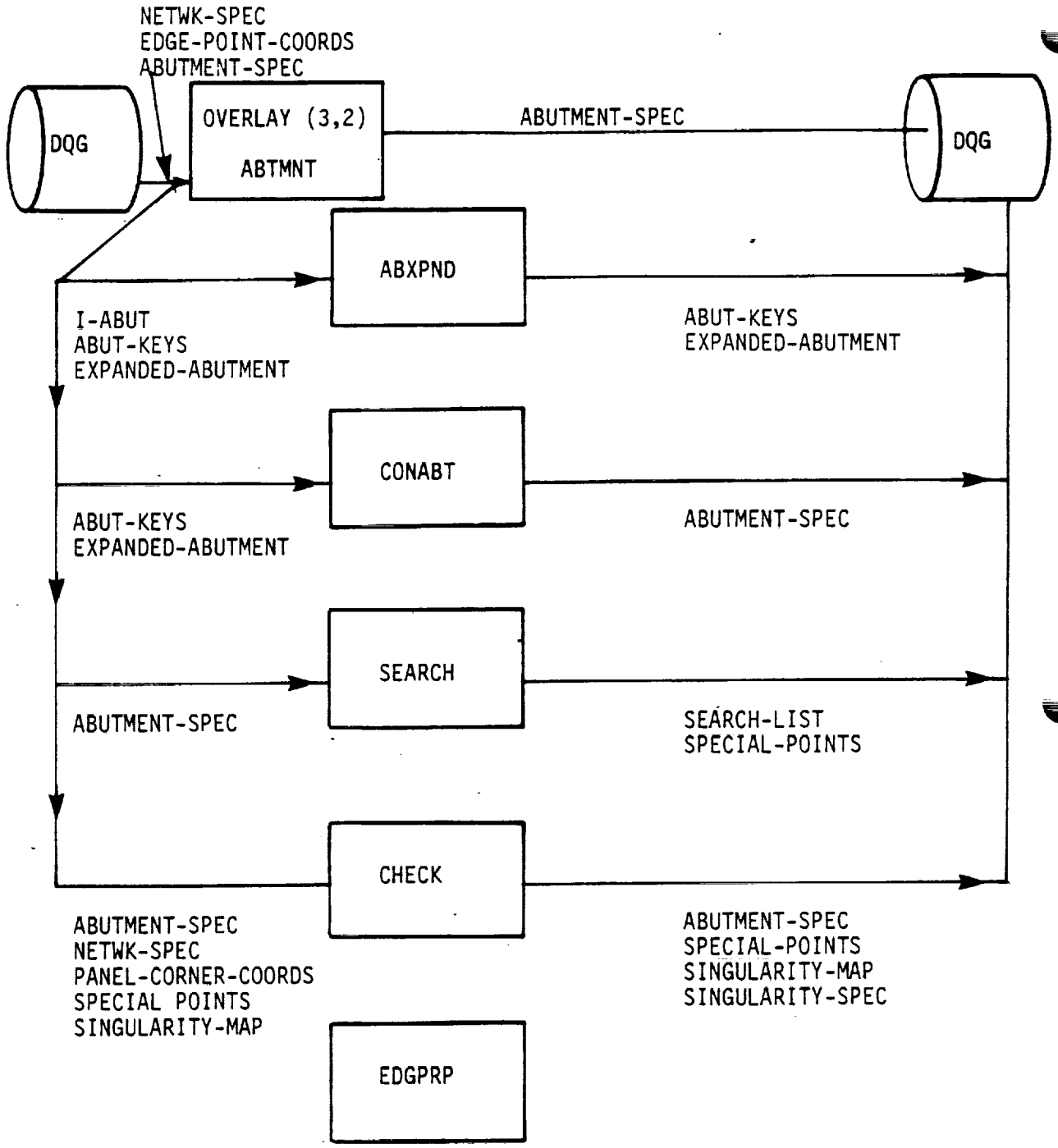


Figure 4.8 - Structure and Data Flow for Overlay (3,2)

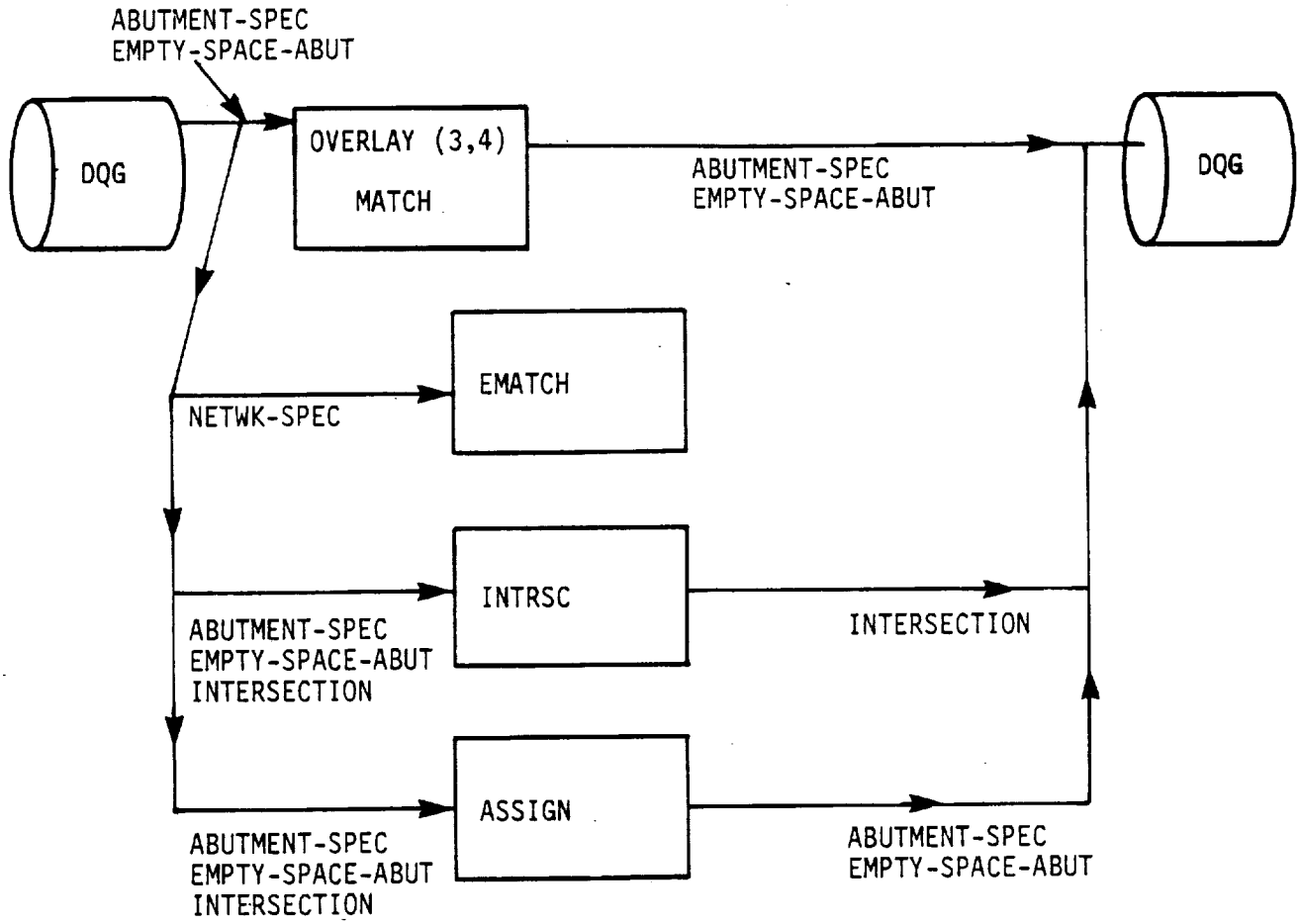


Figure 4.9 - Structure and Data Flow for Overlay (3,4)

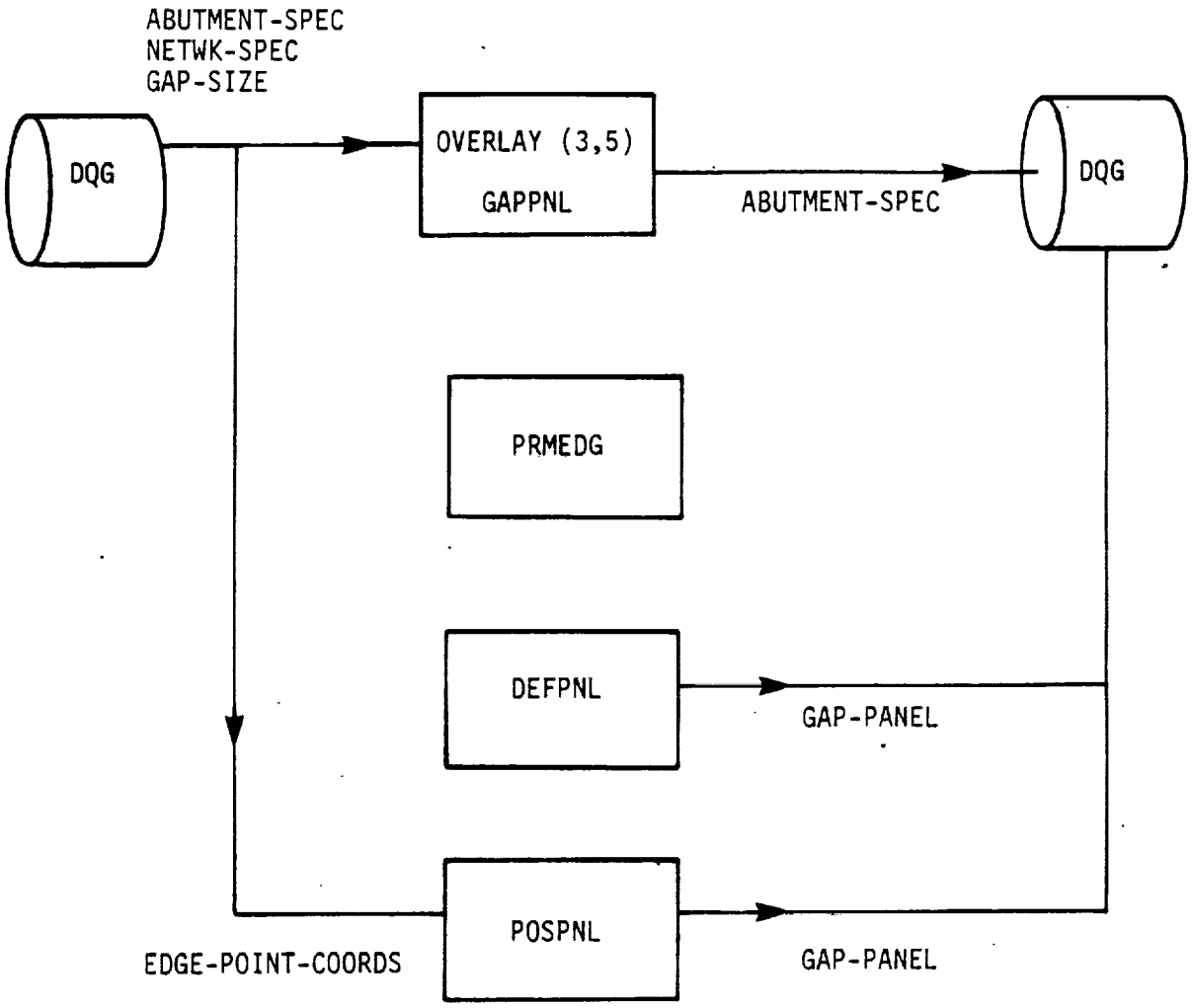


Figure 4.10 - Structure and Data Flow for Overlay (3,5)

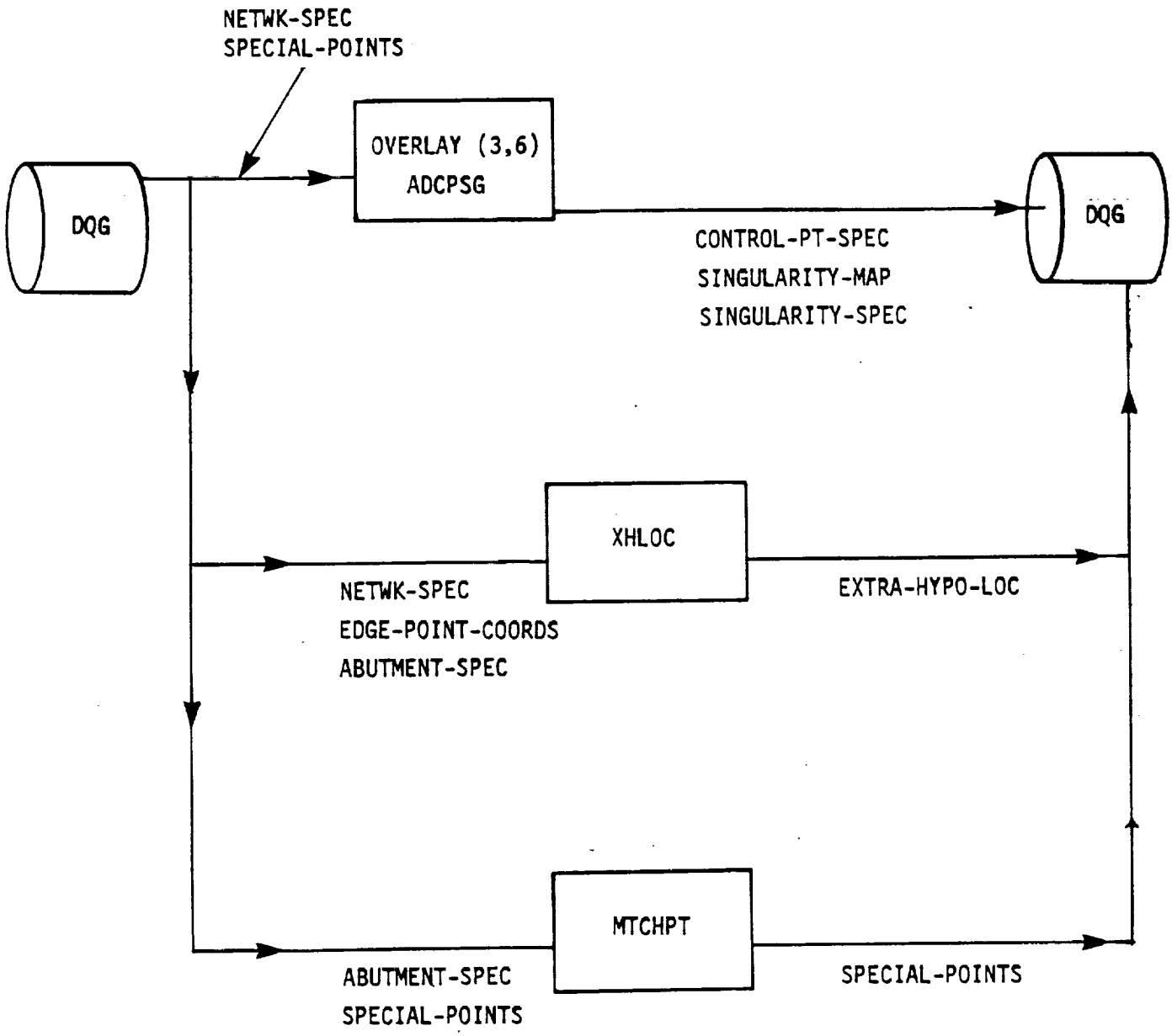


Figure 4.11 - Structure and Data Flow for Overlay (3,6)

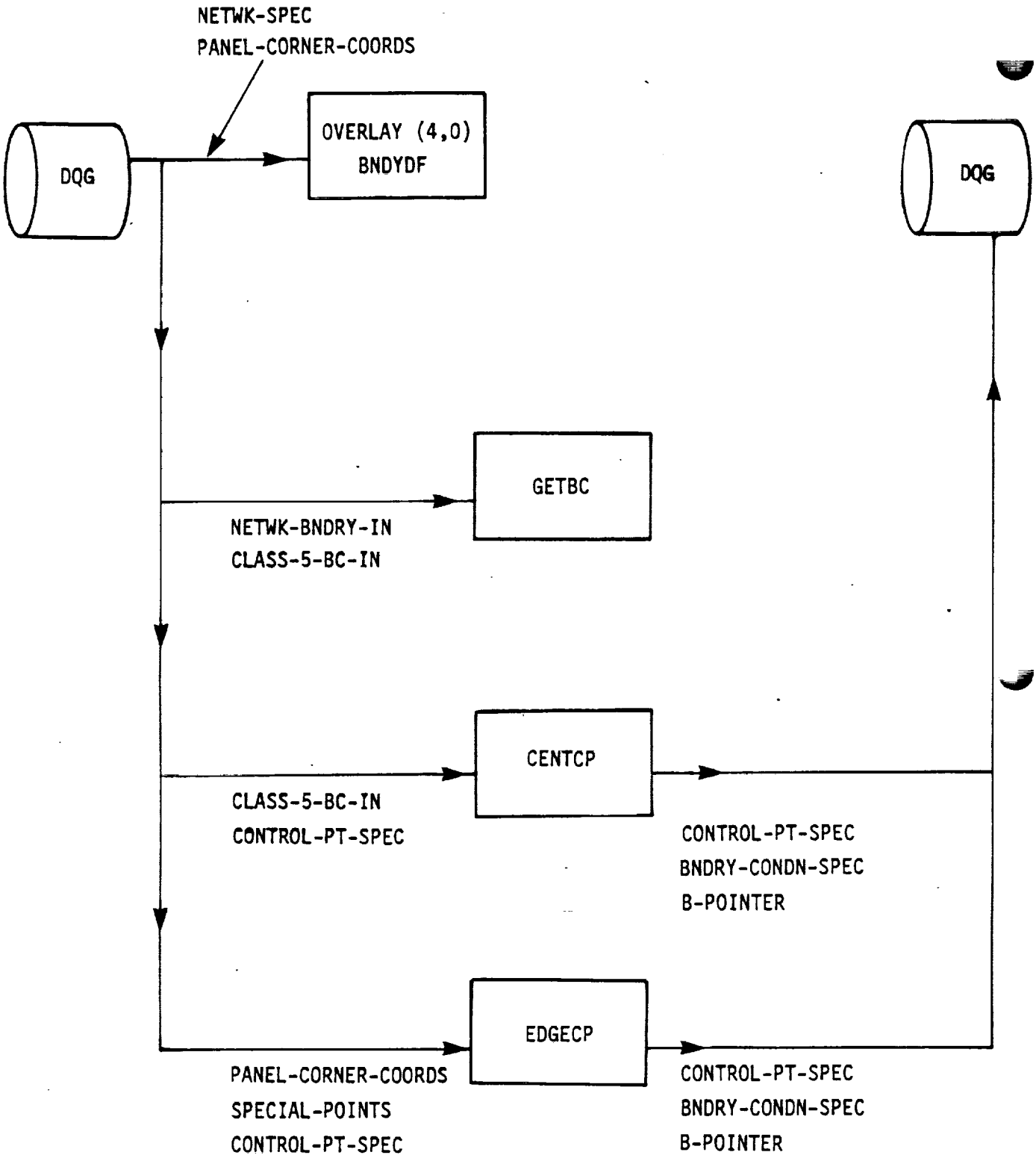


Figure 4.12 - Structure and Data Flow for Overlay (4,0)

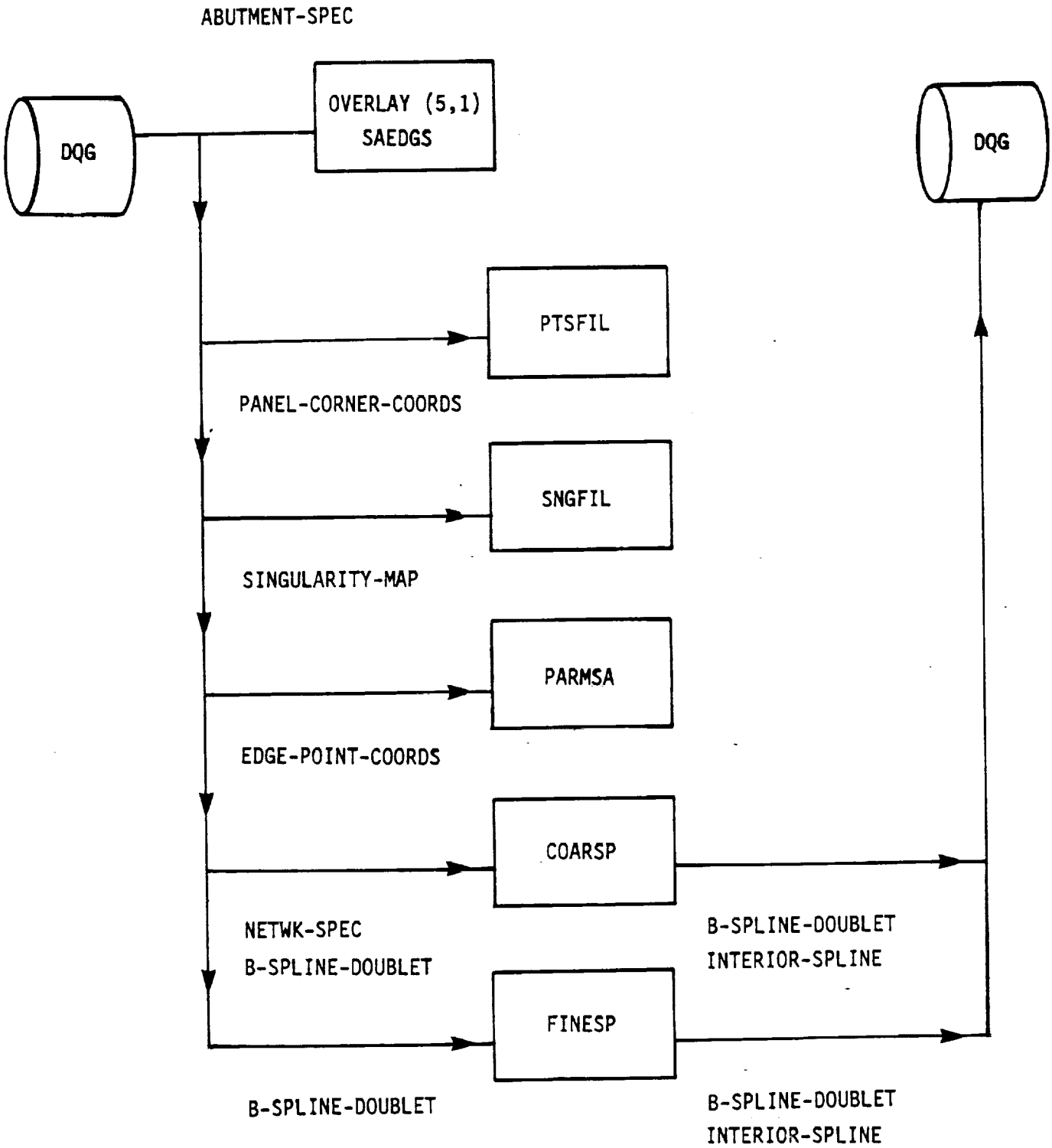


Figure 4.13 - Structure and Data Flow for Overlay (5,1)

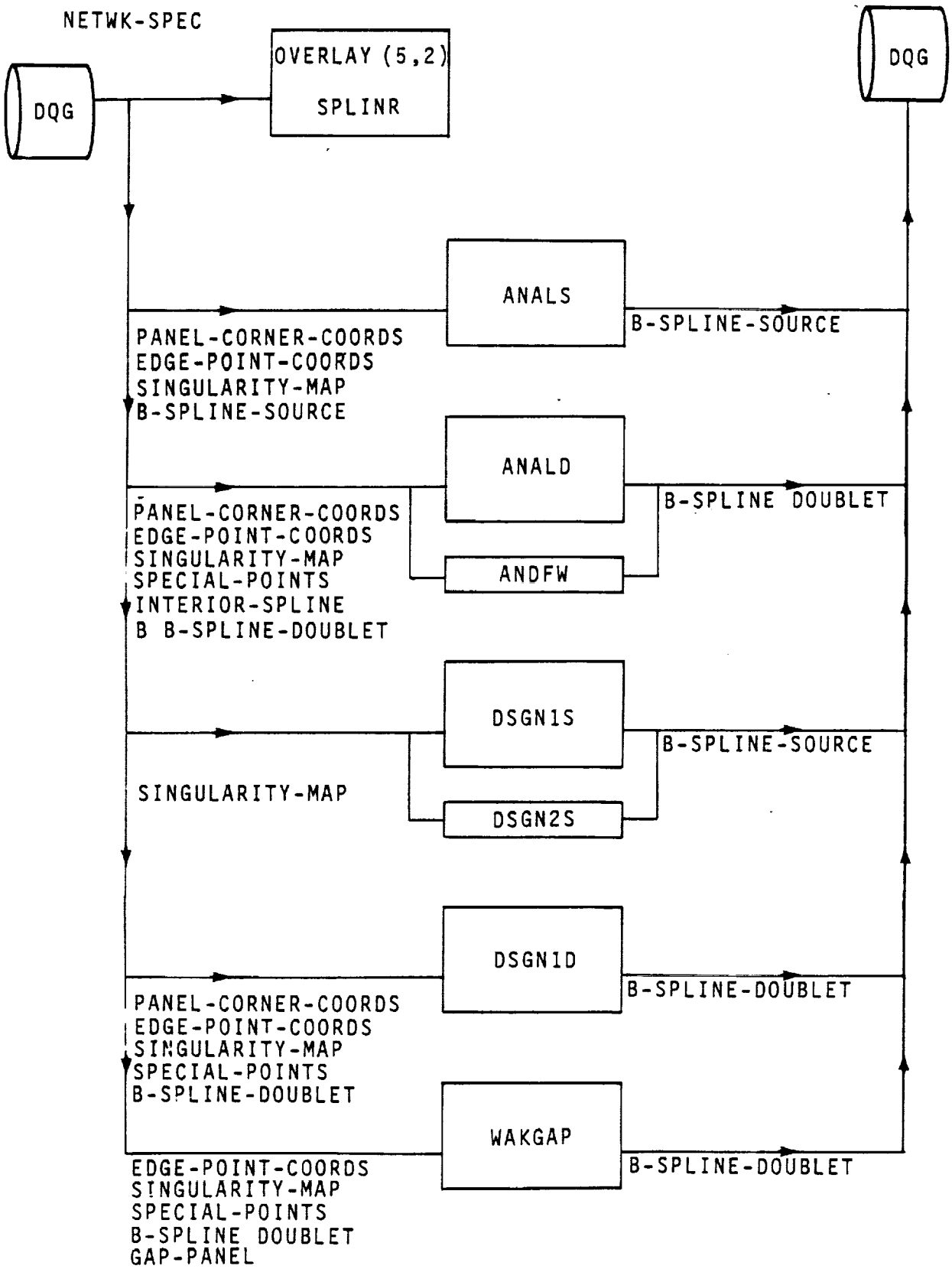


Figure 4.14 - Structure and Data Flow for Overlay (5,2)

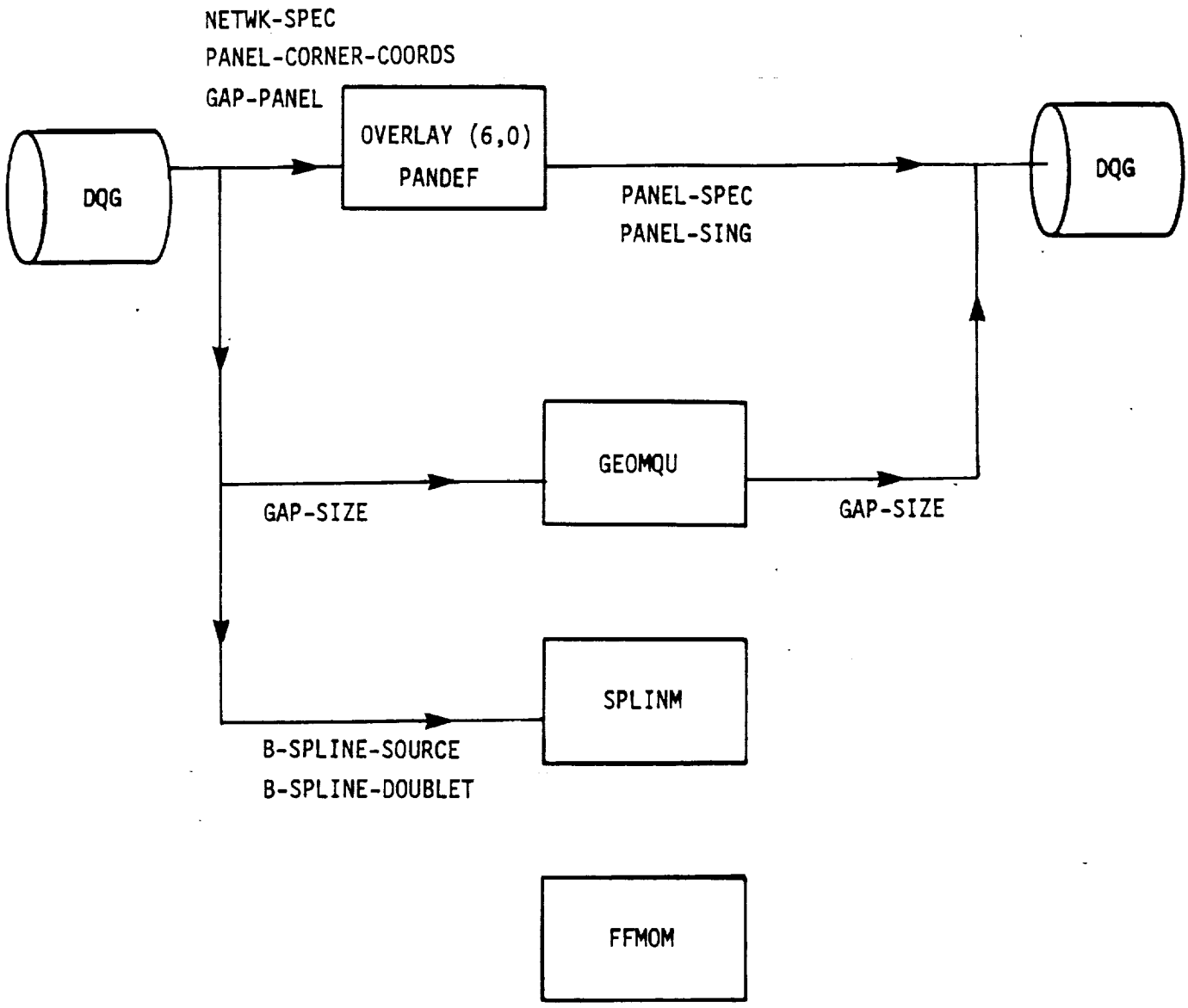


Figure 4.15 - Structure and Data Flow for Overlay (6,0)

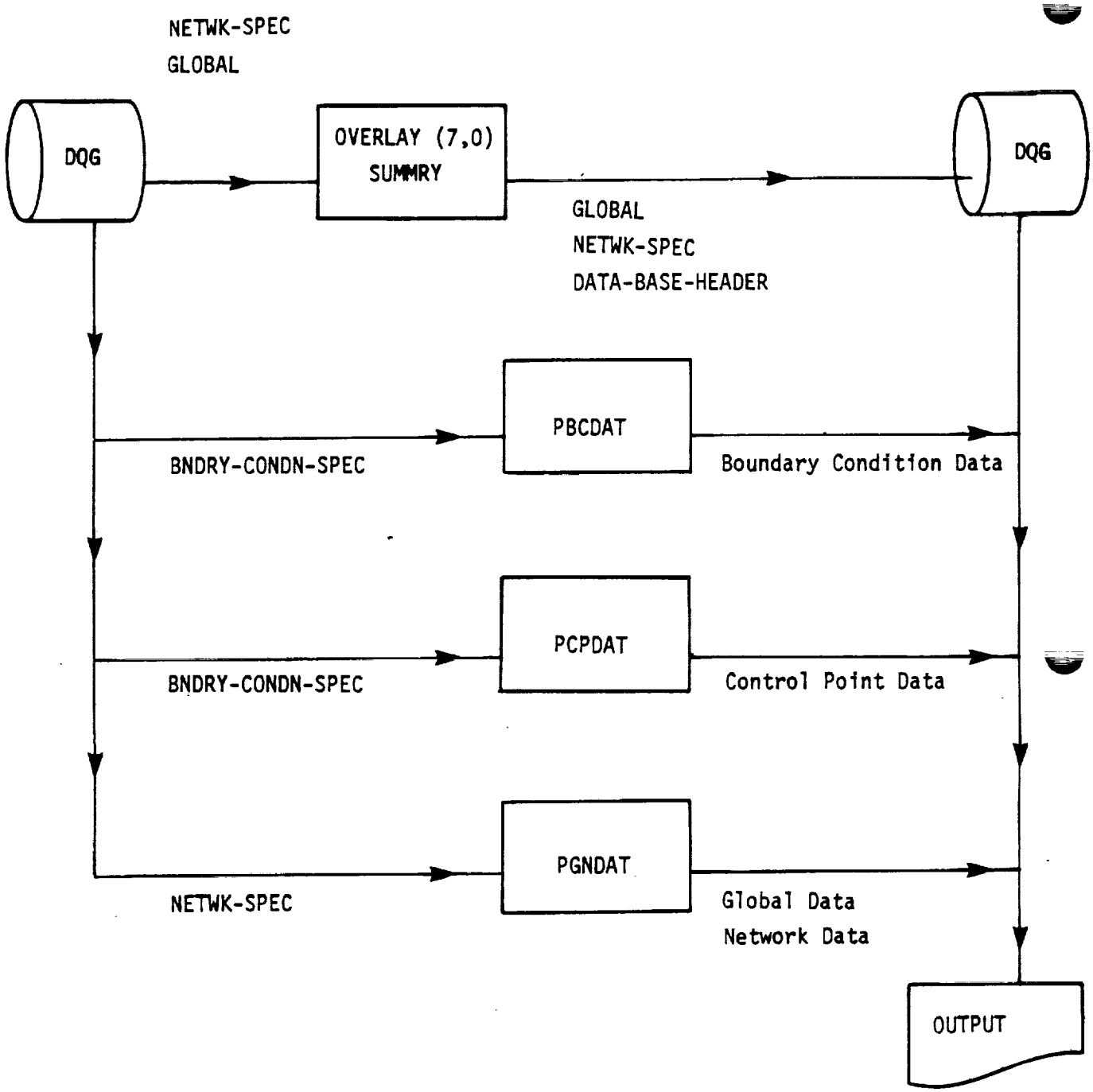


Figure 4.16 - Structure and Data Flow for Overlay (7,0)

APPENDIX 4-A TREE STRUCTURE

The tree structure diagram of the DQG module has been deleted from this document. It is, however, available on the installation tape.

APPENDIX 4-B

FUNCTIONAL DECOMPOSITION OF DQG

This appendix describes the functional decomposition of DQG, that is, an outline form of the modular structure of DQG organized by task. Where a particular task is realized as a subroutine or subroutines in DQG, the subroutine name is listed in parentheses after the textual description of the operations which are performed. It is important to note that the execution sequence of each of the levels of a modular decomposition is not necessarily in accordance with the alphabetical order of the outline form. The control structure of the program (which is described in the comments within the particular subroutine) determines the execution sequence of the submodules. The alphabetical listing of the submodules is merely a listing device to guide the reader to particular sections of the code.

4-B.1 Functional Decomposition of DQG

A. Open Database and Transfer Data from DIP database to DQG Database. (OPENER) [Overlay (1,0)]

- A. Retrieve Information from MEC Database
  - A. Open the MEC Datbase (DBOPEN)
  - B. Define the SDMS Map (DSMAP/DVMAP/ENDMAP)
  - C. Get the Run, User and Problem ID's (ESGET)
  - D. Check the file for the Databases (CHPADB)
  - E. Close the MEC Database (DBCLOS)

- B. Transfer Data from DIP to DQG Database (DIPDAT)
  - A. Define SDMS Maps (DSMAP/DVMAP/SVMAP/ENDMAP)
  - B. Get the Global Data (ESGET)
  - C. Get Network Data and Transform
  - D. Process User Abutment Information (BNDYIN)
  - E. Process Boundary Condition Information
    - A. Define Additional SDMS Maps (DSMAP/DVMAP/SVMAP/ENDMAP)
    - B. Get Network Data (ESGET)
    - C. Clear Boundary Condition Arrays
    - D. Define Class One BC Data
    - E. Transcribe BC Data (XSCRIB)
      - A. Initialize
      - B. Modify Counter with Smear Option (DSCT)
      - C. Define User Class and Number of BC (NBCLAS)
      - D. Get DIP Database Data (ESGET)
      - E. Clear BC Arrays (ZERO)
      - F. Compute Lattice Indices for BC Data (LATBC)
      - G. Copy BC Data to Output Array (COPYBC)
      - H. Write BC Data to DQG Database (ESPOR)
    - F. Transcribe Closure Data (CLOSTR)
      - A. Define Number of Parallel and Perpendicular Panels
      - B. Define Lattice Increments
      - C. Define Lattice Indices
      - D. Clear Closure Arrays
      - E. Define Panel Indices
      - F. Fill Array Indices
      - G. Get Data from DIP (ESGET)
      - H. Define Scale Coefficients
      - I. Get Index of Value Array
      - J. Add Contributions to Mass FLux and Source Terms
      - K. Write Data to Database (ESPUT)

- C. Close Database (PACLOS)

B. Compute Network Defining Quantities (NETDEF) [Overlay (2,0)]

A. Open Database and Define Maps
   A. Open DQG Database (PAOPEN)
   B. Define Maps (MAPB)

B. Get Network Data

C. Define Edge Point Coordinates (DFEDGE)
   A. Initialize
   B. Get Column of Points (ESGET)
   C. Store Corner Points for Edge Four and those Adjacent to Edge Four in Reverse Order
   D. Store First and Second Corner Points for Edge One and its Adjacent Row
   E. Store Last and Next to Last Corner Points for Edge Three and its Adjacent Row (Reverse Order)
   F. Store Corner Points for Edge Two and those Adjacent to Edge Two
   G. Write Edge Point Coordinates to Database (ESPUT)

D. Check Network Edges (EDGCHK)
   A. Calculate Network Edge Length (EDGCAL)
   B. Diagnose Collapsing Edge Error
   C. Collapse the Network Edge (COLAPS)
   D. Diagnose Collapsing Edge Error
   E. Write Changed Coordinates to Database (ESREP)
   F. Diagnose Adjacent Collapsed Edge Errors

E. Find Indicial Center of Network (INDCTR)

F. Get a Columns of Corner Points (ESGET)

G. Extract Panel Corner Points

H. Check for Triangular Panels and High Aspects Ratio (TRICHK)

I. Find Fine Grid Coordinates (FINGRD)

J. Place Fine Grid Points on Database (ESPUT)

K. Define Singularity Indices (SINGDF)
   A. Establish Network Wide Variables
      A. Extract from Common Blocks NETWK and SINGLR
      B. Set to Default Values

   B. Generate Singularity Specifications for All Network Panels (SINGPAN)
   C. Generate Singularity Specifications for Doublet Network Edges
      A. Doublet Analysis Edges (SNGDA)
      B. Doublet Design I Edges (SNGSD1)
      C. Doublet Design II Edges (SNGSD2
      D. Doublet Wake I Edges (SNGDW1)
      E. Doublet Wake II Edges (SNGDW2)
      F. Forward Weighted Doublet Analysis Edges (SNGDFW)

D. Generate Singularity Specifications for Source Network Edges
   A. Source Design I Edges (SNGSD1)
   B. Source Design II Edges (SNGSD2)

E. Nullify Singularities at Collapsed Edges (SNGNUL)

L. Index Control Points (CONTPT)

M. Replace NETWK-SPEC Data Set (ESREP)

N. Print Requested Network Data (PRTNET)

C. Compute Edge Defining Quantities (EDGDEF) [Overlay (3,0)]

A. Find Pairwise Abutments and Define User Abutments (PRABUT) [Overlay (3,1)]

A. Define User Abutments (USEABT)
   A. Initialize Abutment Counters
   B. Get User Abutment Data (ESGET)
   C. Define Reference Network
   D. Define Whole Edge to be in Abutment
   E. Find Closest Corner Point to Start/End Point of Reference Edge
   F. Write Abutment Data to Database (ESPUT)
   G. Define Total Number of Abutments
   H. Define Search List for Automatic and Empty Space Abutment Search (SEARCH)

B. Get Network Data (ESGET)

C. Setup Blank Common Storage (INITIR,STARTR)

D. Get Edge Point Coordinates (ESGET)

E. Search for Network Abutment (NETABT)
   A. Get Search List for Current Reference Edge (ESGET)
   B. Define Candidate Edges (EDGLST)
   C. Define Search Pointers
   D. Compute Minimum Distance from Point to Line Segment
   E. Determine Nearby Segment Found
   F. Start a New Pairwise Abutment
   G. Extend an old Abutment
   H. Terminate the Abutment
   I. Determine Plane of Symmetry Abutment

F. Delete Blank Common Storage (DELETR)

B. Construct Abutments and Check Rules (ABUTMNT) [Overlay (3,2)]

A. Generate Expanded List of Abutments (ABXPND)
   A. Make List of Pairwise Abutments in Which Network Takes Part
   B. Sequence Corner Point Indices
   C. Define Expanded Abutment Arrays
   D. Add Plane of Symmetry to Expanded Abutment Arrays

B. Contract Expanded Abutments to Form Abutment Description (CONABT)

   A. Get Expanded Abutment Data (ESGET)
   B. Find a Reference Edge
   C. Compute Quarter and Three Quarter Point Coordinates (C13QTR)
   D. Computer Distance Between Quarter Points (DISTQT)
   E. Add Network Edge to Abutment
   F. Compute Distance from Quarter Points to Plane of Symmetry
   G. Add Plane of Symmetry to Abutment
   H. Copy Network Edges to Output Array
   I. Copy Plane to Symmetry to Output Array
   J. Clear Associated Information
   K. Write Abutment Data (ESPUT)

C. Get Abutment Data (ESGET)

D. Compute Associated Edge Properties (EDGPRP)

E. Replace Abutment Data (ESREP)

F. Define Search List for Empty Space Abutments (SEARCH)

G. Check Abutment Rules (ESGET)
   A. Get Abutment Data (ESGET)
   B. Check Smooth Abutment Rules
   C. Define Null Singularity Rules
   D. Count Number of Matching Edges
   E. Check Plane of Symmetry Rules (CHKPOS)

C. Compute Gap Sizes (GAPSIZ) [Overlay (3,3)]

   A. Initialize

   B. Get Abutment Data (ESGET)

   C. Define Start and End Pointers

   D. Define Two Panel Corner Points

   E. Find Closest Point on Second Edge to Panel Point on First Edge

   F. Find Minimum Distance to Closest Line Segment

   G. Define Gap Size as Maximum of Minimum Distances

   H. Write Gap Size to Database (ESPUT)

D. Define Empty Space Abutments (MTABUT)

   A. Get Abutment Data (ESGET)

   B. Compute Edge and Point Properties (EDGPRP)

   C. Replace Abutment Data (ESREP)

   D. Define Empty Space Abutment

   E. Write Empty Space Abutment

E. Assign Matching Data to Edges (MATCH) [Overlay (3,4)]

A. Get Abutment Data (ESGET)

B. Assign Network Edge to Abutment (EMATCH)
   A. Get Network Data (ESGET)
   B. Count Matching Edge
   C. Find Leading Edge of Most Downstream Edge Which is also a Supersonic Edge
   D. Choose Finest Network
   E. Define Matching Edge
   F. Diagnose Fatal Error
   G. Diagnose Matching Doublet Pointer for Empty Space Abutment

C. Replace Abutment Data (ESREP)

D. Define Abutment Intersections (INTRSC)
   A. Get Abutment Data (ESGET)
   B. Extend the Abutment List
   C. Extend the Corner Point Map
   D. Extend the Connection List
   E. Sequence Connections by Downstream Parameter
   F. Initialize Arrays for Intersection Description
   G. Assign Entries to Intersection List (NTRLST)
      A. Determine Corner Point Indices from Connection Data
      B. Define New Intersection List
      C. Define Intersection Index
      D. Add to Old Intersection List
      E. Combine Intersections
      F. Change Sign of Abutment Index for Closed Loop
      H. Define Number of Intersections

E. Assign Corner Points for Doublet Matching (ASSIGN)
   A. Get Intersection Data (ESGET)
   B. Count Distinct Abutments
   C. Count Distinct Corner Points and Sequence Them
   D. Diagnose Error in Intersection Data
   E. Initialize Assignment Counter
   F. Prepare Data Needed for Assignment
   G. Choose Corner Point for Assignment (ABTINT)
   H. Assign Corner Points to Abutments (ABASGN)
      A. Define Abutment Index
      B. Get Abutment Data (ESGET)
      C. Define Matching Data
      D. Increment Number Assigned

F. Add Gap Filling Panels (GAPPNL) [Overlay (3,5)]

A. Get Abutment Data (ESGET)

B. Define Panel Limits

C. Define Shift Index for Panel

D. Define Panel Index

E. Set Indicator for Gap Size Exceeded

F. Initialize Gap Panel Construction

G. Parameterize the Edge (PRMEDG)

H. Merge and Sequence Parameterizations

I. Define Gap Panel for Network Edges (DEFPNL)
   A. Initialize
   B. Compute Four Coordinates from a Pair of Parameterizations
   C. Compute Gap Panel Edge Lengths and Count Number of Short Edges
   D. Define Gap Panel Data
   E. Write Gap Panel Element Set (ESPUT)
   F. Error Exit

J. Define Gap Filling Panel for Plane of Symmetry (POSPNL)
   A. Find Plane of Symmetry Reference
   B. Get Coordinate Data (ESGET)
   C. Define Start and End Points and Lattice Indices of First Point
   D. Define Gap Panel Corner Points
   E. Define Gap Panel Data
   F. Count Number of Short Edges
   G. Write Gap Panel Element Set (ESPUT)
   H. Diagnose Error

G. Add Extra Singularities and Control Points (ADCPSG) [Overlay (3,6)]

A. Get Network Data (ESGET)

B. Increment Number of Control Points

C. Assign Control Point Index

D. Add Extra Singularity Parameter

E. Replace Special Points Dataset (ESREP)

F. Assign Extra Hypothetical Locations (XHLOC)
   A. Get Abutment Data (ESGET)
   B. Compute Extra Hypothetical Location for Corner Point
   C. Parameterize Network Edge Segment (PRMEDG)
   D. Define Extra Hypothetical Locations for Edge Midpoints

G. Define Flags for Matching Boundary Conditions
   A. Get Abutment Data (ESGET)
   B. Count Number of Planes of Symmetry in Abutment
   C. Count Distinct Matching Pointers
   D. Find Edge Segment for Matching
   E. Assign Matching Condition
   F. Replace Special Points Dataset (ESREP)
   G. Define Special Points Dataset for Collapsed Edges
      A. Define First and Last Points for Each Edge
      B. Get Special Points Dataset (ESGET)
      C. Check if Matching Pointer Set for Null Control Point
      D. Define Value for Opposite Corner Point on Collapsed Edge
      E. Replace Special Points Dataset (ESREP)
      F. Define Special Points Dataset for Collapsed Edge
      G. Define Matching Value for Point
      H. Write Special Point Database for Collapsed Edge (ESPUT)

D. Compute Control Point and Boundary Condition Data (BNDYDF) [Overlay (4,0)]

A. Open Database and Define Maps (PAOPEN,DSMAP)

B. Get Network Data (ESGET)

C. Get Network-Wide Boundary Conditions (GETBC)
- A. Get BC Data (ESGET)
- B. Compute Average and Difference Coefficients
- C. Check for Nearly Vanishing Coefficients

D. Determine Number of Boundary Conditions Required

E. Copy Second Columns of Corner Points to First Column

F. Compute Center Control Point Data (CENTCP)
- A. Compute BC Data
- B. Compute Geomtric Data
- C. Assign Boundary Conditions (ASGNBC)
  - A. Prepare User Boundary Conditions (ASGNU)
    - A. Initialize Characterization
    - B. Redefine BC Coefficients
    - C. Define C and D Vectors
    - D. Assign Characterizations and Sequence User BC by Hierarchy
  - B. Assign Matching BC (ASGNM)
    - A. Test Point for Closure Point
    - B. Define Vanishing C and D Vectors (ZERO)
    - C. Test Point for Closure Point
    - D. Test Point for Source Matching Point
  - C. Choose Required Boundary Conditions (CHOOSE)
    - A. Clear Output Array (ZERO)
    - B. Initialize BC Pointers
    - C. Copy DQG BC Data to Output Array (DQGOUT)
    - D. Copy User BC Data to Output Array (USROUT)
    - E. Copy Degenerate Data to Output Array (DEGOUT)
    - F. Add Default BC if Insufficient BC Assigned
  - D. Define MAG and PDP Parameters (MPPARM)
  - E. Prepare Degenerate BC Data (DEGPRP)
  - F. Clear Number of DQG Boundary Conditions
  - G. Define Known Singularities (KNOWSP)
- D. Replace Control Point Dataset (ESREP)
- E. Write Boundary Condition Dataset (ESPUT)

G. Compute Boundary Condition Data for Edge (EDGECP)
  A. Get Coordinates (ESGET)
  B. Get Network Wide BC (GETBC)
  C. Compute Geometric Information for Corner Points (CCPGEO)
    A. Define Panel Points
    B. Define Subpanel on Which Point Lies
    C. Compute Normal and Conormal
    D. Compute Recession Vector
    E. Define Control Point Location
    F. Compute Tangent Vector (TANGOP)
    G. Define Remaining Geometric Data
    H. Copy BC Data into User BC Array
      A. Copy Data
      B. Check for Superinclined Subpanel and Modify Boundary Conditions (MODBC)
  D. Assign Boundary Conditions to Point (ASGNBC)
  E. Replace Control Point Dataset (ESREP)
  F. Write Boundary Condition Datasets (ESPUT)
  G. Compute Geometric Data for Edge Midpoints (ECPGEO)
    A. Define Panel Points
    B. Compute Normal and Conormal Vector
    C. Compute Recession Vector
    D. Compute Tangent Vector (TANGOP)
    E. Define Miscellaneous Geometric Data
    F. Check for Superinclined Subpanel and Modify Boundary Conditions (MODBC)
  H. Compute the Kutta Tangent Vector

H. Close Database (PACLOS)

E. Choose Source and Doublet Splines (TOPSPL) [Overlay (5,0)]
  A. Compute Smooth Abutment Splines (SAEDGS) [Overlay (5,1)]
    A. Open Database and Define Maps (PAOPEN,DSMAP)
    B. Get Abutment Data (ESGET)
    C. Choose Closest Network
    D. Parametrize Smooth Abutment (PARAMSA)
    E. Get Coordinates for Smooth Edge (PTSFIL)
    F. Get Singularity Parameter Indices (SNGFIL)
    G. Define Spline Vectors for Coarse Edge
      A. Define Unit Spline Vectors at End Points (UNISPL)
      B. Define Spline Data for Corner Points (SALSQG)
        A. Initialize Counter
        B. Find Closest Corner Point on Fine Network (CCPFN)
        C. Define Least Squares Data (DEFLSQ)
        D. Define and Write Internal Spline Vector (INTERN)
        E. Define Local Two Dimensional Coordinate System (LOC2D)
        F. Computer Transformtion to Local Two Dimensional Coordinate System (SPLTRN)
        G. Compute Weights for Fit (WTLSQ)
      C. Define Lattice Indices for Point (FLIND)
      D. Perform Constrained Least Squares Fit (CQLSF)
      E. Diagnose Error
      F. Print Warning for Poor Fit
      G. Accumulate Spline Vector (VECUNV)
      H. Write Spline Vector to Database (ESPUT)
      I. Define Spline Vector for Edge Midpoint)SALSQE)
        A. Initialize Counter
        B. Find Closest Corner Point on Fine Network (CCPFN)
        C. Find Most Distant Center Point Adjacent to Closeset Corner Point (MDCP)
        D. Define Least Squares Data (DEFLSQ)
        E. Define and Write Internal Spline Vector (INTERN)
        F. Define Local Two Dimensional Coordinate System (LOC2D)
        G. Compute Transformation to Local Two Dimensional Coordinate System (SPLTRN)
        H. Compute Weights (WTLSQ)

    H. Define Spline Vector for Fine Edge (FINESP)
      A. Choose Corner Points on Coarse Edge (CPCSFL)
      B. Compute One Dimensional Quadratic Fit (Q1DFIT)
      C. Accumulate Spline Vector (VECUNV)
      D. Compute Internal Spline Data
      E. Write Spline Vector

    I. Close Database (PACLOS)

B. Compute Spline Vectors for Network (SPLINR) [Overlay (5,2)]
  A. Open Database and Define Maps (PAOPEN,DSMAP)
  B. Get Network Data (ESGET)
  C. Compute Source Analysis Spline Vectors (ANALS)
    A. Initialize
    B. Define and Write Unit Spline Vectors (UNISPL)
    C. Get Arrays of Coordinates
    D. Compute Lattice Indices of Point (LATIND)
    E. Compute Least Squares Spline for Point (SSP13)
      A. Compute Least Squares Data for Point (DATS13)
        A. Initialize
        B. Check for One Dimensional Fit
        C. Increment Counter
        D. Define Least Squares Data for Point (LSQDAT)
        E. Define Xi and Eta Vectors (XIETAV)
        F. Compute Coordinate Transformation (SPLTRN)
        G. Compute Weights (WTLSQ)
        H. Perform One Dimensional Fit
      B. Perform Bilinear Fit (CQLSF)
      C. Diagnose Spline Error
      D. Print Warning Message
      E. Accumulate Spline Vector (VECUNV)
      F. Write Spline Vector (ESPUT)

  D. Compute Spline Vectors for Source Design Network
    A. Compute Fine Grid Lattice Indices
    B. Define Unit Spline Vector (UNISPL)
    C. Define Spline Vector
    D. Get Singularity Index for Point (SNGDEX)
    E. Write Spline Vector (ESPUT)

  E. Compute Doublet Analysis Spline Vectors (ANALD)
    A. Initialize
    B. Calculate Edge Spline Vectors
      A. Define Coarse Lattice Indices
      B. Define Lattice Indices for Point (LATIND)
      C. Define Unit Spline Vectors (UNISPL)
      D. Define Lattice Indices for Last Corner Point on Edge
      E. Write Spline Vector for Point (ESPUT)
      F. Perform Analysis Edge Spline (NTEDGA)
        A. Get Edge Coordinates (ESGET)
        B. Find Edge Segments for Quadratic Fit (EDGSGQ)
        C. Define Spline Vectors for Additional Corner Points
        D. Compute Spline Vectors for Corner Points (CPANAL)
        E. Compute Spline Vectors for Edge Midpoints
        F. Define Spline Vectors for Collapsed Edge
    C. Define Unit Spline Vectors at Center Points (UNISPL)
    D. Get Array of Corner Points (ESGET)
    E. Compute Lattice Indices of Point (LATIND)

F. Compute Spline Vector for Point (SPLA)
  A. Compute Least Squares Data for Surrounding Points (DATANL)
    A. Initialize
    B. Define Lattice Indices
    C. Increment Counter
    D. Define Least Squares Data for Point (LSQDAT)
    E. Define Infinite Weight for Point
    F. Define Xi and Eta Vectors (XIETAV)
    G. Compute Coordinate Transformation (SPLTRN)
    H. Define Weights (WTSLQ)
  B. Perform Constrained Least Squares Fit (CQLSF)
  C. Diagnose Error
  D. Print Warning for Poor Fit
  E. Accumulate Spline Vector (VECUNV)
  F. Write Spline Vector (ESPUT)

G. Compute Doublet Design Spline Vectors (DSGN1D)
  A. Initialize Limit Arrays
  B. Compute Network Edge Spline Vectors
    A. Define Lattice Indices for Point (LATIND)
    B. Define Unit Spline Vector for Point (UNISPL)
    C. Define Lattice Indices for Last Point on Edge
    D. Write Spline Vector for Point
    E. Perform Analysis Edge Spline (NTEDGA)
    F. Perform Design Edge Spline (NTEDGD)
      A. Initialize
      B. Get Edge Coordinates (ESGET)
      C. Find Edge Segments for Quadratic Fit (EDGSGQ)
      D. Define Unit Spline Vector for Extra Points
        E. Compute Start/End Points for Segment (LATEDG)
      F. Parametrize the Segment (PRMEDG)
      G. Compute Intermediate Spline Vector for Edge Midpoints (GAMVEC)
      H. Define Lattice Indices (EDGLAT)
      I. Compute Corner Point Spline Vector (CPDSGN)
      J. Compute Edge Midpoints Spline Vectors (EMDSGN)
  C. Define Fine Grid Lattice Indices
  D. Define and Write Unit Spline Vectors (UNISPL)
  E. Get Arrays of Corner Points
  F. Compute Spline Vectors for Specified Point
    A. Compute Lattice Indices (LATIND)
    B. Define Shifts and Limits
    C. Compute Spline Vector (SPLA)
H. Compute Wake or Gap Panel Spline Vectors (WAKGAP)
  A. Define Corner Point Lattice Indices
  B. Calculate Wake I Edge Spline
    A. Find Matching Edge
    B. Define Unit Spline Vector for Corner Points

C. Perform Edge Spline for Matching Edge (NTEDGA)

C. Calculate Wake II Edge Spline

D. Diagnose Error

E. Define Spline Vectors for Interior of Wake Network

F. Define Spline Vectors for Gap Filling Panels (GAPSPL)

A. Get Spline Vector for Point (ESGET)

B. Define Network Data

C. Compute Lattice Indices (LATIND)

D. Compute Geomtric Weight Matrix

E. Define Weight Vector

F. Accumulate Spline Vector (VECUNV)

G. Write Spline Vector to Datbase (ESPUT)

I. Compute Forward Weighted Doublet Analysis Splines (ANDFW)

A. Initilize

B. Calculate Edge Spline Vectors

A. Define Coarse Lattice Indices

B. Define Lattice Indices for Point (LATIND)

C. Define Unit Spline Vectors (UNISPL)

D. Define Lattice Indices for Last Corner Point on Edge

E. Write Spline Vector for Point (ESPUT)

F. Perform Analysis Edge Spline (NTEDGA)

A. Get Edge Coordinates (ESGET)

B. Find Edge Segments for Quadratic Fit (EDGSGQ)

C. Define Spline Vectors for Additional Corner Points

D. Compute Spline Vectors for Corner Points (CPANAL)

E. Compute Spline Vectors for Edge Midpoints

F. Define Spline Vectors for Collapsed Edge

C. Define Unit Spline Vectors at Center Points (UNISPL)

D. Get Array of Corner Points (ESGET)

E. Compute Lattice Indices of Point (LATIND)

F. Compute Panel Defining Quantities (PANDEF) [Overlay (6,0)]
  A. Open Database, Define Maps and Get Data for Network
  B. Compute Geometric Quantities (GEOMQU)
    A. Compute Panel Defining Points
    B. Compute Panel Geometric Data (PANGEO)
      A. (Not Used)
      B. Compute Normal and Conormal Vector
      C. Compute Panel Diameter and Radius
      D. Define Panel Updatability
      E. Compute Skewness Parameters
      F. Compute Projected Area of Panel and Subpanels
      G. Compute Average Plane Corner Point Coordinates
    C. Compute Subpanel Geometric Data (SUBGEO)
      A. Compute Origin of Subpanel Coordinate System
      B. Compute Normal and Conormal Vectors
      C. Compute Transformation to Subpanel Coordinate System
      D. Compute Subpanel Points in Subpanel Coordinate System
      E. Compute In-Plane Side Normals
      F. Define Zero Value for Data (Error Exit)
    D. Compute Gap Size to Panel Size Ratio (PANSIZ)
      A. Initialize
      B. Compute Unit Normal to Panel Edge
      C. Define Vector from Center to Edge Midpoint
      D. Compute Inner Product of Vectors (VIP)
      E. Get Gap Size for Panel (ESGET)
      F. Compute Gap Size to Panel Size Ratio
      G. Replace GAP SIZE Dataset (ESREP)
    E. Compute Edge Normal in Subpanel Eight Coordinate System
  C. Assemble Spline Matrices (SPLINM)
    A. Construct Outer Spline Matrix (SDSPLM)
      A. (Not Used)
      B. Initialize Counter
      C. Obtain Spline Vector (ESGET or DCSASP)
      D. Accumulate Spline Vector to Form Matrix (VECUNM)
      E. Clear Spline Matrix (ZERO)
    B. Construct Least Squares Defining Point Vectors
    C. Construct Subpanel Spline Matrix (SUBSPL)
      A. Construct the Geometric Matrix
      B. Invert the Geometric Matrix (JORDAN)
      C. Assemble the Extension Matrix
      D. Multiply the Inverse Geomertric and Extension Matrices (CAB)
      E. Define Zero Subpanel Spline Matrices (ZERO)
    D. Construct Kappa Vectors (KAPVEC)
      A. Clear Kappa Vectors
      B. Define Skewness Factors
      C. Compute Kappa Vectors for Subpanels
      D. Scale All Kappa Vectors
    E. Construct Panel Subspline Matrix
      A. Transform Point to Subpanel Eight Coordinate System
      B. Define Weights According to Singularity Type
      C. Compute Constrained Quadratic Least Squares Fit (CQLSF)
      D. Define Zero Matrix (ZERO)
    F. Write Fatal Error
    G. Define Singularity Indices for Panel

D. Compute Far Field Moments (FFMOM)
  A. Compute Basic Far Field Moments
    A. Initialize to Zero (ZERO)
    B. Define Points
    C. Compute Normal Distance from Origin to Line Segment (VIP)
    D. Compute Moments Along Line Segment
    E. Compute G(M,N) Along Line Segment
    F. Accumulate Basic Far Field Moment Contributions
  B. Compute Source Far Field Moments (SRCFFM)
    A. Initialize to Zero (ZERO)
    B. Define D Tensor (DTENSR)
    C. Compute Monopole Term
    D. Compute Dipole Term
    E. Compute Quadrupole Term
    F. Accumulate Contributions
    G. Symmetrize Far Field Moments (SYMFFM)
  C. Compute Doublet Far Field Moments (DBLFFM)
    A. Initialize to Zero (ZERO)
    B. Define D Tensor (DTENSR)
    C. Compute Monopole Term
    D. Compute Dipole Term
    E. Compute Quadrupole Term
    F. Accumulate Contributions
    G. Symmetrize Far Field Moments (SYMFFM)

F. Summarize Execution (SUMMRY)
  A. Print Global and Network Summary (PGNDAT)
    A. Print Header at Top of Page
    B. Write Global Data to Output File
    C. Get Network Data (ESGET)
    D. Write Network Data to Output File

  B. Transform Data Representation
    A. Get Network Data (ESGET)
    B. Redefine Singularity Type
    C. Redefine Edge Type
    D. Replace Network Data (ESREP)

  C. Print Control Point Data (PCPDAT)
    A. Initialize
    B. Get Data (ESGET)
    C. Increment Counter
    D. Print Header
    E. Compute and Write Data to Output File

  D. Print Boundary Condition Data
    A. Get Data (ESGET)
    B. Increment Counter
    C. Print Header
    D. Compute and Write Data to Output File

  E. Close Database (PACLOS)

G. Summarize DQG Operations (SUMMRY) [Overlay (7,0)]
   A. Open Database (PAOPEN)
   B. Define Maps (DSMAP,SVMAP,DUMAP)
   C. End Maps (ENDMAP)
   D. Replace Global Data (ESREP)
   E. Print Global and Network Data (PGNDAT)
   F. Print Control Point Data (PCPDAT)
   G. Print Boundary Condition Data (PBCDAT)
   H. Close Database (PACLOS)

APPENDIX 4-C

DATA BASE COMMUNICATIONS CHART

Tables 4-C.1 through 4-C.3 describe the data flow within DQG. The "First Form" chart (Table 4-C.1) lists the dataset name in alphabetical order by overlay with its corresponding map names used within the overlay and with the destination of the data, usually a common block. Table 4-C.2 is the "Second Form" of the chart. It contains the same information but has it listed under Map Name in alphabetical order. Table 4-C.3 lists the common blocks in alphabetical order by overlay and shows to which dataset the information within the block connects. In the column labelled COMMON BLOCK the word "Dynamic" sometimes appears. In this case the data is not transferred to a common block but is transferred to whatever variables are mentined in the I/O transfer call to ESGET, ESPUT or ESPOR. See Section 13 of this document.

Table 4-C.1 Data Flow for DQG First Form

Overlay (1,0)

DATABASE	DATASET-NAME	MAP NAME	COMMON BLOCK	SUBROUTINE
MEC	DATA-BASE-HEADER	IDS	/RUINDS/	OPENER
MEC	MACRO-OPTIONS	RUNOPT	local	OPENER
DIP	CLOS-COND	DIPCLOSDAT	/GENBCD/	BNDYIN
DIP	COEF-GEN-BC	CGBCMP	/GENBCD/	BNDYIN
DIP	GLOBAL	GLOBAL-IN	/GLOBAL/	DIPDAT
DIP	GLOBAL-PRINTS	PRINT-OPT	Dynamic	DIPDAT
DIP	NETWK-BDC	NETBDC	/NETBDC/	BNDYIN
DIP	NETWK-SPEC	NETMAP	/NETWK/	DIPDAT
DIP	PANEL-COORDS	PAN-COR-PT	/COORDS/	DIPDAT
DIP	TANG-VEC	TVECTCOEFF	/GENBCD/	BNDYIN
DIP	USER-ABUT	USABIN	/ABUT/	DIPDAT
DQG	CLASS-5-BC-DATA	CLASS	/NBCDIN/	BNYDIN
DQG	CLOSURE-DATA-IN	CLOSDIN	/CLOSUR/	BNDYIN
DQG	GLOBAL	GLOB-DYN	/GLOBAL/,Dynamic	DIPDAT
DQG	NETWK-BNDRY-CONDN-IN	BCDATIN	/NBCDIN/	BNDYIN
DQG	NETWK-SPEC	NETMAP	/NETWK/	DIPDAT
DQG	PANEL-CORNER-COORDS	COORDS-GEN	Dynamic	DIPDAT
DQG	USER-ABUT	USABUAT	/ABUT/	DIPDAT

Overlay (2,0)

DATABASE	DATASET-NAME	MAP NAME	COMMON BLOCK	SUBROUTINE
DQG	CONTROL-PT-SPEC	CNTRLPS	/CPGEOM/	MAPB
DQG	EDGE-POINT-COORDS	EDGPTS	Dynamic	MAPB
DQG	FINE-GRID-COORDS	FCNCORDS	Dynamic	MAPB
DQG	NETWK-SPEC	NETMAP	/NETWK/	MAPB
DQG	PANEL-CORNER-COORDS	CORNCOORDS	Dynamic	MAPB
DQG	SINGULARITY-MAP	SINGMAP	/SINGLR/	MAPB
DQG	SINGULAIRTY-SPEC	SINGSPC	/SINGLR/	MAPB
DQG	SINGULARITY-SPEC	SINGSPEC	Dynamic	MAPB

Overlay (3,0)

DATABASE	DATASET-NAME	MAP NAME	COMMON BLOCK	SUBROUTINE
DQG	ABUTMENT-KEYS	ABUTMENTS	Dynamic	ABUTMNT
DQG	ABUTMENT-SPEC	ABUTMENT	/ABUT/	EDGECP
DQG	CONTROL-PT-SPEC	CTLDEXMAP	Dynamic	ZHLOC
DQG	CONTROL-PT-SPEC	CTLSPECDYN	Dynamic	ADCPSG
DQG	EDGE-POINT-COORDS	EDGPTS	Dynamic	EDGDEF
DQG	EMPTY-SPACE-ABUTMENTS	ESABUTMNT	/ABUT/	EDGDEF
DQG	EXPANDED-ABUTMENT	EXPABUT	/EXPAND/	EDGDEF
DQG	EXTRA-HYPO-LOC	XHLOCCP	Local	XHLOC
DQG	GAP-PANEL	GAPPANEL	/GAPANL/	GAPPNL
DQG	GAP-SIZE	GAPSIZE	Dynamic	EDGDEF
DQG	I-ABUT	IABUTMAP	/ABUT/	EDGDEF
DQG	INTERSECTION	CONNECTION	/MATCHD/	MATCH
DQG	NETWK-SPEC	NETMAP	/NETWK/	EDGDEF
DQG	NETWK-SPEC	NETWKS	Dynamic	EDGDEF
DQG	PANEL-CORNER-COORDS	COORDS-GEN	Dynamic	ABUTMNT
DQG	SEARCH-LIST	SEARCHLIST	/LIST/	EDGDEF
DQG	SINGULARITY-MAP	SINGMAP	/SINGLR/	EDGDEF
DQG	SINGULARITY-SPEC	SINGSPC	/SINGLR/	EDGDEF
DQG	SINGULARITY-SPEC	SINGSPEC	Dynamic	EDGDEF
DQG	SPECIAL-POINTS	SPECIALPT	/SPECPT/	EDGDEF
DQG	USER-ABUT	USABUT	/ABUT/	EDGDEF

Overlay (4,0)

DATABASE	DATASET-NAME	MAP NAME	COMMON BLOCK	SUBROUTINE
DQG	BNDRY-CONDY-SPEC	BNDRY	/BCDOUT/	BNDYDF
DQG	B-POINTER	BPOINT	/BCDOUT/	BNDYDF
DQG	CLASS-5-BC-DATA	CLASS5	/NBCDIN/	BNDYDF
DQG	CLASS-5-BC-DATA	XCLASS5D	/XBCDIN/	BNDYDF
DQG	CLOSURE	CLOSURE	/CLOSUR/	BNDYDF
DQG	CLOSURE-DATA-IN	CLOSDIN	/CLOSUR/	BNDYDF
DQG	CONTROL-PT-SPEC	CNTRLPT	/CPGEOM/	BNDYDF
DQG	CONTROL-PT-SPEC	CTLDEXMAP	Dynamic	BNDYDF
DQG	CONTROL-PT-SPEC	INDCTLMP	Dynamic	BNDYDF
DQG	NETWK-BNDRY-CONDY-IN	BCDATIN	/NBCDIN/	BNDYDF
DQG	NETWK-SPEC	NETMAP	/NETWK/	BNDYDF
DQG	PANEL-CORNER-COORDS	COORDS-GEN	Dynamic	BNDYDF
DQG	SINGULARITY-MAP	SINGMAP	/SINGLR/	BNDYDF
DQG	SINGULARITY-SPEC	SINGMAP	Dynamic	BNDYDF
DQG	SPECIAL-POINTS	SPECIALPT	/SPECPT/	BNDYDF

Overlay (5,0)

DATABASE	DATASET-NAME	MAP NAME	COMMON BLOCK	SUBROUTINE
DQG	ABUTMENT-SPEC	ABUTMENT	/ABUT/	SAEDGS
DQG	ABUTMENT-SPEC	ABUTMENT	/ABUT/	SPLTPR
DQG	B-SPLINE-SOURCE	SPLINE-SRC	Dynamic	SPLINR
DQG	B-SPLINE-SOURCE	SSPLINE	/SPLINE/	SPLINR
DQG	B-SPLINE-DOUBLET	DSPLINE	/SPLINE/	SAEDGS
DQG	B-SPLINE-DOUBELT	DSPLINE	/SPLINE/	SPLINR
DQG	B-SPLINE-DOUBLET	SPLINE-DBL	Dynamic	SAEDGS
DQG	B-SPLINE-DOUBLET	SPLINE-DBL	Dynamic	SPLINR
DQG	EDGE-POINT-COORDS	EDGPTS	Dynamic	SAEDGS
DQG	EDGE-POINT-COORDS	EDGPTS	Dynamic	SPLINR
DQG	GAPPANEL	GAP-PANEL	/GAPAN/	SPLINR
DQG	INTERIOR-SPLINE	INTSPLMP	Dynamic	SAEDGS
DQG	INTERIOR-SPLINE	INTSPLMP	Dynamic	SPLINR
DQG	NETWK-SPEC	NETMAP	/NETWK/	SAEDGS
DQG	NETWK-SPEC	NETMAP	/NETWK/	SPLINR
DQG	PANEL-CORNER-COORDS	COORDS-GEN	Dynamic	SAEDGS
DQG	PANEL-CORNER-COORDS	COORDS-GEN	Dynamic	SPLINR
DQG	SINGULARITY-MAP	SINGMAP	/SINGLR/	SAEDGS
DQG	SINGULARITY-MAP	SINGMAP	/SINGLR/	SPLINR
DQG	SPECIAL-POINTS	SPECALPT	/SPECPT/	SPLINR

Overlay (6,0)

DATABASE	DATASET-NAME	MAP NAME	COMMON BLOCK	SUBROUTINE
DQG	B-SPLINE-DBL	SPLINE-DBL	Dynamic	PANDEF
DQG	BSPLINE-SRC	SPLINE-SRC	Dynamic	PANDEF
DQG	GAP-PANEL	GAPPANEL	/GAPANL/	PANDEF
DQG	GAP-SIZE	GAPSIZE	Dynamic	PANDEF
DQG	MAG-PANEL-SPEC	MAGPSPEC	/PANEL/ /SPLINE/	PANDEF
DQG	MDG-PANEL-SPEC	MDGPSPEC	/PANEL/ /FFM/	PANDEF
DQG	NETWK-SPEC	NETMAP	/NETWK/	PANDEF
DQG	PANEL-CORNER-COORDS	COORDS-GEN	Dynamic	PANDEF
DQG	PANEL-CORNER-COORDS	CORNCOORDS	/COORDS/	PANDEF
DQG	PANEL-SING	PANSING	/SPLINE/	PANDEF
DQG	PANEL-SPEC	PANSPEC	/FFM/ /PANEL/ /SPLINE/	PANDEF

Overlay (7,0)

DATABASE	DATASET-NAME	MAP NAME	DATA	SUBROUTINE
DQG	BNDRY-CONDN-SPEC	BCOUTDATA	Dynamic	PBCDAT
DQG	BNDRY-CONDN-SPEC	CTLOUTDATA	Dynamic	PCPDAT
DQG	DATA-BASE-HEADER	DBHEADER	Dynamic	SUMMRY
DQG	GLOBAL	GLOB-DYN	/GLOBAL,Dynamic	SUMMRY
DQG	NETWK-SPEC	NETMAP	/NETWK/	SUMMRY
DQG	SPECIAL-POINTS	SPECIALPT	/SPECPT/	SUMMRY

Table 4-C.2 Data Flow for DQG

Second Form

Overlay (1,0)

DATABASE	MAP NAME	DATASET-NAME	COMMON BLOCK	SUBROUTINE
MEC	IDS	DATA-BASE-HEADER	/RUINDS/	OPENER
MEC	RUNOPT	MACRO-OPTIONS	local	OPENER
DIP	CGBCMP	COEF-GEN-BC	/GENBCD/	BNDYIN
DIP	DIPCLOSDAT	CLOS-COND	/GENBCD/	BNDYIN
DIP	GLOBAL-IN	GLOBAL	/GLOBAL/	DIPDAT
DIP	NETBDC	NETWK-BDC	/NETBDC/	BNDYIN
DIP	NETMAP	NETWK-SPEC	/NETWK/	DIPDAT
DIP	PAN-COR-PT	PANEL-COORDS	/COORDS/	DIPDAT
DIP	PRINT-OPT	GLONAL-PRINTS	DIPDAT	
DIP	TVECTCOEFF	TANG-VEC	/GENBCD/	BNDYIN
DIP	USABIN	USER-ABUT	DIPDAT	
DQG	BCDATIN	NETWK-BNDRY-CONDN-IN	/NISCDIN/	BNDYIN
DQG	CLASS	CLASS-5-BC-DATA	/NBCDIN/	BNYDIN
DQG	CLOSDIN	CLOSURE-DATA-IN	/CLOSUR/	BNDYIN
DQG	COORDS-GEN	PANEL-CORNER-COORDS	Dynamic	DIPDAT
DQG	GLOB-DYN	GLOBAL	/GLOBAL/,Dynamic	DIPDAT
DQG	NETMAP	NETWK-SPEC	/NETWK/	DIPDAT
DQG	USABUAT	USER-ABUT	DIPDAT	

Overlay (2,0)

DATABASE	MAP NAME	DATASET-NAME	COMMON BLOCK	SUBROUTINE
DQG	CORNCOORDS	PANEL-CORNER-COORDS	Dynamic	MAPB
DQG	CNTRLPS	CONTROL-PT-SPEC	/CPGEOM/	MAPB
DQG	EDGPTS	EDGE-POINT-COORDS	Dynamic	MAPB
DQG	FCNCORDS	FINE-GRID-COORDS	Dynamic	MAPB
DQG	NETMAP	NETWK-SPEC	/NETWK/	MAPB
DQG	SINGMAP	SINGULARITY-MAP	/SENCLR/	MAPB
DQG	SINGSPC	SINGULARITY-SPEC	/SINGLR/	MAPB
DQG	SINGSPEC	SINGULARITY-SPEC	Dynamic	MAPB

Overlay (3,0)

DATABASE	MAP NAME	DATASET-NAME	COMMON BLOCK	SUBROUTINE
DQG	ABUTMENT	ABUTMENT-SPEC	/ABUT/	EDGECP
DQG	ABUTMENTS	ABUTMENT-KEYS	Dynamic	ABUTMNT
DQG	CONNECTION	INTERSECTION	/MATCHD/	MATCH
DQG	COORDS-GEN	PANEL-CORNER-COORDS	Dynamic	ABUTMNT
DQG	CTLDEXMAP	CONTROL-PT-SPEC	Dynamic	ZHLOC
DQG	CTLSPECDYN	CONTROL-PT-SPEC	Dynamic	ADCPSG
DQG	EDGPTS	EDGE-POINT-COORDS	Dynamic	EDGDEF
DQG	ESABUTMNT	EMPTY-SPACE-ABUTMENTS	/EXPAND/	EDGDEF
DQG	EXPABUT	EXPANDED-ABUTMENT	/EXPAND/	EDGDEF
DQG	GAPPANEL	GAP-PANEL	/GAPANL/	GAPPNL
DQG	GAPSIZE	GAP-SIZE	Dynamic	EDGDEF
DQG	IABUTMAP	I-ABUT	/ABUT/	EDGDEF
DQG	NETMAP	NETWK-SPEC	/NETWK/	EDGDEF
DQG	NETWKS	NETWK-SPEC	Dynamic	EDGDEF
DQG	SEARCHLIST	SEARCH-LIST	/LIST/	EDGDEF
DQG	SINGMAP	SINGULARITY-MAP	/SINGLR/	EDGDEF
DQG	SINGSPC	SINGULARITY-SPEC	/SINGLR/	EDGDEF
DQG	SINGSPEC	SINGULARITY-SPEC	Dynamic	EDGDEF
DQG	SPECIALPT	SPECTAL-POINTS	/SPECPT/	EDGDEF
DQG	USABUT	USER-ABUT	/ABUT/	EDGDEF
DQG	XHLOCCP	EXTRA-HYPO-LOC	Local	XHLOC

Overlay (4,0)

DATABASE	MAP NAME	DATASET-NAME	COMMON BLOCK	SUBROUTINE
DQG	BCDATIP	NETWK-BNDRY-CONDY-IN	/NBCDIN/	BNDYDF
DQG	BNDRY	BNDRY-CONDY-SPEC	/BCDOUT/	BNDYDR
DQG	BPOINT	B-POINTER	/BCDOUT/	BNDYDF
DQG	CLASS5	CLASS-5-BC-DATA	/NBCDIN/	BNDYDF
DQG	CLOSDIN	CLOSURE-DATA-IN	/CLOSUR/	BNDYDF
DQG	CLOSURE	CLOSURE	/CLOSUR/	BNDYDF
DQG	COORDS-GEN	PANEL-CORNER-COORDS	Dynamic	BNDYDF
DQG	CNTRLPT	CONTROL-PT-SPEC	/CPGEOM/	BNDYDF
DQG	CTLDEXMAP	CONTROL-PT-SPEC	Dynamic	BNDYDF
DQG	INDCTLMP	CONTROL-PT-SPEC	Dynamic	BNDYDF
DQG	NETMAP	NETWK-SPEC	/NETWK/	BNDYDF
DQG	SINGMAP	SINGULARITY-MAP	/SINGLR/	BNDYDF
DQG	SINGMAP	SINGULARITY-SPEC	Dynamic	BNDYDF
DQG	SPECIALIS	SPECIAL-POINTS	/SPECPT/	BNDYDF
DQG	XCLASS5D	CLASS-5-BC-DATA	/XBCDIN/	BNDYDF

Overlay (5,0)

DATABASE	MAP NAME	DATASET-NAME	COMMON BLOCK	SUBROUTINE
DQG	ABUTMENT	ABUTMENT-SPEC	/ABUT/	SAEDGS
DQG	ABUTMENT	ABUTMENT-SPEC	/ABUT/	SPLTPR
DQG	COORDS-GEN	PANEL-CORNER-CORDS	Dynamic	SAEDGS
DQG	COORDS-GEN	PANEL-CORNER-CORDS	Dynamic	SPLINR
DQG	DSPLINE	B-SPLINE-DOUBLET	/SPLINE/	SAEDGS
DQG	DSPLINE	B-SPLINE-DOUBLET	/SPLINE/	SPLINR
DQG	EDGEPTS	EDGE-POINT-COORDS	Dynamic	SAEDGS
DQG	EDGEpts	EDGE-POINT-CORDS	Dynamic	SPLINR
DQG	GAP-PANEL	GAPPANEL	/GAPAN/	SPLINR
DQG	INTSPLMP	INTERCOR-SPLINE	Dynamic	SAEDGS
DQG	INTSPLMP	INTERCOR-SPLINE	Dynamic	SPLINR
DQG	NETMAP	NETWK-SPEC	/NETWK/	SAEDGS
DQG	NETMAP	NETWK-SPEC	/NETWK/	SPLINR
DQG	SCPGMAP	SINGULARITY-MAP	/SENGLR/	SAEDGS
DQG	SLPGMAP	SINGULARITY-MAP	/SENGLR/	SPLINR
DQG	SPECALPT	SPECIAL-POINTS	/SPECPT/	SPLINR
DQG	SPLINE-DBL	B-SPLINE-DOUBLET	Dynamic	SAEDGS
DQG	SPLINE-DBL	B-SPLINE-DOUBLET	Dynamic	SPLINR
DQG	SPLINE-SRC	B-SPLINE-SOURCE	SPLINR	
DQG	SSPLINE	B-SPLINE-SOURCE	SPLINR	

Overlay (6,0)

DATABASE	MAP NAME	DATASET-NAME	COMMON BLOCK	SUBROUTINE
DQG	COORDS-GEP	PANEL-CORNER-COORDS	Dynamic	PANDEF
DQG	CORNCOORDS	PANEL-CORNER-COORDS	/COORDS/	PANDEF
DQG	GAPFILE	GAP-FILE	Dynamic	PANDEF
DQG	GAPPANEL	GAP-PANEL	/GAPANL/	PANDEF
DQG	MAGPSPEC	MAG-PANEL-SPEC	/PANEL/ /SPLINE/	PANDEF
DQG	MDGPSPEC	MDG-PANEL-SPEC	/PANEL/ /FFM/	PANDEF
DQG	NETMAP	NETWK-SPEC	/NETWK/	PANDEF
DQG	PANSINF	PANEL-SING	/SPLINE/	PANDEF
DQG	PANSPEC	PANEL-SPEC	/FFM/	PANDEF
DQG	SPLINE-DBL	B-SPLINE-DOUBLET	Dynamic	PANDEF
DQG	SPLINE-SRL	B-SPLINE-SOURCE	Dynamic /PANEL/ /SPLINE/	PANDEF

Overlay (7,0)

DATABASE	MAP NAME	DATASET-NAME	COMMON BLOCK	SUBROUTINE
DQG	BCOUTDATA	BNDRY-CONDN-SPEC	Dynamic	PBCDAT
DQG	CTLOUTDATA	BNDRY-CONDN-SPEC	Dynamic	PCPDAT
DQG	DBHEADER	DATA-BASE-HEADER	Dynamic	SUMMRY
DQG	GLOB-DYN	GLOBAL	/GLOBAL,Dynamic	SUMMRY
DQG	NETMAP	NETWK-SPEC	/NETWK/	SUMMRY
DQG	SPECIALPT	SPECIAL-POINTS	/SPECPT/	SUMMRY

Table 4-C.3 Data Flow for DQG

Third Form

Overlay (1,0)

COMMON BLOCK	DATABASE	MAP NAME	DATASET-NAME	SUBROUTINE
local	MEC	RUNOPT	MACRO-OPTIONS	OPENER
/RUINDS/	MEC	IDS	DATA-BASE-HEADE	OPENER
/ABUT/	DIP	USABIN	USER-ABUT	DIPDAT
/COORDS/	DIP	PAN-COR-PT	PANEL-COORDS	DIPDAT
Dynamic	DIP	PRINT-OPT	GLOBAL-PRINTS	DIPDAT
/GENBCD/	DIP	DIPCLOSDAT	CLOS-COND	BNDYIN
/GENBCD/	DIP	TVECTCOEFF	TANG-VEC	BNDYIN
/GENBCD/	DIP	CGBCMP	COEF-GEN-BC	BNDYIN
/GLOBAL/	DIP	GLOBAL-IN	GLOBAL	DIPDAT
/NETBDC/	DIP	NETBCD	NETWK-BDC	BNDYIN
/NETWK/	DIP	NETMAP	NETWK-SPEC	DIPDAT
/ABUT/	DQG	USABUAT	USER-ABUt	DIPDAT
Dynamic	DQG	COORDS-GEN	PANEL-CORNER-COORDS	DIPDAT
/CLOSUR/	DQG	CLOSDIN	CLOSURE-DATA-IN	BNDYIN
/GLOBAL/,Dynamic	DQG	GLOB-DYN	GLOBAL	DIPDAT
/NBCDIN/	DQG	CLASS	CLASS-S-BC-DATA	BNYDIN
/NETWK/	DQG	NETMAP	NETWK-SPEC	DIPDAT
/NISCDIN/	DQG	BCDATIN	NETWK-BNDRY-CONDN-IN	BNDYIN

Overlay (2,0)

COMMON BLOCK	DATABASE	MAP NAME	DATASET-NAME	SUBROUTINE
Dynamic	DQG	CORNCOORDS	PANEL-CORNER-COORDS	MAPB
Dynamic	DQG	EDGOTS	EDGE-POINT-COORDS	MAPB
Dynamic	DQG	FCNCORDS	FINE-GRID-COORDS	MAPB
Dynamic	DQG	SINGSPEC	SINGULARITY-SPEC	MAPB
/CPGEOM/	DQG	CNTRLPS	CONTROL-PT-SPEC	MAPB
/NETWK/	DQG	NETMAP	NETWK-SPEC	MAPB
/SINCLR/	DQG	SINGMAP	SINGULARITY-MAP	MAPB
/SINGLR/	DQG	SINGSPC	SINGULARITY-SPEC	MAPB

Overlay (3,0)

COMMON BLOCK	DATABASE	MAP NAME	DATASET-NAME	SUBROUTINE
/ABUT/	DQG	ABUTMENT	ABUTMENT-SPEC	EDGECP
/ABUT/	DQG	ESABUTMNT	EMPTY-SPACE-ABUTMENTS	EDGDEF
/ABUT/	DQG	IABUTMAP	I-ABUT	EDGDEF
/ABUT/	DQG	USABUT	USER-ABUT	EDGDEF
Dynamic	DQG	ABUTMENTS	ABUTMENT-KEYS	ABUTMNT
Dynamic	DQG	COORDS-GEN	PANEL-CORNER-COORDS	ABUTMNT
Dynamic	DQG	CTLDEXMAP	CONTROL-PT-SPEC	ZHLOC
Dynamic	DQG	CTLSPECDYN	CONTROL-PT-SPEC	ADCPSG
Dynamic	DQG	EDGPTS	EDGE-POINT-COORDS	EDGDEF
Dynamic	DQG	GAPSIZE	GAP-SIZE	EDGDEF
Dynamic	DQG	NETWKS	NETWK-SPEC	EDGDEF
Dynamic	DQG	SINGSPEC	SINGULARITY-SPEC	EDGDEF
/EXPAND/	DQG	EXPABUT	EXPANDED-ABUTMENT	EDGDEF
/GAPANL/	DQG	GAPPANEL	GAP-PANEL	GAPPNL
/LIST/	DQG	SEARCHLIST	SEARCH-LIST	EDGDEF
Local	DQG	XHLOCCP	EXTRA-HYPO-LOC	XHLOC
/MATCHD/	DQG	CONNECTION	INTERSECTION	MATCH
/NETWK/	DQG	NETMAP	NETWK-SPEC	EDGDEF
/SPECPT/	DQG	SPECIALPT	SPECIAL-POINTS	EDGDEF
/SINGLR/	DQG	SINGMAP	SINGULARITY-MAP	EDGDEF
/SINGLR/	DQG	SINGSPC	SINGULARITY-SPEC	EDGDEF

Overlay (4,0)

COMMON BLOCK	DATABASE	MAP NAME	DATASET-NAME	SUBROUTINE
/BCDOUT/	DQG	BNDRY	BNDRY-CONDY-SPEC	BNDYDR
/BCDOUT/	DQG	BPOINT	B-POINTER	BNDYDF
/CLOSUR/	DQG	CLOSURE	CLOSURE	BNDYDF
/CLOSUR/	DQG	CLOSDIN	CLOSURE-DATA-IN	BNDYDF
/CPGEOM/	DQG	CNTRLPT	CONTROL-PT-SPEC	BNDYDF
Dynamic	DQG	CTLDEXMAP	CONTROL-PT-SPEC	BNDYDF
Dynamic	DQG	INDCTLMP	CONTROL-PT-SPEC	BNDYDF
Dynamic	DQG	COORDS-GEN	PANEL-CORNER-COORDS	BNDYDF
Dynamic	DQG	SINGMAP	SINGULARITY-SPEC	BNDYDF
/NBCDIN/	DQG	BCDATIP	NETWK-BNDRY-CONDY-IY	BNDLYDF
/NBCDIN/	DQG	CLASS5	CLASS-5-BC-DATA	BNDYDF
/NETWK/	DQG	NETMAP	NETWK-SPEC	BNDYDF
/SINGLR/	DQG	SINGMAP	SINGULARITY-MAP	BNDYDF
/SPECPT/	DQG	SPECIALIS	SPECIAL-POINTS	BNDYDF
/XBCDIN/	DQG	XCLASS5D	CLASS-S-BC-DATA	BNDYDF

Overlay (5,0)

COMMON BLOCK	DATABASE	MAP NAME	DATASET-NAME	SUBROUTINE
/ABUT/	DQG	ABUTMENT	ABUTMENT-SPEC	SAEDGS
/ABUT/	DQG	ABUTMENT	ABUTMENT-SPEC	SPLTPR
Dynamic	DQG	COORDS-GEN	PANEL-CORNER-COORDS	SAEDGS
Dynamic	DQG	COORDS-GEN	PANEL-CORNER-COORDS	SPLINR
Dynamic	DQG	SPLINE-SCR	B-SPLINE-SOURCE	SPLINR
Dynamic	DQG	SPLINE-DBL	B-SPLINE-DOUBLET	SPLINR
Dynamic	DQG	SPLINE-DJL	B-SPLINE-DOUBLET	SAEDGS
Dynamic	DQG	EDGPTS	EDGE-POINT-COORDS	SAEDGS
Dynamic	DQG	EDGPTS	EDGE-POINT-COORDS	SPLINR
Dynamic	DQG	INTSPLMP	INTERCOR-SPLINE	SAEDGS
Dynamic	DQG	INTSPLMP	INTERCOR-SPLINE	SPLINR
/GAPAN/	DQG	GAP-PANEL	GAPPANEL	SPLINR
/NETWK/	DQG	NETMAP	NETWK-SPEC	SAEDGS
/NETWK/	DQG	NETMAP	NETWK-SPEC	SPLINR
/SENGLR/	DQG	SCPGMAP	SINGULARITY-MAP	SAEDGS
/SENGLR/	DQG	SLPGMAP	SINGULARITY-MAP	SPLINR
/SPECPT/	DQG	SPECALPT	SPECIAL-POINTS	SPLINR
/SPLINE/	DQG	SSPLINE	B-SPLINE-SOURCE	SPLINR
/SPLINE/	DQG	DSPLINE	B-SPLINE-DOUBLET	SAEDGS
/SPLINE/	DQG	DSPLINE	B-SPLINE-DOUBLET	SPLINR

Overlay (6,0)

COMMON BLOCK	DATABASE	MAP NAME	DATASET-NAME	SUBROUTINE
/COORDS/	DQG	CORNCOORDS	PANEL-CORNER-COORDS	PANDEF
Dynamic	DQG	COORDS-GET	PANEL-CORNER-COORDS	PANDEF
Dynamic	DQG	GAPFILE	GAP-FILE	PANDEF
Dynamic	DQG	SPLINE-DBL	B-SPLINE-DOUBLET	PANDEF
Dynamic	DQG	SPLINE-SRL	B-SPLINE-SOURCE	PANDEF
/FFM/	DQG	MDGPSPEC	MDG-PANEL-SPEC	PANDEF
/FFM/	DQG	PANSPEC	PANEL-SPEC	PANDEF
/GAPANL/	DQG	GAPPANEL	GAP-PANEL	PANDEF
/NETWK/	DQG	NETMAP	NETWK-SPEC	PANDEF
/PANEL/				
/PANEL/	DQG	MAGPSPEC	MAG-PANEL-SPEC	PANDEF
/PANEL/	DQG	MDGPSPEC	MDG-PANEL-SPEC	PANDEF
/SPLINE/	DQG	MAGPSPEC	MAG-PANEL-SPEC	PANDEF
/SPLINE/	DQG	PANSING	PANEL-SING	PANDEF
/SPLINE/				

Overlay (7,0)

COMMON BLOCK	DATABASE	MAP NAME	DATASET-NAME	SUBROUTINE
Dynamic	DQG	BCOUTDATA	BNDRY-CONDN-SPEC	PBCDAT
Dynamic	DQG	CTLOUTDATA	BNDRY-CONDN-SPEC	PCPDAT
Dynamic	DQG	DBHEADER	DATA-BASE-HEADER	SUMMRY
/GLOBAL/Dynamic	DQG	GLOB-DYN	GLOBAL	SUMMRY
/NETWK/	DQG	NETWK	NETWK-SPEC	SUMMRY
/SPECPT/	DQG	SPECIALPT	SPECIAL-POINTS	SUMMRY

APPENDIX 4-D MASTER DEFINITION

The data base master definition listing of the DQG module has been deleted from this document. It is produced from the PAN AIR tape during installation.

APPENDIX 4-E

ERROR MESSAGES IN DQG

The following pages list the messages that accompany all diagnosed errors in DQG. Section 8. of the PAN AIR User's Manual (Reference 2) discusses interpretation of the messages and suggests causes and remedies.

```
                PROGRAM OPENER

******** FATAL ERROR
         RUN,PROBLEM, AND USER IDS NOT FOUND
         ON THE MEC DATABASE


                SUBROUTINE DIPDAT

******** ERROR
         NO NETWORKS DEFINED
******** ERROR
         ZERO LENGTH ABUTMENT
           USER ABUTMENT INDEX          3
             NETWORK      EDGE      START PT      END PT
                   1         1            3           3
******** ERROR
         INVALID SOURCE/DOUBLET TYPE FROM DIP
            NETWORK  UPPER-WING
            SOURCE TYPE     1  DOUBLET TYPE    9


                SUBROUTINE NETDEF

******** FATAL ERROR
         1 COLUMN OR 1 ROW SOURCE DESIGN II
         NETWORK ENCOUNTERED. NETWORK NO =   4
         EXECUTION WILL BE TERMINATED.
******** THE FATAL ERROR LIMIT OF         10 WAS EXCEEDED.
         EXECUTION WILL BE TERMINATED


                SUBROUTINE DFEDGE

******** FATAL ERROR
         NETWORK     3 COLUMN     1 OF CORNER
         POINTS NOT AVAILABLE ON DATABASE.


                SUBROUTINE EDGCHK

******** FATAL ERROR
         NETWORK (UPPER-WING)  EDGE    3
         SOURCE DESIGN I NETWORK CAN NOT HAVE A COLLAPSED EDGE.
******** FATAL ERROR
         NETWORK (UPPER-WING)  EDGE    3
         SOURCE DESIGN II NETWORK CAN NOT HAVE A COLLAPSED EDGE.
******** FATAL ERROR
         NETWORK (UPPER-WING          )  EDGE   4
         AVERAGE PANEL LENGTH EXCEEDS TOLERANCE
         BUT THE MINIMUM DOES NOT.  THE EDGE
         CANNOT BE COLLAPSED.
******** FATAL ERROR
         TWO ADJACENT EDGES HAVE ZERO LENGTH.
         NETWORK UPPER-WING           EDGES   1  2
```

```
            SUBROUTINE SNGPAN

******** FATAL ERROR
         SINGULARITY TYPE NOT FOUND FOR NETWORK     3

            SUBROUTINE TRICHK

******** FATAL ERROR
         INTERIOR PANEL IS TRIANGULAR
         NETWORK UPPER-WING           PANEL COLUMN     5 AND ROW    2
******** FATAL ERROR
         ZERO DENOMINATOR FOR ASPECT RATIO OF
         NETWORK UPPER-WING           PANEL COLUMN     1 AND ROW    6
******** FATAL ERROR
         ASPECT RATIO =  0.6934E+06
         NETWORK UPPER-WING           PANEL COLUMN     3 AND ROW    8
 * * * * WARNING
         ASPECT RATIO =  0.6394E+04
         NETWORK UPPER-WING           PANEL COLUMN     3 AND ROW    9


            SUBROUTINE SEARCH

******** ERROR
          ERRONEOUS USER ABUTMENT DATA
          OVERLAPPING ABUTMENTS
            NETWORK  UPPER-WING              EDGE          3
            OVERLAP FROM COLUMN          3  ROW          1
            TO COLUMN                    7  ROW          1

            SUBROUTINE EDGLST

******** WARNING
          TOO MANY NEARBY NETWORK EDGES
          SOME ABUTMENTS MAY BE MISSED
            NETWORK FIN                    EDGE     1

            PROGRAM PRABUT

******** ERROR
          INSUFFICIENT CORE MEMORY FOR AUTOMATIC ABUTMENT SEARCH
          NUMBER OF EXTRA CORE MEMORY NEEDED   738
          OR APPROXMATELY  2000 OCTAL
******** ERROR
          ERROR IN REQUESTING BLANK COMMON IN SUB PRABUT
          ERROR NUMBER    2

            SUBROUTINE USEABT

******** ERROR
          ERRONEOUS USER ABUTMENT DATA
            USER ABUTMENT NUMBER      3
          NETWORK    EDGE    START-X    START-Y    END-X        END-Y
               1       5          1          1        7          1
               2       3          7          1        7          7
```

```
******** ERROR
         NETWORK EDGES TO FAR APART FOR ABUTMENT
           USER ABUTMENT NUMBER      4
         NETWORK    EDGE    START-X    START-Y    END-X        END-Y
               1       1          1          1        3          1
               5       1          6          1        1          1
                2 TH NETWORK EDGE IN LIST GT  1.357E+15
                FROM FIRST NETWORK EDGE IN LIST
******** ERROR
        KUTTA TANGENT IS NOT PERPENDICULAR TO PLANE-OF-SYMMETRY NORMAL
        NETWORK      EDGE      DQGCP    POS
           3           4        37       1
******** ERROR
         ABUTMENT POINTS NOT ON NETWORK EDGE
           USER ABUTMENT NUMBER      7
         NETWORK    EDGE    START-X    START-Y    END-X        END-Y
               3       1          3          3        5          7
           NUMBER OF ROWS IN NETWORK =      5
           NUMBER OF COLUMNS IN NETWORK =      7
******** ERROR
         ERRONEOUS USER ABUTMENT DATA
         ZERO LENGTH ABUTMENT
           USER ABUTMENT NUMBER      8
         NETWORK    EDGE    START-X    START-Y    END-X        END-Y
               7       2          5          1        5          1
******** ERROR
         ERRONEOUS USER ABUTMENT DATA
         COLLAPSED EDGE IN ABUTMENT
           USER ABUTMENT NUMBER      3
         NETWORK    EDGE    START-X    START-Y    END-X        END-Y
               1       5          1          1        7          1
               2       3          7          1        7          7
         ZERO LENGTH ABUTMENT

             SUBROUTINE ABXPND

******** ERROR
         TOO MANY NETWORK EDGES IN AN ABUTMENT
         THIS MAY ARISE EITHER FROM HAVING TOO
         MANY NETWORK EDGES COMING TOGETHER IN
         A SINGLE ABUTMENT OR FROM THE SAME
         NETWORK EDGES TAKING PART IN TOO MANY
         ABUTMENTS.
           NETWORK       EDGE
                     1          1
                     2          1
                     3          2
                     4          1
                     5          3
                     6          4
                     7          1
                     8          4
                     9          1
                    10          1
                    11          2
```

```
            SUBROUTINE CHECK

******** ERROR
         UPDATABLE NETWORK EDGE ABUTTING
         A NONUPDATABLE NETWORK EDGE
           ABUTMENT INDEX      3
         NETWORK    EDGE    START-X    START-Y    END-X        END-Y
              1       1          1          1        5          1
              2       3          3          3        1          1
           UPDATABLE FLAG      1       0
******** WARNING
         UPDATABLE NETWORK EDGE ABUTTING
         A NONUPDATABLE NETWORK EDGE
           ABUTMENT INDEX      3
         NETWORK    EDGE    START-X    START-Y    END-X        END-Y
              1       1          1          1        5          1
              2       3          3          3        1          1
           UPDATABLE FLAG      1       0
******** WARNING
         MORE THAN TWO NETWORKS IN SMOOTH ABUTMENT.
         SMOOTH ABUTMENT TREATED AS NORMAL ABUTMENT.
           ABUTMENT INDEX      4
         NETWORK    EDGE    START-X    START-Y    END-X        END-Y
              1       2          4          1        4          5
              2       1          1          1        4          1
              3       2          5          1        5          5
******** WARNING
         SMOOTH ABUTMENT DEFINED WITH DESIGN NETWORK
         SMOOTH ABUTMENT TREATED AS NORMAL ABUTMENT.
           ABUTMENT INDEX      7
         NETWORK    EDGE    START-X    START-Y    END-X        END-Y
              4       1          1          1        6          1
              5       1          6          1        1          1
******** ERROR
         ERRONEOUS ABUTMENT DATA
         EDGE OUT OF RANGE
           ABUTMENT INDEX      8
         NETWORK    EDGE    START-X    START-Y    END-X        END-Y
              3       5          3          3        1          3
              4       1          1          1        3          1
******** ERROR
         MORE THAN ONE MATCHING EDGE IN ABUTMENT
           ABUTMENT INDEX      9
         NETWORK    EDGE    START-X    START-Y    END-X        END-Y
              5       1          1          1        7          1
              6       1          1          1        3          1
******** WARNING
         MATCHING EDGE ABUTS A PLANE OF SYMMETRY.
         RESULTS DEPEND UPON THE CONFIGURATION.
         THE AIC MATRIX MAY BE UNDER CONSTRAINED,
         OVER-CONSTRAINED, SINGULAR OR REASONABLY
         CORRECT.  OTHER ERRORS MAY BE TRIGGERED
         BUT PROCESSING WILL CONTINUE AND A
         SOLUTION WILL BE ATTEMPTED
           DOUBLET MATCHING IMPOSED AT ABUTMENT.
```

```
******** WARNING
          NETWORK HAS TOO FEW PANELS FOR SMOOTH ABUTMENT
            ABUTMENT INDEX      9
          NETWORK    EDGE    START-X    START-Y    END-X        END-Y
               5       1          1          1        7          1
               6       1          1          1        3          1
            INDEX OF SMALL NETWORK     5
******** WARNING
            VELOCITY OPTIONS NOT COMPATIBLE
            ABUTMENT INDEX
          NETWORK    EDGE    START-X    START-Y    END-X        END-Y
               5       1          1          1        7          1
               6       1          1          1        3          1
            VELOCITY COMP METHODS     1      2


              SUBROUTINE CHKPOS

******** ERROR
          NETWORK ENCOUNTERED WHICH PARTIALLY LIES
          ON A PLANE OF SYMMETRY.
            NETWORK  PLANER-BODY           PLANE OF SYMMETRY      1
            NUMBER OF POINTS OFF P-O-S     20
            NUMBER OF POINTS ON P-O-S     10
******** WARNING
          NETWORK ENCOUNTERED WHICH PARTIALLY LIES
          ON A PLANE OF SYMMETRY.
            NETWORK  PLANER-BODY           PLANE OF SYMMETRY      1
            NUMBER OF POINTS OFF P-O-S     20
            NUMBER OF POINTS ON P-O-S     10


              SUBROUTINE CONABT

******** ERROR
          TOO MANY NETWORKS IN ABUTMENT
               NETWORK     EDGE     START-X   START-Y   END-X     END-Y
                  1          1          1         1        3         1
                  2          1          6         1        1         1
                  3          1          4         1        1         1
                  4          1          1         1        5         1
                  5          1          7         1        1         1
                  6          1          8         1        1         1
******** WARNING
          AUTOMATIC ABUTMENT SEARCH FINDS
          EMPTY SPACE ABUTMENT IN MIDDLE OF
          NETWORK EDGE.  CHECK EMPTY SPACE
          ABUTMENT DESCRIPTIONS IF USER
          DID NOT SPECIFY THE ABUTMENT.
               NETWORK     EDGE     START-X   START-Y   END-X     END-Y
                  1          1          1         1        5         1
```

```
            SUBROUTINE GAPSIZE

******** ERROR
          PROGRAM ERROR.  ZERO LENGTH ABUTMENT.
            ABUTMENT NUMBER       1
          NETWORK   EDGE    START-X    START-Y    END-X        END-Y
                1      1          1          1        1          1
                2      2          3          1        3          4

            SUBROUTINE ABASGN

******** ERROR
          NO MATCHING ASSIGNMENT POSSIBLE
            INTERSECTION NUMBER     10
            ABUTMENT INDEX     33
          NETWORK    EDGE    START-X    START-Y    END-X        END-Y
                1       1          1          1        8          1
                2       1          8          1        1          1
          CORNER POINT MAP INDEX     73
          NETWORK   EDGE     START-X    START-Y    END-X        END-Y
                1      2           8          1

            SUBROUTINE ASSIGN

******** ERROR
          ONLY ONE ABUTMENT IN AN INTERSECTION
            INTERSECTION NUMBER      3
            WITH ABUTMENTS
               2
******** ERROR
          NORMAL VECTOR NOT PERPENDICULAR TO P-O-S
          FOR A NETWORK THAT LIES ON P-O-S
            NETWORK    3    COLUMN    1    ROW    1
            NORMAL VECTOR   8.782E-01   0.000E+00   0.000E+00
            V DOT N   0.000E+00
******** ERROR
          INSUFFICIENT NUMBER OF CORNER POINTS ASSIGNED
          TO IMPOSE DOUBLET MATCHING.
          INTERSECTION NUMBER      6
          NUMBER ASSIGNED      1
          NUMBER REQUIRED      2
          WITH ABUTMENTS
              14     4     7
         A PROGRAM ERROR HAS OCCURRED DQG IS ABORTED.
******** WARNING
          INSUFFICIENT NUMBER OF CORNER POINTS ASSIGNED
          TO IMPOSE DOUBLET MATCHING.
          INTERSECTION NUMBER      7
          NUMBER ASSIGNED      1
          NUMBER REQUIRED      2
          WITH ABUTMENTS
            2005     6  1002
         SEE TABLE 8-17 OF PAN AIR USER"S MANUAL.
```

```
******** ERROR
         TOO MANY ABUTMENTS IN AN INTERSECTION
         INTERSECTION NUMBER      11
         WITH ABUTMENTS
              1     2   2003  2004     6     7  2008     9    11    17    21

             SUBROUTINE EMATCH

******** ERROR
         MORE THAN ONE MATCHING EDGE IN ABUTMENT
          ABUTMENT        3
           NUMBER OF MATCHING EDGES       2
           EDGE POINTERS       1        2
         NETWORK    EDGE    START-X    START-Y    END-X        END-Y
              1       1          1          1        7            1
              2       1          1          1        6            1
******** WARNING
         NO DOUBLET MATCHING AT NETWORK EDGE
           ABUTMENT INDEX       9
         NETWORK    EDGE    START-X    START-Y    END-X        END-Y
             10       4          1          1        1            7
             11       2          3          1        3            7

             SUBROUTINE INTRSC

******** ERROR
         MISSING ABUTMENTS IN PILOT CODE CONNECTION
         FOR CORNER POINT ON NETWORK      9    COLUMN     1    ROW    11
         CORNER POINT LABEL 9001011    ABUTMENTS       4      14

             SUBROUTINE NTRLST

******** ERROR
         TOO MANY ABUTMENTS INTERSECT
         INTERSECTION NUMBER      5
             ABUTMENT          CP        CP
                 1             1         2
                 2             2         3
                 3             3         4
                 4             3         4
                30            53         2
                31             1         3


             SUBROUTINE GAPPNL

******** ERROR
         PROGRAM ERROR. ZERO LENGTH ABUTMENT.
           ABUTMENT NUMBER       1
         NETWORK    EDGE    START-X    START-Y    END-X    END-Y
              1       1          1          1        1        1
              2       1          1          1        3        1
```

```
              SUBROUTINE DEFPNL

******** ERROR
          FACTOR FOR GAP PANEL .GT. 1.0
          NUMERATOR= 1.000E+00  DENOMINATOR= 1.000E-01
            ABUTMENTS INDEX=    5
            POINT INDEX=    13
            NETWORK LOOP INDEX=     1
            T12 ARRAY INDEX=      7
    T(I,1) ARRAY
 0.000E00  1.000E-01 1.000E+00
    T(I,2)ARRAY
 0.000E+00 1.000E+01 1.000E+00
    T12(I) ARRAY
0.000E+00  1.000E-01 1.000E+00 1.000E+01 1.000E+00
******** ERROR
          PROGRAM ERROR
          ABNORMAL LOOP TERMINATION
            ABUTMENT INDEX=     4
            POINT INDEX=     1
            NETWORK LOOP INDEX=     2
    T(I,1) ARRAY
 0.000E+00 1.000E+00
    T(I,2) ARRAY
 0.000E+00 0.000E+00
    T12(I) ARRAY
 0.000E+00 0.000E+00 0.000E+00 1.000E+00
            T12 ARRAY INDEX=     8


              SUBROUTINE POSPNL

******** WARNING
          GAP FILLING PANELS REQUIRED AT
          ABUTMENT WITH NETWORK EDGE AND
          TWO PLANES OF SYMMETRY. THIS
          SITUATION IS BEYOND CURRENT
          CAPABILITIES OF DQG.
            SITUATION OCCURS FOR ABUTMENT     7


              SUBROUTINE ASGNU

******** ERROR
          MACH INCLINED PANEL AND/OR SUBPANEL
            NETWORK FIN
            PANEL COLUMN     3
            PANEL ROW        6
            NORMAL-CONORMAL INNER PRODUCT  1.378E-13
```

```
******** ERROR
          VANISHINGLY SMALL INNER AND OUTER SUBPANELS
            NETWORK WING
            EDGE        2
            CORNER PT COLUMN        8
            CORNER PT ROW           1
            SUBPANEL NUMBER         2
            PT 1.000E-11   2.000E-11   3.000E-11   1.000E-11   2.000E-11
               0.000E 00   0.          0.          0.000E-11   1.000E-11
               0.          0.          0.          0.          0.
******** ERROR
          NON-CONVEX PANEL WITH CORNER POINT
          CLOSE TO PANEL CENTER POINT.
            NETWORK WING
            EDGE        3
            CORNER PT COLUMN       5
            CORNER PT ROW          3
            SUBPANEL NUMBER        3
            2+SKEW(1)+SKEW(2)  1.000E-03


              SUBROUTINE CCPGEO

******** ERROR
          TANGENT VECTOR PROJECTION TO PANEL
          IS LESS THAN HALF OF TANGENT VECTOR MAGNITUDE.
            NETWORK WING
            EDGE        4
            CORNER PT COLUMN       1
            CORNER PT ROW          7
            SUBPANEL               4
            TANGENT VECTOR UPPER
******** ERROR
          TANGENT VECTOR MAGNITUDE TOO SMALL
            NETWORK WING
            EDGE        1
            CORNER PT COLUMN        3
            CORNER PT ROW           1
            SUBPANEL                1
            TANGENT VECTOR RHS


              SUBROUTINE CENTCP

******** ERROR
          TANGENT VECTOR PROJECTION TO PANEL
          IS LESS THAN HALF OF TANGENT VECTOR MAGNITUDE
            NETWORK UPPER-WING
            PANEL COLUMN       3
            PANEL ROW          2
            USER CLASS
            TANGENT VECTOR  UPPER
```

```
          SUBROUTINE CHOOSE

******** ERROR
         TANGENT VECTOR MAGNITUDE TOO SMALL
           NETWORK LOWER-WING
           PANEL COLUMN      4
           PANEL ROW         1
           USER CLASS
           TANGENT VECTOR  AVERAGE
******** ERROR
         INSUFFICIENT NUMBER OF USER-SPECIFIED BOUNDARY CONDITIONS
           NETWORK  UPPER-WING
           PANEL COLUMN      3
           PANEL ROW         1
           TOTAL NUMBER OF BOUNDARY CONDITIONS REQUIRED    2
******** WARNING
         INSUFFICIENT NUMBER OF USER-SPECIFIED BOUNDARY CONDITIONS
           PROGRAM WILL ADD BOUNDARY CONDITION OF ZERO
           PERTUBATION MASS FLUX.  IF THIS BOUNDARY
           CONDITION IS UNSATISFACTORY, THE USER MUST ADD
           A BOUNDARY CONDITION FOR THIS PANEL BY
           INVOKING CLASS FIVE BOUNDARY CONDITIONS INPUT
           TO MODULE DIP FOR THIS NETWORK.  IF THIS IS
           A WAKE NETWORK, NO USER ACTION IS ADVISED.
           IF THIS NETWORK LIES ON A PLANE OF SYMMETRY.
           BE SURE AT LEAST ONE BOUNDARY CONDITION IS
           OF THE FORM NORMAL MASS FLUX, POTENTIAL OR
           TANGENTIAL VELOCITY (ALL AVERAGE QUANTITIES).
           NETWORK  RIGHT-WAKE
           PANEL COLUMN      1
           PANEL ROW         1
           TOTAL NUMBER OF BOUNDARY CONDITIONS REQUIRED    2


          SUBROUTINE ECPGEO

******** ERROR
         TANGENT VECTOR PROJECTION TO PANEL
         IS LESS THAN HALF OF TANGENT VECTOR MAGNITUDE
           NETWORK MID-WING
           CORNER PT COLUMN      3
           CORNER PT ROW         1
           SUBPANEL NUMBER       5
           TANGENT VECTOR  DIFFERENCE
******** ERROR
         TANGENT VECTOR MAGNITUDE TOO SMALL
           NETWORK MIDWING
           CORNER PT COLUMN      5
           CORNER PT ROW         1
           SUBPANEL NUMBER       5
           TANGENT VECTOR  DIFFERENCE
```

```
            SUBROUTINE GETBC

******** ERROR
         NO USER-SPECIFIED BOUNDARY CONDITIONS
         NETWORK OUTER-WING
         FINE GRID COLUMN INDEX     2
         FINE GRID ROW INDEX     3

            SUBROUTINE SPLTRN

******** ERROR
         INCORRECT SELECTION OF XI-ETA VECTORS.
         PROGRAM ERROR
              XI          ETA        ZETA        POINT        PO        VECTOR
         0.000E+00   1.414E-01  0.000E+00   3.786E+01   3.681E+01   1.050E+
         0.          3.735E-01  0.          4.138E+00   3.147E+00   9.990E-
         0.          0.         0.          1.791E+01   1.790E+01   1.000E-


            SUBROUTINE CPCSEL

******** ERROR
         PROGRAM ERROR.
           POINT NUMBER      5    6    4  CORNER PT T-VALUE-1.368E-01

            SUBROUTINE PTSFIL


******** ERROR
         ERRONEOUS ABUTMENT DESCRIPTION
           ABUTMENT ARRAY      1      5      1      1      7      1


            SUBROUTINE SNGFIL

******** ERROR
         ERRONEOUS ABUTMENT DESCRIPTION
           ABUTMENT ARRAY      1      5      1      1      7      1


            SUBROUTINE CPDSGN

******** ERROR
         PROGRAM/DATA ERROR.
         ZERO DENOMINATOR FOR CORNER POINT WEIGHT
         IN DOUBLET DESIGN EDGE SPLINE.
           NETWORK MID-WING
           EDGE      4
           CORNER POINT NUMBER      1
           DISTANCES TO ADJACENT EDGE MIDPOINTS
                      1.472E-21
                      4.216E-21
                      5.688E-21
```

```
            SUBROUTINE NTEDGA

******** ERROR
          PROGRAM ERROR.
          INCORRECT CALLING ARGUMENT FOR EDGE INDEX
            NETWORK UPPER-TAIL
            EDGE      7


            SUBROUTINE ONDFIT

******** ERROR
          PROGRAM ERROR
          SINGULAR ONE DIMENSIONAL FIT
            NETWORK NUMBER      3
            LATTICE INDEX-X     4
            LATTICE INDEX-Y     6


            SUBROUTINE POINT

******** ERROR
          REQUIRED POINT COORDINATE NOT IN CORE
            NETWORK INDEX      7
            LATTICE INDEX-X    8
            LATTICE INDEX-Y    6
              COLUMNS IN CORE
                 4     5     6     7     8


            SUBROUTINE SPLA

******** ERROR
          SINGULAR LEAST SQUARES FIT
            NETWORK WING-TIP
            LATTICE INDEX-X      7
            LATTICE INDEX-Y      3
              DEVIATION FROM UNITY   1.337E+00
******** WARNING
          POOR LEAST SQUARES FIT.
            NETWORK WING-TIP
            LATTICE INDEX-X      7
            LATTICE INDEX-Y      5
              DEVIATION FROM UNITY   5.369E-07


            SUBROUTINE SSP13

******** ERROR
          SINGULAR LEAST SQUARES FIT
            NETWORK LOWER-WING
            LATTICE INDEX-X      2
            LATTICE INDEX-Y      3
              CHISQUARE=  2.471E+10
```

```
******** WARNING
          POOR LEAST SQUARES FIT.
            NETWORK LOWER-WING
            LATTICE INDEX-X      1
            LATTICE INDEX-Y      4
              CHISQUARE=  4.149E+02

              SUBROUTINE WAKGAP

******** ERROR
          ERROR IN CALLING ARGUMENTS
          WAKE SPLINE CALL FOR NON-WAKE NETWORK
            NETWORK PRE-WAKE-WING-EDGE
            DOUBLET TYPE DA

              SUBROUTINE CBLFFM

******** ERROR
          SINGULAR INVERSE FOR SUBPANEL XFM MATRIX
          DUE TO INVALID MACH NUMBER
          ONE MINUS MACH NUMBER SQUARED = 3.791E-16

              SUBROUTINE DBLFFM

******** ERROR
          SINGULAR INVERSE FOR SUBPANEL XFM MATRIX
          DUE TO INVALID MACH NUMBER
          ONE MINUS MACH NUMBER SQUARED = 3.799E-16

              SUBROUTINE PANGEO

******** ERROR
          MACH-INCLINED PANEL DISCOVERED
            NETWORK  UPPER-PLATE
            PANEL COLUMN  3
            PANEL ROW     5
******** WARNING
          CRITICALLY INCLINED PANEL DISCOVERED
            NETWORK  UPPER-PLATE
            PANEL COLUMN  3
            PANEL ROW     6
            ANGLE WITH RESPECT TO MACH CONE = -3.697E-03
******** WARNING
          NON-CONVEX PANEL DISCOVERED
            NETWORK  UPPER-PLATE
            PANEL COLUMN  7
            PANEL ROW     3
******** WARNING
          NEARLY NON-CONVEX PANEL DISCOVERED
           NETWORK
           PANEL COLUMN  7
           PANEL ROW     4
```

```
******** WARNING
         SUBPANEL AREA SET TO ZERO (BY PANGEO), SUBPANEL =  3
          NETWORK RIGHT-TOP-WING
          PANEL COLUMN   2
          PANEL ROW   2

              SUBROUTINE PANSIZ

******** ERROR
          PANEL SIZE VANISHES
          NETWORK  WING
          EDGE  3
          PANEL INDEX ALONG EDGE          6
          (CLOCKWISE DIRECTION)
******** WARNING
          GAP SIZE EXCEEDS PANEL SIZE
          NETWORK  BODY
          EDGE  1
          PANEL INDEX ALONG EDGE          9
            GAPSIZE/PANEL SIZE = 3.691E+00

              SUBROUTINE PANSUB

******** ERROR
          LEAST SQUARES ERROR IN PANEL SUBSPLINE
          NETWORK  BODY
          PANEL COLUMN   9
          PANEL ROW   6
******** WARNING
          POOR LEAST SQUARES FIT IN PANEL SUBSPLINE
            NETWORK  BODY
            PANEL COLUMN   7
            PANEL ROW   6

              SUBROUTINE SPLINM

******** ERROR
          PANEL DEPENDENT ON TOO MANY PARAMETERS
            NETWORK  MID-BODY
            PANEL COLUMN  1
            PANEL ROW  3
            NUMBER OF SOURCE PARAMETERS =  10
            NUMBER OF DOUBLET PARAMETERS =  22
******** ERROR
          SINGULAR SOURCE SUBPANEL SPLINE MATRIX
            NETWORK  MID-BODY
            PANEL COLUMN  1
            PANEL ROW  4
            SUBPANEL NUMBER  5
******** ERROR
          SINGULAR DOUBLET SUBPANEL SPLINE MATRIX
            NETWORK MID-BODY
            PANEL COLUMN  5
            PANEL ROW  8
            SUBPANEL NUMBER  6
```

```
            SUBROUTINE SRCFFM

******** ERROR
          SINGULAR INVERSE FOR MATRIX
          DUE TO INVALID MACH NUMBER.
          ONE MINUS MACH NUMBER SQUARED = 6.425E-18


            SUBROUTINE SUBGEO

******** ERROR
          MACH INCLINED SUBPANEL DISCOVERED
            NETWORK  TAIL
            PANEL COLUMN  9
            PANEL ROW    3
            SUBPANEL INDEX 7
******** WARNING
          CRITICALLY INCLINED SUBPANEL DISCOVERED
            NETWORK  TAIL
            PANEL COLUMN  6
            PANEL ROW    3
            SUBPANEL INDEX 1
            ANGLE WITH RESPECT TO MACH CONE = -3.769E-04
```

APPENDIX 4-F

ADDITIONAL DIAGNOSTIC OUTPUT

During maintenance activities, additional diagnostic output may be desired from DQG. This may be to investigate code errors or to better understand the analysis of a particularly complex configuration by tailoring the output for that configuration. If the DEFINE directive is available with the UPDATE program then DQG can be instrumented with additional output code in an efficient and straightforward manner. The redundancies of adding output code in several routines can be reduced and the original code can also be left unaffected. The general approach is outlined below.

First, all changes to the DQG program library should be surrounded by an IF directive as shown below.

```
*IF DEF,DIAGNOS

(additional output code)

*ENDIF
```

When DQG is updated prior to compilation with *DEFINE DIAGNOS, the output code will be instrumented. If DQG is updated without the DEFINE directive, the compiled code remains the same.

Second, two special COMDECKs are called from every routine in DQG. They are DECLAR and ENDECL. Changes (enclosed by IF-ENDIF directives) to these common blocks will be propagated to every DQG routine that is recompiled. This assumes that the DEFINE directive was used. Only specification statements should be placed in DECLAR. ENDECL may contain any data statements followed by executable statements. The executable statements at the end of ENDECL will be executed immediately upon entry to the subroutine. As an additional aid the name of each subroutine is data loaded into the local variable SUBNAM.

APPENDIX 4-G

SAMPLE OUTPUT FROM DQG

An example and a discussion of the output from DQG is contained in the PAN AIR User's Manual, Section 8 (Reference 2).

APPENDIX 4-H

INDEXING SCHEMES IN DQG

4-H.0 Introduction

Indexing schemes are a basic part of DQG. Data organization and pattern recognition or identification are its essence. Most algorithms in DQG depend upon the availability of one or several indexing systems.

This section describes the important indexing schemes used in DQG.

4-H.1 The Panel.

The basic geometrical unit in PAN AIR is the panel. A panel (shown in Figure 4-H.1) is an arbitrarily shaped quadrilateral. It is defined by its four corner points. These are indexed in a counter clockwise sense (when viewed from above the upper surface) as shown in the figure. Five additional derived points are indexed. They are the four edge mid-points of the panel and the center point of the panel. The set of nine points define eight triangular subpanels. The subpanels are indexed as shown in Figure 4-H.2. Figure 4-H.3 shows the numbering scheme for the points in the subpanel. These indexing schemes are used primarily in the sixth overlay of DQG.

4-H.2 The Network.

Collections of panels make up a network. Networks are defined by a set of rectangularly indexed corner points. Figure 4-H.4 shows the upper surface of a network and the manner in which the previously discussed panel indexing fits into the network.

The location of a corner point in the network is defined by a pair of coarse grid lattice indices (Figure 4-H.5). These are a pair of indices which indicate the position of the point in terms of a two dimensional lattice of points.

Adding edge midpoints and center points to the panels in a network defines the fine grid of points (Figure 4-H.6). These are referenced by the fine grid lattice indices. These are similar to the coarse grid lattice indices. Note that center points have (even, even) lattice indices, column edge midpoints have (even, odd) lattice indices, row edge midpoints have (odd, even) lattice indices, and corner panels have (odd, odd) indices in the fine grid lattice coordinate system. In fact, if (I_C, J_C) are the coarse grid lattice indices of a corner point, the fine grid indices of the point are $(2I_C-1, 2J_C-1)$. On the perimeter of the network, the corner points are also referred to by a sequential point index in a counterclockwise sense. Figure 4-H.7 illustrates the edge indexing. Subroutines LATEDG and EDGLAT are used to transform coarse grid lattice indices to sequential edge indices (LATEDG) and vice versa (EDGLAT). There is also a lattice indexing system for panels. Figure 4-H.8 shows the panel lattice indices of panels in the network. This lattice indexing system is used mostly for internal processing. Error or warning messages sometimes list panel column and panel row as an aid in identifying where in the network the problem has occurred. These column and row indices correspond to the panel lattice indexing coordinate system.

4-H.3 Control Points.

Control points are located at every panel center point, at every edge midpoint on a network edge, and at every corner point on a network edge which is either a start point or an end point of an abutment. These last points include at least the four network corner points.

Starting with the first network in the processing sequence, the control points at panel centers are indexed first. Then the first corner point on the first edge is assigned an index followed by an assignment to every edge midpoint on the edge. This proceeds around the network in a counterclockwise direction. After all networks have been processed, any additional control points which were added because an abutment began or ended in the middle of an edge receive an index. Figure 4-H.9 illustrates the indexing scheme.

4-H.4 Singularity Parameters.

Singularity parameters (λ^S, λ^D) are located at different places in a network depending on the source and doublet type of the network. The scheme used for assigning a global index to singularity parameters follows the general scheme of the control point indexing (see Appendix 4-H, Section 4-H.3). The varying locations of singularity parameters introduces some complications. See PAN AIR Theory Document, Section D.1 (Reference 1).

The process of indexing singularity parameters is synonomous with creating the SINGULARITY-MAP and SINGULARITY-SPEC datasets in the DQG database. These datasets (see Appendix 4-D) contain information about where in the network the singularity lies, whether it is a source or a doublet parameter and whether it is a known singularity.

The general approach is to loop over panels and assign an index first to a source parameter (if any) and then to a doublet parameter (if any). Any singularity parameters that are on an edge of the network are not indexed at this time. After the loop on panels ends, singularity parameters on the edges are indexed. First doublet parameters on the four edges are indexed in a counter clockwise sense. Then source parameters are indexed.

In Figure 4-H.14 it is clear that two singularity parameters are assigned to each center point in the network since there are two indices associated with each center point in the network. There is only one singularity parameter located at the edge midpoints on the perimeter of the network and only one parameter at each of the four network corner points. By examining Table 4-H.1 we can see that singularity parameters 1 and 2 are source and doublet parameters respectively, which (from Figure 4-H.14) are located at the center point of the panel in the lower left corner of the network.

Figures 4-H.10 to 4-H.28 illustrate the indexing scheme for all combinations of networks. (Since singularities locations for Doublet Forward Weighted networks are identical to those of Doublet Analysis networks separate figures are not given for doublet forward weighted networks.) Tables 4-H.1 to 4-H.12 label the indices as source or doublet for the dual networks in Figures 4-H.14 to 4-H.28.

4-H.5 Some Useful Conversions.

Several different indexing schemes can be employed to describe the same quantity. Often a need arises to convert from one indexing system to another. This section provides a list of a number of algorithms which define these conversions:

Coarse grid lattice indices denoted by (I_c, J_c) to fine grid lattice indices

$(I_c, J_c) \rightarrow (2I_c-1, 2J_c-1)$

Panel lattice, panel point to fine grid lattice indices

$(I_p, J_p), N_p \rightarrow (2I_p-1, 2J_p-1) + (I_x(N_p), I_y(N_p))$

where the panel index and the corresponding lattice index within the panel $(I_x(N_p), I_y(N_p))$ may have the following values

N_p	I_x	I_y	N_p	I_x	I_y
1	0	0	5	1	0
2	2	0	6	2	1
3	2	2	7	1	2
4	0	2	8	0	1
			9	1	1

Coarse grid indices to sequential counter clockwise edge index (Refer to Figures 4-H.5 and 4-H.7)

$(I_c, J_c) \rightarrow$	I_c	Edge 1
	J_c	Edge 2
	N_c-I_c+1	Edge 3
	N_R-J_c+1	Edge 4

where N_c = number of corner point columns and
N_R = number of corner point rows.

Table 4-H.1 Source/Doublet Parameters for Source Analysis/Doublet Analysis Network (Figure 4-H.14)

Index	S/D	Index	S/D	Index	S/D
1	S	36	D	71	D
2	D	37	S	72	D
3	S	38	D	73	D
4	D	39	S	74	D
5	S	40	D	75	D
6	D	41	S	76	D
7	S	42	D	77	D
8	D	43	S	78	D
9	S	44	D	79	D
10	D	45	S	80	D
11	S	46	D	81	D
12	D	47	S	82	D
13	S	48	D		
14	D	49	S		
15	S	50	D		
16	D	51	S		
17	S	52	D		
18	D	53	S		
19	S	54	D		
20	D	55	S		
21	S	56	D		
22	D	57	D		
23	S	58	D		
24	D	59	D		
25	S	60	D		
26	D	61	D		
27	S	62	D		
28	D	63	D		
29	S	64	D		
30	D	65	D		
31	S	66	D		
32	D	67	D		
33	S	68	D		
34	D	69	D		
35	S	70	D		

Table 4-H.2 Source/Doublet Parameters for
Source Design I/Doublet Analysis Network (Figure 4-H.15)

Index	S/D	Index	S/D	Index	S/D
1	D	36	S	71	D
2	D	37	D	72	D
3	D	38	S	73	S
4	D	39	D	74	S
5	D	40	D	75	S
6	S	41	S	76	S
7	D	42	D	77	S
8	S	43	S	78	S
9	D	44	D	79	S
10	S	45	S	80	S
11	D	46	D	81	S
12	D	47	D	82	S
13	S	48	D	83	S
14	D	49	D	84	S
15	S	50	D	85	S
16	D	51	D	86	S
17	S	52	D	87	S
18	D	53	D	88	S
19	D	54	D	89	S
20	S	55	D	90	S
21	D	56	D	91	S
22	S	57	D	92	S
23	D	58	D	93	S
24	S	59	D	94	S
25	D	60	D		
26	D	61	D		
27	S	62	D		
28	D	63	D		
29	S	64	D		
30	D	65	D		
31	S	66	D		
32	D	67	D		
33	D	68	D		
34	S	69	D		
35	D	70	D		

Table 4-H.3 Source/Doublet Parameters for
Source Analysis/Doublet Design I Network (Figure 4-H.16)

Index	S/D	Index	S/D
1	S	36	S
2	S	37	D
3	S	38	S
4	S	39	D
5	S	40	S
6	S	41	S
7	D	42	D
8	S	43	S
9	D	44	D
10	S	45	S
11	D	46	D
12	S	47	D
13	S	48	D
14	D	49	D
15	S	50	D
16	D	51	D
17	S	52	D
18	A	53	D
19	S	54	D
20	S	55	D
21	D	56	D
22	S	57	D
23	D	58	D
24	S	59	D
25	D	60	D
26	S	61	D
27	S	62	D
28	D	63	D
29	S	64	D
30	D	65	D
31	S	66	D
32	D	67	D
33	S	68	D
34	S	69	D
35	D	70	D

Table 4-H.4 Source/Doublet Parameters for Source Design I/Doublet Design I Network (Figure 4-H.17)

Index	S/D	Index	S/D	Index	S/D
1	S	36	D	71	S
2	D	37	D	72	S
3	S	38	D	73	S
4	D	39	D	74	S
5	S	40	D	75	S
6	D	41	D	76	S
7	S	42	D	77	S
8	D	43	D	78	S
9	S	44	D	79	S
10	D	45	D	80	S
11	S	46	D	81	S
12	D	47	D	82	S
13	S	48	D		
14	D	49	D		
15	S	50	D		
16	D	51	D		
17	S	52	D		
18	D	53	D		
19	S	54	D		
20	D	55	D		
21	S	56	D		
22	D	57	D		
23	S	58	D		
24	D	59	D		
25	S	60	D		
26	D	61	S		
27	S	62	S		
28	D	63	S		
29	S	64	S		
30	D	65	S		
31	S	66	S		
32	D	67	S		
33	S	68	S		
34	D	69	S		
35	S	70	S		

Table 4-H.5 Source/Doublet Parameters for
Source Analysis/Doublet Wake I Network (Figure 4-H.20)

Index	S/D	Index	S/D
1	S	36	D
2	S	37	D
3	S		
4	S		
5	S		
6	S		
7	S		
8	S		
9	S		
10	S		
11	S		
12	S		
13	S		
14	S		
15	S		
16	S		
17	S		
18	S		
19	S		
20	S		
21	S		
22	S		
23	S		
24	S		
25	S		
26	S		
27	S		
28	S		
29	D		
30	D		
31	D		
32	D		
33	D		
34	D		
35	D		

Table 4-H.6 Source/Doublet Parameters for Source Analysis/Doublet Wake II Network (Figure 4-H.21)

Index	S/D
1	S
2	S
3	S
4	S
5	S
6	S
7	S
8	S
9	S
10	S
11	S
12	S
13	S
14	S
15	S
16	S
17	S
18	S
19	S
20	S
21	S
22	S
23	S
24	S
25	S
26	S
27	S
28	S
29	D

Table 4-H.7 Source/Doublet Parameters for Source Design I/Doublet Wake I Network (Figure 4-H.22)

Index	S/D	Index	S/D
1	S	36	S
2	S	37	S
3	S	38	S
4	S	39	S
5	S	40	S
6	S	41	S
7	S	42	S
8	S	43	S
9	S	44	S
10	S	45	S
11	S	46	S
12	S		
13	S		
14	S		
15	S		
16	S		
17	S		
18	S		
19	D		
20	D		
21	D		
22	D		
23	D		
24	D		
25	S		
26	S		
27	S		
28	S		
29	S		
30	S		
31	S		
32	S		
33	S		
34	S		
35	S		

Table 4-H.8 Source/Doublet Parameters for
Source Design I/Doublet Wake II Network (Figure 4-H.23)

Index	S/D	Index	S/D
1	S	36	S
2	S	37	S
3	S	38	S
4	S	39	S
5	S	40	S
6	S	41	S
7	S		
8	S		
9	S		
10	S		
11	S		
12	S		
13	S		
14	S		
15	S		
16	S		
17	S		
18	S		
19	D		
20	S		
21	S		
22	S		
23	S		
24	S		
25	S		
26	S		
27	S		
28	S		
29	S		
30	S		
31	S		
32	S		
33	S		
34	S		
35	S		

Table 4-H.9 Source/Doublet Parameters for Source Design II/Doublet Analysis (Figure 4-H.25)

Index	S/D	Index	S/D	Index	S/D
1	D	36	D	71	D
2	S	37	S	72	D
3	D	38	D	73	D
4	S	39	S	74	D
5	D	40	D	75	D
6	S	41	S	76	S
7	D	42	D	77	S
8	D	43	D	78	S
9	S	44	S	79	S
10	D	45	D	80	S
11	S	46	S	81	S
12	D	47	D	82	S
13	S	48	S	83	S
14	D	49	D	84	S
15	D	50	D	85	S
16	S	51	D	86	S
17	D	52	D	87	S
18	S	53	D	88	S
19	D	54	D	89	S
20	S	55	D		
21	D	56	D		
22	D	57	D		
23	S	58	D		
24	D	59	D		
25	S	60	D		
26	D	61	D		
27	S	62	D		
28	D	63	D		
29	D	64	D		
30	S	65	D		
31	D	66	D		
32	S	67	D		
33	D	68	D		
34	S	69	D		
35	D	70	D		

Table 4-H.10 Source/Doublet Parameter for Source Design II/Doublet Design I Network (Figure 4-H.26)

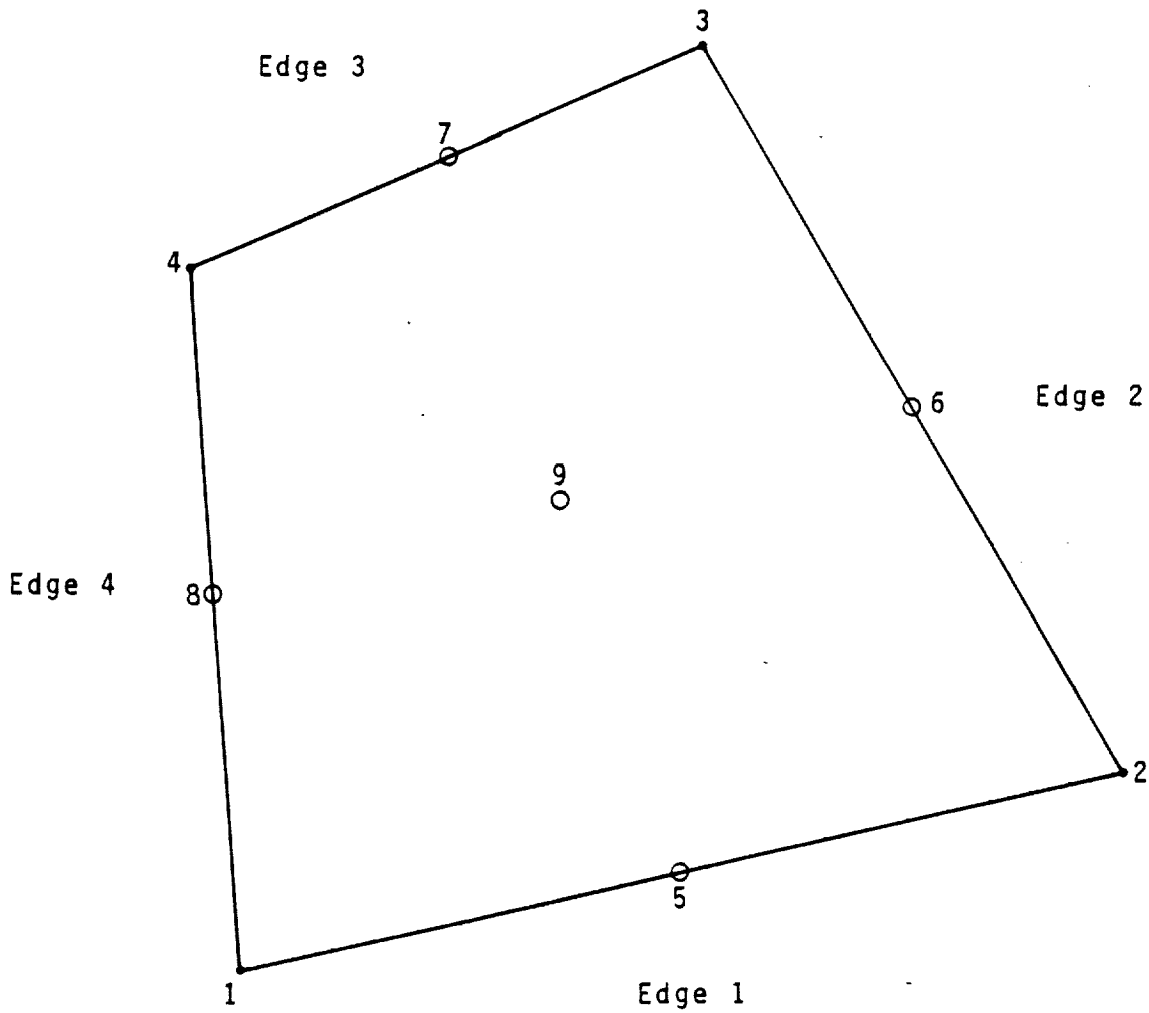
Index	S/D	Index	S/D	Index	S/D
1	S	36	S	71	S
2	S	37	D	72	S
3	S	38	S	73	S
4	S	39	D	74	S
5	D	40	D	75	S
6	S	41	D	76	S
7	D	42	D	77	S
8	S	43	D		
9	D	44	D		
10	S	45	D		
11	D	46	D		
12	S	47	D		
13	D	48	D		
14	S	49	D		
15	D	50	D		
16	S	51	D		
17	D	52	D		
18	S	53	D		
19	D	54	D		
20	S	55	D		
21	D	56	D		
22	S	57	D		
23	D	58	D		
24	S	59	D		
25	D	60	D		
26	S	61	D		
27	D	62	D		
28	S	63	D		
29	D	64	S		
30	S	65	S		
31	D	66	S		
32	S	67	S		
33	D	68	S		
34	S	69	S		
35	D	70	S		

Table 4-H.11 Source/Doublet Parameters for Source Design II/Doublet Wake I (Figure 4-H.27)

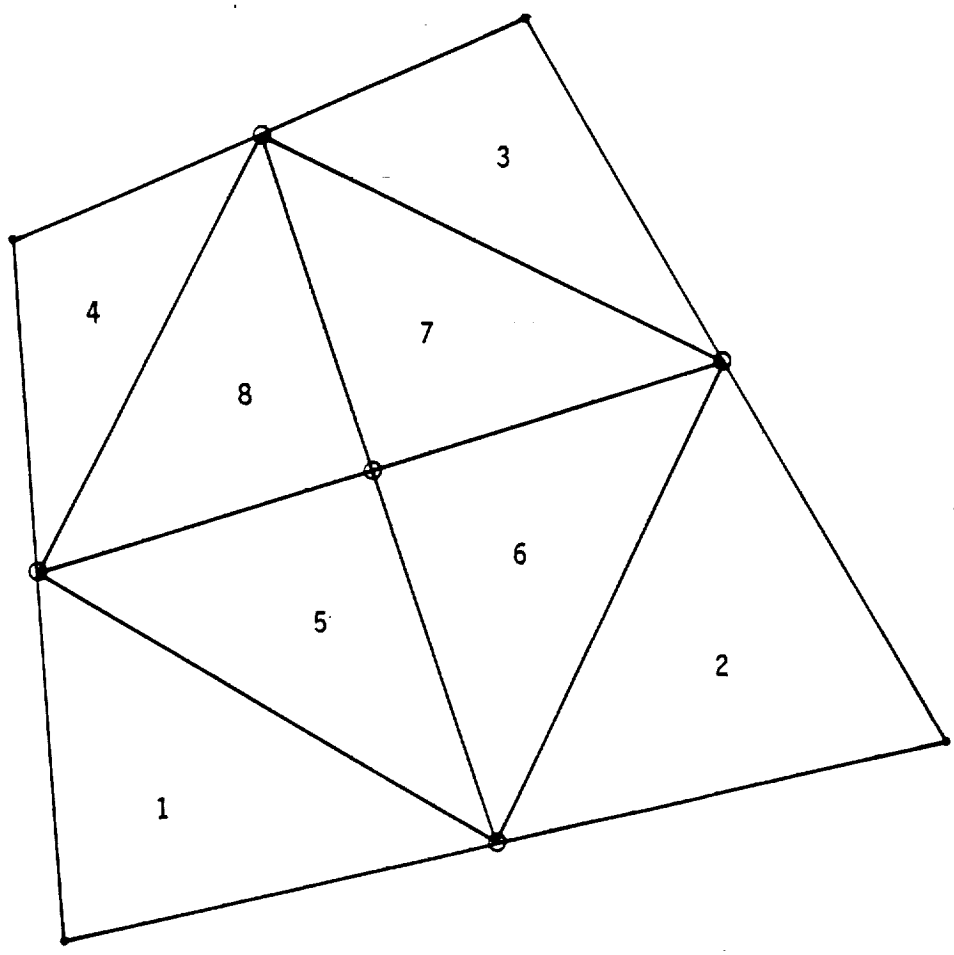
Index	S/D	Index	S/D
1	S	36	S
2	S	37	S
3	S	38	S
4	S		
5	S		
6	S		
7	S		
8	S		
9	S		
10	S		
11	S		
12	S		
13	S		
14	S		
15	S		
16	S		
17	S		
18	S		
19	S		
20	S		
21	S		
22	S		
23	S		
24	S		
25	D		
26	D		
27	D		
28	D		
29	D		
30	D		
31	S		
32	S		
33	S		
34	S		
35	S		

Table 4-H.12 Source/Doublet Parameters for Source Design II/Doublet Wake II (Figure 4-H.28)

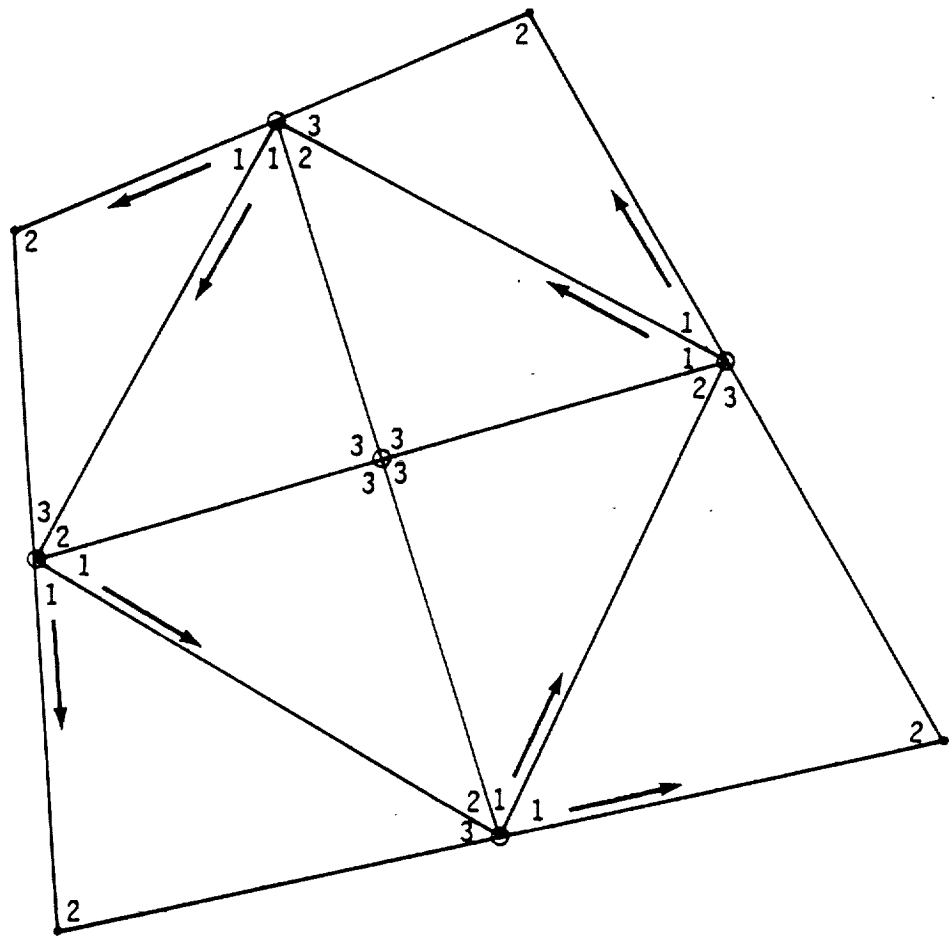
Index	S/D
1	S
2	S
3	S
4	S
5	S
6	S
7	S
8	S
9	S
10	S
11	S
12	S
13	S
14	S
15	S
16	S
17	S
18	S
19	S
20	S
21	S
22	S
23	S
24	S
25	D
26	S
27	S
28	S
29	S
30	S
31	S
32	S
33	S



4-H.1 - The Panel



4-H.2 - The Subpanels



4-H.3 - Indexing of Subpanel Points

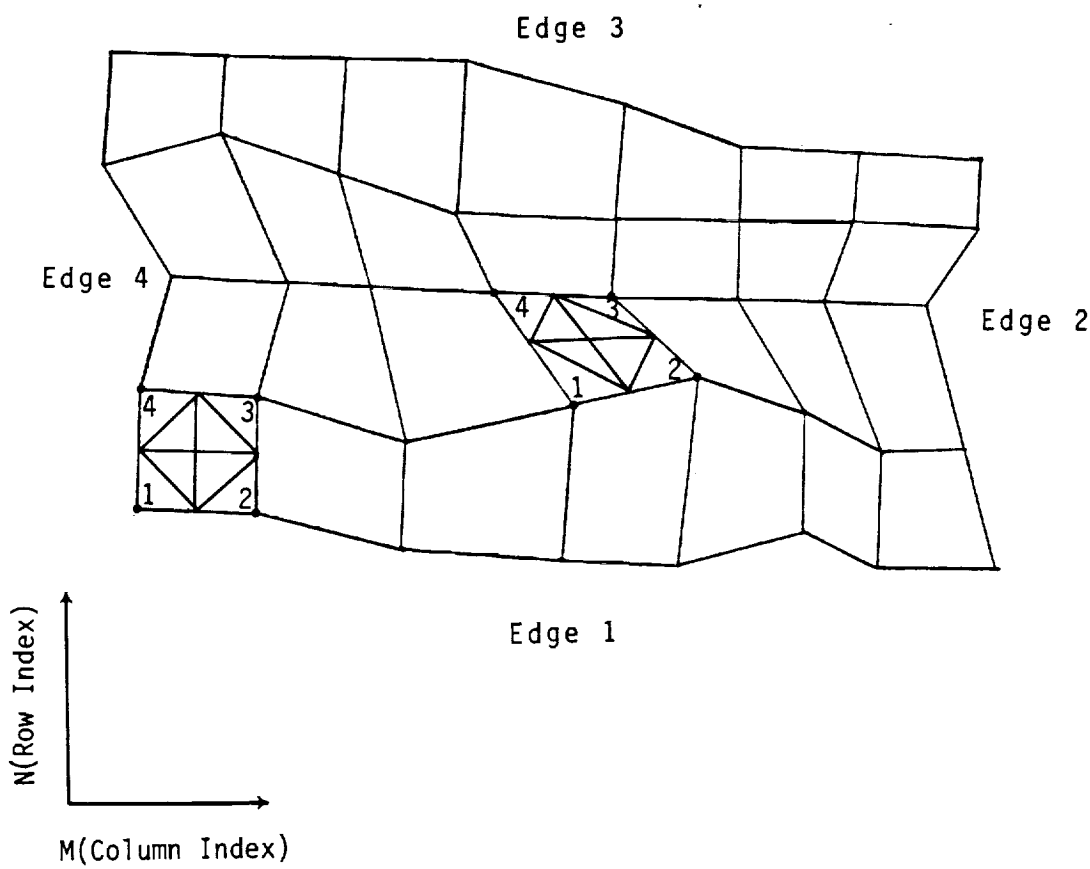


Figure 4-H.4 - A Network

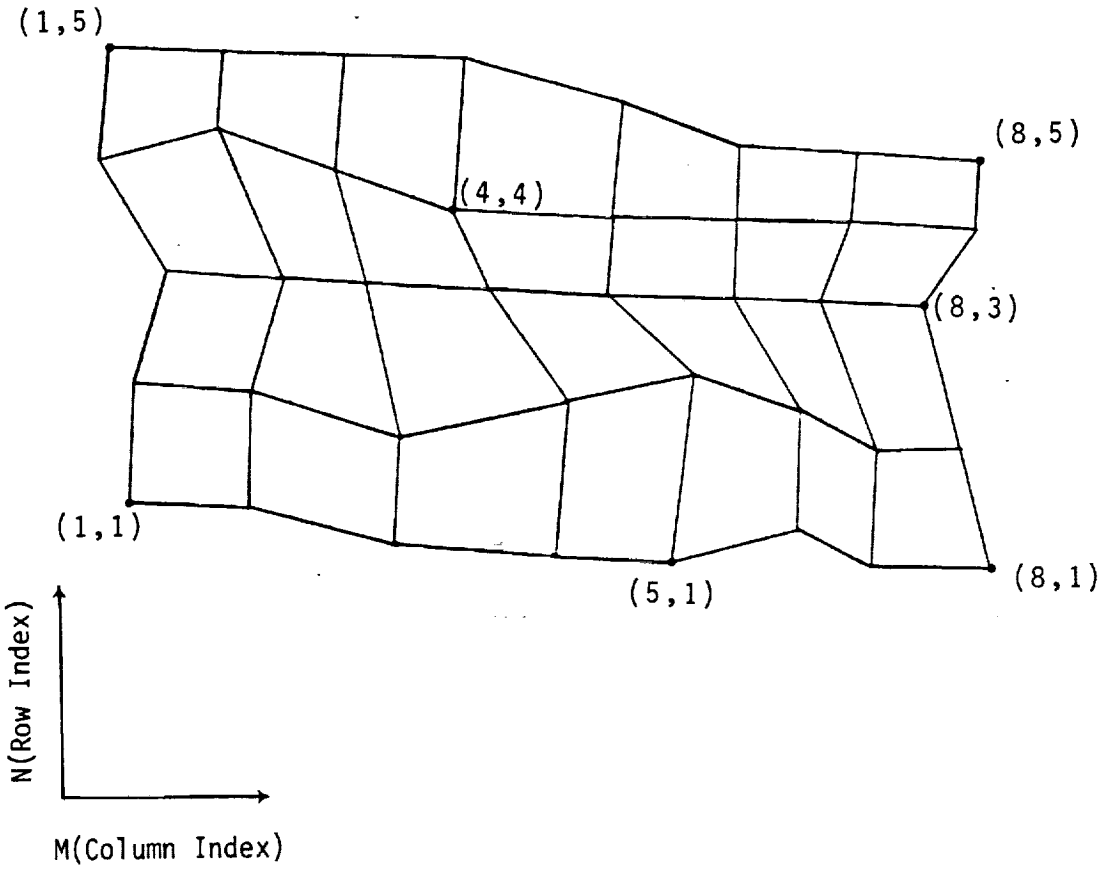


Figure 4-H.5 - Coarse Grid Lattice Indices (M,N)

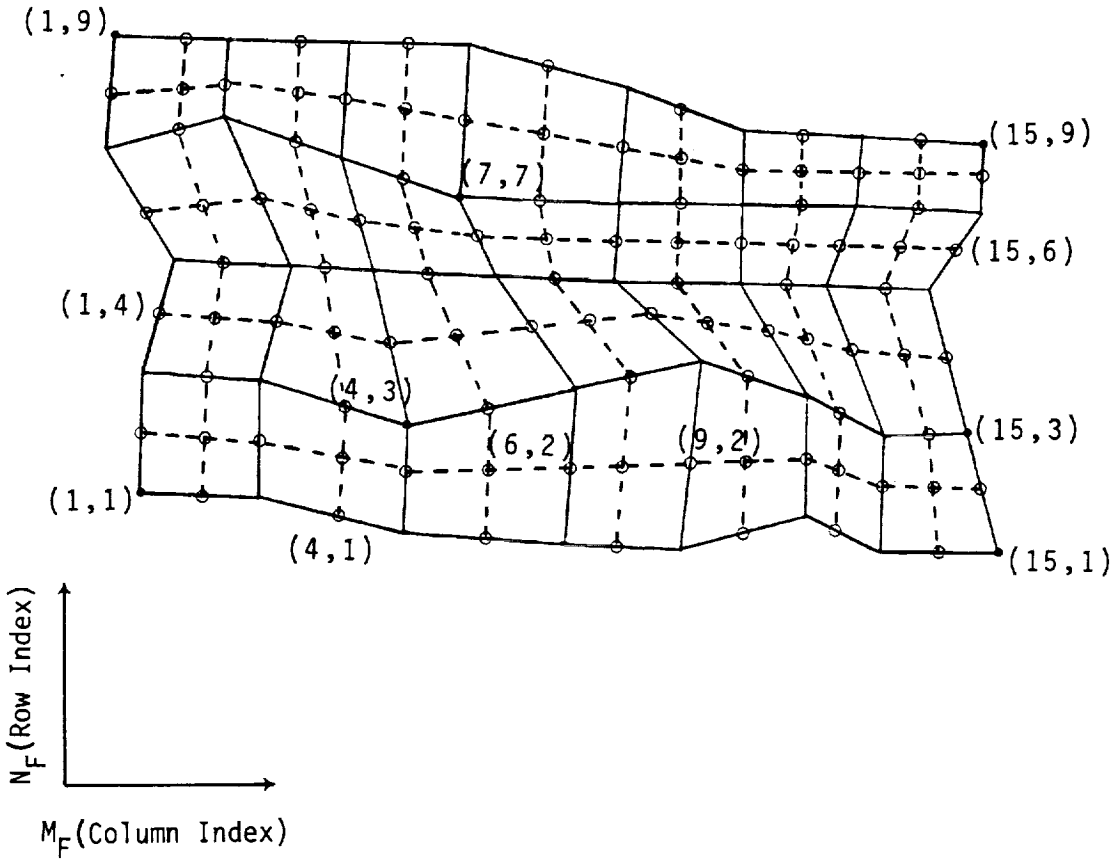


Figure 4-H.6 - Fine Grid Lattice Indices ($M_F N_F$)

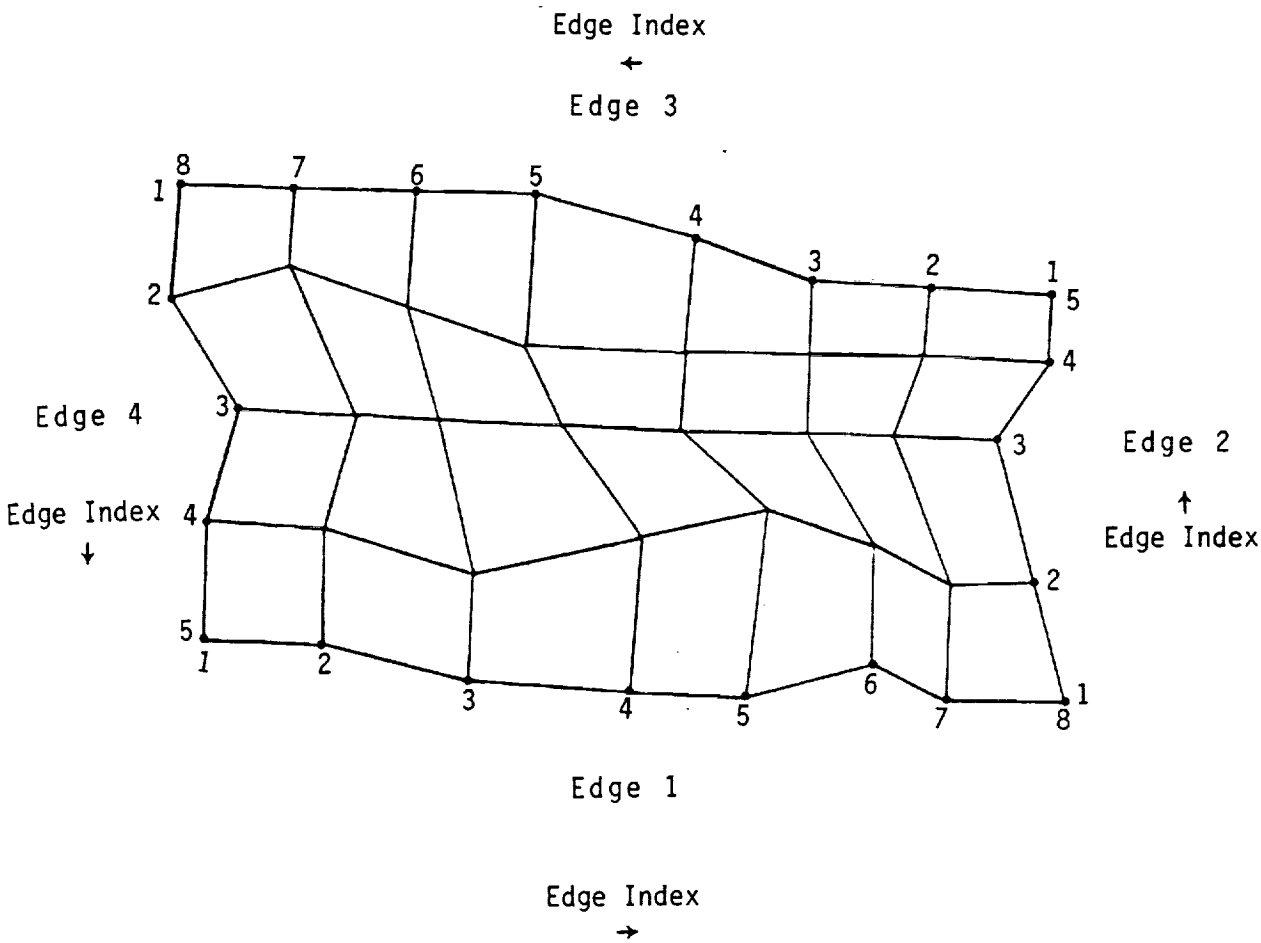


Figure 4-H.7 - Indexing at Edge Points

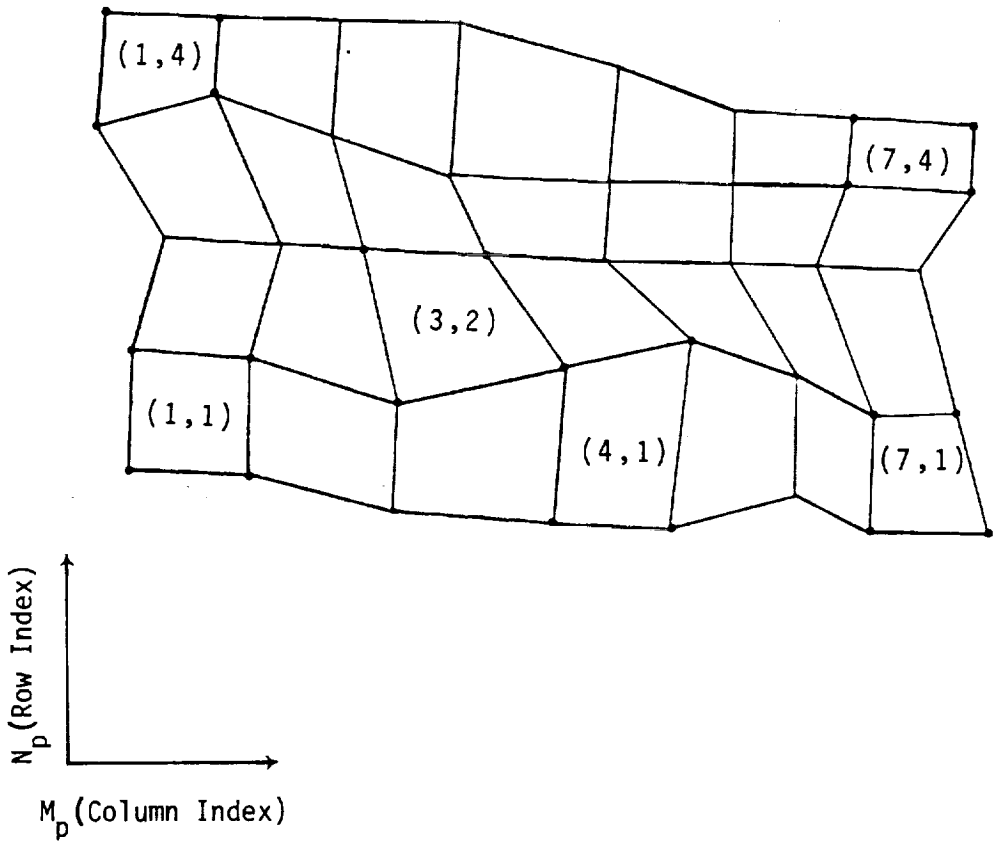


Figure 4-H.8 - Panel Lattice Indices (M_pN_p)

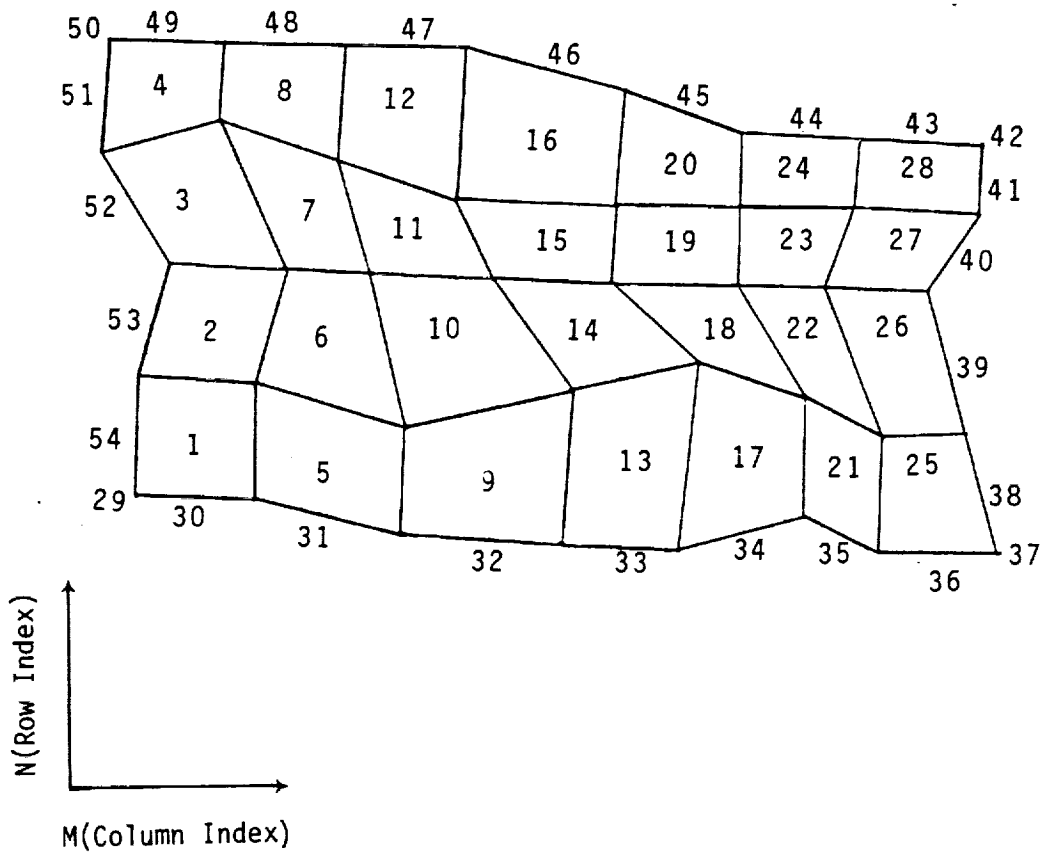


Figure 4-H.9 - Control Point Indexing

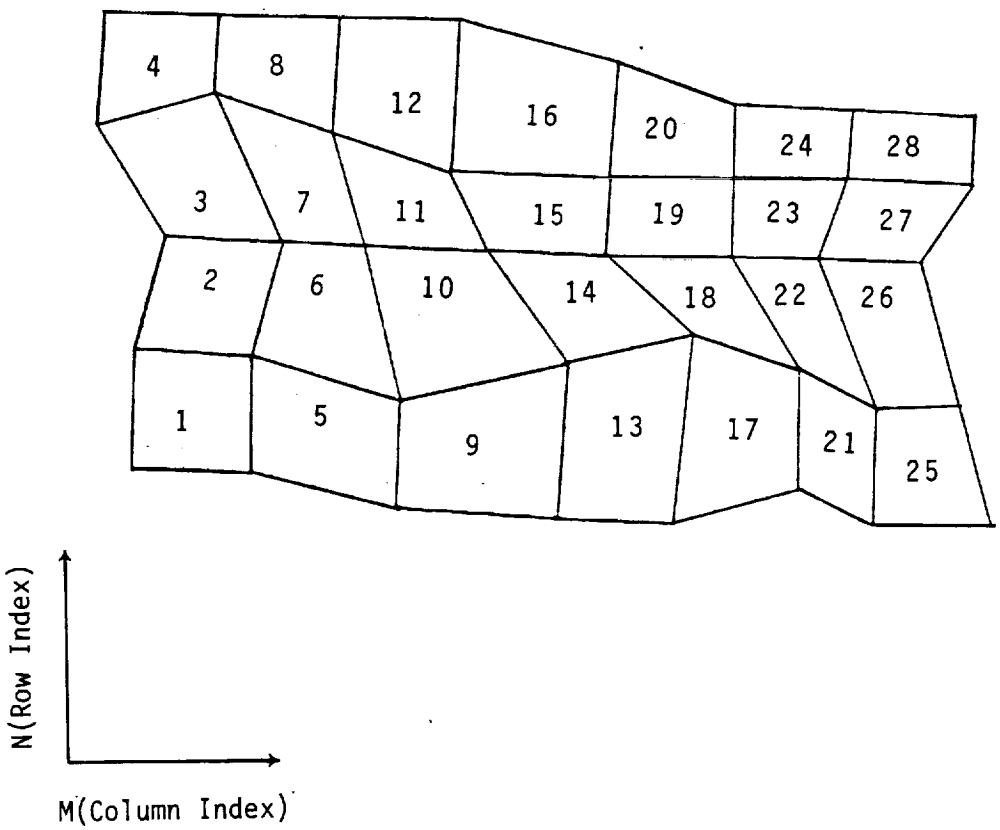


Figure 4-H.10 - Indexing of Singularity Parameters λ_i^s on a Source Analysis, Doublet-Null Networks

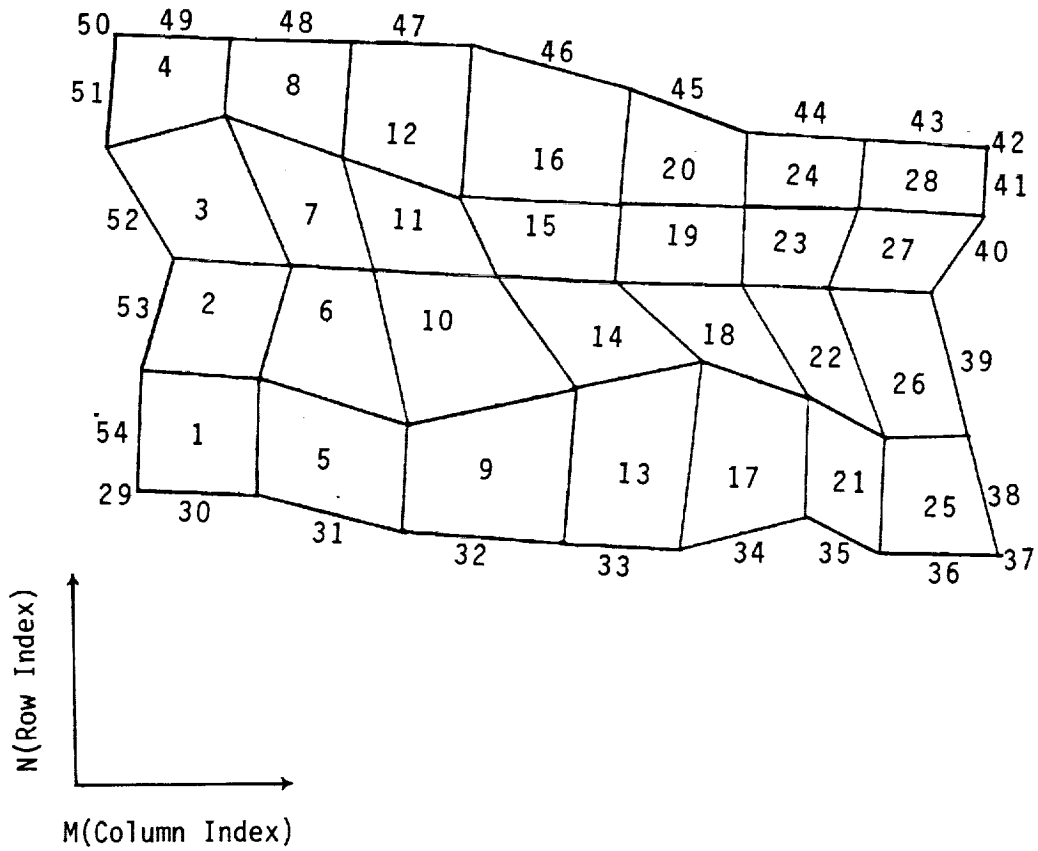


Figure 4-H.11 - Indexing of Singularity Parameters λ_i^D on Source Null, Doublet Analysis Networks

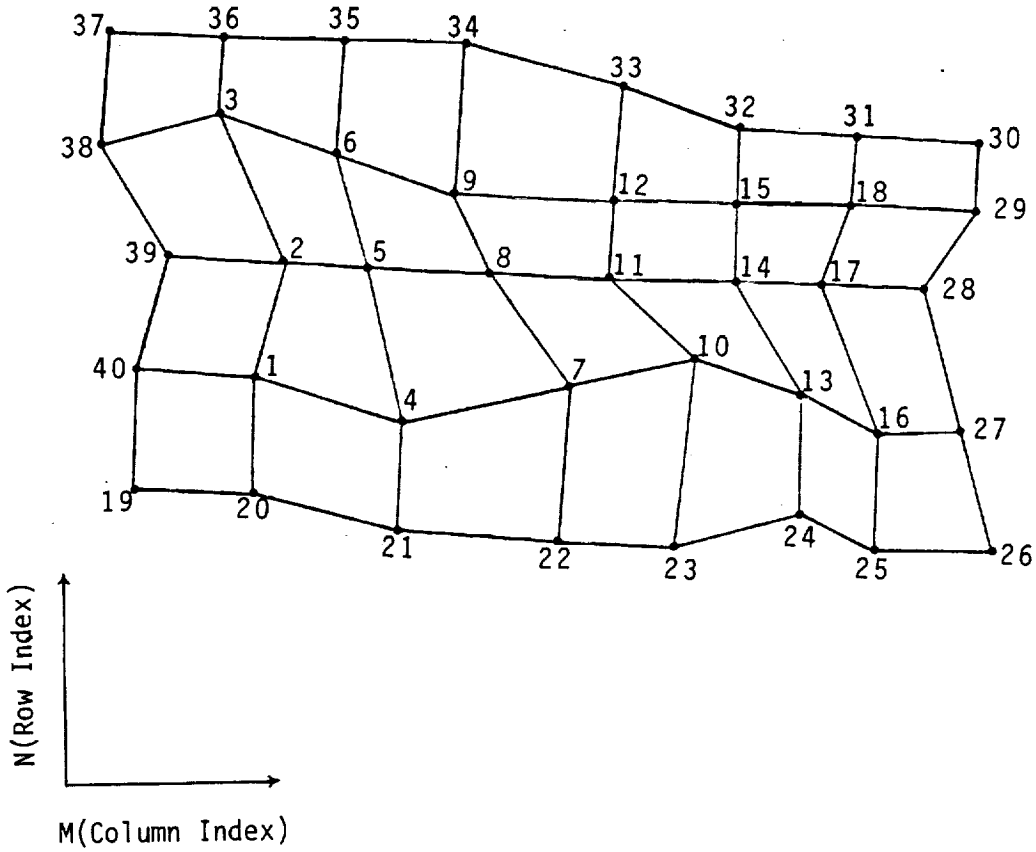


Figure 4-H.12 - Indexing of Singularity Parameters λ_i^s on Source Design I, Doublet Null Network

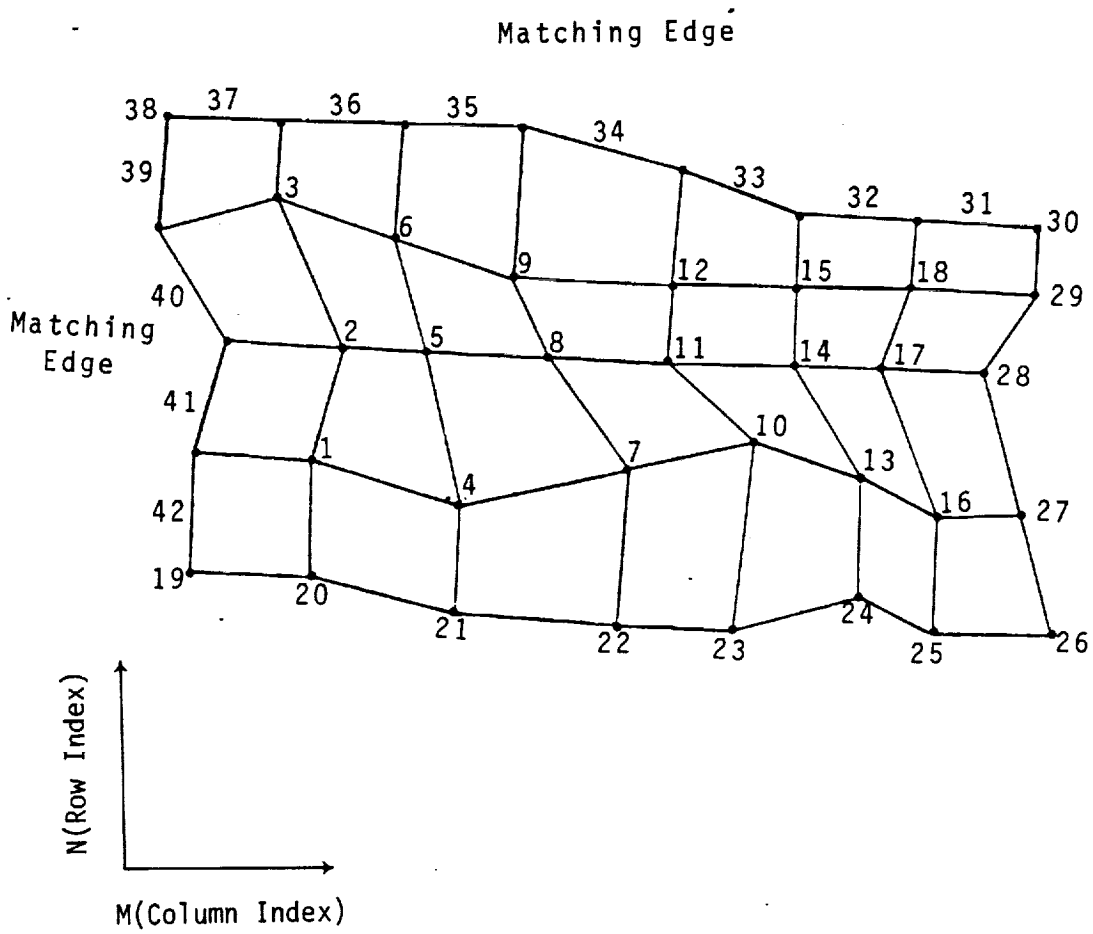


Figure 4-H.13 - Indexing of Singularity Parameters λ_i^D for a Source Null, Doublet Design I Network

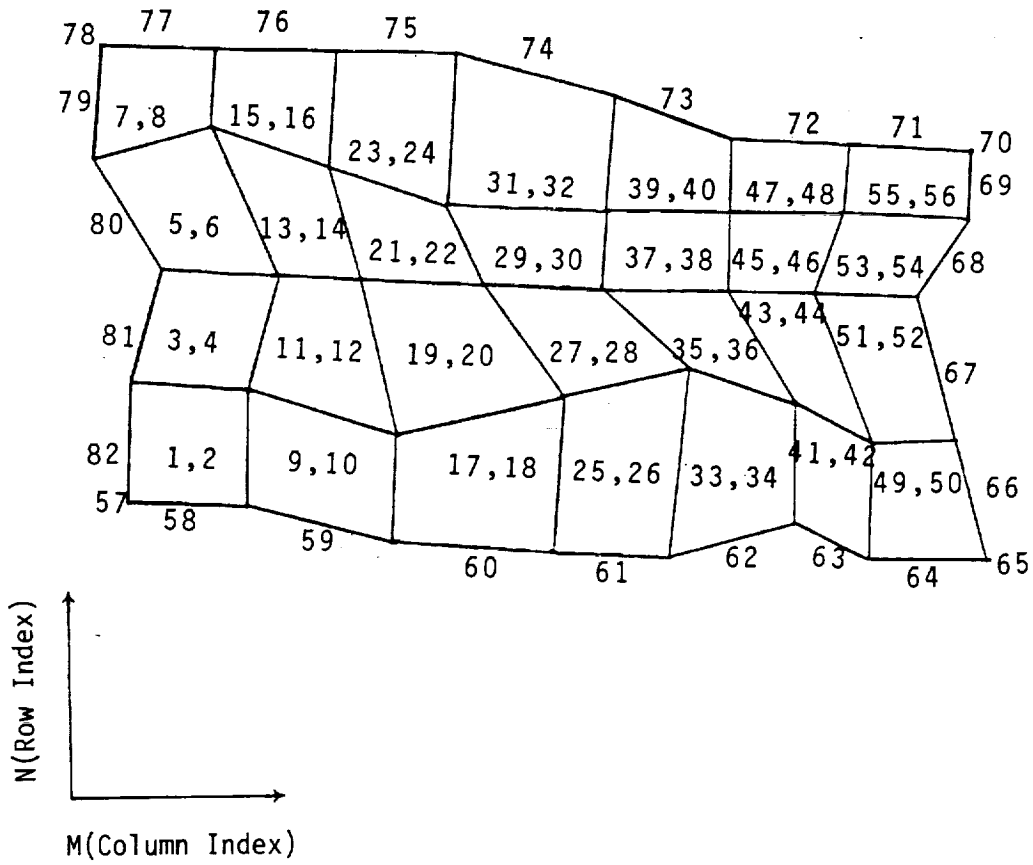


Figure 4-H.14 Indexing of Singularity Parameters λ_i^S and λ_i^D on a Source Analysis Doublet Analysis Network. Odd Indices Between 1 and 55 are Source Parameters. Even Indices Between 2 and 56 and all Indices Between 57 and 82 are Doublet Parameters.

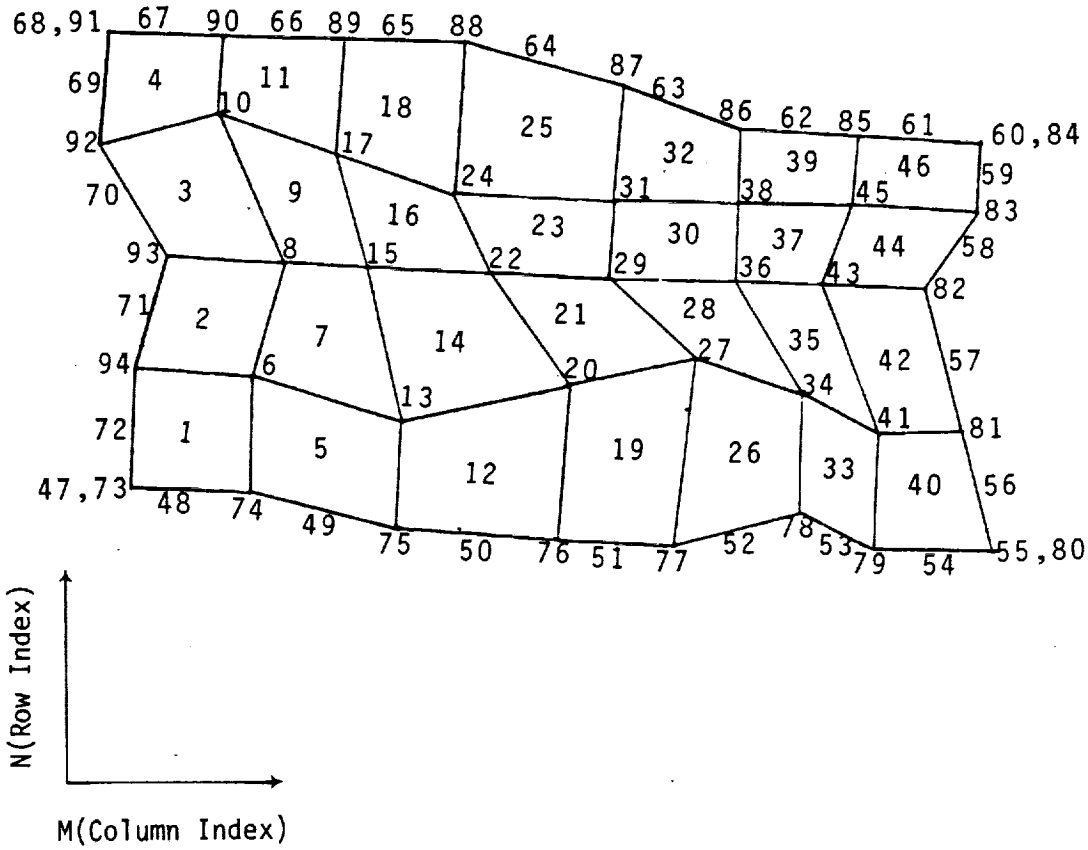


Figure 4-H.15 - Indexing of Singularity Parameters λ_i^s and λ_i^D on a Source Design I, Doublet Analysis Network

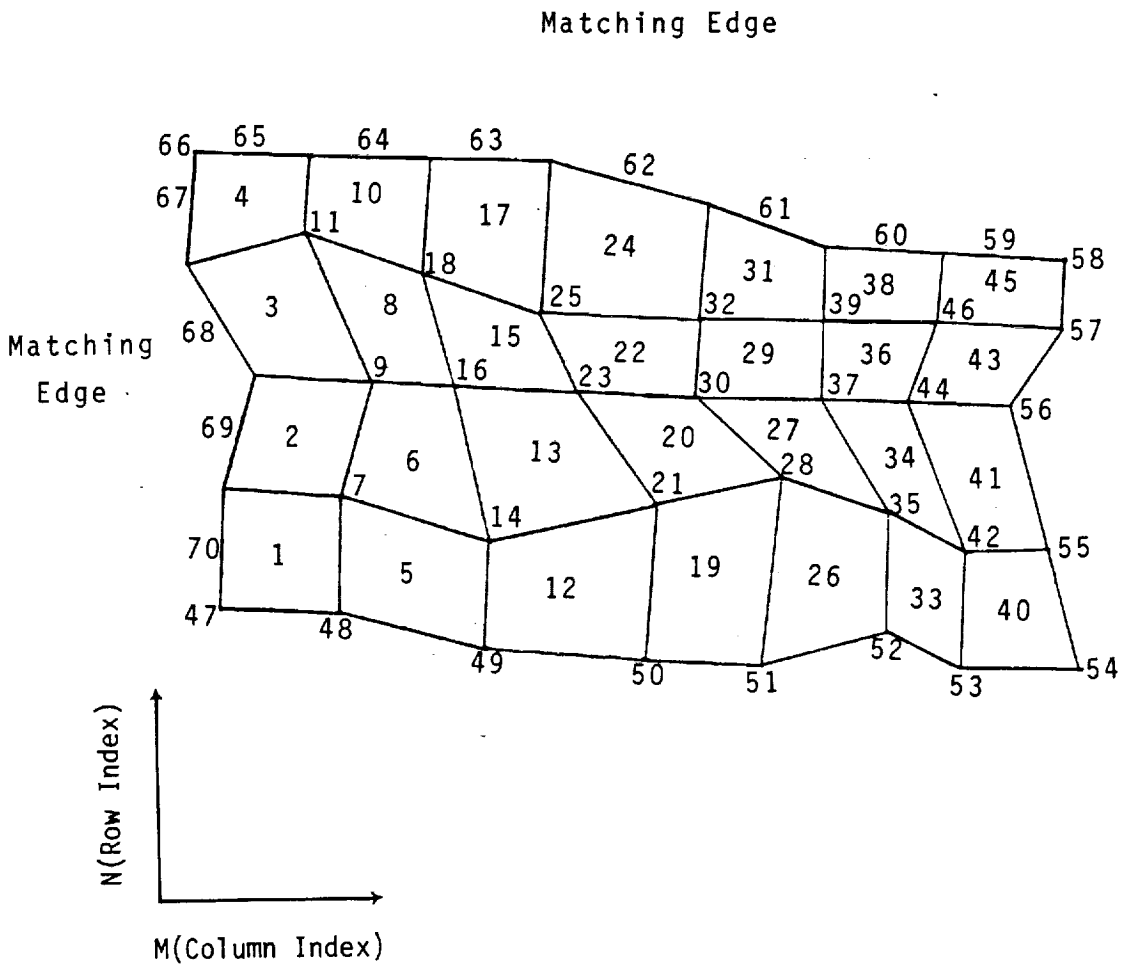


Figure 4-H.16 Indexing of Singularity Parameters λ_i^S and λ_i^D for Source Analysis Doublet Design I Networks

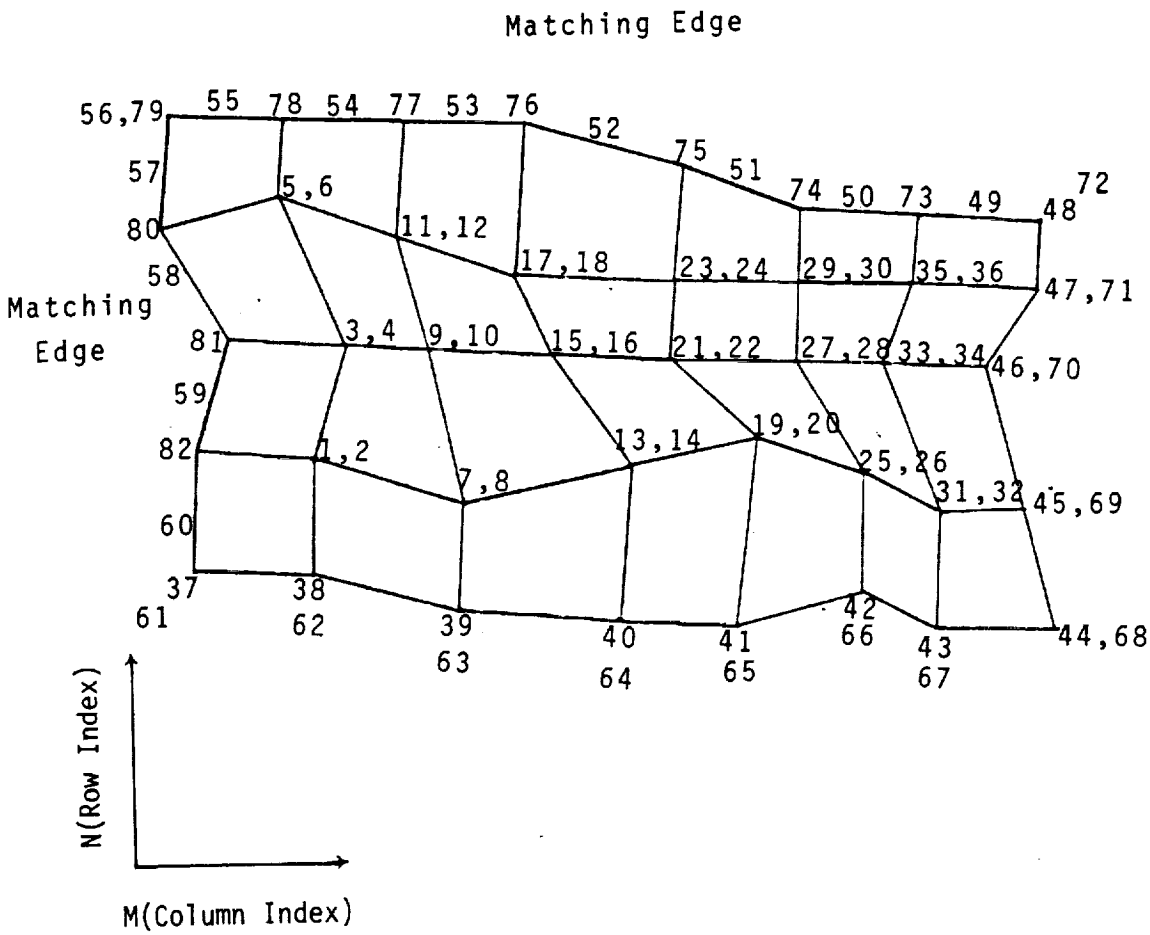


Figure 4-H.17 Indexing of Singularity Parameters λ_i^S and λ_i^D for Source Design I, Doublet Design I Networks

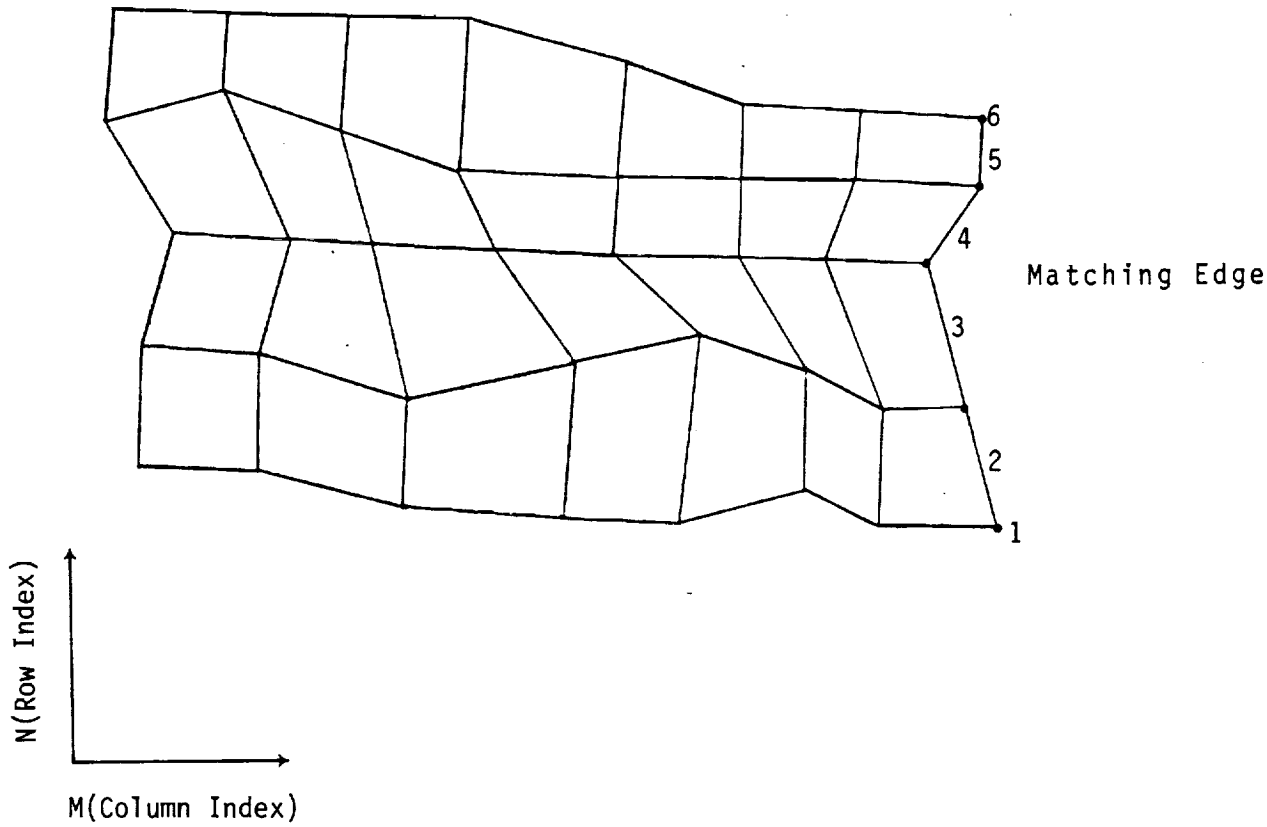


Figure 4-H.18 Indexing of Singularity Parameter λ_i^D for Source Null, Doublet Wake I Network

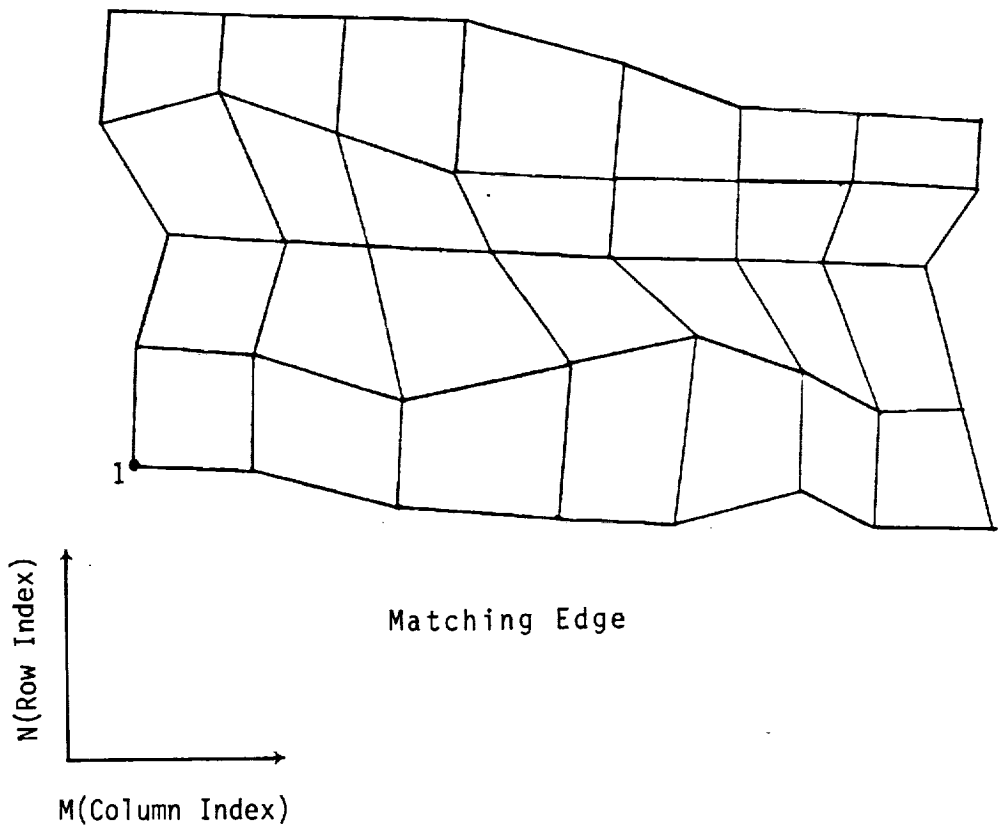


Figure 4-H.19 Indexing of Singularity Parameters λ_i^D for Source Null, Doublet Wake II Networks

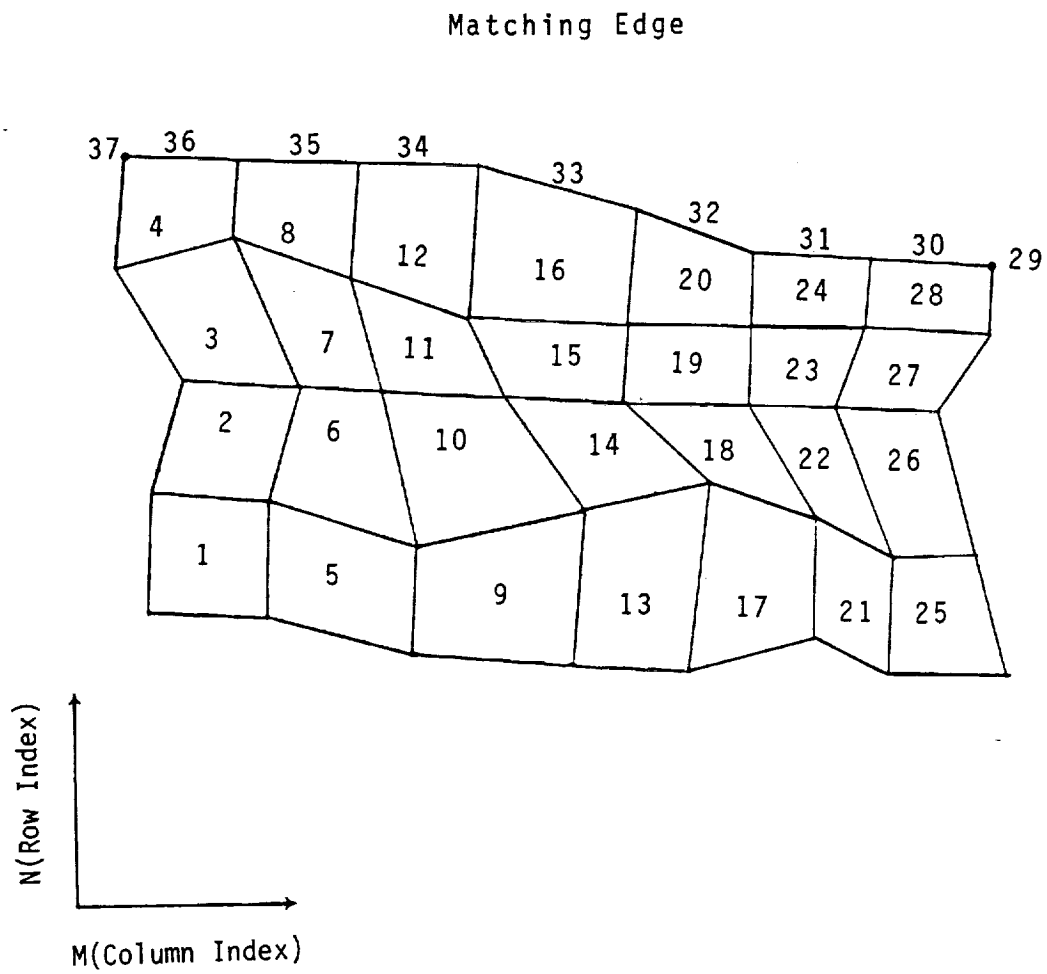


Figure 4-H.20 Indexing of Singularity Parameters λ_i^S and λ_i^D for Source Analysis, Doublet Wake I Network

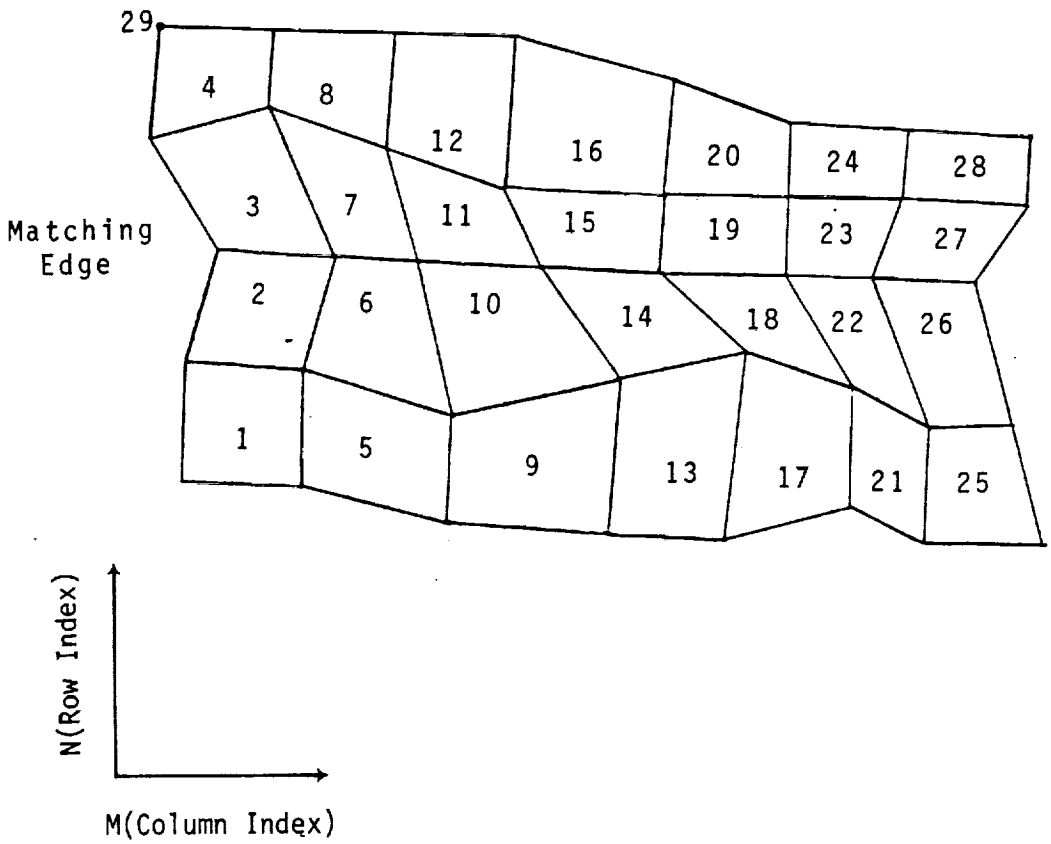


Figure 4-H.21 Indexing of Singularity Parameters λ_i^S and λ_i^D for Source Analysis, Doublet Wake II Network

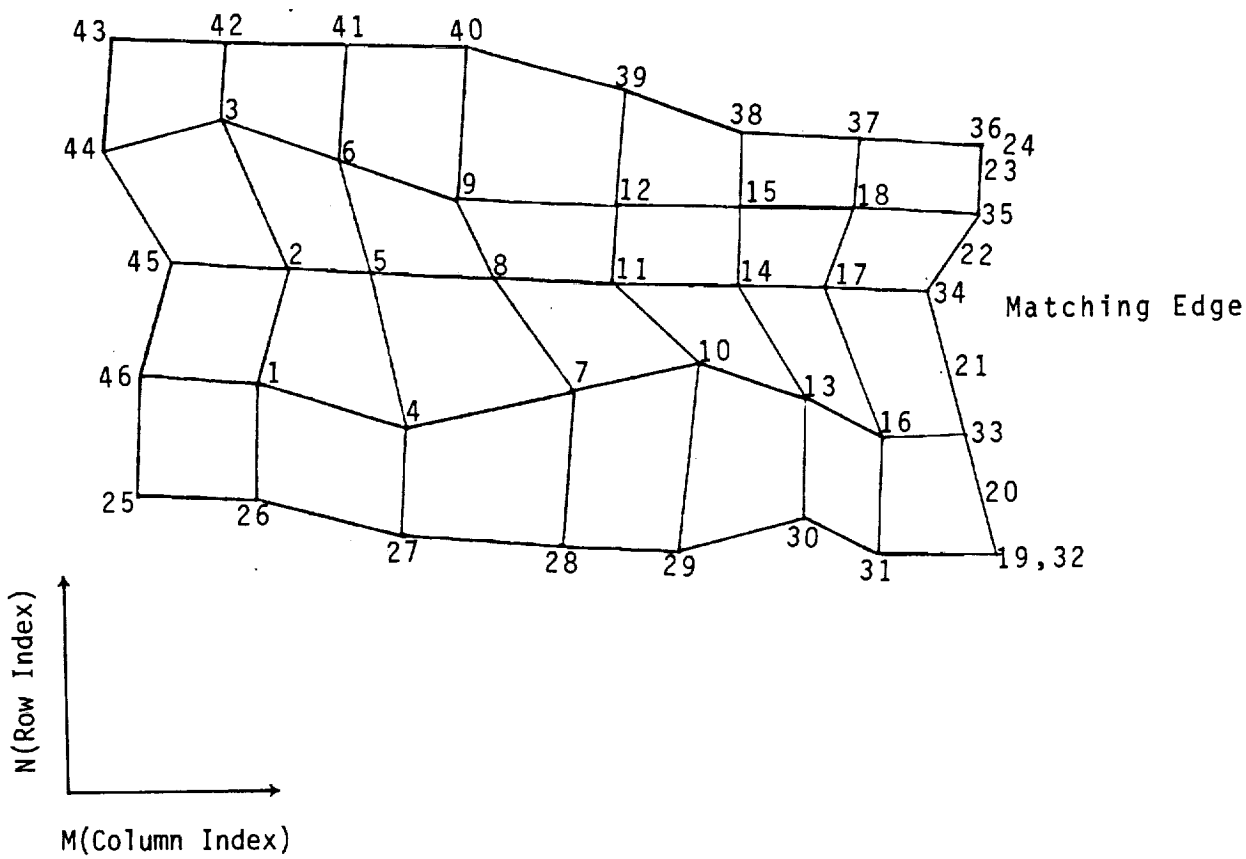


Figure 4-H.22 Indexing Singularity Parameters λ_i^S and λ_i^D for Source Design I, Doublet Wake I Network

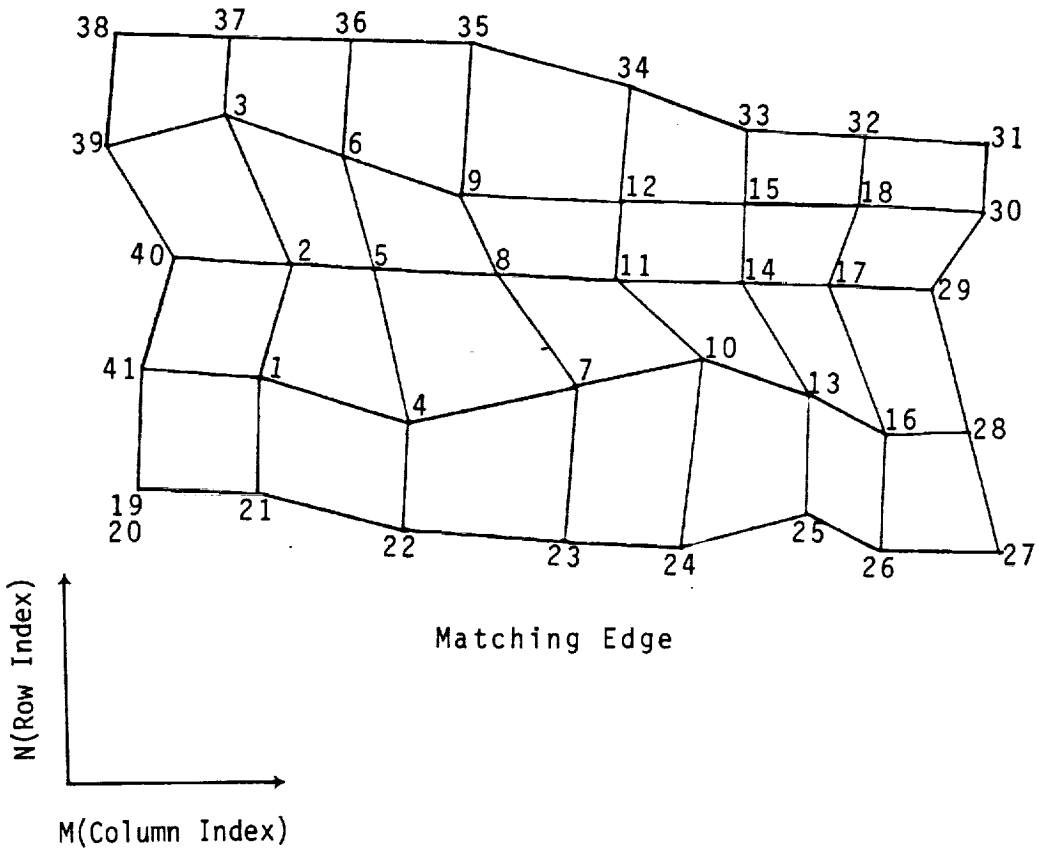


Figure 4-H.23 Indexing of Singularity Parameters λ_i^S and λ_i^D for Source Design I and Doublet Wake II Networks

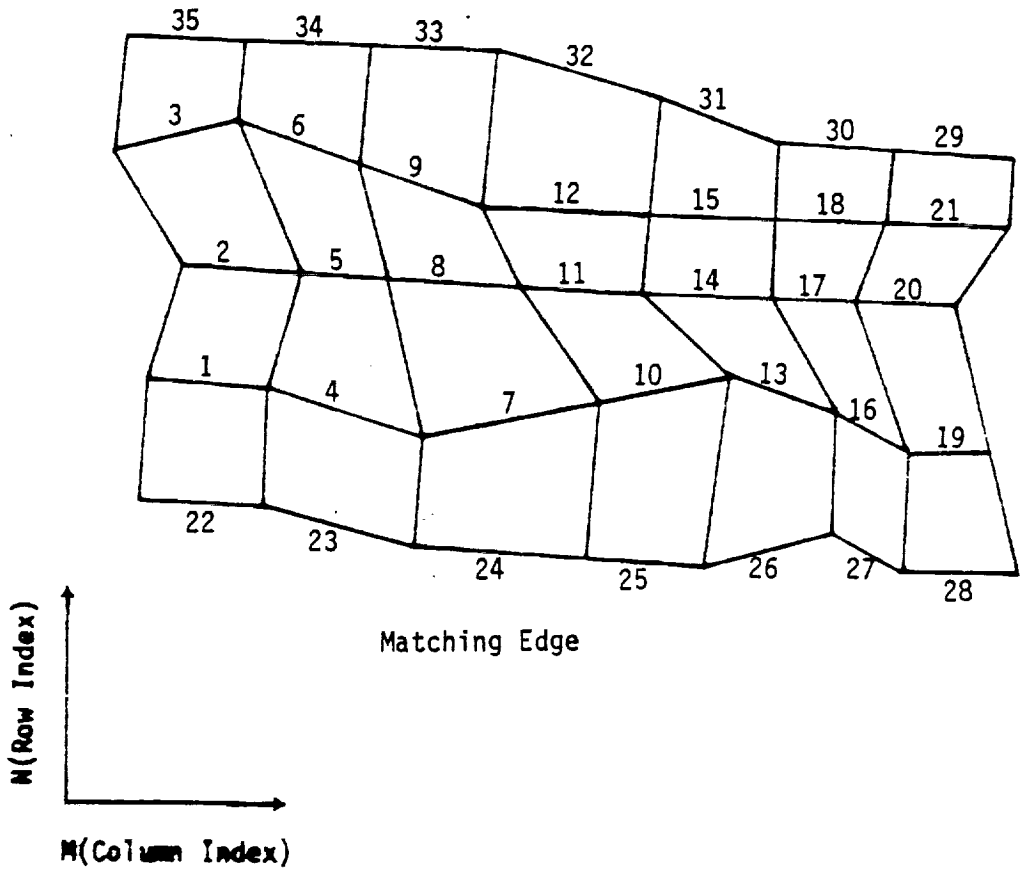


Figure 4-H.24 - Indexing of Singularity Parameters λ_i^S on a Source Design II, Doublet-Null Network Matching Edge

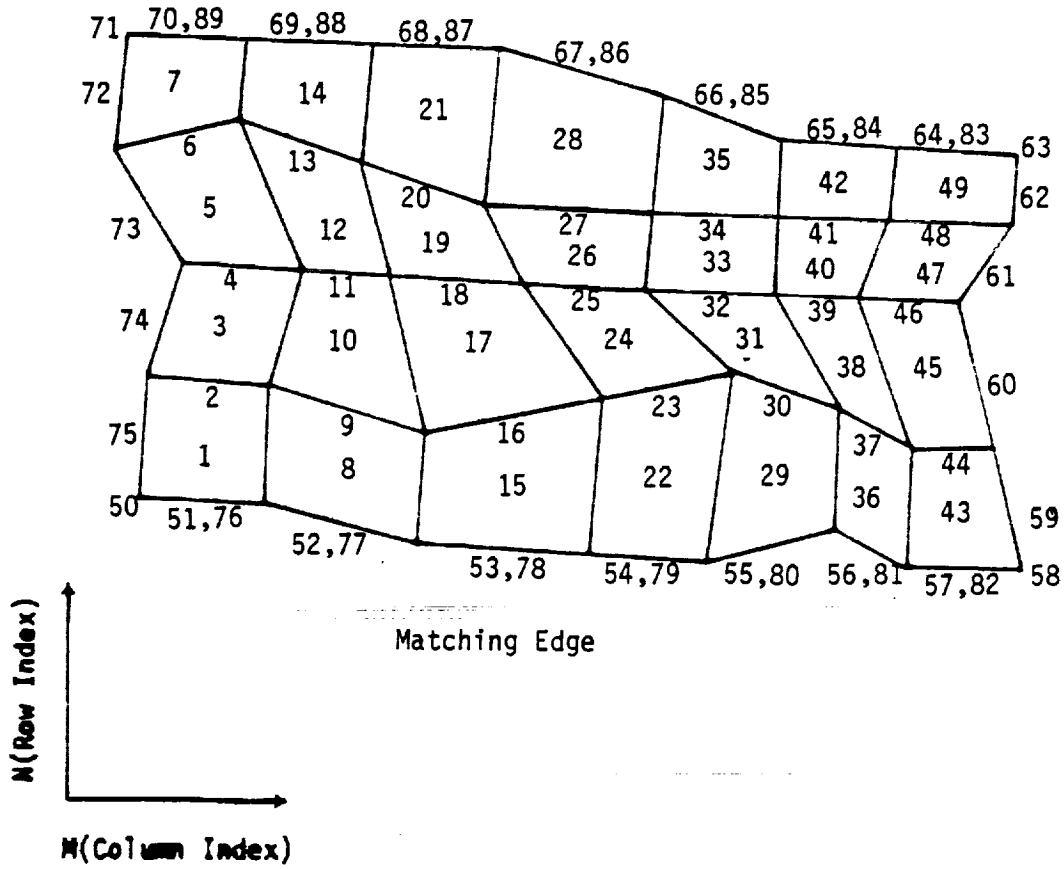


Figure 4-H.25 - Indexing of Singularity Parameters λ_i^S and λ_i^D on a Source Design II, Doublet Analysis Network

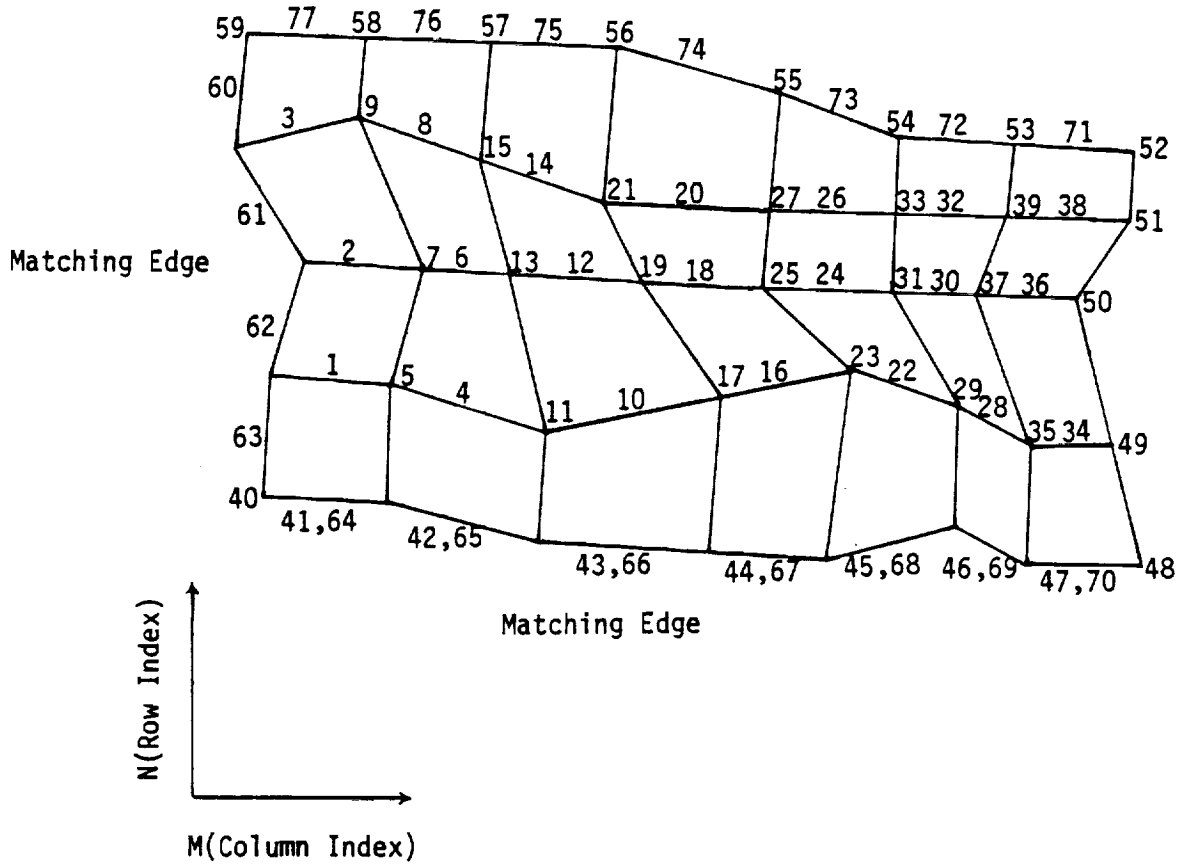


Figure 4-H.26 - Indexing of Singularity Parameters λ_i^S and λ_i^D on a Source Design II, Doublet Design I Network

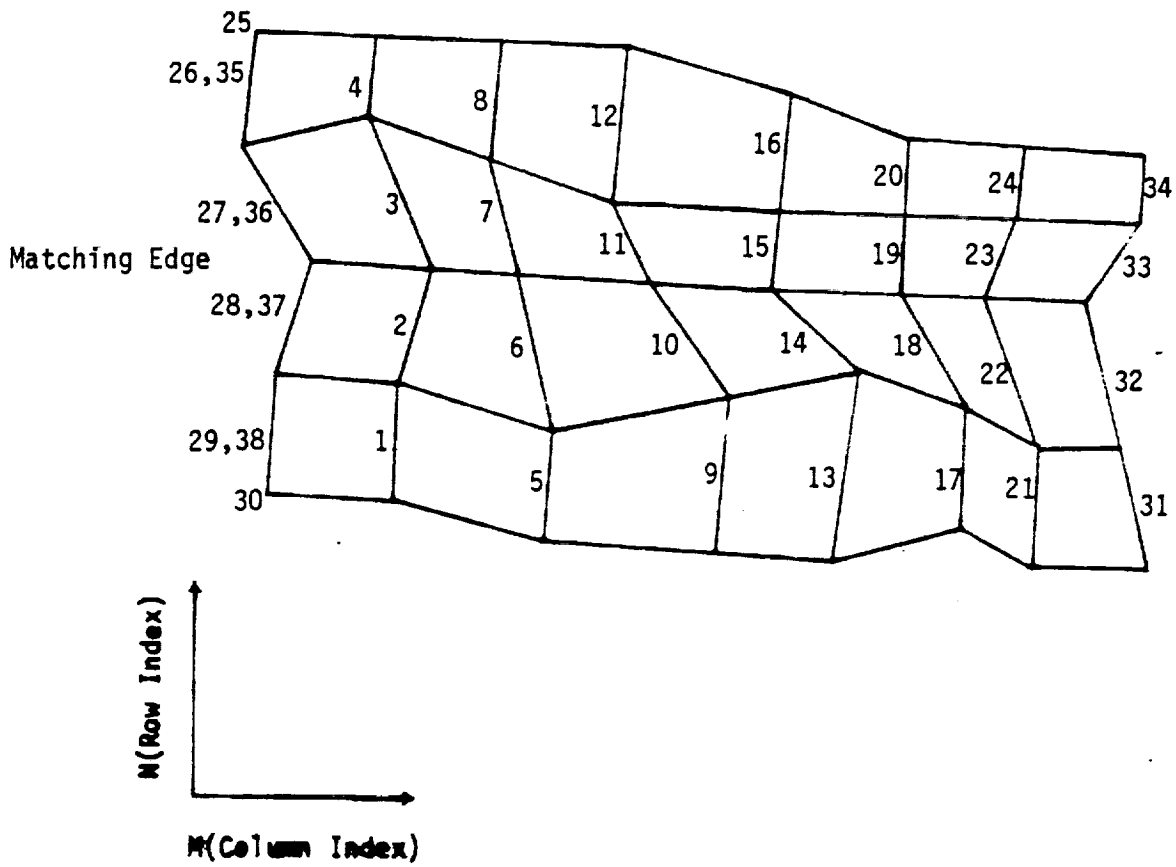


Figure 4-H.27 - Indexing of Singularity Parameters λ_i^S and ^D_i on a Source Design II, Doublet Wake I Network

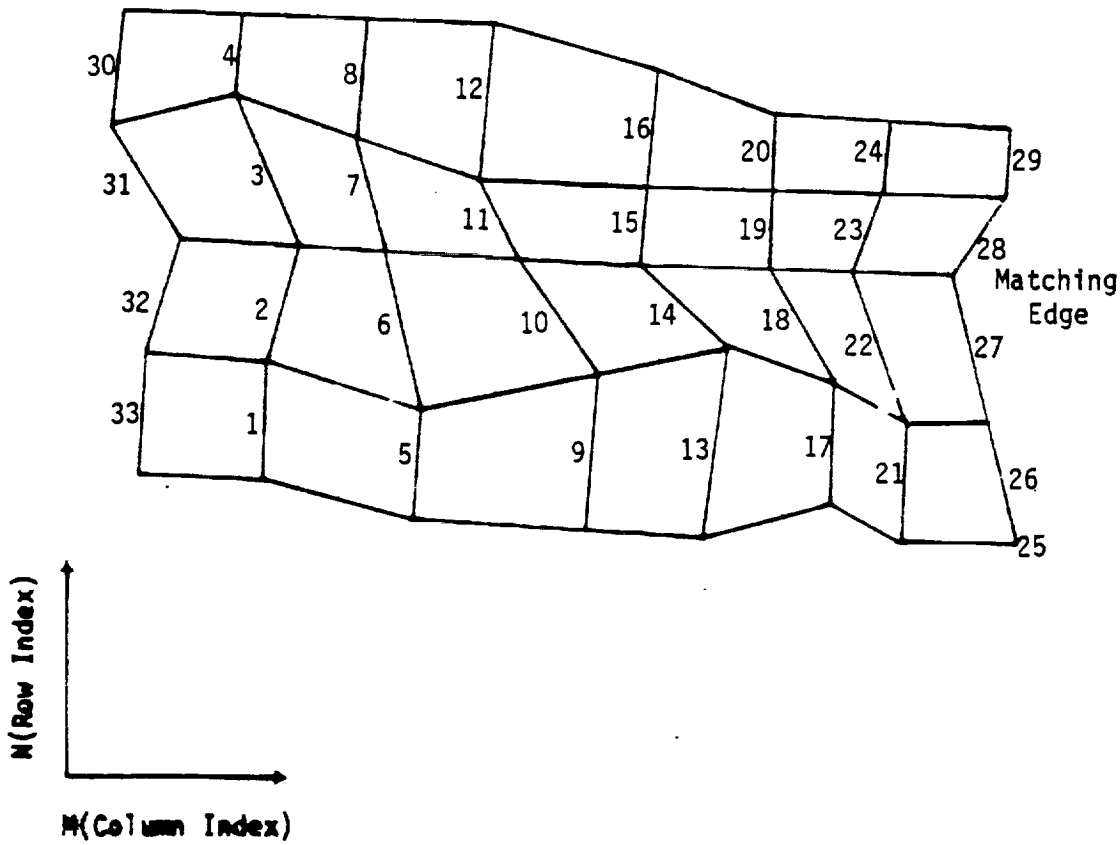


Figure 4-H.28 - Indexing of Singularity Parameters λ_i^S and λ_i^D on a Source Design II, Doublet Wake II Network

APPENDIX 4-I

AUTOMATIC ABUTMENT SEARCH

4-I.1 General Discussion.

DQG offers an option to the user that greatly simplifies the tedious job of specifying abutments, that is, describing which network edges meet. DQG will search the configuration geometry for places where the distance between network edges is less than some specified tolerance distance and will define abutments there. This appendix describes the process which DQG employs to identify abutments. The automatic abutment search is performed in the (3,1) and (3,2) overlays of DQG.

Figure 4-I.1 shows a configuration consisting of five networks. There are two kinds of abutments: network edge abutments and empty space abutments. Network edges which meet other network edges or planes of symmetry are called network edge abutments. Empty space abutments are those places where a network edge does not meet another network edge or plane of symmetry. In the figure there are six network edge abutments (A1 through A6) and eight empty space abutments (E1 through E8).

The automatic definition of abutments occurs in three stages. First, all relations of the form "network A, edge N from point B to point C lies near network D, edge P" are established. These are called pairwise abutment descriptions. In the second stage, all such descriptions for one network edge are examined and a list of all network edges which a particular segment of the one edge lies near is compiled. This is called the expanded abutment description. The final process consists of contracting the expanded description. In this procedure, start and end points of different network edges in the abutment are defined consistently and the abutment assembly is transferred to an output array (the WEABUT or IESABT array for network or empty space abutments) and written as the ABUTMENT-SPEC or EMPTY-SPACE-ABUT datasets.

The user has the option of completely specifying all abutments or specifying part of them and allowing DQG to find the rest or of allowing DQG to find them all. Any abutments specified by the user are not disturbed by the automatic search.

4-I.2 Data Representations.

An understanding of the content and structure of certain arrays is required to understand the manipulations of the automatic abutment search. In this section the arrays are defined.

The IABUT(8) array contans the pairwise abutment description. The first six entries define the edge segment (as in IESABT). The seventh and eighth entries are the network and edge of the other network which the segment lies near.

The IXPAND array contains the expanded abutment description. This is a list of all network edges which lie near one another. It is closely related to the abutment description in the array WEABUT (see below), except that it might contain several network abutments. It is dimensioned 10 by 6. The first index ranges over network edge segments which take part in an

abutment. The values of the second index indicate (1) network index, (2) edge index, (3) and (4) start point column and row indices (coarse grid lattice indices) and (5) and (6) stop point column and row indices (coarse grid lattice indices). This structure is similar to that of the array WEABUT (see below).

The WEABUT array is a 5 by 6 array which contains the abutment description for network abutments, (that is , abutments which involve two or more network edges, or one network edge and a plane of symmetry). A maximum of five network edge segments are permitted in the abutment. The first index of the array ranges over the network edge segments in the abutment. The second index ranges from 1 to 6 and describes the network edge segmenet. WEABUT(I,1) contains the network index of the Ith network in the abutment, WEABUT(I,2) contains the edge index, WEABUT(I,3) and WEABUT(I,4) contain the coarse grid column and row indices of the start point of the edge segment, and WEABUT(I,5) and WEABUT(I,6) contain the coarse grid column and row indices of the end point of the edge segment.

ISRCH is called the search list. It describes the portion of a particular network edge which has not been defined by the user to form an abutment. It is a two dimensional array which is dimensioned 20 by 4. The first index ranges over individual edge segments along a single network edge (a network edge may take part in up to twenty separate network or empty space abutments). The second index ranges from 1 to 4 and defines the column and row indices of the start and of the end point of the segment respectively.

LISTAB is a vector of dimension 20. It is a list of the pairwise abutments in which a particular network edge takes part. This array is used for diagnostic purposes. It is not essential to the automatic abutment search.

LISTCP is a list of the columns and rows of the start and end points of the pairwise abutment segments in which a particular network edge takes part. It is dimensioned 20 by 4. The first index ranges over edge segments on the edge and the second index ranges over column and row indices of the start and end point each edge segment.

SEQCP is a two dimensional array dimensioned 40 by 2. It contains the same data as in LISTCP except that it is sequenced in increasing coarse grid lattice index order.

ILIST is a list of all network edges which lie near one another. It is a two dimensional array, 10 by 2. ILIST(I,1) is the network index of the Ith network edge segment in the abutment and ILIST(I,2) is the edge index of the edge segment.

MSHARY is a two dimensional array dimensioned 2 by 100. It contains the mesh size of each network in the configuration.

EGLNTH is a two dimensional array dimensioned 4 by 100. It contains the edge length of each of the 4 edges of the networks in the configuration.

BLANK is a blank common array used to store all of the edge coordinates. Its dimension is variable and is dependent on the configuration.

LPT is a vector of length 401. Its first element contains a zero. The remaining elements contains, for each network edge, a cumulative count of the number of edge points whose coordinates are stored in the blank common array.

4-I.3 Program Execution.

In the (3,1) overlay of DQG program, program PRABUT calls subroutine USEABT to define any user-provided abutments. At the end of this process USEABT calls SEARCH to define the search list for abutments (Array ISRCH). PRABUT then reads into memory the edge coordinates of all networks and PRABUT sets up a bookkeeping vector, LPT, to keep track of the storage location of the edge coordinates. Finally PRABUT calls subroutine NETABT, which constructs the pairwise abutment arrays.

For each network edge a list is made of all network edges which are not so far away that they are unlikely to take part in an abutment with the given edge. This is done in subroutine EDGLST. A maximum distance is defined equal to the larger of the edge length of the network whose pairwise abutments are being constructed (the reference network edge) and the edge length of the network being examined. Then if a point on the reference edge is closer to either the first or last point on the network edge under examination than the maximum distance, the edge under examination is added to the edge list.

When all network edges have been examined, the pairwise abutment arrays are constructed for the reference edge. For each point on the reference edge which is also in the search list, (ISRCH), the minimum distance to each line segment (segment between two successive corner points on the edge) on the edge of a network in the edge list is computed. If this distance is less than the global tolerance distance, it means a pairwise abutment is found. The reference network, edge and coarse lattice indices are added to the IABUT arrays as well as the network and edge index of the edge under examination. This begins the pairwise abutment.

A similar computation is made for the next point on reference edge. If it is also close to a line segment, the point is defined as the end point of the pairwise abutment. This extends the pairwise abutment.

The extensions continue until there are no more points on the reference edge or until there is a point on the reference edge which is not close enough to the edge under examination. In either of these cases this signals the termination of the pairwise abutment. A check is made to assure there are at least two distinct corner points in the IABUT array and it is written to the data base.

The process continues over all network edges in the edge list and for each network edge in the configuration. Note that collapsed edges are never used in the automatic abutment search.

After the pairwise abutments are defined, they are expanded in subroutine ABXPND in the (3,2) overlay. For each network edge a list is made of all the pairwise abutments in which it appears as the reference network (array ABLIST). At the same time the network and edge which the segment lies near (IABUT(7) and IABUT(8)) are added to the array LSTNET. The start and end point lattice indices are transferred to the array LISTCP. These are sequenced in increasing lattice index without duplication in the SEQCP array. Now a determination is made as to which of the network edges in the LSTNET array all lie near which of the line segments of the reference network edge. The successive entries in the SEQCP array define edge segments which lie near a common set of network edges.

The average of two successive indices in SEQCP are computed. Then the average indices are compared with the start and end points in LISTCP. If the average lies within the start/end interval then the corresponding network edge in NETLST lies near the edge segment defined by the successive entries in SEQCP. The network and edge are added to the ILIST array and the edge segment data is transferred to an array JXPND. After all entries in LISTCP are examined, the reference network edge is added to the ILIST array. The entries in ILIST are sequenced by the network edge constant. ILIST is transferred to an array called IKEY. This is used as a key set to get any pre-existing expanded abutment data. If the data is found, the new data in JXPND is added to the expanded abutment data and the information is written to the disk. If no data is present, this defines a new expanded abutment. The new key array is written to the data base as the ABUT-KEYS data and the new expanded abutment data is written to the EXPANDED-ABUTMENT data set.

The process continues until all network edges in the configuration have been processed.

At the end of subroutine ABXPND, all of the edge segments of every network edge which appears in a single abutment should be contained in some EXPANDED-ABUTMENT element set. The only remaining tasks are to identify start and end points of the edge segments in a consistent fashion and, for the rare case shown in Figure 4-I.3, separate the two abutments which occur on common edge segments. These are performed in CONABT.

In subroutine CONABT an expanded abutment description is read from the disk. A single network edge from the IXPAND array is chosen to establish the start and end points of the abutment. This edge is parameterized (see PAN AIR Theory Document, Appendix F, Section F.6 (Reference 1)) and the coordinates of the points 1/4 and 3/4 of way along the edge are computed. Also a reference distance is defined as the maximum of twice the global tolerance distance and one tenth of the distance between th one-quarter and three-quarter points. Then each other network edge in the expanded abutment is parameterized and the distance from the 1/4 and 3/4 points of the first edge to the 1/4 and 3/4 points of the other edge. If both the 1/4-1/4 and 3/4-3/4 distances are less than the reference distance or if both the 1/4-3/4 and 3/4-1/4 distances are less than the reference distance, the network edge is transferred to the WEABUT array and written to the data base. If neither condition is satisfied, the other edge is skipped and the rest of the edges in the expanded description are checked.

After all expanded abutments are processed, a call is again made to subroutine SEARCH. This again scans all abutments and writes to the DQG data base all those network edges which do not take part in an abutment. These are used in subroutine MTABUT to define empty space abutments.

4-I.4 An Example

Figure 4-I.2 shows a configuration which will be used to illustrate the operations described in the previous section.

Before searching for pairwise abutments the program PRABUT determines that there is sufficient core memory available to store the coordinates of all the edge points. After reading the coordinates of an edge PRABUT stores in array LPT the cumulative number of edge points which have been read into memory. The edges are read in order of network number and edge number. Using the array LPT the program may keep track of the storage locations of the edge point coordinates.

Assume that no abutments have been defined by the user. The automatic abutment search will then be executed for the whole configuration. The search for pairwise abutments begins with the first edge of the first network. The configuration is sufficiently small that all network edges are close enough together to be considered for an abutment with the first edge of network one. Thus all edges in the configuration are searched for pairwise abutments. The first network edge which lies within the global tolerance distance of the first point on edge 1 of network 1 is network 4 edge 3. This causes the first entries to be made in the IABUT array. The network edge and start point column and row indices are listed in IABUT(1), IABUT(2), IABUT(3) and IABUT(4) respectively. The network and edge which lie near the reference network edge (network 1 edge 1), namely network 4 edge 3, is stored in IABUT (7) and IABUT(8) respecitvely. The end point column and row indices of the pairwise abutment (IABUT(5) and IABUT(6) are set equal to the start point indices. The IABUT array then looks like:

```
                I  =   1, 2, 3, 4, 5, 6, 7, 8
          IABUT(I) = ( 1, 1, 1, 1, 1, 1, 4, 3)
                       :  :  :  :  :  :  :  :-Edge index
                       :  :  :  :  :  :  :     (other network)
                       :  :  :  :  :  :  :
Network index----------:  :  :  :  :  :  :----Network index
   (ref network)          :  :  :  :  :  :     (other network)
                          :  :  :  :  :  :
Edge index (ref network)---:  :  :  :  :-------Row index, last pt
                             :  :  :
Col index, start pt----------:  :  :----------Col index, last pt
                                :
Row index, start pt-------------:
```

Then the (2,1) point (that is, column 2 and row 1) of network 1 is examined. It too lies near edge 3 of network 4. Thus the end pont is redefined (IABUT(5)=2 and IABUT(6)=1) and the IABUT array now looks like:

```
                I  =   1, 2, 3, 4, 5, 6, 7, 8
          IABUT(I) = ( 1, 1, 1, 1, 2, 1, 4, 3 )
```

The process continues and the end point is redefined until the last point on edge 1 of network 1 is reached. At this point the search stops momentarily and the IABUT array is written to the DQG database as dataset I-ABUT with a key which is a cumulative index of the number of pairwise abutments discovered, beginning with one. The rest of the network edges are processed in a similar fashion. At the end of program PRABUT, there are fourteen

element sets in the I-ABUT dataset. The contents of the I-ABUT dataset after the end of PRABUT execution are summarized in Table 4-I.1. Note that the fourth pairwise abutment involves the fourth edge of network 1 with the first plane of symmetry. (Planes of symmetry are indicated by a negative network index.) Note that the start and end points in Table 4-I.1 are the coarse grid lattice indices of the points on the network edges (see Appendix 4-H of this manual).

After all the pairwise abutments have been defined, the expanded abutment list is generated. A loop over network edges defines each network edge in turn as a reference network edge. Then each pairwise abutment is examined to see if the reference network edge is in the IABUT array that defines the pairwise abutment (i.e., that the reference network edge is in IABUT(1) and IABUT(2)). For each of these pairwise abutment, the other network edge (entries IABUT(7) and IABUT(8)) identifiers are extracted and added to the LSTNET array. The start and end points of the reference edge are recorded in array LISTCP. To illustrate the process, consider network 2 edge 4 as the reference network edge in the example introduced above. Examination of Table 4-I.1 shows that network 2 edge 4 appears as the entry in IABUT(1) and IABUT(2) in three pairwise abutments. In the first network 2 edge 4 from point (1,3) to (1,6) lies near network 1 edge 2. This defines LSTNET(1,1)=1 and LSTNET(1,2)=2 and LISTCP(1,I) = (1,3,1,6). The second pairwise abutment in which network 2 edge 4 takes part states that points (1,1) to (1,6) lie near network 3 edge 2. Thus LSTNET(2,1)=3 and LSTNET(2,2)=2 and LISTCP(2,I) = (1,1,1,6). Finally the third pairwise abutment that the reference edge appears in states that points (1,1) to (1,3) lie near network 4 edge 2. Thus LSTNET(3,1)=4 and LSTNET(3,2)=2 with LISTCP(3,I) = (1,1,1,3).

After all pairwise abutments have been examined the points in LISTCP are rearranged in increasing lattice index order. The order of the points after the rearrangement is:

(1,1)
(1,1)
(1,3)
(1,3)
(1,6)
(1,6)

Duplicate entries in the list are deleted as the list is moved into the array SEQCP. The results of these operations are summarized by the contents of the arrays LSTNET, LISTCP and SEQCP in Table 4-I.2.

After the indices are copied into SEQCP, the average of each pair of successive indices is computed. This is listed as "AVERAGE" in the SEQCP portion of Table 4-I.2. Now the expanded abutment description is assembled. For each of the entries in the array LISTCP, if the average index of the two adjacent values of SEQCP lies between the start and end points in LISTCP, the corresponding network and edge indices of the array LSTNET are added to the array ILIST. For the first set of average values we have (Table 4-I.2) (1,2). Comparing this with entries in LISTCP we see the point does not lie between the start and end points of the first entry (that is between (1,3) and (1,6)), but it does lie between the second and third entries (namely (1,1) to (1,6) and (1,1), to (1,3) respectively). Thus the corresponding network and edge indices (network 3, edge 2 and network 4 edge 2) are added to the ILIST

array. After all entries in LISTCP have been examined for a particular average value, the reference network is added to the array ILIST. Then the entries in ILIST are resequenced in increasing network and edge index order. This result is shown in Table 4-I.3.

The sequenced ILIST array is used as a key set for the expanded abutment data which is contained in the EXP-ABUT dataset. An attempt to read the data with that particular key set is made. If no data is found, the expanded abutment data is defined from the reference network edge (network 2 edge 4) and from the entries in SEQCP which defined the average value (namely (1,1) and (1,3)). Thus the array IXPAND(1,I) = (2,4,1,1,1,3). The number of edges in the expanded abutment is set to one and the data is written to the EXP-ABUT dataset with a key set equal to (24,32,42). (Note that the network and edge data from ILIST are combined into one index for each network edge by multiplying the network index by 10 and adding the edge index.)

If the expanded abutment description is already found on the database, the number of edges in the expanded abutment is increased by one and the network edge and start and end corner point data is added to the existing IXPAND array. Then the data is written to the database.

This proceeds until all networks and edges have been defined as the reference edge in a pairwise abutment. At the end of the subroutine ABXPND, the EXP-ABUT dataset contains the five element sets described in Table 4-I.4 which are addressed by the key set as indicated in the same table. The reader who desires a full comprehension of these steps is urged to work through the rest of the problem by setting up the LSTNET, LISTCP, SEQCP and ILIST arrays for each network edge in this simplified configuration and thus verify Table 4-I.4.

Finally subroutine CONABT reads in the expanded abutment data and from it defines the abutment description. The last network edge segments description which is not a plane of symmetry is chosen to establish the start and end points of the abutment. This is referred to in the code as the reference network edge. The network, edge, start and stop indices are copied into an intermediate storage array called TWEBUT and the refenence network index in the array IXPAND is set to zero so that this edge will not be selected again in an attempt to define another abutment. Then the subroutine C13QTR finds the coordinates of the points one quarter and three quarters along the edge of the reference segment. This is accomplished by parameterizing the edge segment (see PAN AIR Theory Document, Section 6 of Appendix F (Reference 1)), and then finding the successive corner points whose parameterizations span 0.25 and 0.75. The coordinates of the point on the line segment between these points are then computed in an obvious fashion (by interpolation). The quarter three-quarter point coordinates are used to assure that the start and end points of each network edge are appropriately mated to the start and end points of the reference edge.

After these coordinates are defined, each edge segement in the expanded abutment description other than the reference edge is examined. The distances between the quarter point of the reference edge and the quarter point of the segment under examination, between the three quarter point and the quarter point, between the quarter point and the three quarter point and the quarter point and finally between the three quarter point and the three quarter point are computed and compared with a reference distance. The reference distance

is the larger of twice the global tolerance distance and one tenth of the distance between the quarter point and the three quarter point of the reference edge. If both the distances between the two quarter points and the two three-quarter points is less than the reference distance, then the segment under examination is copied into the TWEBUT array without interchanging the start and end points. If the distances between the quarter point of one side and the three quarter point of the other and vice-versa are less than the reference distance then the data is copied into the TWEBUT array but the start and end indices are interchanged. If neither condition holds, the edge segment under examination does not form an abutment with the reference edge. (This occurs in the configuration illustrated in 4-I.3.) After all edge segments have been considered, the data stored in the TWEBUT array is copied to the WEABUT array and the abutment description is written to the DQG database.

In the example discussed above, the reference edge segment in CONABT is network 4 edge 2 from point (3,1) to (3,3). In this case the other to edge segments in the expanded abutment lie sufficiently close to the reference edge that they are included in the final abutment description. Table 4-I.5 contains the abutment descriptions for the configuration as they would appear at the conclusion of the (3,2) overlay of DQG.

The information presented here covers the basic operations of the automatic abutment search. Any additional information will be obtained by examining the code itself. Having understood the discussion in this appendix the code should be easy to comprehend.

Table 4-I.1 IABUT Array

Network	Edge	Start (Col, Row)	End (Col,Row)	Network	Edge
1	1	(1,1)	(3,1)	4	3
1	2	(3,1)	(3,3)	3	2
1	2	(3,1)	(3,3)	2	4
1	4	(1,1)	(1,3)	-1	0
2	4	(1,3)	(1,6)	1	2
2	4	(1,1)	(1,6)	3	2
2	4	(1,1)	(1,3)	4	2
3	2	(1,2)	(1,3)	1	2
3	2	(1,1)	(1,3)	2	4
3	2	(1,1)	(1,2)	4	2
4	2	(3,1)	(3,3)	2	4
4	2	(3,1)	(3,3)	3	2
4	3	(3,3)	(1,3)	1	1
4	4	(3,1)	(1,1)	-1	0

Table. 4-I.2 Generation of Expanded Abutments for network 2 edge 4

LSTNET(I,J)

I/J	1	2
1	1	2
2	3	2
3	4	2

LISTCP(I,J)

I/J	1	2	3	4
1	1	3	1	6
2	1	1	1	6
3	1	1	1	3

SEQCP(I,J)

I/J	1	2	AVERAGE
1	1	1	(1,2)
2	1	3	(1,4)
3	1	6	

Table 4-I.3 The ILIST array

ILIST(I,J)

I/J	1	2
1	3	2
2	4	2
3	2	4

ILIST(I,J) After Sequencing

I/J	1	2
1	2	4
2	3	2
3	4	2

IXPAND = (2, 4, 1, 1, 1, 3)

Table 4-I.4 IXPAND Arrays

Element Set	Key Set	Network	Edge	Start (Col, Row)	End (Col, Row)
1	(11,43)	1	1	(1,1)	(3,1)
		4	3	(3,3)	(1,3)
2	(12,24,32)	1	2	(3,1)	(3,3)
		2	4	(1,3)	(1,1)
		3	2	(1,2)	(1,3)
3	(-10,14)	-1	0	(0,0)	(0,0)
		1	4	(1,1)	(1,3)
4	(24,32,42)	2	4	(1,1)	(1,3)
		3	2	(1,1)	(1,2)
		4	2	(3,1)	(3,3)
5	(-10,44)	-1	0	(0,0)	(0,0)
		4	4	(1,3)	(1,1)

Table 4-I.5 Final Abutment Description

Network Abutments

Abutment	Network	Edge	Start Point (Col, Row)	End Point (Col, Row)
1	1	1	(1,1)	(3,1)
	4	3	(1,3)	(3,3)
2	2	4	(6,1)	(3,1)
	3	2	(1,3)	(1,2)
	1	2	(3,3)	(3,1)
3	2	4	(1,1)	(3,1)
	3	2	(1,1)	(2,1)
	4	2	(3,1)	(3,3)
4	1	4	(3,1)	(1,1)
	-1	0	(0,0)	(0,0)
5	4	4	(3,1)	(1,1)
	-1	0	(0,0)	(0,0)

Empty Space Abutments

1	1	3	(3,3)	(1,3)
2	2	1	(1,1)	(3,1)
3	2	2	(3,1)	(3,6)
4	2	3	(1,6)	(3,6)
5	3	1	(1,1)	(3,1)
6	3	4	(1,3)	(1,1)
7	4	1	(1,1)	(3,1)

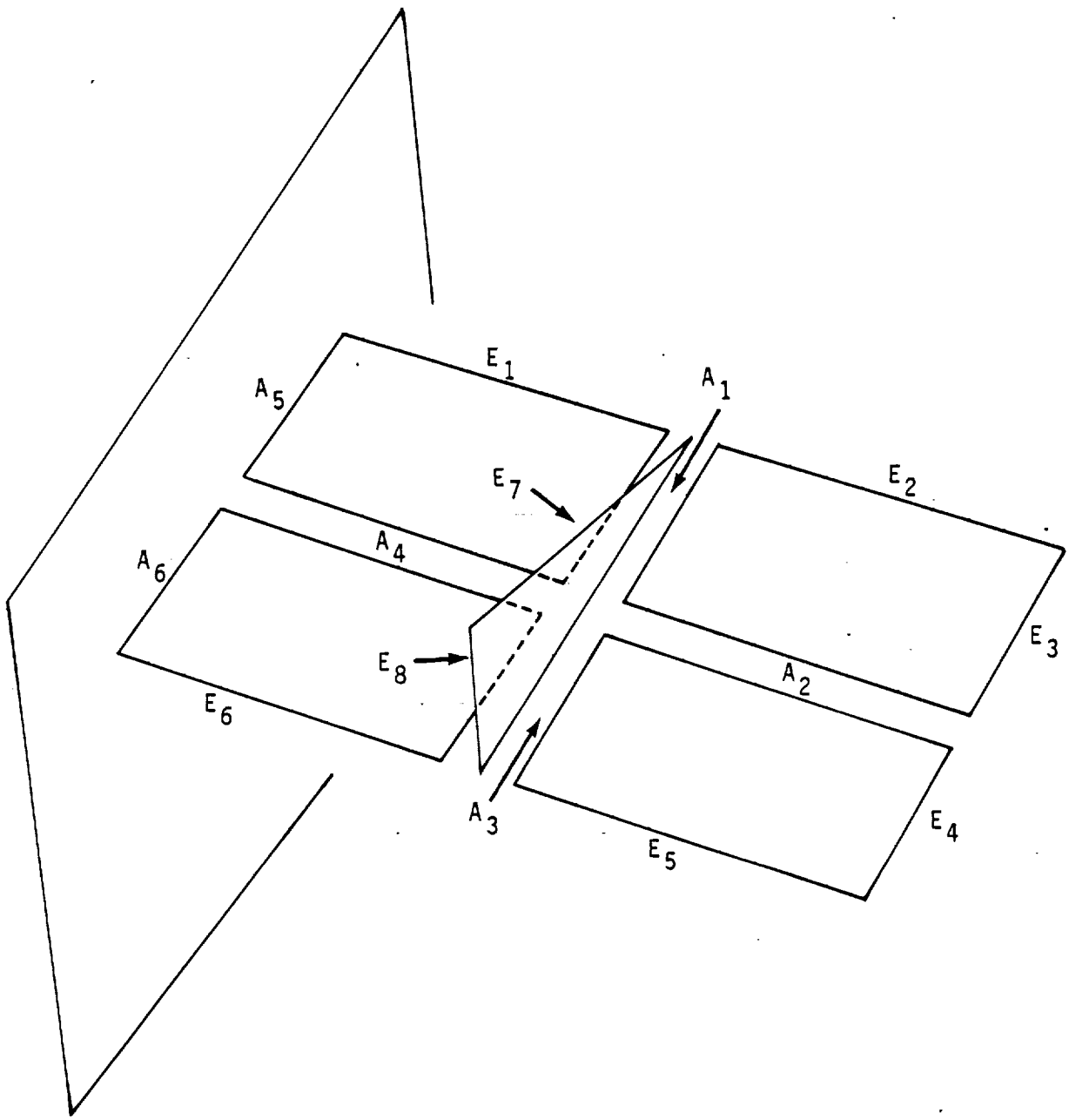


Figure 4-I.1 - Sample Configuration Illustrating Abutments

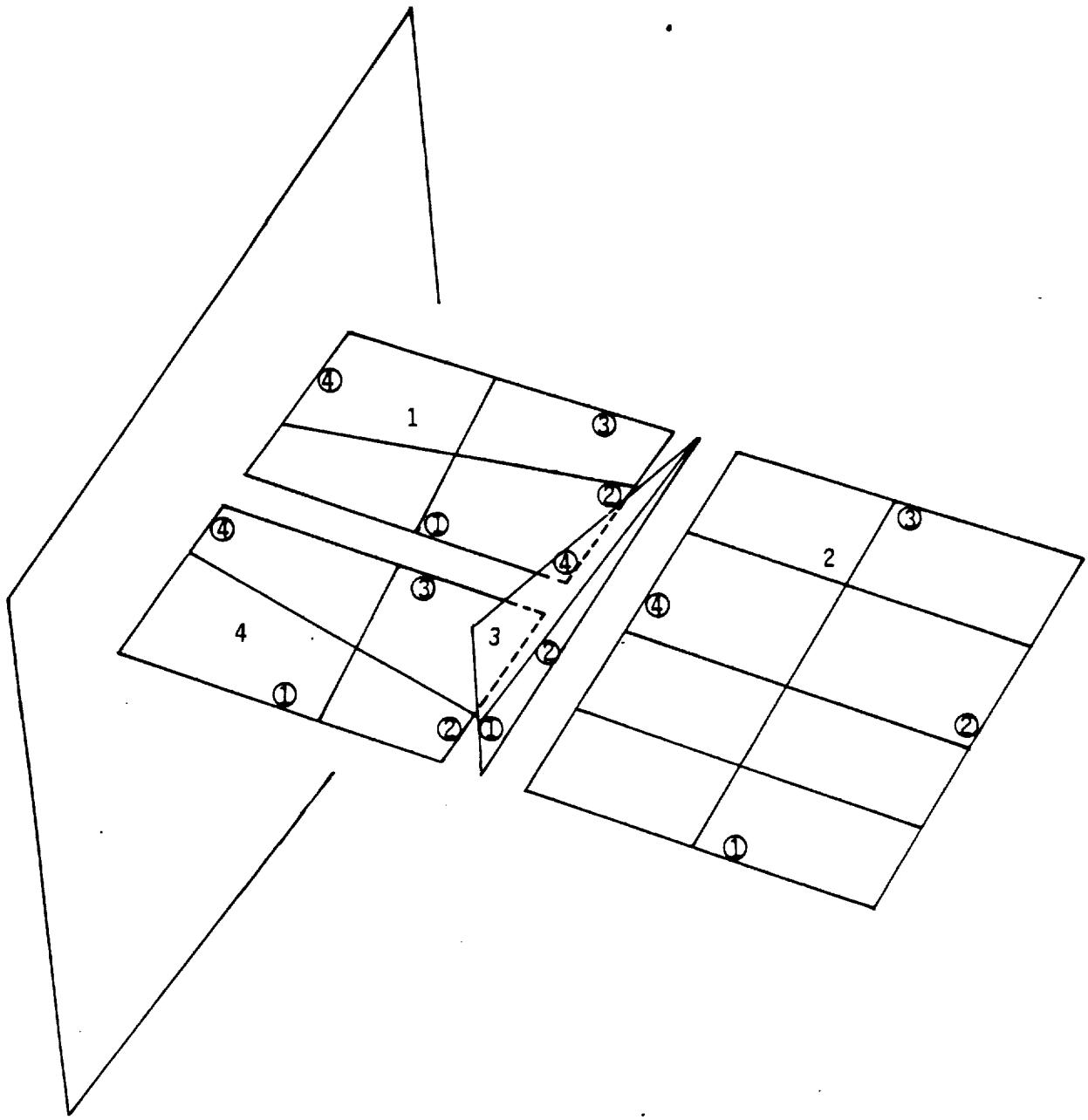
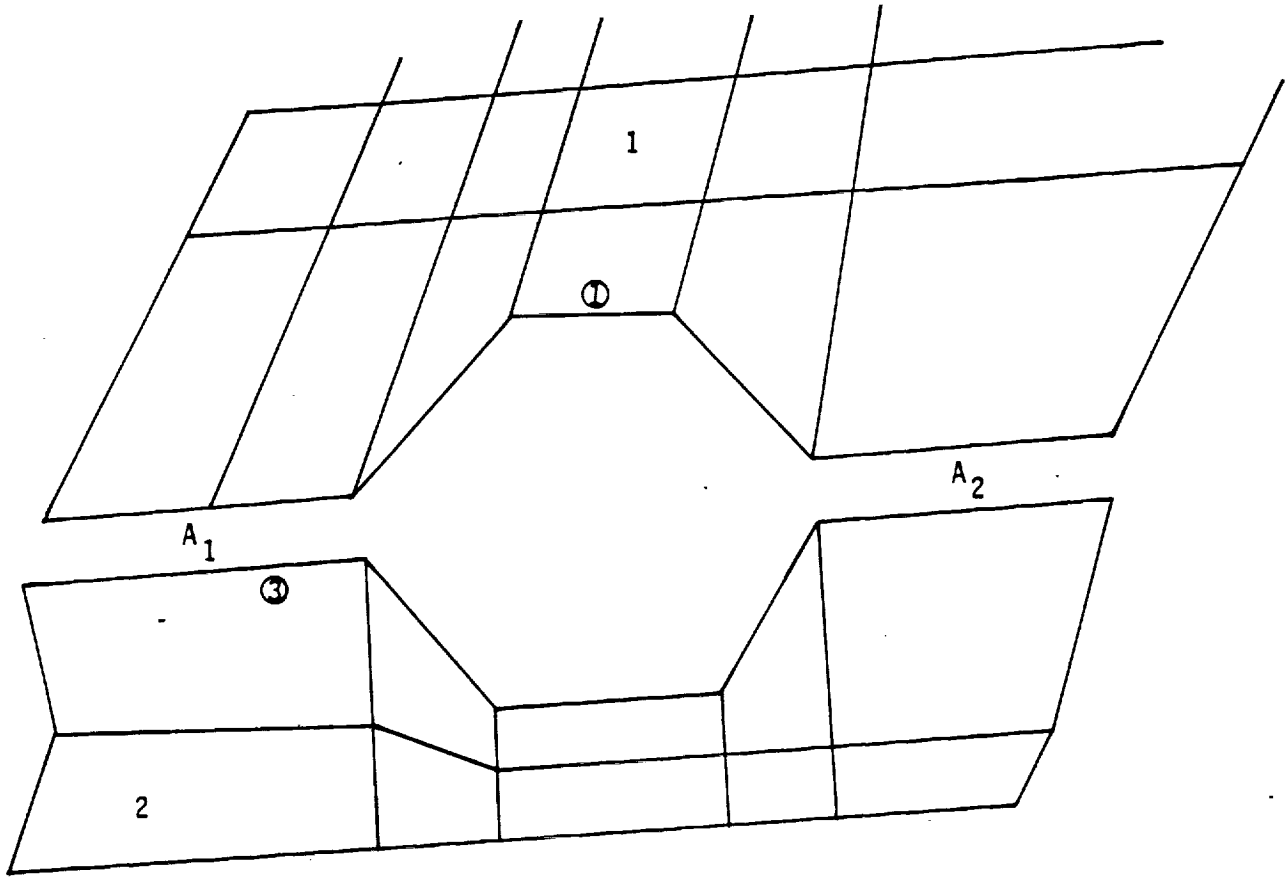


Figure 4-I.2 - Configuration for Example Discussed in Paragraph 4-I.4



Keyset:

	1	1		
	1	1		
	2	3		
	2	3		

IXPAND.

	1	1	(1,1)	(3,1)
	2	3	(5,3)	(6,3)
	1	1	(6,1)	(7,1)
	2	3	(1,3)	(2,3)

A_1

1	1	(1,1)	(3,1)
2	3	(1,3)	(2,3)

A_2

1	1	(6,1)	(7,1)
2	3	(5,3)	(6,3)

Figure 4-I.3 - A Special Case Treated Correctly by Subroutine CONABT

APPENDIX 4-J

ABUTMENT INTERSECTION SEARCH

4-J.1 General Discussion.

In the PAN AIR system the continuity of doublet strength accross network boundaries can be met under those conditions specified in PAN AIR Theory Document, Section 4 of Appendix F (Reference 1). This is achieved through the introduction of matching boundary conditions. One of the most difficult problems which had to be solved in DQG was the determination of how to impose these conditions at abutment intersections.

An abutment (see Appendix 4-I) is a place along which two network edges meet (see Figure 4-J.1). An abutment intersection is the region where abutments come together (Figures 4-J.2, 4-J.3 and 4-J.4). The matching condition at the intersection means the doublet strengths at the two adjacent corner points on the abutting network edges are equal to one another.

The PAN AIR Theory Document, Appendix F, Section F.5 (Reference 1), shows that if N abutments come together at an intersection and N-1 corner points are assigned to match doublet strength, then this is a necessary and sufficient condition to assure doublet continuity at the intersection without redundant equations if the correct N-1 corner points are assigned.

The problem faced by DQG is to select the correct N-1 corner points subject to the following conditions:

(1) Some corner points on design or wake networks are "matching" points, i.e., they must be chosen for matching boundary conditions.

(2) Some corner points on some networks are "non-matching" points which must not be used to match doublet strength.

(3) The selection of corner points to receive matching conditions shall not cause a redundant system of equations.

(4) At collapsed edges of networks, at most one of the two corner points can be used for matching doublet strength.

(5) Doublet strength must be continuous across a plane of symmetry at corner points that lie on the plane of symmetry and doublet strength at a corner point lying on an empty space abutment must be matched to zero.

(6) A corner point can be assigned to at most one abutment and each abutment can receive at most one corner point.

The PAN AIR Theory Document, Appendix F, Section F.5 (Reference 1) introduces a graphical abstraction which summarizes the geometrical situation. Figures 4-J.3 and 4-J.4 provide some additional examples of certain geometric configurations and Figure 4-J.5 a, b and c provide the corresponding graphical equivalents. The reduction of the geometrical configuration to its graphical equivalent and the use of an algorithm from graph theory yields the solution to the problem. The (3,4) overlay of DQG performs the operations which result in the solution to the problem.

4-J.2 Solution to a Problem in Graph Theory

A graph is a collection of points called nodes connected by a set of lines called branches. See Figure 4-J.6. An irreducible subgraph is the set of all nodes which are connected by some set of branches to any other node in the set (Figure 4-J.7). The problem of graph theory is to find all irreducible subgraphs of a given graph.

The solution is accomplished in a very ingenious fashion. First all branches and nodes are assigned an index. See Figure 4-J.8. Then all connections are enumerated. A connection is a list of a branch and the two nodes which match its end points. Table 4-J.1 lists the connections in Figure 4-J.8. Now the connection list is sorted in the following particular fashion.

A table is constructed which has its columns labeled by connection index and whose rows are labeled by node index (Table 4-J.2). The leftmost column is filled with zeroes to initialize the table. For each connection the rows corresponding to the nodes in the previous connection column are examined.

If both contain zero, the connection defines a new subgraph. New entries in the connection columns are made with an integer labeling the new subgraph for the two new nodes. All other nodes' entries are carried over without change. This occurs for the first four connections in Table 4-J.2.

The first connection in Table 4-J.1 has branch 1 connecting node 2 with node 3. The initial column in the sorting table contains only zero entries corresponding to node 2 and node 3 (Table 4-J.2), so the next column in the table (labelled by the index of the first connection, namely 1), is the same as the previous column except that in the row corresponding to node 2 and node 3 there is a 1 entered. This indicates that nodes 2 and 3 belong to the first irreducible subgraph. A similar process generates columns labelled 2, 3 and 4 in Table 4-J.2.

When the fifth connecton in Table 4-J.1 is examined (branch 9 connecting node 2 and node 5), it is discovered that there is a non-zero entry already for node 2. Now something different happens.

If one entry contains zero while the other contains an integer labeling a subgraph, the connection extends the existing subgraph. The zero is replaced by the index of the subgraph and all other indices are copied from the previous column. In the example of Figure 4-J.8, this situation arises for connections 5, 6, 7, 8, 10, 12, and 13.

For connection 5 in Table 4-J.2, node 5 has a zero entry in the previous column while node 2 has an entry equal to 1. So in the column for connection 5, node 5 receives an entry equal to 1 and all other indices are copied over without change.

If both entries contain non-zero but different indices, the two subgraphs are connected to one another. All entries containing one subgraph index are changed to the other. For consistency, we always change the larger index. All other entries are carried over without change. Connection 9 in Table 4-J.1 consists of branch 4 connecting nodes 10 and 11. In Table 4-J.2 the column labelled 8 has a 3 in row 10 and a 2 in row 11. Thus connection 9 causes subgraph 3 and subgraph 2 to be connected. In column 9 of Table 4-J.2,

all "3's" in column 8 are changed to "2's" and the remaining entries are copied directly. This situation occurs both for connection 9 and connection 14.

If both entries contain the same non-zero subgraph index, a closed loop has been discovered, i.e., a subgraph which connects to itself. This situation arises for connection 11.

Connection 11 of Table 4-J.1 consists of branch 8 connecting node 3 and node 4. In the column labelled 10 in Table 4-J.2, both row 3 and 4 contain the same entry "1". Thus connection 11 causes a closed loop to be formed in the irreducible subgraph with index "1". This is noted at the bottom of the table.

When all connections have been processed, the entries in the final column define the irreducible subgraphs of the problem. In Table 4-J.2, the last column has entries with "1" and "2" in them. Thus there are two irreducible subgraphs in the example (see Figure 4-J.8). The first contains nodes 1, 2, 3, 4, 5, 6, 7 and 8 and the second contains noded 9, 10, 11, 12, 13, 14 and 15. The first subgraph contains a closed loop. Thus we have found through the use of an algorithm that Figure 4-J.8 contains two irreducible subgraphs, one of whch has a closed loop, a conclusion which is obvious from examination of the figure.

4-J.3 Application to Abutment Intersection Problem.

The solution to the problem in graph theory can be applied to the abutment intersection problem by identifying abutment intersections with irreducible subgraphs, abutments with nodes, and corner points with branches. The details of the graphical representation of an abutment intersection are given in Appendix F, Section F.5.1 of the PAN AIR Theory Document (reference 1), and we shall not discuss it here any further.

4-J.3.1 Data Representations.

In this section we describe the data storage arrays and their meanings.

IABUTS (300) is an array which contains the index of the abutment. This index is modified if the abutment is an abutment with a plane of symmetry (IABUTS = 2,000 + abutment index) or an empty space abutment (IABUTS = 1,000 + abutment index).

ICPMAP (5,600) contains information about where a corner point at the start or end of an abutment is located as well as a flag indicating its special properties:

```
ICPMAP(1,I) = Network index
ICPMAP(2,I) = Edge index
ICPMAP(3,I) = Coarse grid lattice indices of point (column)
ICPMAP(4,I) = Coarse grid lattice indices of point (row)
ICPMAP(5,I) = +1 if matching point
               0 normal unspecified point
              -1 if non-matching point
```

ICPMAP(I,J) thus indexes all network corner points in the problem (the J-index of ICPMAP is a global corner point index) and describes where they are (the I-index of ICPMAP).

CONNCT (3,600) defines the connection between two corner points by an abutment.

CONNCT(1,I) = Abutment index
CONNCT(2,I) = Global corner point index of one corner point
CONNCT(3,I) = Global corner point index of other corner point

For an abutment with two network edges, there is only one connection at the start and one at the end of the abutment. If there are N network edges and planes of symmetry in the abutment, there are N(N-1)/2 connections. If one of the edges is a plane of symmetry, this is a special case. The plane of symmetry corner point index is defined conventionally to be zero. An abutment with empty space is another special case. It has only one corenr point in its connection. In this case the single corner point index is listed as both nodes in the connection.

The connection list is sequenced in a special order. First, all connections which have one or more corner points which are matching points appear in the list. Then all connections with empty space abutments appear. Finally, all remaining connections are sequenced by the greater of the two "downstream parameters" of the two corner points. (The downstream parameter is a measure of whether the point is upstream or downstream of the network interior. It is defined as

$$D = \frac{\vec{V} \cdot \hat{c}_0}{V}$$

where $\hat{c}_0$ is the compressibility vector and $\vec{V}$ is the vector from the corner point to the diagonally opposite panel-corner point if the corner point is a network corner point. If the corner point is an extra control point added by DQG then V is the vector from the corner point to the next interior panel-corner point on the same column or row.

This sequencing of connections is to assure that connections with matching corner points will not be the ones that will be found to form a closed loop. They will tend to create new subgraphs rather than extend existing ones. By the same token, the more upstream corner points will tend to form additions to subgraphs which will not form closed loops. This means that more upstream corner points will be selected to impose matching conditions for the network, a situation which is empirically preferred for stabiity reasons when solving design problems. See PAN AIR Theory Document, Appendix F, Section F.4 (Reference 1).

On collapsed edges of a network, special consideration is required. Collapsed edges do not appear as an abutment. For this reason the procedures to be described would find for the example in Figure 4-J.9 that abutment A_1 corner point C_1 did not take part in any abutment intersections, while abutments A_2 and A_3 along with corner points C_2, C_3, and C_4 were part of an intersection.

To allow the addition of abutment A_1 to the intersection, the contents of the array CONNCT are modified. All nodes which are indexed by the global index of the last corner point on the collapsed edge (in a counter clockwise sense around the edge) are changed to -1 x the global index of the first corner point on the collapsed edge. At the same time in array CECPN (600) (initialized to zero) the entry under the global index of the first point is set equal to the global index of the last point on the edge.

This will allow the procedure to find A1 connected to A2 and A3 . The negative value for the node index flags the corner point to indicate that if the node in the connection is selected for a matching assignment, the alternate reference (the last point on the edge, obtained from array CECPN) is the true location for the matching assignment.

CPLST (600) is an array which is used as the table constructed in Section 4-J.2. It is initialized to zero. At the end of the procedure, each entry in the array contains an index of the abutment intersection (subgraph) to which the corner point belongs.

PCCT (3,600) is an integer array which contains information compliment to CONNECT. PCCT defines the connection between two abutments by a corner point.

PCCT(1,I) = corner point location for Ith corner point. It contains network number times 1,000,000 plus column number times 1000 plus row number of the corner point.

PCCT(2,I) = abutment index for one of the two abutments connected to the Ith corner point.

PCCT(3,I) = abutment index for the other abutment connected to the Ith corner point.

ABTSYM(300) is an integer array that contains the abutment tangent symmetry descriptor. Its value may range from 0 to 3.

ABTSYM(I) = 0 if abutment I lies away from both POS
ABTSYM(I) = 1 if abutment I lies in first POS
ABTSYM(I) = 2 if abutment I lies in second POS
ABTSYM(I) = 3 if abutment I lies in both POS

NETDBT(100) contains the network doublet type for each network in the configuration.

After all abutment intersections are found, the matching assignments must be made. In the process of performing the assignment, some additional arrays are used. These arrays contain information concerning a particular abutment intersection.

ABCPCP (3,30) contains all the connections which make up any one abutment intersection (subgraph). These are stored on the disk keyed by intersection (subgraph) number. If a connection establishes a closed loop, then the abutment index in the connection description ABCPCP (1, K) is multiplied by -1. Thus a negative abutment index in the connection list of an abutment intersection indicates that the intersection has a closed loop.

CPLIST (60) contains the global corner point index of all corner points in the abutment intersection. As corner points are assigned matching conditions, the value of the location which held the global index of the point is set to zero. This removes it from consideration in future assignments.

ABLIST (30) contains the modified abutment indices (see description of the IABUTS array) of all abutments in the intersection.

LNOD(15) CONTAINS THE ABUTMENT TANGENT STATUS EXTRACTED FROM ABTSYM for the abutments which meet at the abutment intersection.

PNOD(15) contains the abutment indices. These indicies may be positive or negative integers. The sign indicates the direction of the abutment. A positive sign indicates that the abutment points away from the abutment intersection and a negative sign indicates that the abutment points toward the abutment intersection.

PQ(2,30) is a list of pairs of abutments. The indices of the two abutments adjacent to Ith corner point are contained in PQ(1,I) and PQ(2,I).

KSEG(30) is the doublet matching status of corner points.

KSEG(I) = 0	Ith corner point is on a source alone network
KSEG(I) = 1	not used
KSEG(I) = 2	doublet network but no control point for matching
KSEG(I) = 3	reserved for plane of symmetry
KSEG(I) = 4	regular corner control point
KSEG(I) = 5	matching corner control point

LSEG(30) indicates whether corner point lies in a plane of symmetry.

LSEG(I) = 0	Ith corner point not in any POS
LSEG(I) = +1	Ith corner point in first POS
LSEG(I) = ∓2	Ith corner point in second POS

Positive sign indicates that the network normal is parallel to the POS normal and negative sign indicates that the normals are anti-parallel.

WSEG(30) contains the upstream downstream parameters of corner points

NFGSEG(30) is set up to help maintenance programmers to diagnose errors. NEGSEG(I) contains network number times 1,000,000 plus column number times 1000 plus row number of Ith corner point in an abutment intersection.

4-J.3.2 Program Execution.

The assignment of matching conditions at abutment intersections occurs in two steps. First all abutment intersections are found and written to the data base (subroutine INTRSC). Then each intersection is assigned matching conditions (subroutine ASSIGN).

4-J.3.2.1 Abutment Intersections.

Figure 4-J.10 illustrates the process which finds all abutment intersections in the configuration. Subroutine INTRSC reads abutment data and from it constructs the IABUTS, ICPMAP, CPMAP, PCCT and CONNCT arrays. Subroutine COLCPT modifies entries in CONNCT if there are collapsed network edges and generates array CECPN. Then subroutine NTRLST performs an analysis of entries in CONNCT similar to that presented in Section 4-J.2; and defines abutment intersections. These are written onto the DQG data base in data set INTERSECTION.

4-J.3.2.2 Matching Assignments.

Figure 4-J.11 illustrates the subroutines and data flow which occur in the process of assigning matching conditions at abutment intersections. Subroutine ASSIGN reads the intersection data from the data base and fills array ABCPCP with it. From this data the arrays ABLIST, CPLIST, LNOD, PNOD, PQ, KSEG, LSEG, WSEG and NFGSEG are then created. Subroutine ABTINT, which is a PAN AIR library routine, examines the graph of the abutment intersection and assigns corner points to abutments for doublet matching. If there are N abutments in the intersections, N-1 corner points must be assigned to insure doublet matching at the intersection. See PAN AIR Theory Document, Appendix F, Section F.5 (Reference 1) for a discussion of this important point.

The indices of the matching corner points and the indices of the associated abutments are passed to subroutine ABASGN. ABASGN obtains the abutment data and checks ICPMAP(I,ICP) (where ICP is the corner point index), I = 1, 3 and 4 for agreement with the network and coarse grid lattice indices of the start or end point in an edge segment in the abutment. If the point is found in the Nth network edge in the abutment, the flag in the array DSVMCH (part of the abutment data) corresponding to doublet corner point matching for start or end points (DSVMCH(I,2,1) and DSVMCH(I,3,1) respectively) is set equal to N. Then the abutment data is replaced on the data base, the number of assignments made is incremented and the routine returns. In this way all corner points where doublet matching is to occur are labelled in the abutment description. Later in the (3,5) overlay, subroutine MATCHPT reads the abutment data and copies these matching flags into the SPECIAL-POINTS dataset for use in the (4,0) overlay of DQG where the matching boundary conditions are actually imposed.

Matching assignments for abutment intersections lying on one or more planes of symmetry present additional complications. PAN AIR takes the approach to treat each symmetry condition as a separate problem. Therefore, whether a matching condition should be assigned at a corner control point lying on a plane of symmetry depends on the symmetry condition and which plane of symmetry the control point lies. For corner control points lying on a plane of symmetry the matching assignments described in the last paragraph are made for each symmetry condition. The matching pointers for Ith symmetry condition is stored in DSVMCH(I,2,1) and DSVMCH(I,3,1) for start and end corner points.

For corner points not lying on a plane of symmetry and for edge interior control points the matching boundary conditions are independent of symmetry condition and therefore, matching pointers are needed only for the first symmetry condition.

4-J.3.3 An Example.

Figure 4-J.12 shows a configuration for which we present a detailed example of the operations discussed in this section. Tables 4-J.3 and 4-J.4 contain the abutment data. The abutments are labeled in Figure 4-J.13. (The configuration contains a collapsed edge (network 4), plane of symmetry abutments, a matching edge, and a non-matching edge. This example will illustrate nearly all features of the program.)

From the data in Tables 4-J.3 and 4-J.4 we construct the IABUTS arrays and the ICPMAP arrays shown in Table 4-J.5 and 4-J.6. Note that the empty space abutments' entry in the IABUT array is the empty space abutment index plus 1,000. Also note the offset of 2,000 added to the plane of symmetry abutments.

Figure 4-J.14 shows the index assigned to each corner point and abutment. These indices correspond to the columns labelled index in Table 4-J.6 and 4-J.5 respectively.

Table 4-J.7 contains the connection description. Note that first in the connection list appears the connections with matching corner points. The entry (0 0 0) in the connection array separates the matching connections from the empty space connections (connection index 3), and the empty space connections from all the other connections (connection index 20). The connections at the end of the list are sequenced by downstream parameter. Then subroutine COLCPT finds that edge 3 of network 4 collapsed. Thus connection number 18 is changed from (14, 20, 20) to (14,-10,-10). The reference to corner point number 20 appears in array CECPN.

The search procedure begins in NTRLST. Table 4-J.8 describes the transition of the array CPLST as the connections are examined for subgraph formations. This data is similar to Table 4-J.2 discussed in section 4-J.2. Table 4-J.9 contains the resulting intersection data. Note that a closed loop is discovered at connections numbers 23 and 29 for the first intersection and at connection number 33 for the eighth intersection. Note also that several intersections (13, 12, 10, and 6) are discovered to be connected to another intersection (1, 10, 8, and 3 respectively) as the intersection search proceeds.

At the conclusion of the search, the DQG data set INTERSECTION contains the same data as Table 4-J.9. Thus there are ten abutment intersections is the example of Figure 4-J.12. One of them has two closed loops (intersection 1) formed by abutments 3 and 4, and another has one closed loop (intersection 8) formed by abutment 3.

After the intersections are defined, doublet matching assignments are made. Subroutine ASSIGN is called. This subroutine reads the intersection data and sets up the data which describe the directed graph corresponding to the abutment intersection. Subroutine ABTINT is called to make the matching assignments for each abutment intersection. ABTINT returns the array NODSEG. If the Ith element of NODSEG is less than or equal to zero then the Ith corner point in CPLIST is not used for matching. However, if the Ith element of NODSEG is a positive integer P then the Ith corner point in CPLIST is used to match doublet across Pth abutment in PNOD. Each pair of matching corner point and the corresponding abutment is passed to subroutine ABASGN which updates the matching assignment on the DQG database.

In this discussion we will only examine the assignment process for the first intersection. The arrays PNOD, CPLIST and PQ for this intersection are shown in Tables 4-J.10 and 4-J.11, together with the associated network numbers. The direction of the abutments are shown by arrows in Figure 4-J.15.

The array NODSEG as outputed by subroutine ABTINT is given in Table 4-J.12. The first element contains 3. Therefore, the first corner point in CPLIST, corner point number 10, is selected to match across the abutment contained in the third element of PNOD or abutment number 14. The abutment index is then used by ABASGN to read the abutment data. The corner point index is used along with the array ICPMAP to determine whether the matching corner point is the start point or the end point of the abutment. In this case the lattice indices of corner point number 10 does not correspond to the lattice indices of either the start point or the end point. But corner point number 10 is on a collapsed edge (as indicated by a negative sign in abutment intersection data and array PCCT). Therefore, array CECPM contains an alternate corner point number which is 20. Corner point number 20 turns out to be the end point of abutment number 14. The matching pointer is then set to the appropriate control point and the abutment data is updated on the database.

This process continues until all matching pointers are set. Notice that the fourth element of NODSEG contains a zero. This means that the fourth corner point in CPLIST, corner point number 12, does not have a matching boundary condition. Processing of this abutment intersection is therefore skipped to the fifth element in NODSEG.

Figure 4-J.15 shows the doublet matching assignments after all intersections have been processed. Note that no assignment is made for points 11 and 16 since they are "no matching" points. Also no matching assignment is made for point 12 since if it were assigned to either abutment 3 or 4, it would produce a redundant set of equations for matching at the first intersection.

Thus all abutment intersections have been identified and matching assignments have been made without producing a redundent set of constraints and without missing an assignment. Later in the third overlay in subroutine MATCHPT of the (3,6) overlay, the matching flags are read from the abutment data and transferred to the SPECIAL-POINTS dataset. This dataset is read by the (4,0) overlay of DQG where the presence of a matching flag produces a DQG generated boundary condition of doublet matching. The (4,0) overlay then selects from among the user-defined boundary conditions and the DQG-defined boundary conditions to determine what constraints are actually imposed at the control points (see Appendix 4-M).

Table 4-J.1 Connections in the graphs of Figure 4-J.8 (see Section 4-J.2)

Cumulative Index	Branch	Node	Node
1	1	2	3
2	3	13	14
3	5	9	10
4	12	7	8
5	9	2	5
6	11	14	15
7	10	12	13
8	13	11	12
9	4	10	11
10	7	4	5
11	8	4	3
12	6	7	6
13	14	1	2
14	2	5	6

Table 4-J.2 Sorting table whose generation determines the number of irreducible subgraphs in a graph (see Section 4-J.2).

	Connnection Index														
Node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
2	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
5	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
6	0	0	0	0	0	0	0	0	0	0	0	0	4	4	1
7	0	0	0	0	4	4	4	4	4	4	4	4	4	4	1
8	0	0	0	0	4	4	4	4	4	4	4	4	4	4	1
9	0	0	0	3	3	3	3	3	3	2	2	2	2	2	2
10	0	0	0	3	3	3	3	3	3	2	2	2	2	2	2
11	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2
12	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2
13	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2
14	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2
15	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2
												Closed Loop			

Table 4-J.3
Network Edge Abutments for the example in Figure 4-J.12.
This example is discussed in Section 4-J.3.3.

1	Network	Edge	Start	End
	1	2	(3,1)	(3,3)
	2	4	(1,1)	(1,3)
2	Network	Edge	Start	End
	2	1	(1,1)	(3,1)
	3	3	(1,3)	(3,3)
3	Network	Edge	Start	End
	3	4	(1,1)	(1,3)
	4	2	(3,1)	(3,3)
	5	2	(3,1)	(3,3)
4	Network	Edge	Start	End
	5	3	(1,3)	(3,3)
	1	1	(1,1)	(3,1)
5	Network	Edge	Start	End
	1	4	(1,3)	(1,1)
	-1	0		
6	Network	Edge	Start	End
	5	4	(1,3)	(1,1)
	-1	0		

Table 4-J.4
Empty Space Abutments for the example in Figure 4-J.13.
The example is discussed in section 4-J.3.3.

Empty Space Abutment Index	Network	Edge	Start	End
1	1	3	(3,3)	(1,3)
2	2	3	(3,3)	(1,3)
3	2	2	(1,3)	(3,3)
4	3	2	(1,3)	(3,3)
5	3	1	(1,1)	(3,1)
6	5	1	(1,1)	(3,1)
7	4	1	(1,1)	(3,1)
8	4	4	(1,3)	(1,1)

Table 4-J.5
IABUTS Array for the example of Figure 4-J.12 and 4-J.13.
The example is discussed in Section 4-J.3.3.

Index	Array
1	1
2	2
3	3
4	4
5	2005
6	2006
7	1001
8	1002
9	1003
10	1004
11	1005
12	1006
13	1007
14	1008

Table 4-J.6
ICPMAP array for the example in Figure 4-J.12 and 4-J.13.
The example is discussed in Secton 4-J.3.3.

Index	Network	Edge	Start/stop Point (Col, Row)	Matching Flag	CPMAP
1	1	2	(3,1)	0	0.51
2	1	2	(3,3)	0	-0.51
3	2	4	(1,1).	0	-0.70
4	2	4	(1,3)	0	0.71
5	2	1	(3,1)	0	0.72
6	3	3	(1,3)	1	-0.66
7	3	3	(3,3)	1	-0.66
8	3	4	(1,1)	0	0.65
9	4	2	(3,1)	0	0.56
10	4	2	(3,3)	0	-0.88
11	5	2	(3,1)	-1	0.42
12	5	2	(3,3)	0	-0.46
13	5	3	(1,3)	0	-0.44
14	1	1	(1,1)	0	0.51
15	1	4	(1,3)	0	-0.51
16	5	4	(1,1)	-1	0.42
17	2	3	(3,3)	0	-0.70
18	3	2	(3,1)	0	0.65
19	4	1	(1,1)	0	0.57
20	4	4	(1,3)	0	-0.88

Table 4-J.7 The list of Connections for the example of Figure 4-J.12
The example is discussed in Section 4-J.3.3.

Connection Index	CONNCT Array (Initial)			CONNCT Array (After Sequencing)			CONNCT Array (After COLCPT Execution)		
	A	Cp	Cp	A	Cp	Cp	A	Cp	Cp
1	2	3	6	2	3	6	2	3	6
2	2	5	7	2	5	7	2	5	7
3	0	0	0	0	0	0	0	0	0
4	7	2	2	7	2	2	7	2	2
5	7	15	15	7	15	15	7	15	15
6	8	17	17	8	17	17	8	17	17
7	8	4	4	8	4	4	8	4	4
8	9	5	5	9	5	5	9	5	5
9	9	17	17	9	17	17	9	17	17
10	10	7	7	10	7	7	10	7	7
11	10	18	18	10	18	18	10	18	18
12	11	8	8	11	8	8	11	8	8
13	11	18	18	11	18	18	11	18	18
14	12	16	16	12	16	16	12	16	16
15	12	11	11	12	11	11	12	11	11
16	13	19	19	13	19	19	13	19	19
17	13	9	9	13	9	9	13	9	9
18	14	20	20	14	20	20	14	-10	-10
19	14	19	19	14	19	19	14	19	19
20	0	0	0	0	0	0	0	0	0
21	1	1	3	3	6	10	3	6	10
22	1	2	4	3	6	12	3	6	12
23	3	8	9	3	10	12	3	10	12
24	3	8	11	6	13	13	6	13	13
25	3	6	10	6	16	16	6	16	16
26	3	6	12	1	1	3	1	1	3
27	3	9	11	4	13	14	4	13	14
28	3	10	12	5	15	15	5	15	15
29	4	13	14	4	12	1	4	12	1
30	4	12	1	5	14	14	5	14	14
31	5	15	15	3	9	11	3	9	11
32	5	14	14	3	8	9	3	8	9
33	6	13	13	3	8	11	3	8	11
34	6	16	16	1	2	4	1	2	4

<u>CECPN Array</u>

CECPN(I) = 0 $\quad$ I $\neq$ 10 $\quad$ I = 1,20
CECPN(10) = 20

Table 4-J.8

Transition of array CPLST during abutment intersection search in subroutine NTRLST for example of Figure 4-J.12. The example is discussed in Section 4-J.3.3.

Connection Index

Corner Point Index	Initial	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
1	0											1	1	1	1	1	1	1	1	1															
2	0				3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	0							6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
5	0		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
6	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	0		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
8	0												8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
9	0																	12	12	12	12	12	12	12	12	12	12	12	12	12	12	10	8	8	8
10	0																		13	13	13	1	1	1	1	1	1	1	1	1	1	1	1	1	1
11	0															10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	8	8	8
12	0																						1	1	1	1	1	1	1	1	1	1	1	1	8
13	0																								14	14	14	14	14	14	14	14	14	14	14
14	0																											14	14	14	14	14	14	14	14
15	0					4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
16	0														9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
17	0						5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
18	0											7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
19	0																11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
20	0																							Closed Loop						Closed Loop				Closed Loop	

Table 4-J.9
Abutment Intersections in example of Figure 4-J.12.
The example is discussed in Section 4-J.3.3

Intersection Index	Connections		
1	2	3	6
	3	0	10
	14	-10	-10
	3	6	12
	-3	10	12
	1	1	3
	-4	12	1
2	2	5	7
	10	7	7
	9	5	5
3	7	2	2
	8	4	4
	1	2	4
4	7	15	15
	5	15	15
5	8	17	17
	9	17	17
6	Appended to 3		
7	10	18	18
	11	18	18
8	11	8	8
	3	8	9
	12	11	11
	3	9	11
	13	9	9
	-3	8	11
9	12	16	16
	6	16	16
10	Appended to 8		
11	13	19	19
	14	19	19
12	Appended to 10		
13	Appended to 1		
14	6	13	13
	4	13	14
	5	14	14

Table 4-J.10 PNOD Array for First Intersection

INDEX	PNOD(INDEX)
1	-2
2	-3
3	14
4	-1
5	4

Table 4-J.11 Corner Point and Abutment Arrays for First Intersections

Index	After Sequencing by Downstream Parameter CPLIST(INDEX)	PQ(1,INDEX)	PQ(2,INDEX)	NFGSEG(INDEX)
1	10	-3	14	4003003
2	3	-2	-1	2001001
3	6	-2	-3	3001003
4	12	4	-3	5003003
5	1	-1	4	1003001

Table 4-J.12 NODSEG Array for First Intersection

INDEX	NODSEG(INDEX)
1	3
2	1
3	2
4	0
5	4

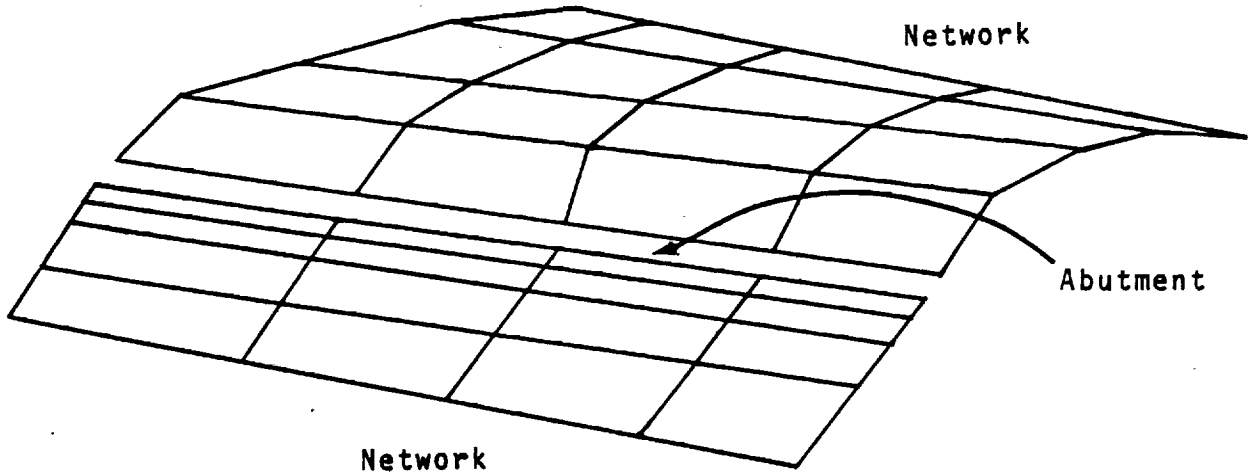
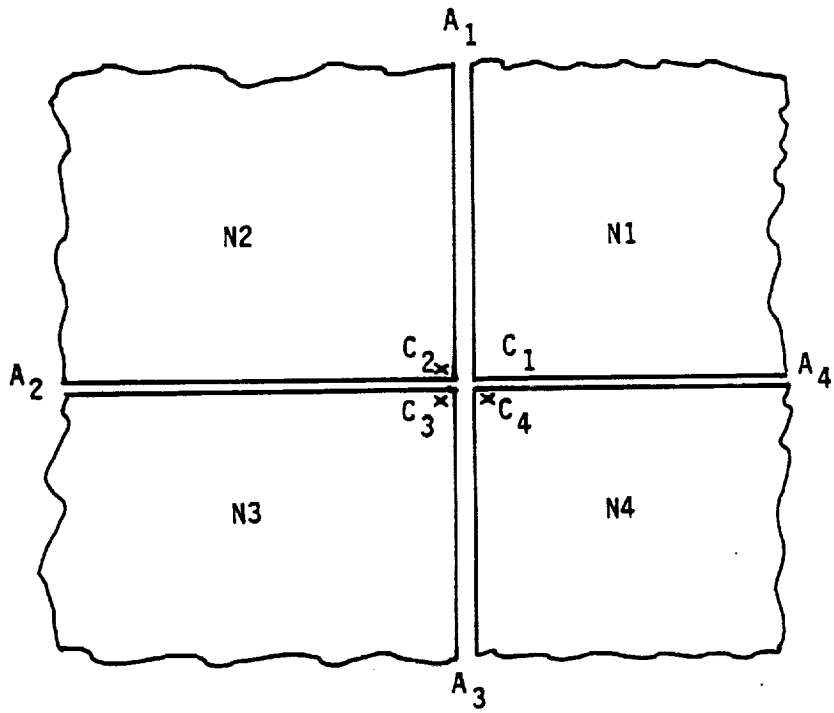


Figure 4-J.1 Example of an Abutment Between Two Network Edges



N = Network
A = Abutment
C = Corner Point
at Abutment Intersection

Figure 4-J.2 Example of an Abutment Intersection

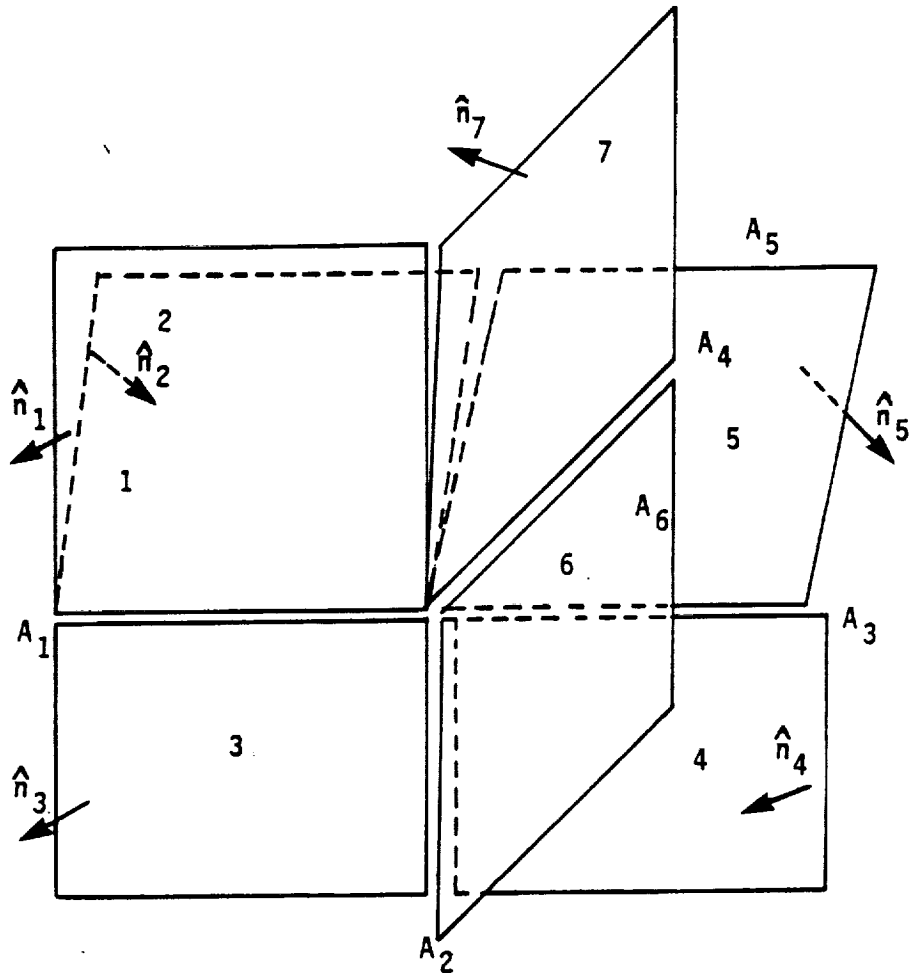


Figure 4-J.3 An Abutment Intersection with 6 Abutments

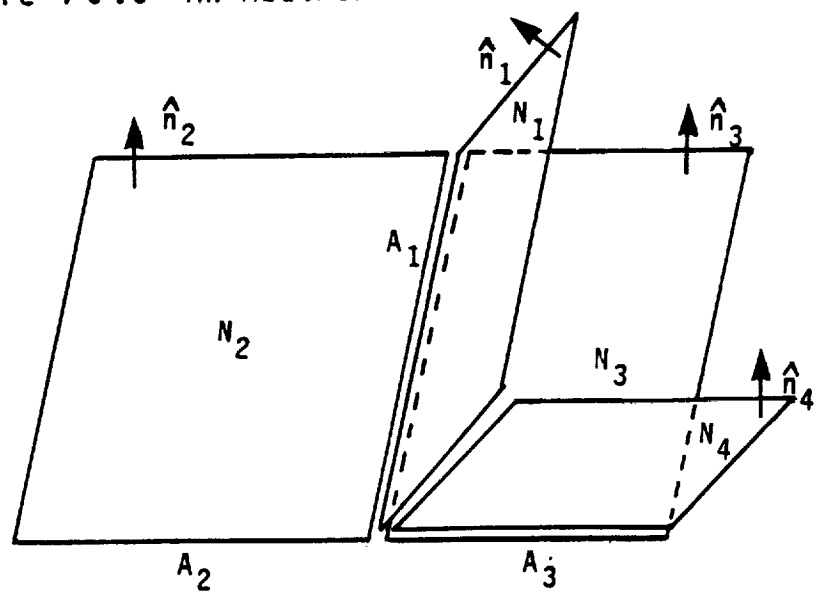


Figure 4-J.4 Another Abutment Intersection with 4 Abutments

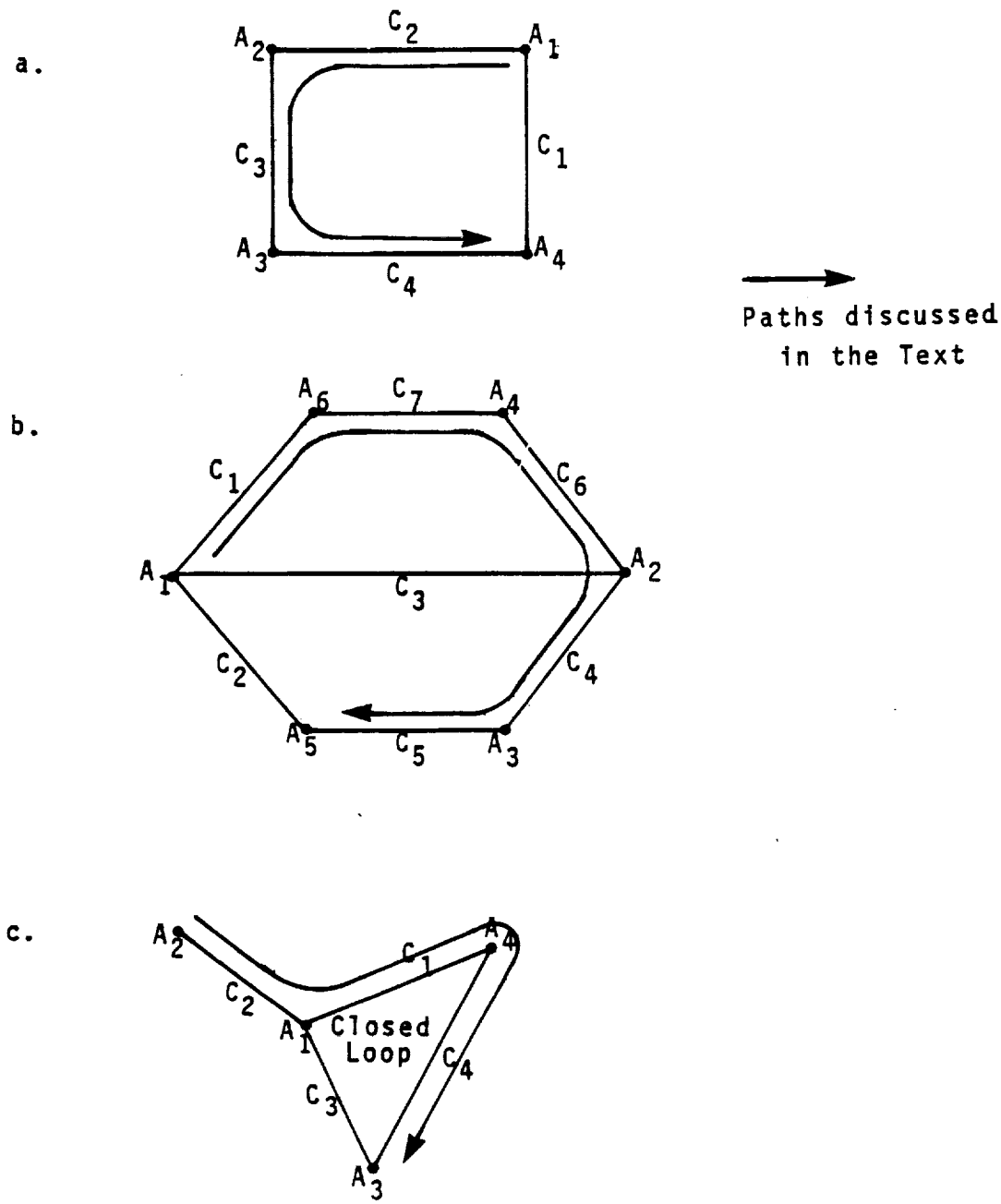


Figure 4-J.5 Line Segment and Point Diagrams Corresponding to Three Abutment Intersections

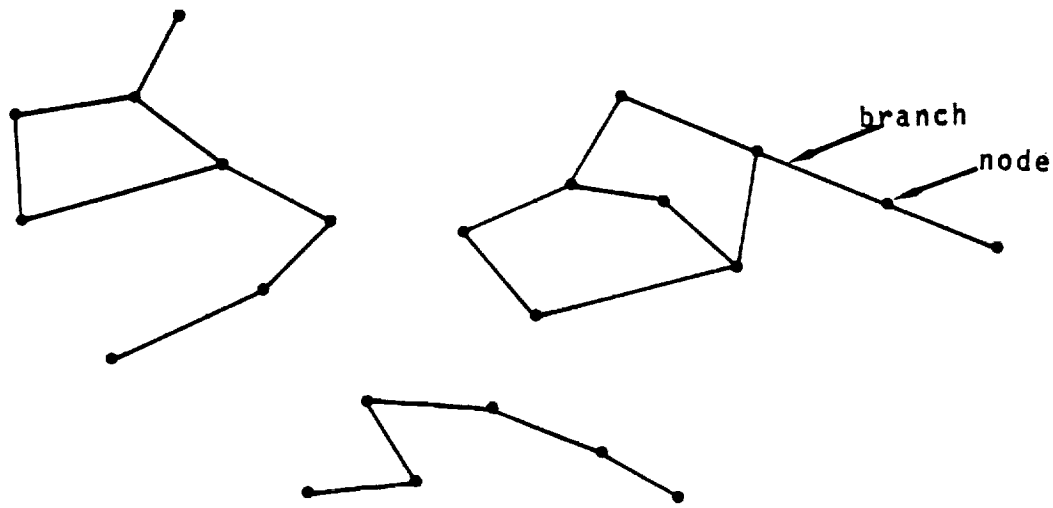


Figure 4-J.6 An Example of a Graph

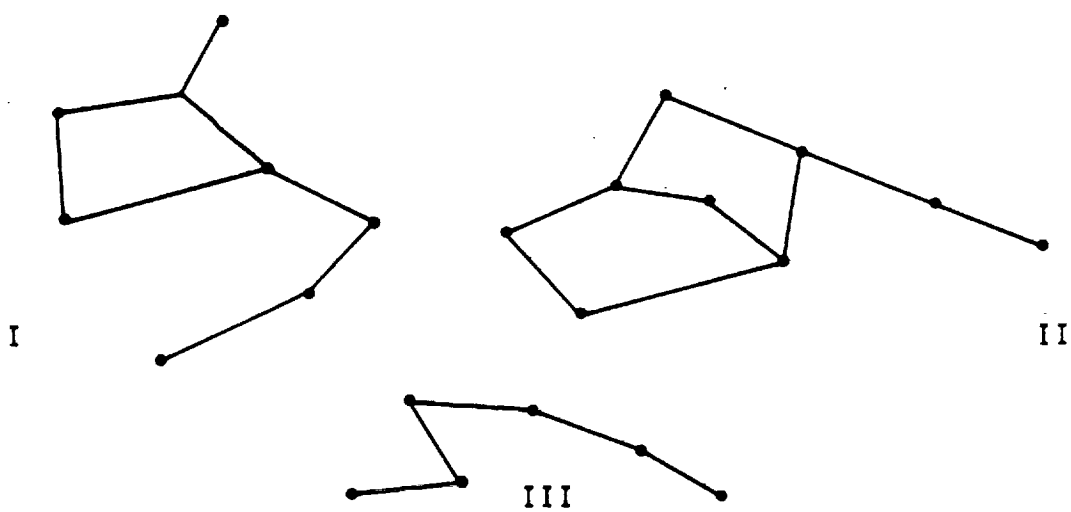
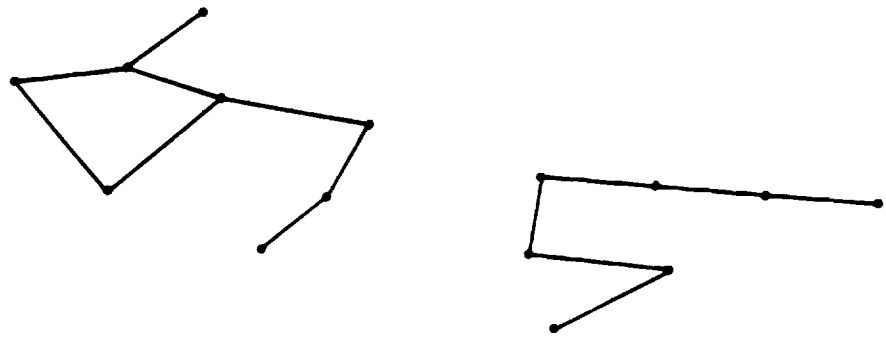
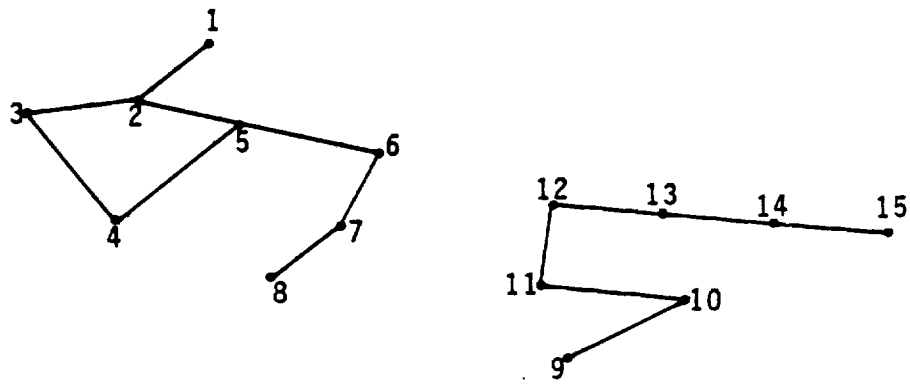


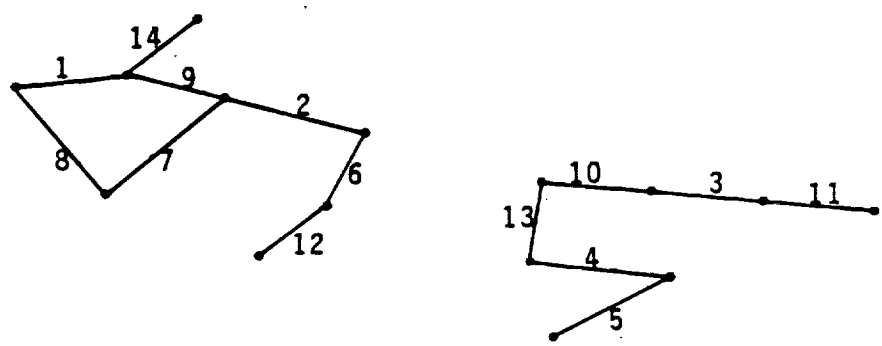
Figure 4-J.7 Illustration of Irreducible Subgraphs



a) The Unlabeled Graph



b) Labeling of Nodes



c) Labeling of Branches

Figure 4-J.8 Assignment of an Index to All Branches and Nodes of a Graph

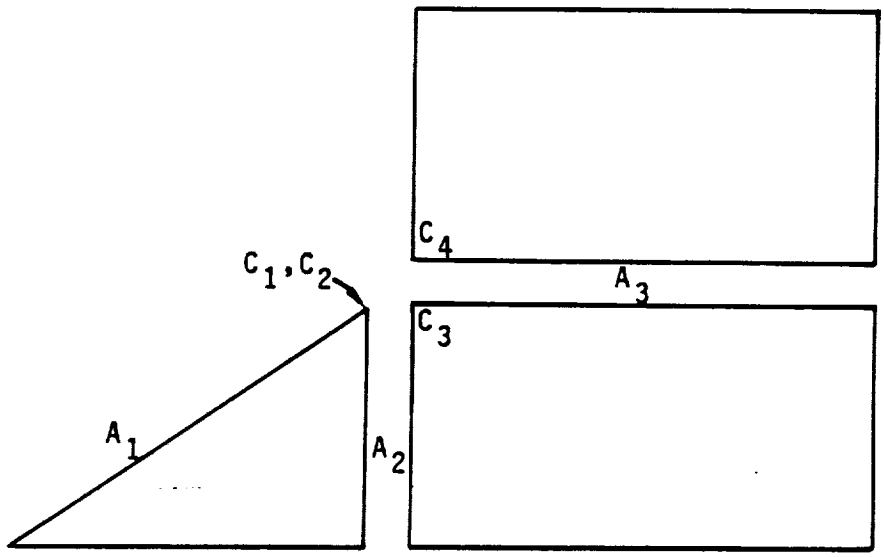


Figure 4-J.9 Abutment Intersections at Collapsed Edges of Networks. Without special processing the intersection of abutments A1 with A2 and A3 would be missed. The special processing is described in Section 4-J.3

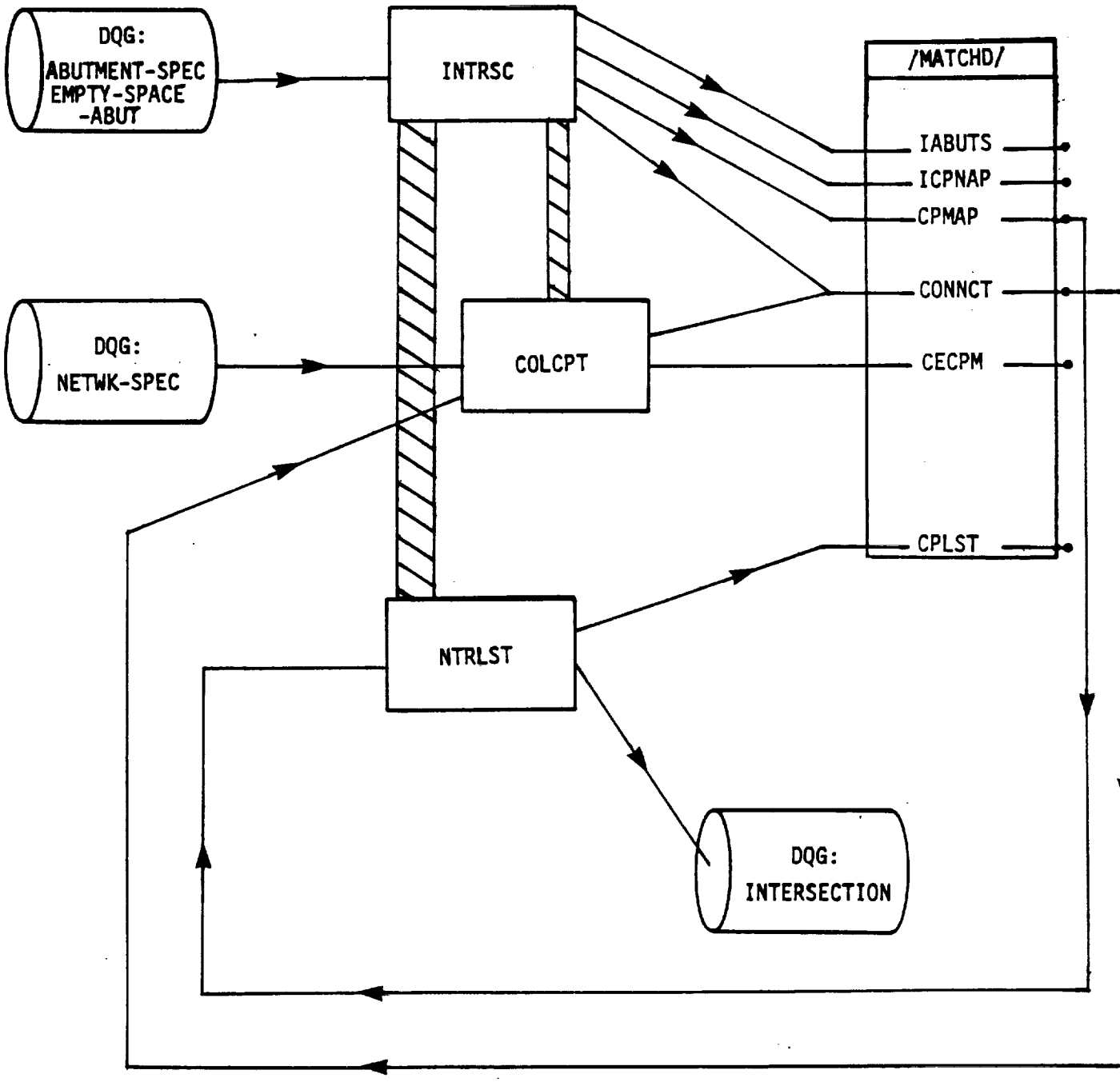


Figure 4-J.10 Data Flow and Program Operation for Intersection Construction

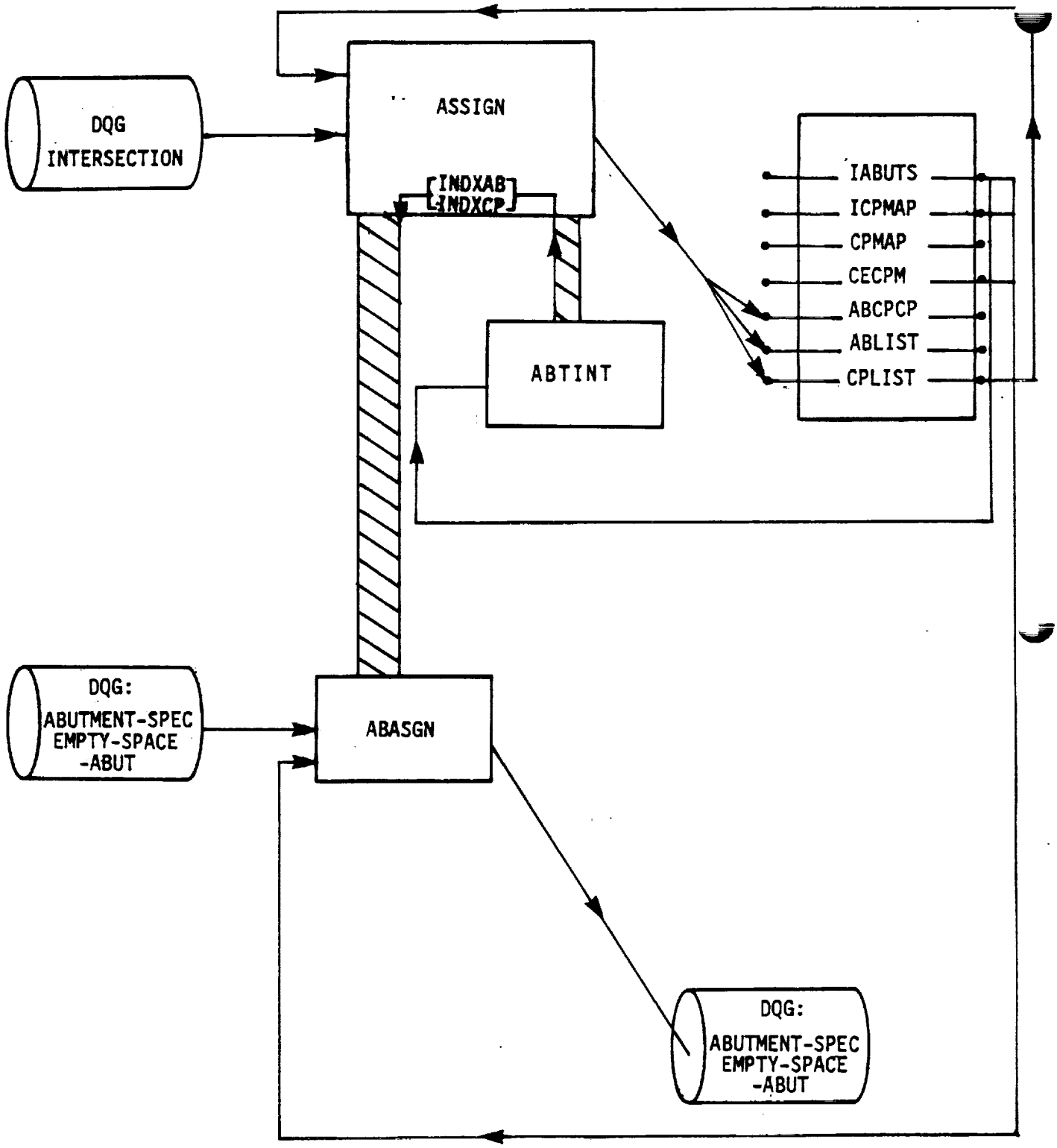


Figure 4-J.11 Data Flow and Program Operation for Matching Assignment

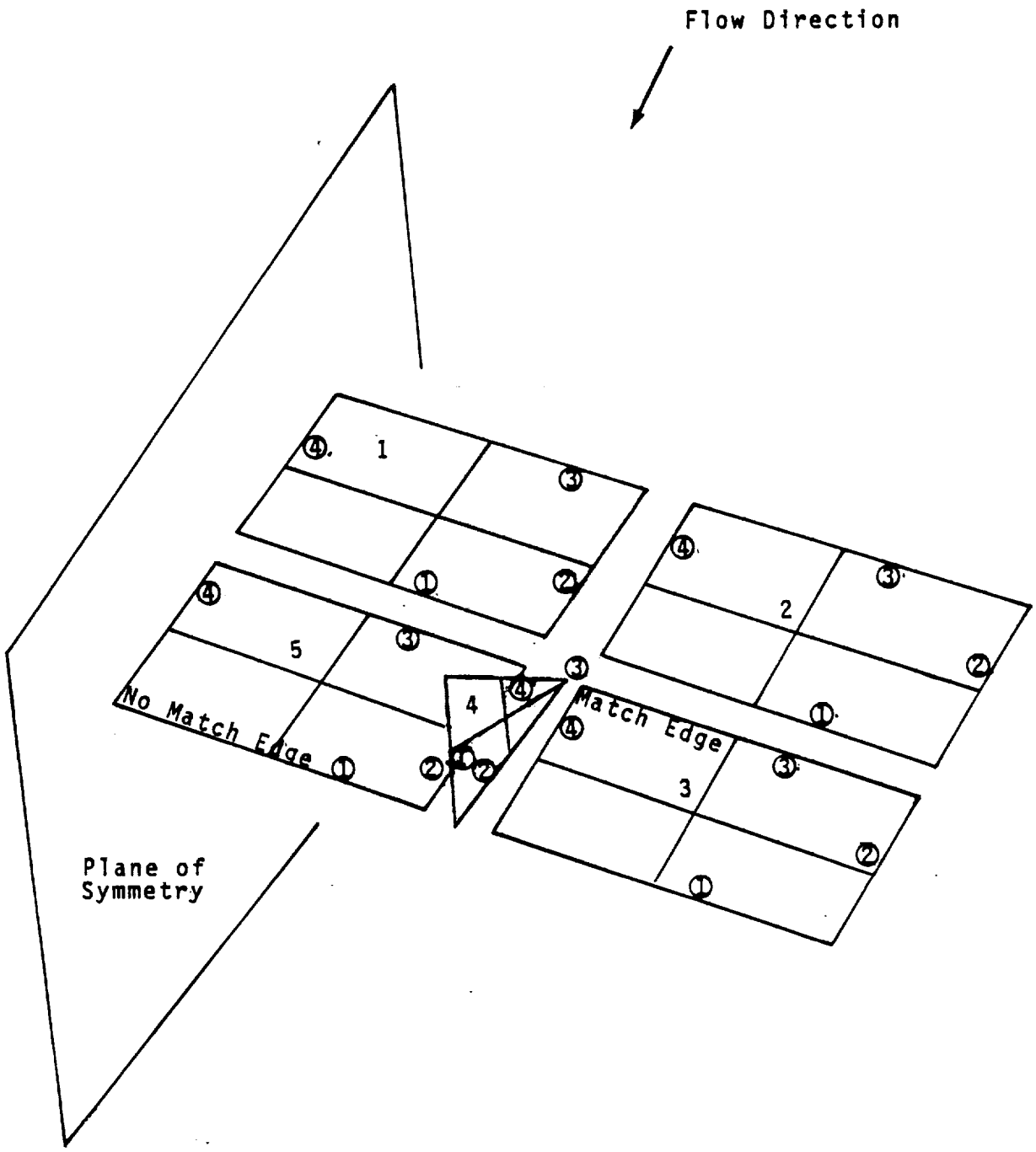


Figure 4-J.12 Configuration for Example of Abutment Intersection Search

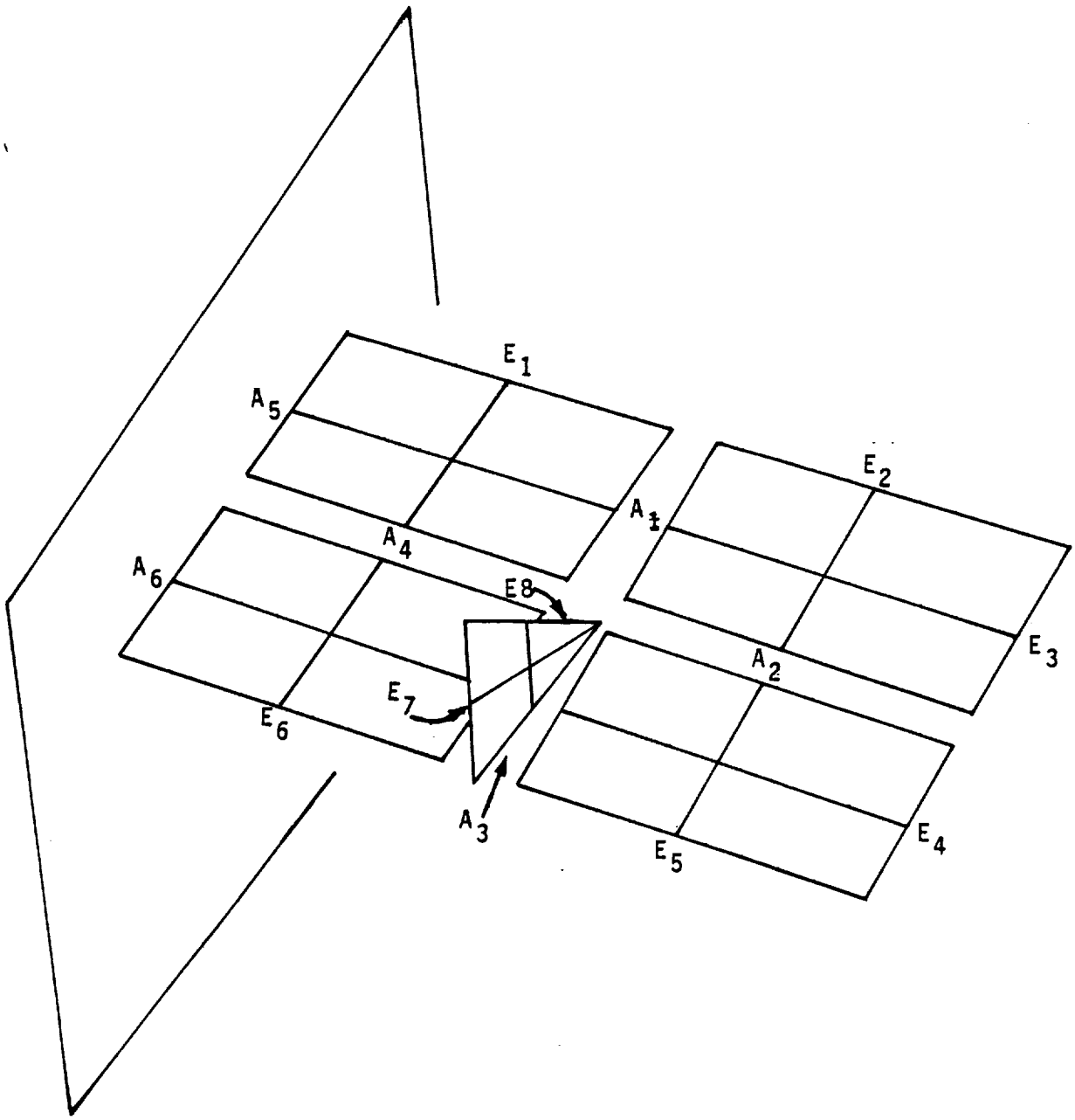


Figure 4-J.13 Abutments in Example Configuration

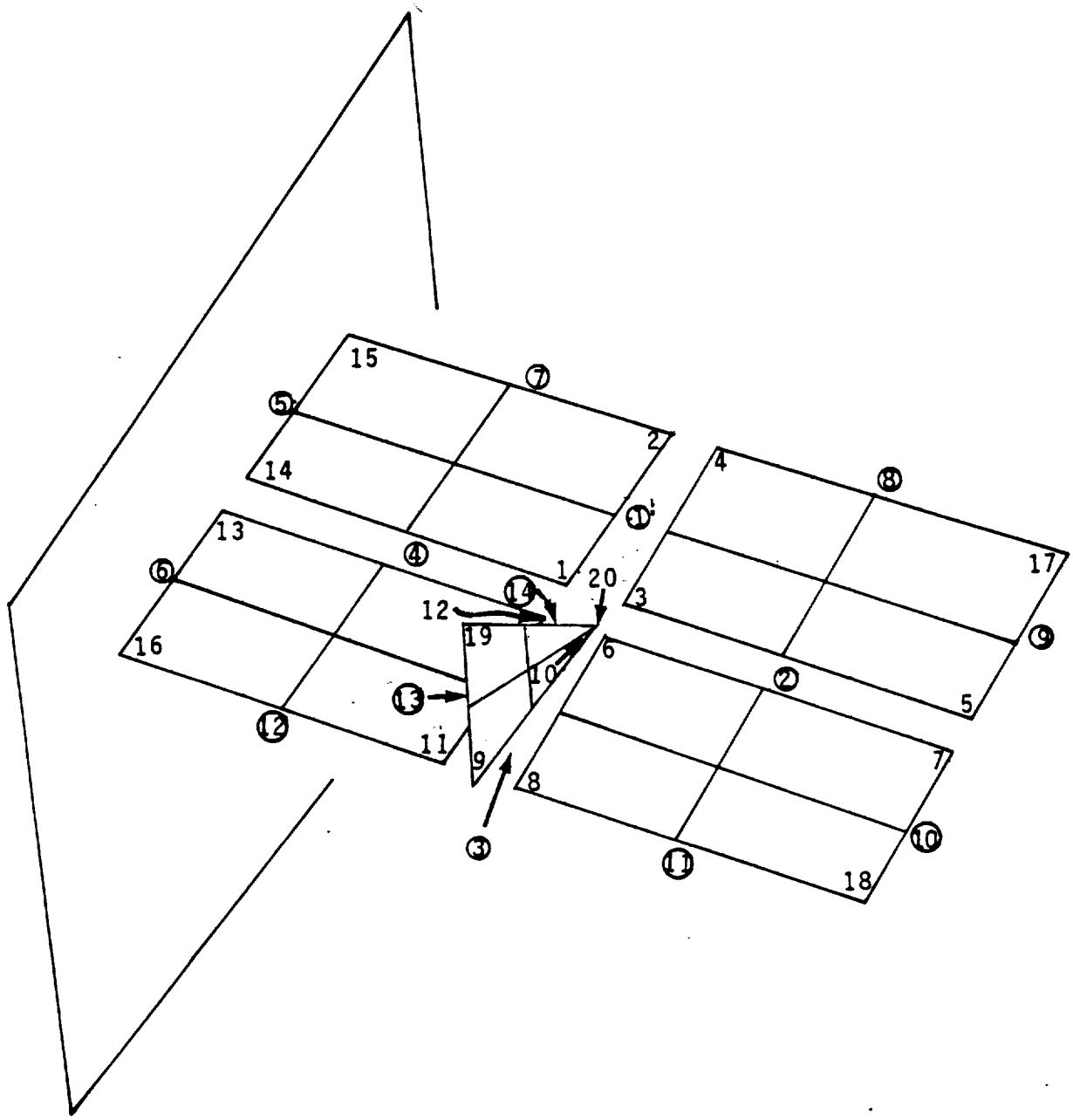


Figure 4-J .14 Abutment and Corner Point Indexing

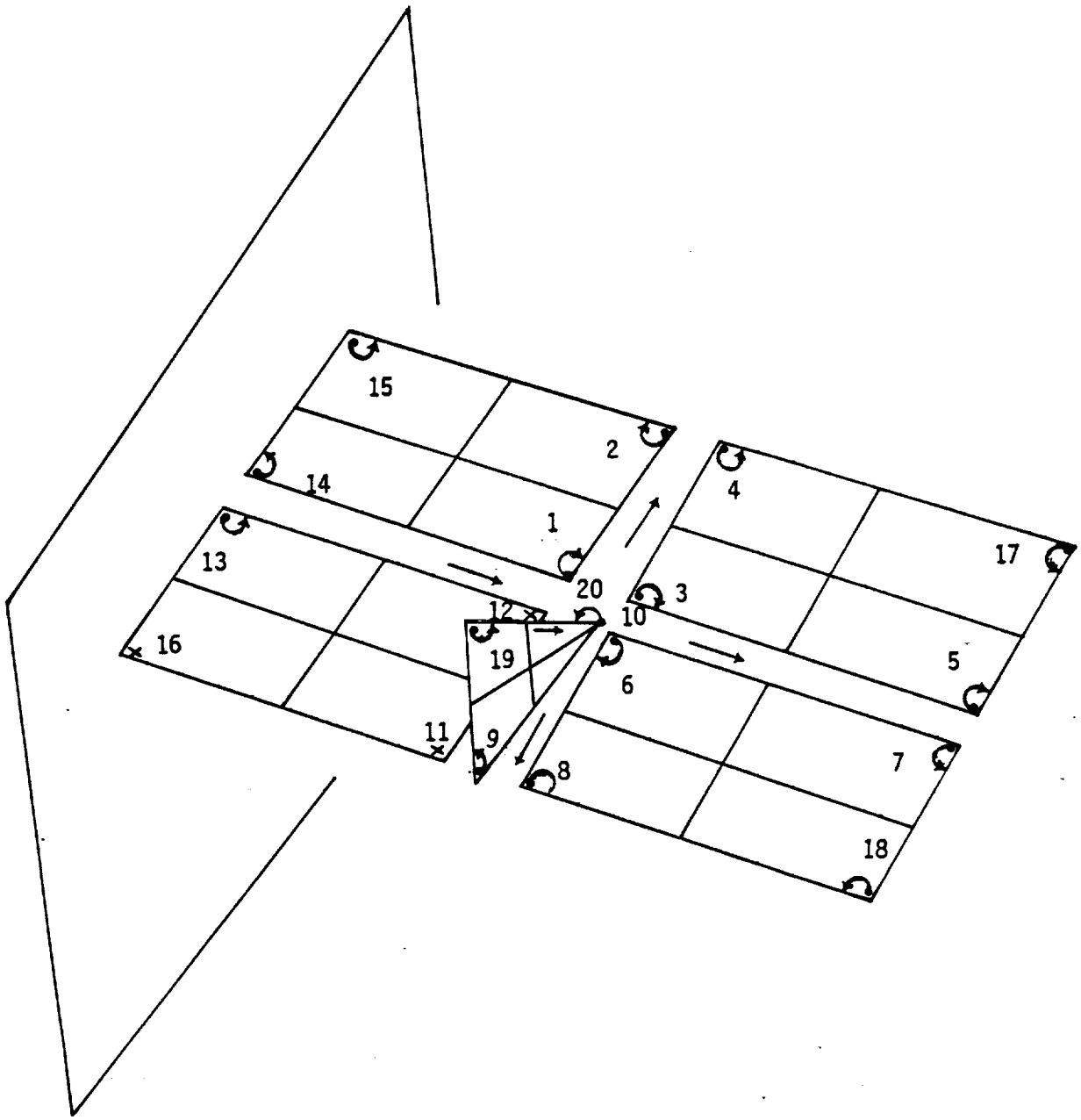


Figure 4-J.15 Doublet Matching Assignments at the Conclusion of the Abutment Intersection Analysis at the End of the (3,4) Overlay of DQG.

APPENDIX 4-K

OUTER SPLINE CONSTRUCTION

4-K.0 Introduction

The fifth overlay of DQG computes the [SP] vectors (see PAN AIR Theory Document, Appendix I (Reference 1)) at every corner point, center point and edge mid-point in all of the networks of the configuration. These vectors define source and doublet strength at the nine defining points of any particular panel in a network in terms of the singularity parameters located in that network, or, (in the case of a smooth abutment) in terms of singularity parameters located in an adjacent network. In the sixth overlay of DQG these spline vectors are used to assemble the spline matrices ($[B^S]$ and $[B^D]$ matrices). For source analysis networks the $[B^S]$ matrix is computed separately. It does not use a unique [SP] vector for each grid point and therefore does not impose source continuity across panel boundaries. Also in the sixth overlay the subpanel spline matrices ($[SPSPL^S]$ and $[SPSPL^D]$) are computed from the panel geometry.

Source and doublet strengths over the surface of a network are defined by a complicated series of spline operations which are discussed from a theoretical point of view in the PAN AIR Theory Document, Appendix I (Reference 1). The fifth and part of the sixth overlays of DQG compute the splines in several steps.

This appendix discusses mainly the calculation of the outer spline vectors $[SP^S]$ and $[SP^D]$. Section 4-K.1 also discusses, however, how the outer spline vectors are assembled to form the spline matrix for a panel. The coding of the subpanel spline construction is straightforward and its implementation is not discussed.

Appendix 4-K.1 discusses some general concepts and also discusses how the outer spline vectors [SP] are assembled into the spline matrix for a panel. Appendix 4-K.2 discusses the computation of doublet splines on network edges. Appendix 4-K.3 discusses the computation of doublet splines in network interiors. Appendix 4-K.4 discusses the construction of source spline vectors.

The concepts presented in this appendix are difficult. The readers are encouraged to study the PAN AIR Theory Document, Appendix I (Reference 1), where a more detailed discussion is given, in order to gain a more complete understanding of this appendix.

4-K.1 General Concepts

Before discussing the splining operations further, it is useful to introduce some definitions of items which will be referred to throughout the succeeding sections.

A spline vector is an one dimensional array with a dimension between one to twenty possible components. Its inner product with a vector consisting of values of singularity parameters in the vicinity of a point gives the value of singularity strength at that point. Associated with a spline vector is an index vector with the same dimension whose components are the singularity parameter indices (see Appendix 4-H) of the surrounding points.

A unit spline vector is usually defined for each point where a singularity parameter is located. It is a vector of dimension one with its component equal to unity and with its associated index vector equal to the index of the singularity parameter located at the point.

The splining process takes values of source or doublet strength at discrete surrounding points and defines the source or doublet strength at the point whose singularity strength is required as a linear combination of the strengths at the surrounding points. The coefficients of the linear combination are determined from a least squares fit (see PAN AIR Theory Document, Appendix I, (Reference 1)). Splines are sometimes computed to surrounding points whose doublet strength is not due to one singularity parameter but is itself another spline vector. The process of computing the spline vector which includes this more general case is called accumulating the spline vector.

Subroutine VECUNV performs the accumulation of the spline vectors. The input to the subroutine includes the number of surrounding points to which the spline is being performed, a spline vector for each of the surrounding points (usually of dimension one), an index vector for each singularity parameter (discussed below), the dimension of each spline vector and the set of coefficients from the least squares fit. The output consists of a spline vector, its dimension and an index vector for the spline vector. The index vector associated with a particular spline vector tells which singularity parameter to use to determine the value of singularity strength at the point. An example should clarify this concept.

Suppose that we have a spline vector [SP] = (0.3, 0.25, 0.8) with an index vector [ISP] = (23, 45, 21). The index vector means that the value of singularity strength at the point which the spline vector refers to is 0.3 multiplied by the value of singularity parameter number 23 plus 0.25 times the value of singularity parameter 45 plus 0.8 times the value of singularity parameter 21. Of course the values of the singularity parameters are not known until the AIC matrix has been inverted and applied to the right hand side in module RHS. Thus, it is necessary to maintain the list of index vectors to keep track of which singularity values to use to evaluate the singularity strength at an arbitrary point.

The problem to be solved in the abstract is the determination of the union of a set of vectors lying in separate but possibly overlapping subspaces. For example, if we have the three vectors:

$\llcorner X \lrcorner = (2, 0, 0)$

$\llcorner XY \lrcorner = (1, 4, 0)$

$\llcorner XZ \lrcorner = (1, 0, 2)$

Instead of the above representation each vector is separated into two vectors. One contains the non-zero components of the original vector; the other indicates which components are non-zero. Therefore, the above vectors are represented by:

$\llcorner X \lrcorner = (2)$	$\llcorner IX \lrcorner = (\text{"x"})$
$\llcorner XY \lrcorner = (1, 4)$	$\llcorner IXY \lrcorner = (\text{"x"}, \text{"y"})$
$\llcorner XZ \lrcorner = (1, 2)$	$\llcorner IXZ \lrcorner = (\text{"x"}, \text{"z"})$

where "X", "Y", or "Z" indicates the corresponding component is non-zero.

The construction of the union vector [R] consists of the linear combination:

$$\llcorner R \lrcorner = C_1 \llcorner X \lrcorner + C_2 \llcorner XY \lrcorner + C_3 \llcorner XZ \lrcorner$$

This is called the accumulation of the vectors. This union vector is constructed by examining the index of each component of each input spline vector in order to see if it already exists in the union vector. If it already exists in the union vector then the coefficient associated with the input vector is multiplied by the component of the vector and the result is added to the existing component of the union vector. If it does not exist in the union vector, a new component is added to the union vector and the coefficient times the component is added to the new component of the union vector. In the example discussed above the union vector [R] initially looks like:

$$\llcorner R \lrcorner = (2C_1) \quad \text{with} \quad \llcorner IR \lrcorner = (\text{"x"})$$

After processing the vector $\llcorner XY \lrcorner$ the union vector $\llcorner R \lrcorner$ looks like:

$$\llcorner R \lrcorner = (2C_1 + C_2,\ 4C_2) \quad \text{with} \quad \llcorner IR \lrcorner = (\text{"x"}, \text{"y"})$$

Finally after the third vector is processed the union vector looks like:

$$\llcorner R \lrcorner = (2C_1 + C_2 + C_3,\ 4C_2,\ 2C_3) \quad \text{with} \quad \llcorner IR \lrcorner = (\text{"x"}, \text{"y"}, \text{"z"})$$

Thus the union vector is as we would have expected if we had kept all the components of the vectors $\llcorner X \lrcorner$, $\llcorner XY \lrcorner$ and $\llcorner XZ \lrcorner$ and simply performed vector addition.

The construction of the spline matrix ($[B^S]$ and $[B^D]$) from the spline vectors ($\llcorner SP^S \lrcorner$ and $\llcorner SP^D \lrcorner$) in subroutine VECUNM is similar to the process of accumulating spline vectors except that the separate entries do not get added together, but rather are used to define a separate row of a matrix. An entry whose index does not appear in the matrix causes a new column to be added to the matrix. By special convention the last row of the matrix is used to

define the indices of the singularity parameters. Using the data in the previous example, the spline matrix [B] constructed from the ⌊X⌋, ⌊XY⌋ and ⌊XZ⌋ vectors is defined in the following steps. First, after processing the first vector ⌊X⌋, the matrix is a 1 x 1 matrix of the form:

$$[B] = \underset{\text{"x"}}{\begin{bmatrix} 2 \end{bmatrix}}$$

Then after processing vector ⌊XY⌋, the matrix gets an additional column:

$$[B] = \begin{array}{cc} \begin{bmatrix} 2 & 0 \\ 1 & 4 \end{bmatrix} \\ \text{"x"} \quad \text{"y"} \end{array}$$

Finally after the ⌊XZ⌋ vector is processed, the matrix becomes:

$$[B] = \begin{array}{c} \begin{bmatrix} 2 & 0 & 0 \\ 1 & 4 & 0 \\ 1 & 0 & 2 \end{bmatrix} \\ \text{"x"} \quad \text{"y"} \quad \text{"z"} \end{array}$$

Note that zeroes fill out columns of previous rows when a new column is created. The process is accomplished by subroutine VECUNM in the (6,0) overlay of DQG.

4-K.1.1 Local Coordinate Tranformations for Splining

The spline computations define variation of source or doublet strength over a two dimensional surface. Networks and panels do not have to be flat, however. One needs to define a coordinate transformation from the three dimensional space to the two dimensional surface on which subpanels are defined. Since the value of a spline vector at a point depends on the distances to the surrounding singularity parameters it is essential to define the coordinate transformation in a manner that approximately preserves the distance between two points.

This kind of coordinate transformation is required for all spline calculations at interior points of networks and on the coarse edge of smooth abutments. This is done in subroutines XIETAV, SPLTRN and LOC2D in the fifth overlay of DQG.

The relative two dimensional coordinate variation is defined in two stages. First a local coordinate system is defined with its origin at the point whose spline vector is required. The coordinate axes are defined in a manner to be described. They consist in general of a skewed set of two dimensional coordinates that lie in some local average plane of the network and a third component which is prependicular to the local plane. Subroutine XIETAV computes these vectors for most doublet splines and for all source splines. Subroutine LOC2D computes these vectors for smooth abutment doublet splines.

Having defined the local coordinate system, the three dimensional coordinates of the singularity parameter are transformed into two dimensional coordinates by transforming to the local plane coordinate system, truncating the coordinate normal to the local plane, and scaling the remaining two coordinates so that if a point lies a certain three dimensional distance from the origin, the two dimensional distance computed from the scaled two dimensional coordinates is the same as the three dimensional distance. This is done in subroutine SPLTRN.

The skewed axes of the local coordinate system are defined by defining two vectors from the four fine grid points adjacent to the point at which the spline vector is required. If the fine grid lattice indices of the point are (I,J), a vector XI is defined from point (I-1,J) to (I+1,J) and a vector ETA is defined from point (I,J-1) to (I,J+1). If the I+1 or J±1 points do not lie on the network, the lattice index I or J is used instead. These two vectors are not normalized to unit magnitude. A vector along the third coordinate axis, ZETA, is defined by the cross product of XI and ETA. The vector ZETA is defined as

$$\{ZETA\} = \frac{\{XI\} \times \{ETA\}}{SQRT\ (ABS\ (\{XI\} \times \{ETA\})**3/2)}$$

The coordinate transformation of a point P, is then defined as described in the PAN AIR Theory Document, Appendix I, Section I.1.2.3 (Reference 1).

4.K.1.2 Spline Computations

The computation of the spline involves three steps. The first is the selection of surrounding points to which the spline is performed. The second is the computation of a least squares fit to the surrounding points using a particular function. The final step is the accumulation of terms to define the spline vector at the required point. This first step is discussed in sections 4-K.2 and 4-K.3. The last step has been discussed in the previous section. The computation of the least squares fit is straight forward in most respects. This section discusses these straightforward concepts.

The particular funciton which is used in the fit depends on the singularity type of the spline. For example, a bilinear function of the form

$$S = a + bx + cy + dxy$$

is used for source splines. For doublets, a quadratic function

$$D = a + bx + cy + (1/2)dx**2 + exy + (1/2)fy**2$$

is used. The number of points to be fit, the x and y coordinates of the points to be fit, a weight for each point to be fit and the information about what functional form to use are provided as inpout to subroutine CQLSF of PALIB, the PAN AIR Library (see Section 12 of this Manual). The weighting of each point is determined as discussed in the PAN AIR Theory Document Appendix I(Reference 1). Certain points are assigned to be fit exactly. By convention, any point with a weight greater than 1. is fit exactly. The operations of CQLSF are discussed fully in Section 12 of this document.

4-K.2 Doublet Edge Splines

With regard to doublet splines there are four types of edges in PAN AIR: smooth edges, analysis edges, design edges and wake edges. Smooth edges define doublet distributions which have continuous derivatives across network boundaries. They are discussed in Section 4-K.2.1. Analysis edges are discussed in Section 4-K.2.2 and design edges in 4-K.2.3. These edges differ in the locations of their singularity parameters. There are two types of wake edges. The first, Wake I, is identical to analysis edges. Wake II edges are simply a constant distribution of doublet strength all along the edge. Wake II edges are discussed in Section 4-K.2.4.

4-K.2.1 Smooth Edge Splines

Smooth edge splines achieve approximate continuity of doublet derivative across network edges by defining spline vectors at points on the network edges which depend on doublet parameters in both networks. Only doublet analysis networks are permitted to take part in smooth abutments. All of the singularity parameters at edge midpoints on the two network edges are declared "null" and are not used in computing the smooth edge splines. Only singularities located at the panel centers (and at the corner points on the edges which mark the start and end of the smooth abutment) are used for defining spline vectors.

Figure 4-K.1 illustrates the situation at a typical smooth edge. One of the two network edges is declared the coarse edge, i.e., the one with fewest panels. Spline vectors for the panel corner points along the network edge and for the edge midpoints along the network edge are computed for the coarse edge in terms of singularity parameters located at center points in both networks. The other edge (the fine edge) then has spline vectors defined for it at each of its corner points and edge midpoints in terms of the spline vectors already computed at three adjacent points on the coarse edge.

Section 4-K.2.1.1 discusses some data structures used in performing the calculations. Section 4-K.2.1.2 describes how the spline vectors on the coarse edge are computed and Section 4-K.2.1.3 discusses how the spline vectors for the finer edge are computed.

4-K.2.1.1 Data Storage for Smooth Abutment Splines

Program SAEDGS, the (5,1) overlay of DQG, controls the computation of smooth edge splines. The first operation it performs is the storage of coordinates of corner point on and near the edges of the smooth abutment. These are stored in array CORPT (I,J,K). For K = 1 to 3, this array contains the coordinates of the first three rows (or columns) of corner points adjacent and parallel to the coarse edge. For K = 4 and 5, it contains the corner points on and adjacent to the fine edge (see Figure 4-K.2).

The second operation is performed by SNGFIL. This subroutine stores the singularity indices of the center points required for the fitting procedures. SINGDX(I,1) and SINGDX(I,2) contain the singularity indices of the two center point rows adjacent to the coarse edge. SINGDX(I,3) contains the indices of singularities adjacent to the fine edge. See Figure 4-K.2.

After storing this data, SAEDGS calls subroutine PARMSA. This subroutine

parameterizes both edges in the abutments. See PAN AIR Theory Documents, Appendix F, Section F.6 (Reference 1).

4-K.2.1.2 Coarse Edge Splines

The coarse edge spline is controlled by subroutine COARSP. First COARSP defines unit spline vectors (see section 4-K.1) at the start and end points. Then it computes spline vectors for all corner points on the coarse edge (except the start and end points). Figure 4-K.3 illustrates the surrounding points which are used in the spline calculations. If one of the required points run beyond the start or end point of the abutment, then the start or end corner point is used in its place. The points in the other network which are used in the fit are the two center points adjacent to the corner point on the fine edge which is closest to the point whose spline vector is required (see Figure 4-K.3). Four points are fit exactly: the two points on the finer network and the two closest center points on the coarse edge.

After the corner point splines are computed, the splines for the edge-midpoints are computed. Figure 4-K.4 shows the points selected for fitting on edge midpoint. To find the points on the finer edge, for each adjacent corner point on the coarse edge the closest corner point on the fine edge is found. Then for each center point adjacent to the corner point on the fine edge, the one most distant from the edge midpoint is used for the fit. Three points are fit exactly. For the first and last edge midpoint, the point selection runs outside of the start or end of the smooth abutment. In place of the point in the first row, the start or end point of the abutment is used. The point in the second row is simply omitted from the fit.

Subroutines SALSQC and SALSQE select the surrounding points for the corner points and edge midpoints. The process of selection means that the spline vector at that point (in this case all unit spline vectors) is stored in the array BSPL and INDX (coefficient and singularity index), the coordinate of the point is stored in COORD, a local two dimensional coordination system is defined by LOC2D and the coordinates of the selected points are transformed into the local coordinate system. Finally an upstream weighing factor is defined for those points which are part of the least squares fit. This information is stored in common block /LSQ/.

A library subroutine CQLSF is used to compute the solution: the coefficient corresponding to each point which is fit (see Section 4-K.1.5). Then subroutine VECUNV multiplies each spline vector by its coefficient and accumulates the product into array BSPLIN. After the spline vector is accumulated by VECUNV, it is written to the DQG data base with a key set consisting of the network index of the coarse network and the fine grid lattice indices of the point which has been splined (see Section 4-H.2).

In addition, a second data set is written to the DQG data base. This contains an alternate spline vector for use when computing spline vectors for interior points close to the smooth edge. Figure 4-K.5 illustrates points which would use one of the corner points or edge midpoints on a smooth edge for the calculation of its spline vector. If the spline vector for the point on the smooth edge were used, panels in the vicinity of the interior point would depend on too many doublet parameters. (The Partitioned Random Column Method employed in module MAG for the Influence Coefficient matrix construction assumes that panels depend on at most thirty-one singularity

parameters. See Section 5 of this document.) The alternate spline vector for the point on the smooth edge is the spline vector of the closer of the two center points in the fine network which were used to define the original spline vector for the point on the edge. The alternate spline vector data set includes the coordinates of the center point on the fine edge. Section 4-K.3 describes the splining procedures on the network interior.

4-K.2.1.3 Fine Edge Splines

After all spline vectors on the coarse edge have been computed, subroutine FINESP computes spline vectors for points on the fine edge in terms of those on the coarse edge. For each point on the fine edge, two adjacent corner points on the coarse edge are found whose parameterizations span the parameterization of the point on the fine edge (Figure 4-K.6). The spline vectors of these two points and of the edge midpoint between them are used to construct the spline vector for the point on the fine edge. A one dimensional quadratic fit is made to the three points on the edge by subroutine Q1DFIT. This yields a set of three solution coefficients which are used to scale the spline vectors for the three points on the coarse edge. The result is accumulated by VECUNV to form the spline vector for the point.

An alternate spline vector (see Section 4-K.2.1.2) is written for each point on the fine edge. The alternate point is the center point of the panel on the coarse edge whose points were used to define the spline vector for the fine edge point (see Figure 4-K.6).

4-K.2.2 Analysis Edge Splines

Analysis edges have a singularity parameter located at edge midpoints of network edge segments and at the start and end points of the edge segment. (An edge segment is that portion of a network edge which takes part in a single abutment. It can be as small as one panel (two corner points) or as large as the whole network edge.) See Figure 4-K.7.

The spline procedure is relatively simple. A unit spline vector is written for the start and end corner points of the segment. Then a spline vector is constructed for each other corner point in the segment. This spline vector is constructed from the singularity parameters located at the two adjoining edge midpoints. If d_1 and d_2 are the distances to the two edge midpoints (Figure 4-K.8), the spline vector and its index vector are:

$$B = \left(\frac{d_2}{d_1+d_2}, \frac{d_1}{d_1+d_2}\right)$$

$$I = (I_1, I_2)$$

(I_1 and I_2 are the singularity indices of the edge midpoints.)

After spline vectors are written for all the corner points on the edge segment, the edge midpoint spline vectors are constructed. The spline vectors for the two corner points adjacent to the edge midpoint are read from the data base and a unit spline vector is defined for the singularity parameter located at the edge midpoint. The three spline vectors are accumulated with coefficients of 1/4, 1/4, and 1/2 for the two corner points and the edge midpoint respectively. This defines the spline vector for the edge midpoint. Thus the spline vectors at corner points depend on two singularity parameters

and the spline vectors at edge midpoints depend on three singularity parameters (see Figure 4-K.8).

If a network edge collapses to a point, an alternate procedure is employed. The unit spline vector for the first corner point on the collapsed edge (in a counterclockwise sense) is read from the data base and is written as the spline vector for every point on the collapsed edge. Figure 4-K.9 illustrates which point is used for the unit spline vector.

There is one subtle matter regarding analysis edge splines. This has to do with the relationship between the lattice index system of labeling points and the sequential, counterclockwise sense, manner of numbering points on the edge. The edge segment is described by the coarse grid lattice indices of its start and end points. These may be in an increasing lattice index direction or a decreasing lattice index directon from start to end. The coordinates of the corner points on the edge are stored in counterclockwise sense sequential order. Thus both the fine grid lattice index of the corner point and the counterclockwise sequential index of the point are passed to subroutine CPANAL.

Within CPANAL is an array IADJEM which, when added to the lattice indices for the corner point, gives the lattice indices of the edge midpoint next to the corner point in a clockwise sense. The first edge midpoint coordinates in the distance calculation are those for the edge midpoint just before the corner point, i.e., the point next to the corner point in a clockwise sense.

Thus the point selection proceeds in a counterclockwise direction starting with the most clockwise point. This procedure is necessary so that the singularity indices for the edge midpoint are assigned to the right component of the spline vector.

Figure 4-K.10 illustrates the order in which the edge midpoint singularity indices are obtained for points on the four edges.

4-K.2.3 Design Edge Splines

Design edges have a singularity parameter located at every corner point on the edge segment (Figure 4-K.11). The spline procedure for design edges is a bit more involved than that for analysis edges. This is necessary for reasons of stability (see PAN AIR Theory Document, Appendix I (Reference 1)).

Unit spline vectors are defined for the start and end points of the segment. Then the edge segment is parameterized (see PAN AIR Theory Document, Section 6 of Appendix F.6 (Reference 1)). Then an intermediate spline vector is computed for each edge midpoint in the segment. The intermediate spline vector is called the gamma vector (see PAN AIR Theory Document, Section 1.4 of Appendix I (Reference 1)). It is computed by performing a one dimensional quadratic fit to the four singularity parameters located at the four adjacent corner points on the edge (see Figure 4-K.12). If an edge midpoint is too close to the start or end point of the segment, then only three parameters are used in the fit. The two closest points to the edge midpoint are fix exactly. The end points (when they exist) are fit in a least squares sense. The weights defined for the extreme points are from the same upstream weighing algorithm described in the PAN AIR Theory Document, Section 1.2.4 of Appendix I (Reference 1), with the origin defined to be the coordinate of the edge midpoint.

After the intermediate spline vectors have been written to the data base, spline vectors for corner points are defined in a manner similar to the definition for analysis edges except that instead of using singularity parameters at edge midpoints in the spline, the intermediate spline vectors at the edge midpoints are used. Finally, after all corner point spline vectors have been written to the data base, the intermediate spline vectors at each edge midpoint is replaced by the "true" edge midpoint spline vector which is calculated in a fashion similar to the analysis edge spline. It is the accumulation of 1/4 times the spline vector for each adjacent corner point and 1/2 times the intermediate spline vector at the edge midpoint.

This design edge spline has not been proven to be stable (see PAN AIR Theory Document, Appendix I (Reference 1)). For this reason an alternate splining technique has been defined within DQG by means of the DEFINE option of UPDATE (see Section 4-E). If a *DEFINE RESERV is inserted in an UPDATE and compilation of DQG, subroutines NTEDGD and CPDSGN will be modified so that the intermediate edge midpoint spline vector will become the final spline vector and unit spline vectors will be written for each corner point on the edge.

If this version of design splines is used, it should ony be employed for problems where any network edge which meets a design edge has the same paneling density.

4-K.2.4 Wake Edge Splines

There are two types of wake networks in PAN AIR. The first, WAKE I, has singularity parameters located on one edge called the matching edge. The locations of the singularity parameters is the same as for an analysis edge. The construction of spline vectors for this edge proceeds in the same fashion as in Section 4-K.2.2. The second kind of wake network, WAKE II, has a single singularity parameter located at one of the four network corner points. The edge for which that point is the first corner point in a counterclockwise sense is called a matching edge. A unit spline vector is defined for the corner point with the singularity parameter and the same unit spline vector is written for each corner point and edge midpoint on the matching edge. This defines a constant doublet strength along the edge.

4-K.3 Doublet Splines for Network Interiors

4-K.3.1 Analysis and Design Network

Figures 4-K.13 to 4-K.18 illustrate the general pattern of point selection for the interior of analysis and design networks. Figures 4-K.19 to 4-K.24 illustrate how the point selection proceeds if the required point is not on the network, that is, if skipping the required number of steps in the column and row index directions moves a point on or over the edge of the network.

A single procedure was developed to handle the point selection for all of these cases. It makes use of the fine grid lattice indices of points in the network (see Section 4-H). The different cases are handled by passing to the subroutine DATANL different values of arguments.

The arguments are: ISH(2), a set of "shift" lattice indices; LIMXY(2), a pair of limits in the X and Y lattice directions; OVF(2), a set of pointers to be used in the case that the lattice indices overflow or underflow; MAXXY(2),

the limits of the fine grid lattice indices for the network; and LATXY(2), the fine grid lattice indices of the point whose spline vector is required.

A reference point lattice index is defined as the vector sum of the lattice indices of the point whose spline is required with the array ISH. (It does not matter if this point is off the network.) See Figure 4-K.25. A loop on I and J is set up from 1 to LIMXY(1) and from 1 to LIMXY(2) respectively. Lattice indices of the surrounding point are defined as:

```
LATX = IREF(1) + 2*(I-1)
LATY = IREF(2) + 2*(J-1)
```

If the loop indices are 1 or LIMXY(1) (for I), or 1 or LIMXY(2) (for J), the point is skipped. No spline data is defined for it. Otherwise, the LATX and LATY variables are passed to LSQDAT which generates the spline vector at that point, its index vector and the coordinate of the point.

If the variables LATX and LATY are less than one (underflow) or greater than MAXXY(1) or MAXXY(2) respectively (overflow), they are set to either 1 or MAXXY(1) or MAXXY(2). The overflow (OVF(1) or OVF(2)) is added to the lattice indices. In some cases, if a point overflows it is omitted from the fit. This is indicated in Figures 4-K.14 to 4-K.24. If both loop indices are not equal to 1 and not equal to LIMXY, the point is defined to be fit exactly.

Table 4-K.1 defines the correct values for the arrays ISH, OVF and LIMXY in the six cases. The reader is urged to verify that the values yield the required set of points in the various situations.

Several exceptions are introduced to the procedure for handling some special cases.

If an edge of a design network collapses, the splines for center points in the column or row adjacent to the collapsed edge will be singular since two identical points are fit exactly. For this reason, if a point is the second point on a collapsed edge, no exact constraint is defined for this point. (Identical points may appear in the least squares portion of the fit without difficulty.)

If the fitting procedure were followed for one column or one row network, the spline would also be singular for simple geometries since all points would lie on two straight lines. Additional points are added to the spline to prevent this (see Figure 4-K.26).

If a point on the network edge which is selected as one of the surrounding points and that point is part of a smooth abutment, then alternate spline vector data are read from the data base. These alternate data defines a spline vector and coordinate of a point on the other network with which the smooth abutment is formed (see Section 4-K.2.1.3). If this procedure were not followed, panels near the smooth edge would depend upon too many singularity parameters (| 31) for the Partitioned Random Column method in the MAG module to function.

4-K.3.2 Wake Networks

Wake networks have constant doublet strength in a direction perpendicular to the matching edge. Thus for each point on the matching edge, the spline

vector is read from the DQG data base. Then for every point in the column (row) perpendicular to the matching edge row (column), the spline vector for the edge point is written with a key set containing the fine grid lattice indices (see Appendix 4-H) of the interior point.

4-K.4 Source Spline Point Selection

Selection of points for source splines is much simpler than the doublet procedures. Figure 4-K.27 illustrates the points selected for source analysis network splines. Point selection is similar to that in Section 4-K.3. A reference point is defined by shifting the lattice indices of the point whose spline is required. Then spline vectors for the four surrounding points are obtained for the fit. Note that if any of the points overflow or underflow, the reference point is shifted so that all of the points are on the network. Spline vectors for source analysis networks are defined for corner points and center points.

One panel column or one panel row networks are a special case. Splines are constructed from the two nearest center points. In this case the source strength is a linear instead of bilinear function. See Figure 4-K.28. Source design I networks have singularity parameters located at every corner point. Unit spline vectors are written for every corner point. Spline vectors for center points are defined to be of length four with equal amplitudes of 1/4, and index vector consisting of the four singularity parameter indices at each corner of the panel.

Source design II networks have singularity parameters located at panel edge midpoints. One panel column or one panel row networks are not allowed as Source design II networks. Spline vectors for center points are of length two and each component has an amplitude of 1/2. The spline vectors for edge midpoints located on panel edges transverse to the matching edge are defined by the four surrounding singularity parameters. Figure 4.K-29 shows the pattern of point selection for the spline vectors of a source design II network.

Table 4-K.1 Values for arrays ISH, LIMXY and OVF.
These are used to select surrounding singularity parameters for design and analysis networks and for the varouts types of points for which splines are computed.

	.....Analysis.....			.....Design.....		
	Corner	Column Edge Midpt	Row Edge Midpt	Center	Column Edge Midpt	Row Edge Midpt
ISH	(-3, -3)	(-2, -3)	(-3, -2)	(-3, -3)	(-3, -2)	(-2, -3)
LIMXY	(4, 4)	(3, 4)	(4, 3)	(4, 4)	(4, 3)	(3, 4)
OVF	(1, 1)	(0, 1)	(1, 0)	(0, 0)	(0, 0)	(0, 0)

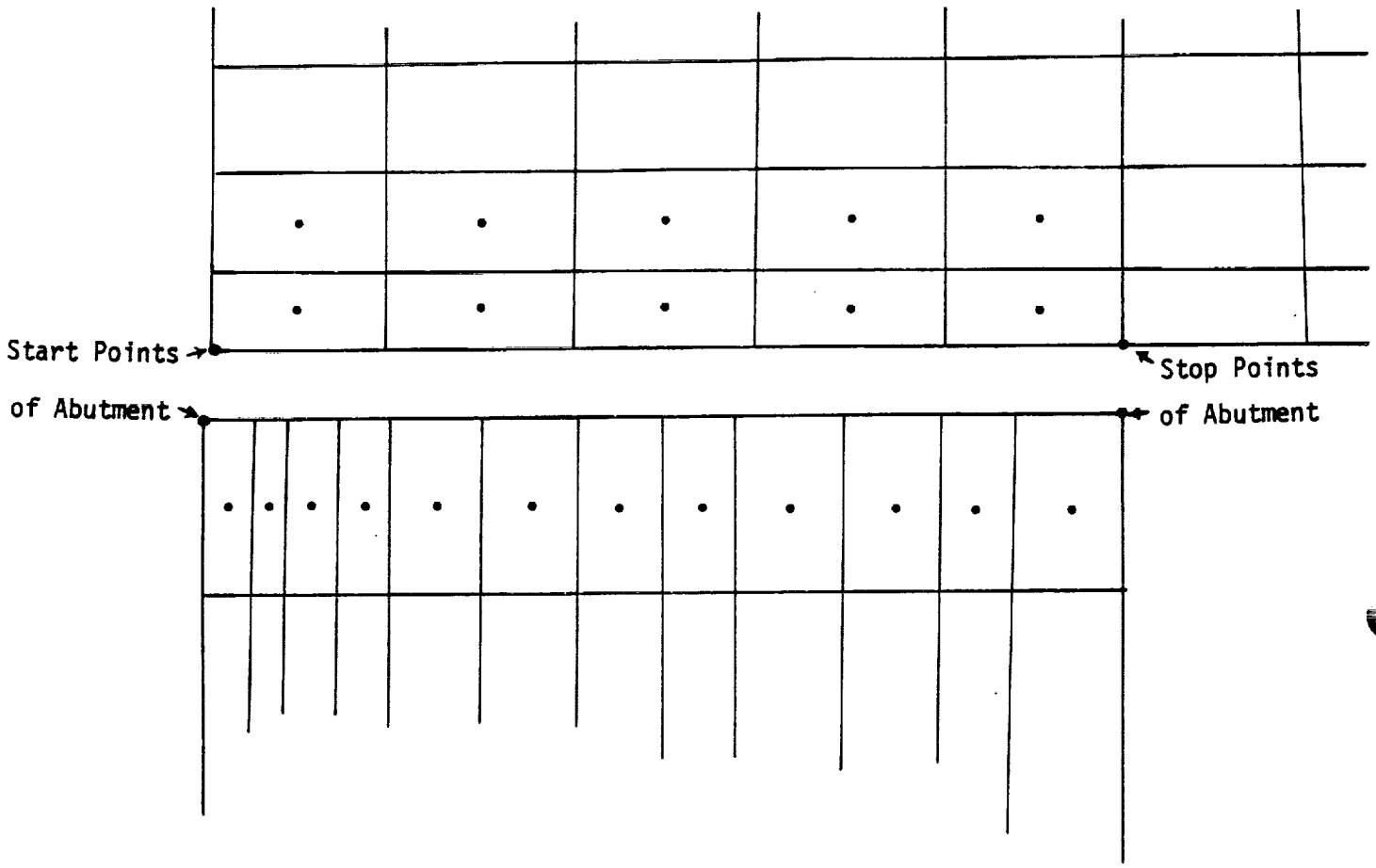
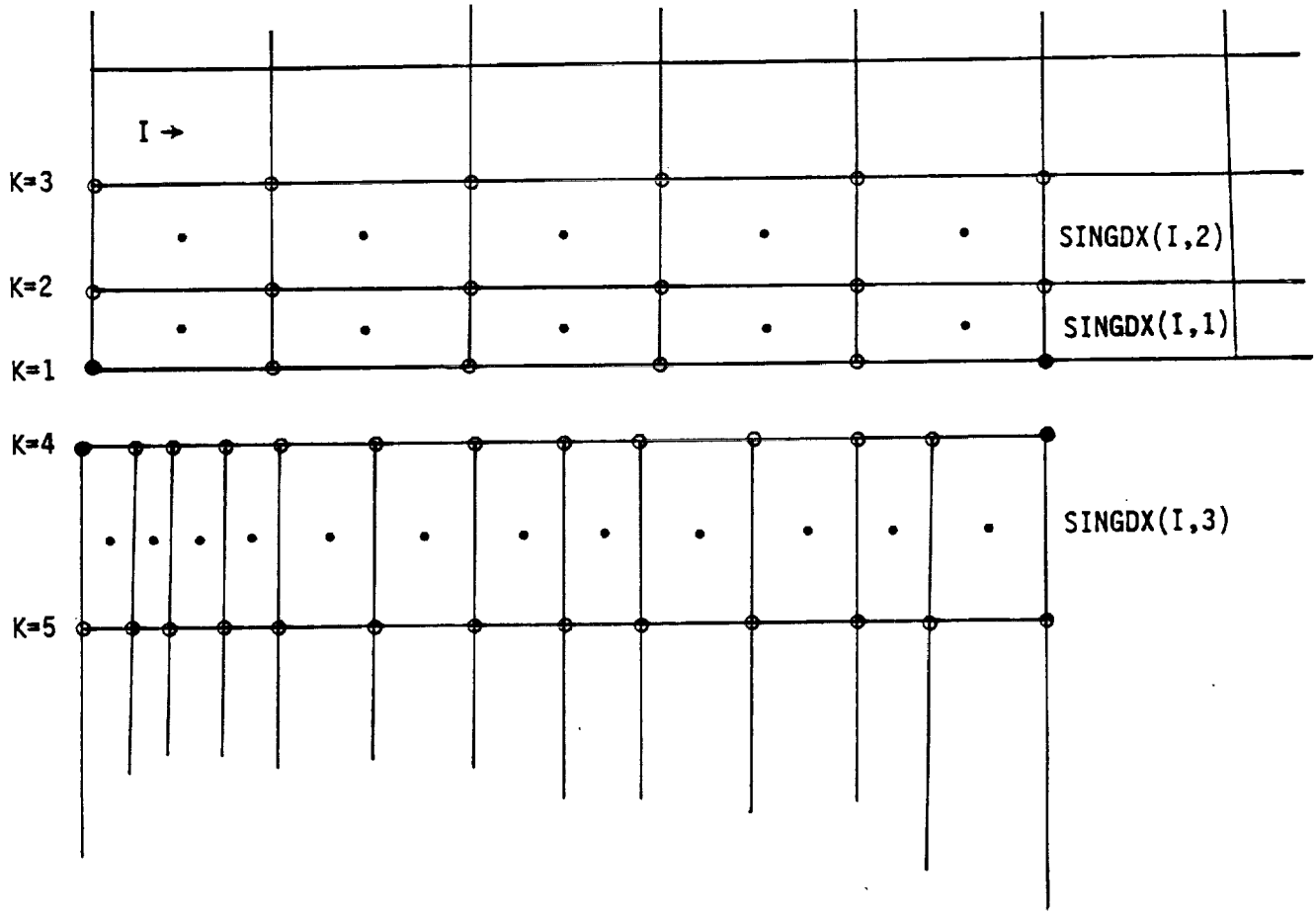


Figure 4-K.1 Singularity Parameters Used for Smooth Abutment Spline Calculations

Coarse Edge Network



Fine Edge Network

● Singularity Parameter Stored in Array SINGDX(I,J)

○ Corner Points Stored in Array CORPT(I,J,K)

Figure 4-K.2 Storage at Corner Point Coordinates and Singularity Parameter Indices

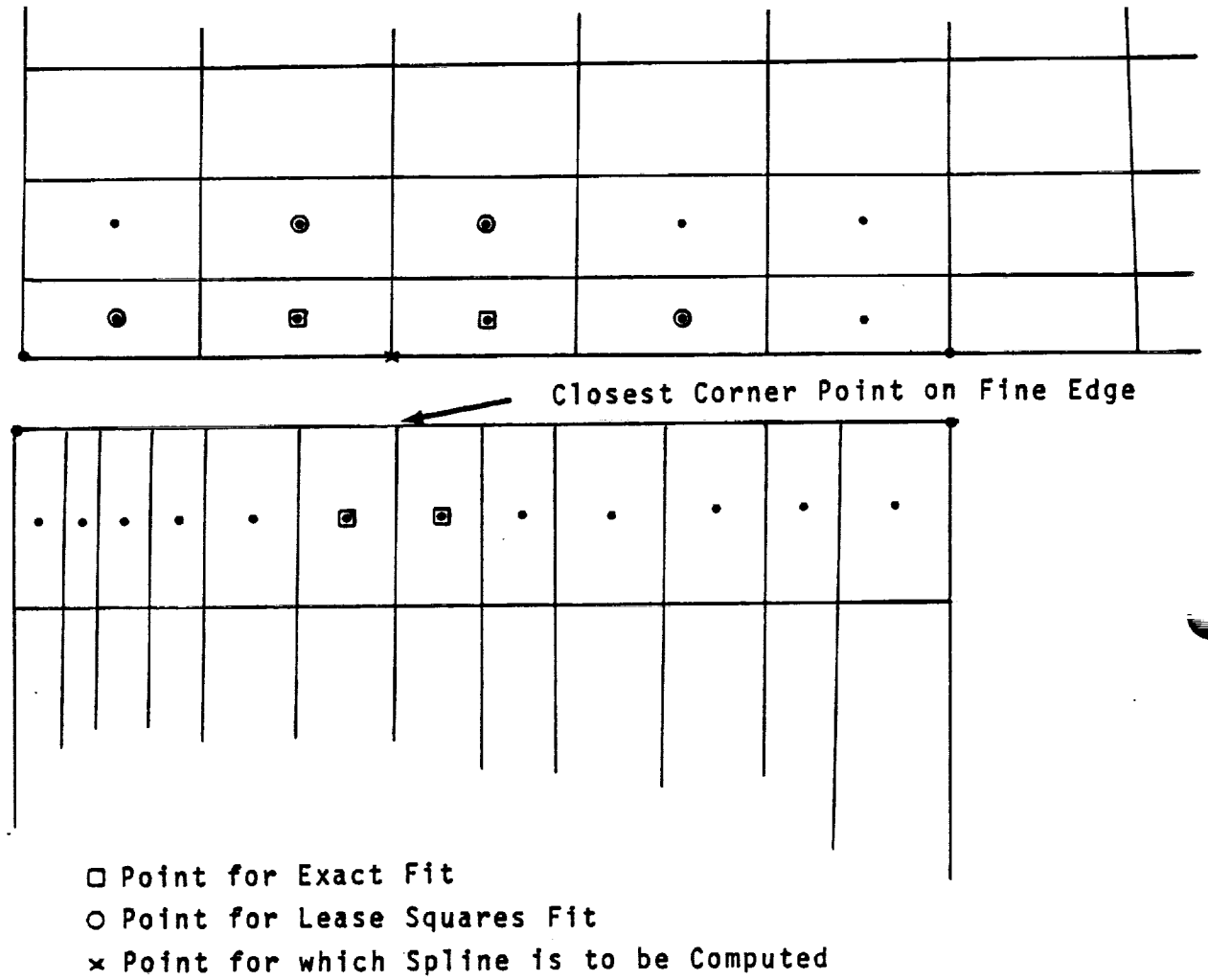
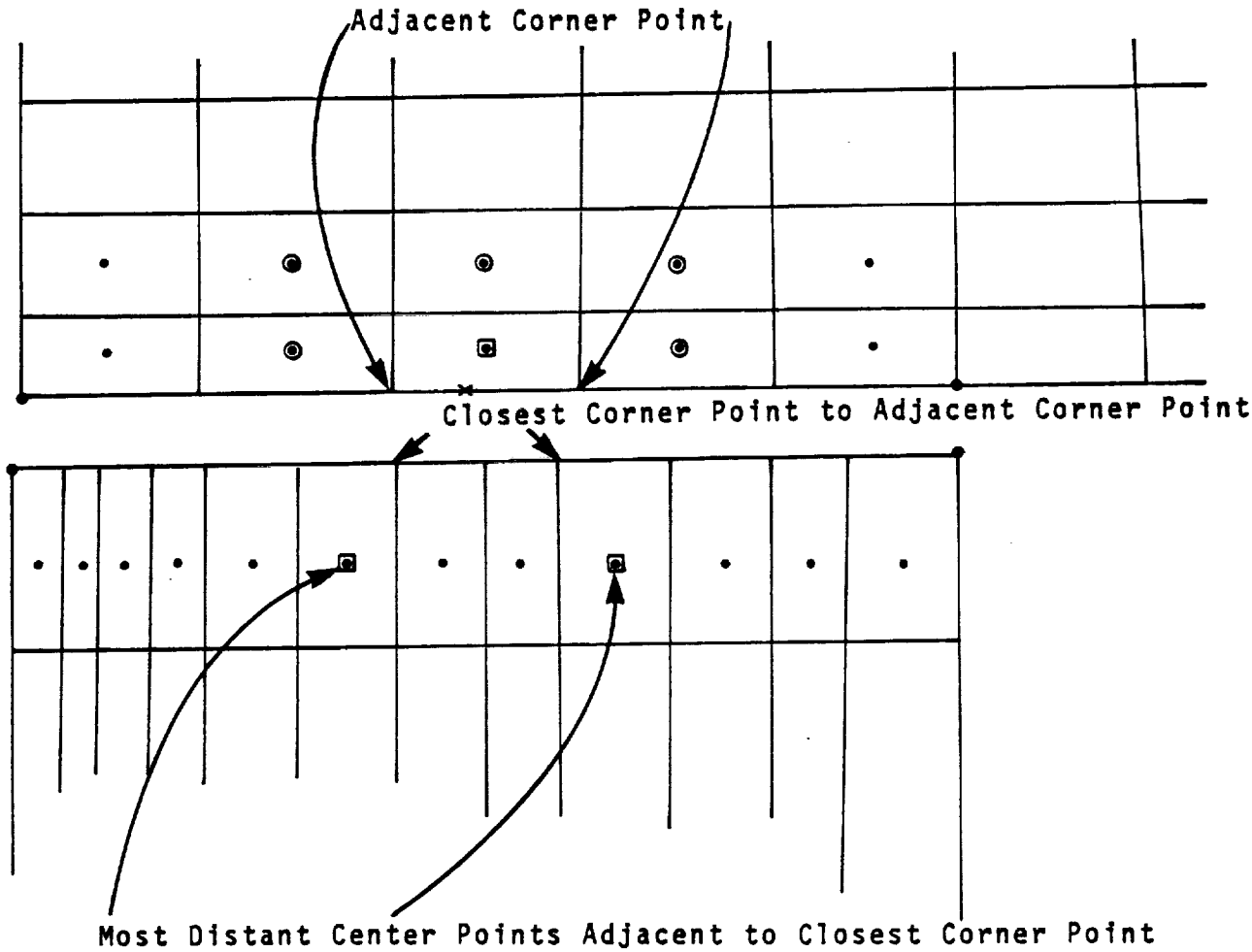


Figure 4-K.3 Surrounding Singularities for Corner Point Spline Computation on Smooth Edge

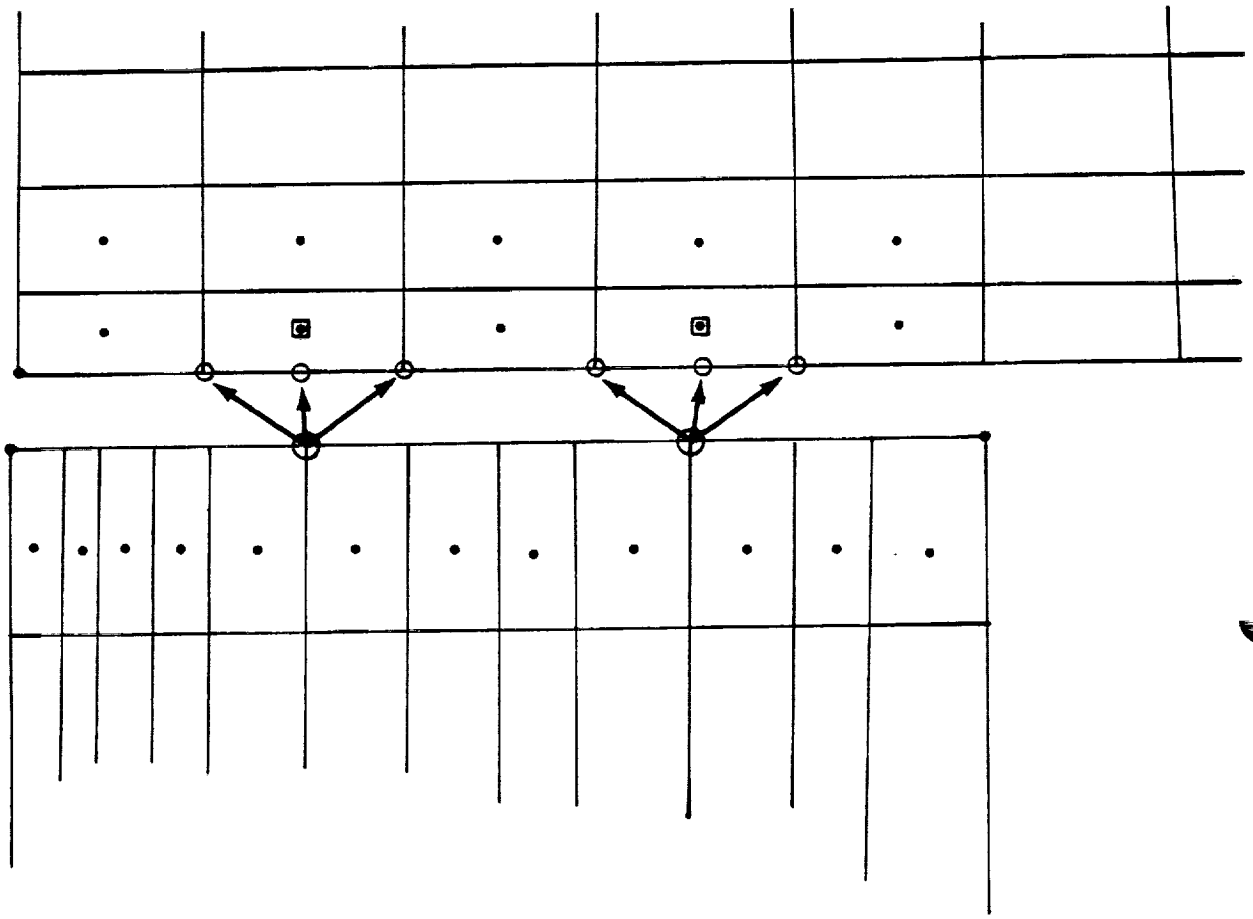


□ Point for Exact Fit

○ Point for Least Squares Fit

× Point for which Spline Vector is to be Computed

Figure 4-K.4 Surrounding Singularities for Edge Midpoint Spline Computation on Smooth Edge

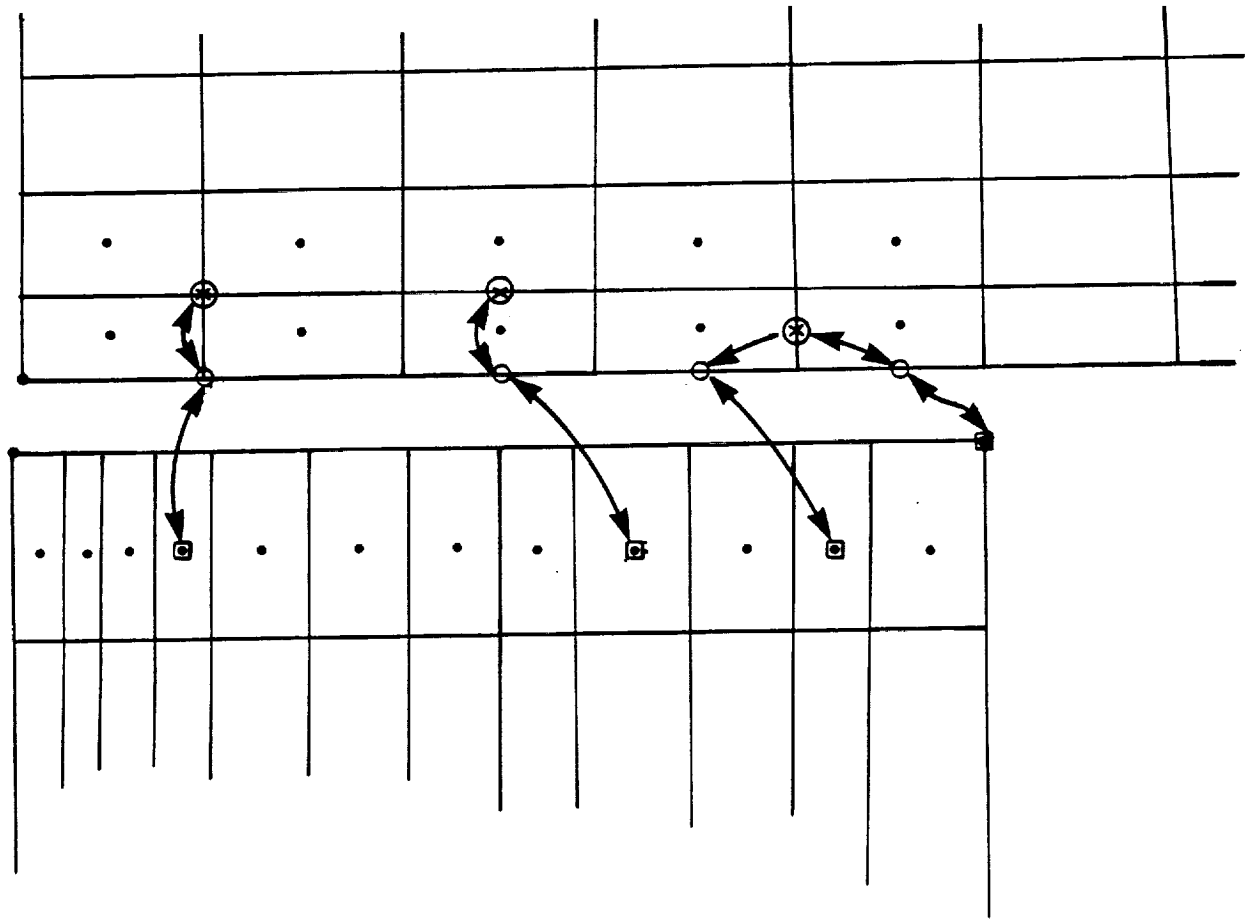


● Interior Point which Uses Alternate Spline Vector

○ Point Chosen for Spline if Edge is Not Smooth

□ Location of Point Used as Alternate Spline Vector

Figure 4-K.5 Alternate Spline Vector Selection



● Point for which Spline Vector is Required
○ Point Used for Quadratic Fit
□ Point Used for Alternate Spline Vector

Figure 4-K.6 Point Selection for Alternate Spline Vectors

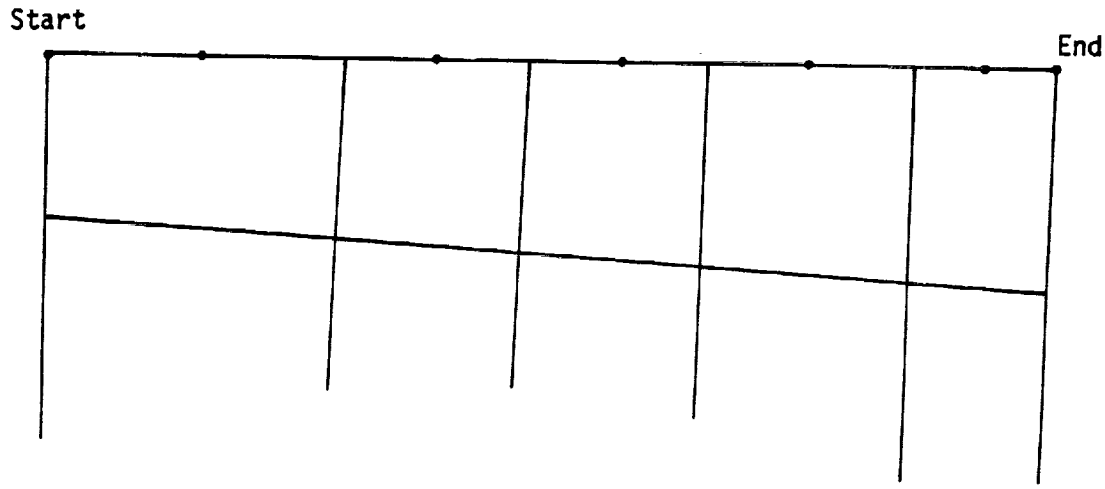


Figure 4-K.7 Location of Doublet Parameters on an Analysis Edge

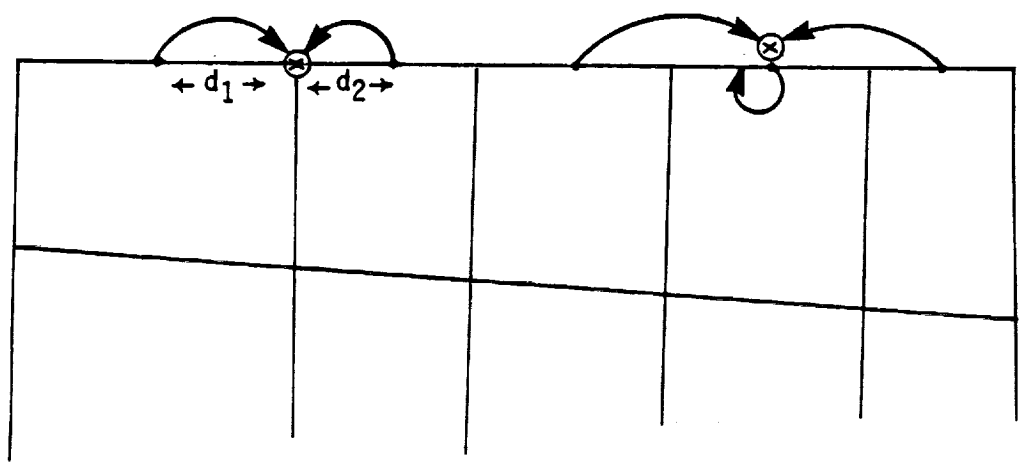


Figure 4-K.8 Dependence of Spline Vectors for Analysis Edges on Surrounding Singularity Parameters

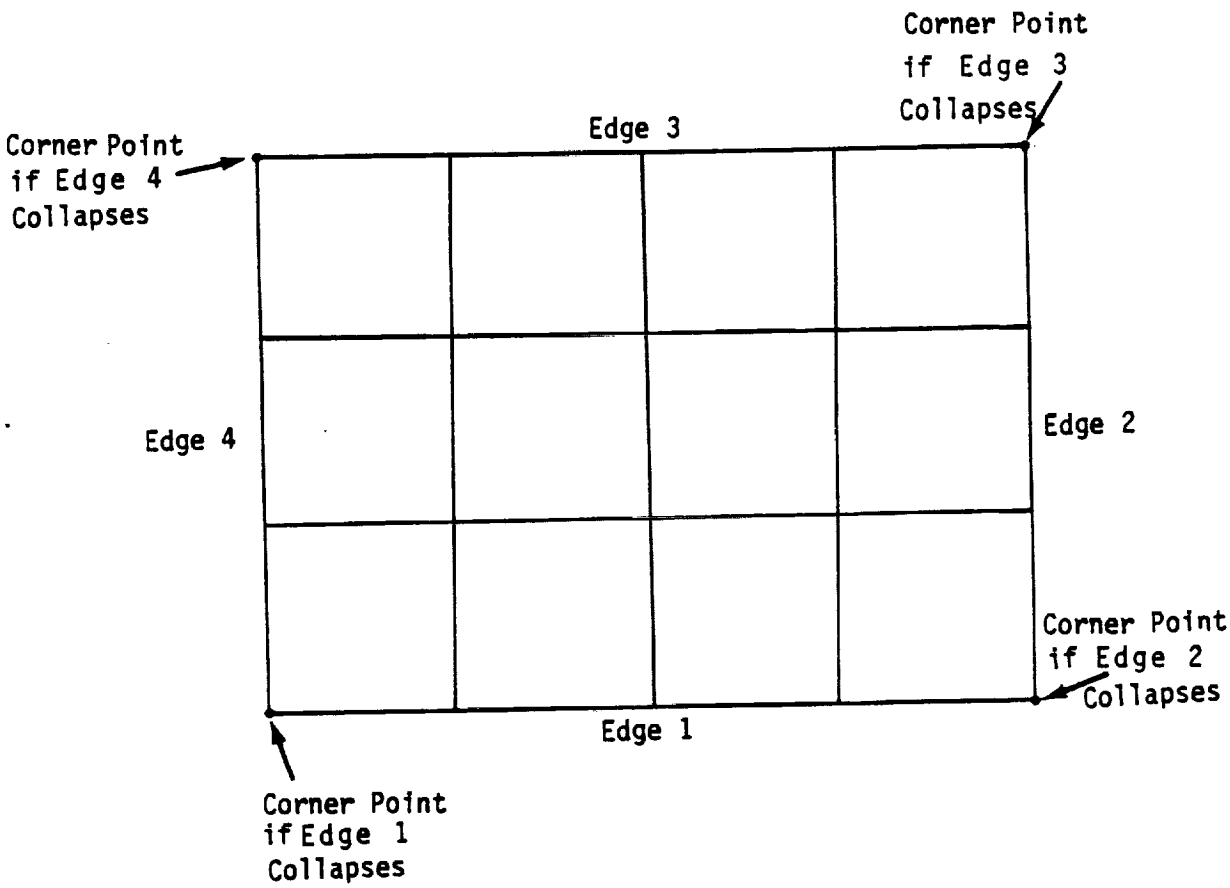
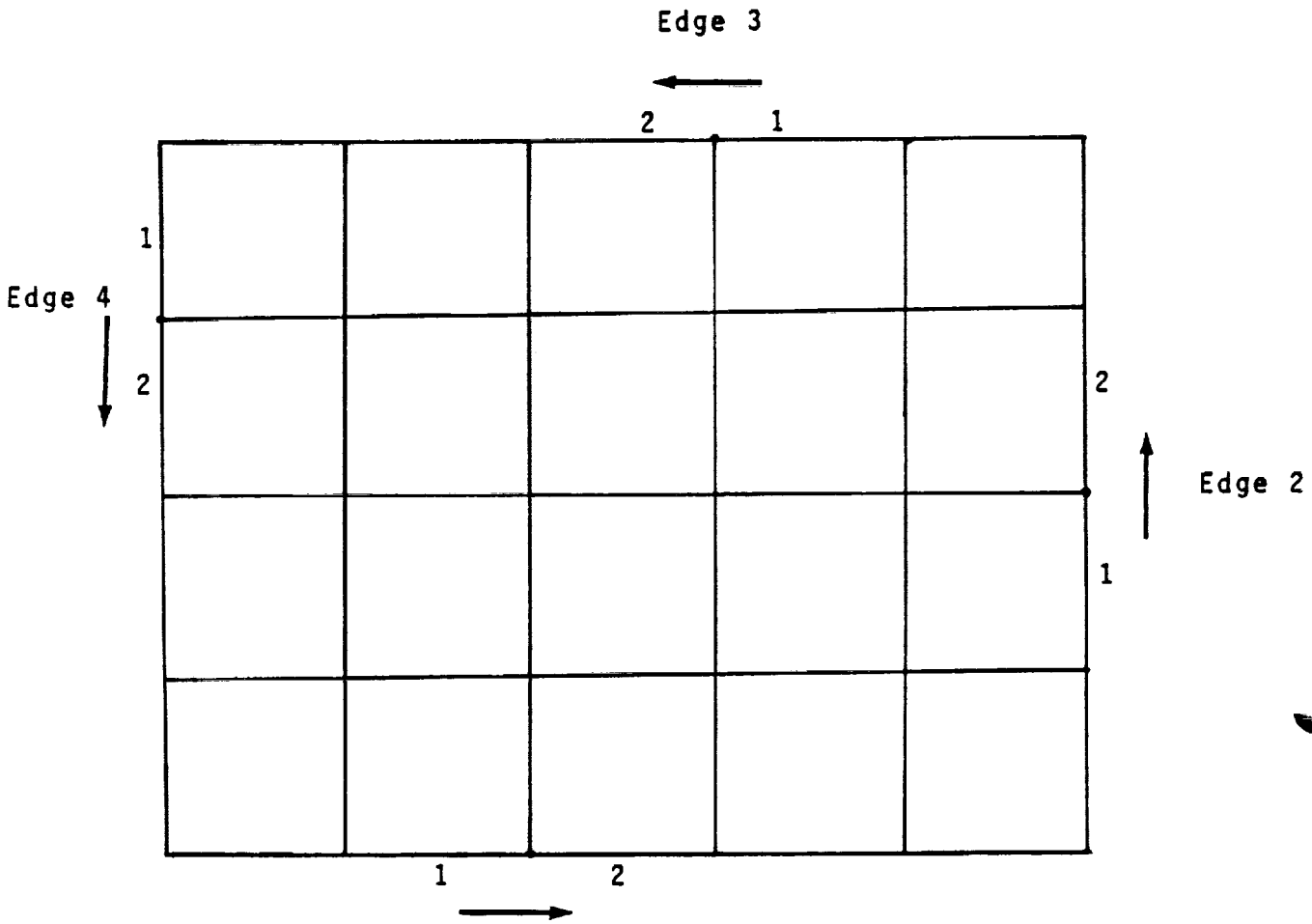
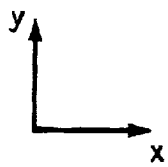


Figure 4-K.9 Unit Spline Point for Collapsed Network Edge





Increasing lattice index direction

Figure 4-K.10 Sequence of Edge Midpoint Selection for Splining Corner Points on Analysis Edges

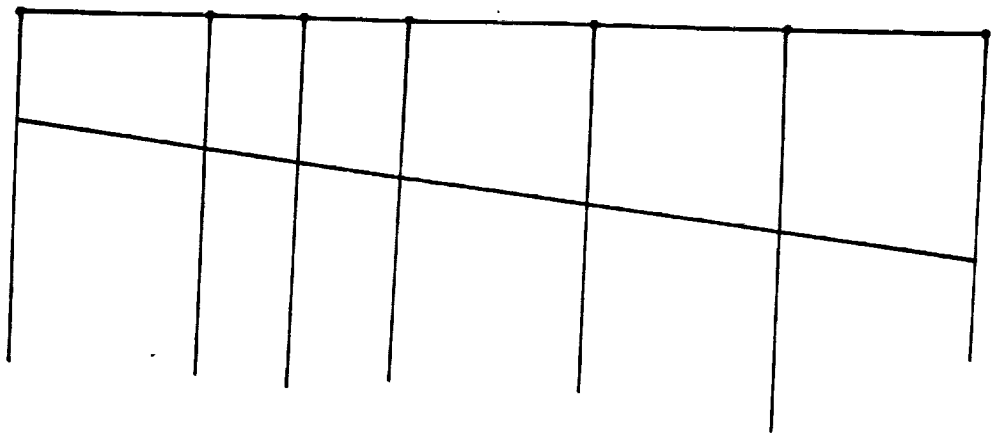


Figure 4-K.11 Singularity Parameter Locations for Design Edges of Networks

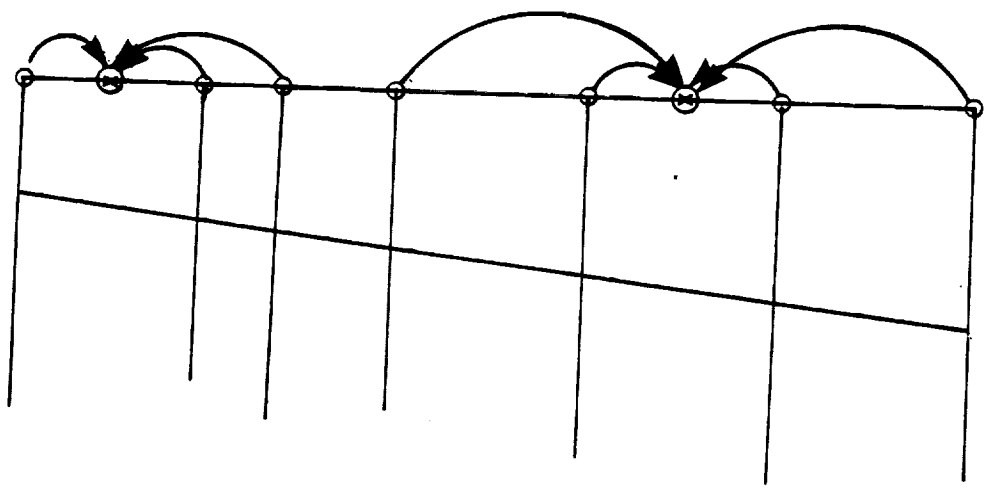
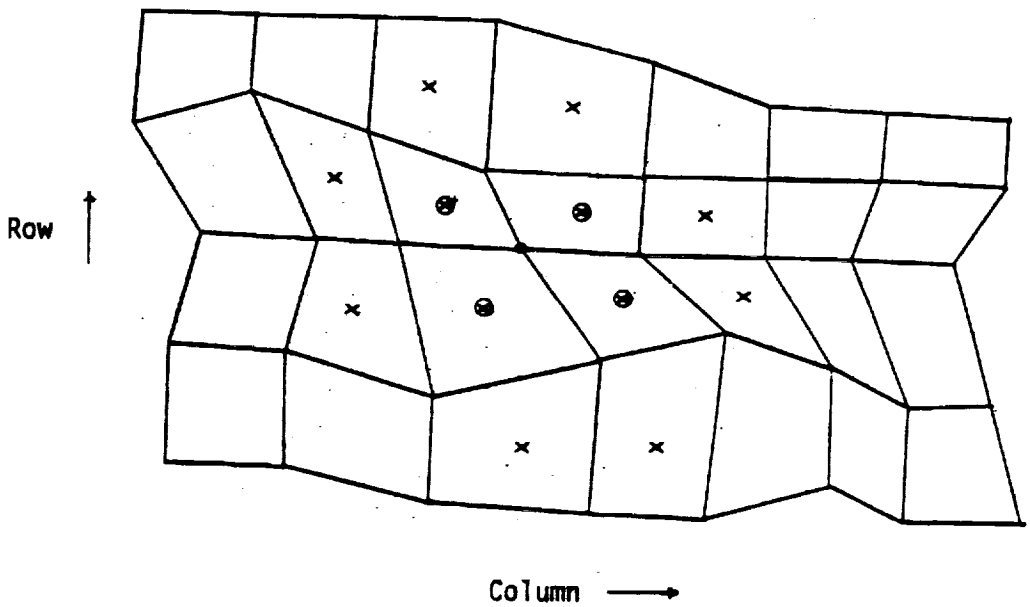


Figure 4-K.12 Singularity Parameters for Intermediate Spline Vector Construction

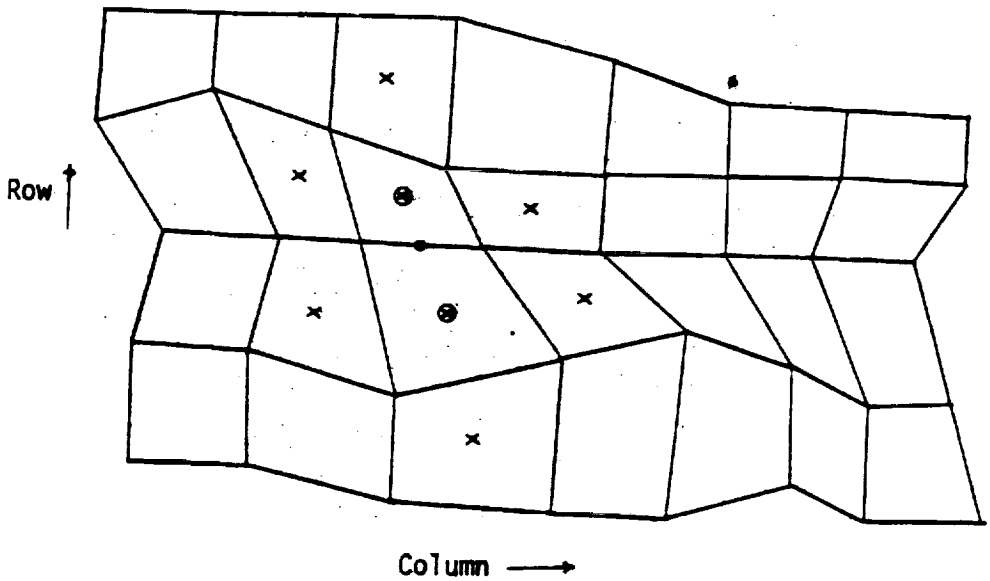


● Point where spline vector is required

⊗ Point for exact fit

× Point for least squares fit

Figure 4-K.13 Surrounding Point Locations for Corner Splines for Doublet Analysis Network

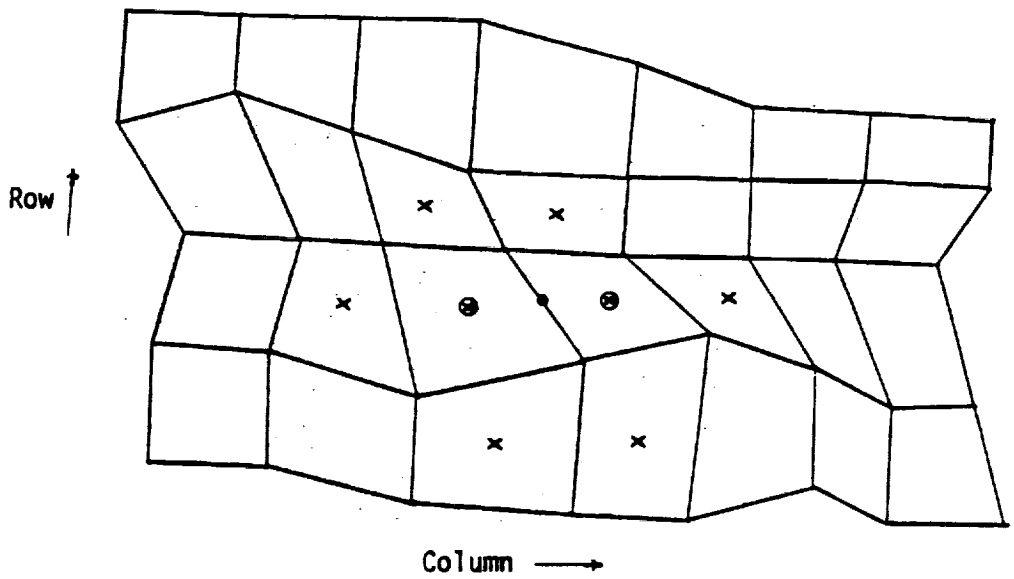


● Point where spline vector is required

⊗ Point for exact fit

× Point for least squares fit

Figure 4-K.14 Surrounding Point Locations for Column Edge Midpoint Splines of Doublet Analysis Network

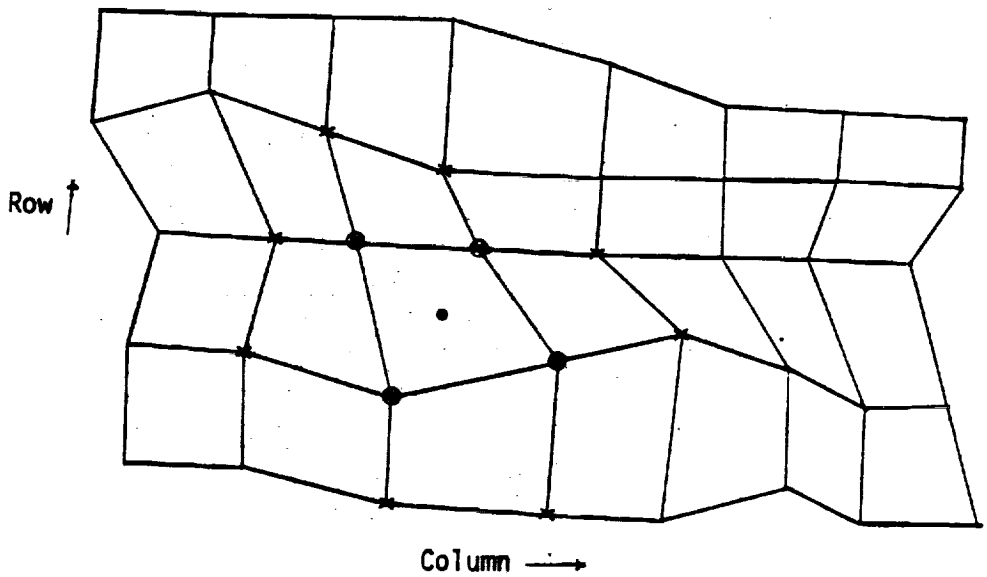


● Point where spline vector is required

⊗ Point for exact fit

× Point for least squares fit

Figure 4-K.15 Surrounding Point Locations for Row Edge Midpoint Splines for Doublet Analysis Network

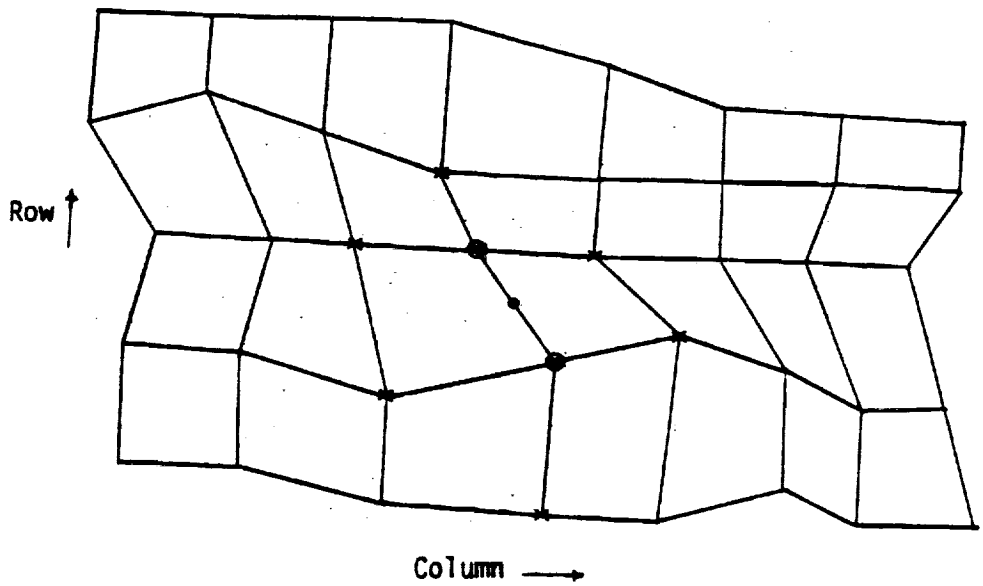


● Point where spline vector is required

⊗ Point for exact fit

× Point for least squares fit

Figure 4-K.16 Surrounding Point Locations for Corner Point Splines for Doublet Design I Networks

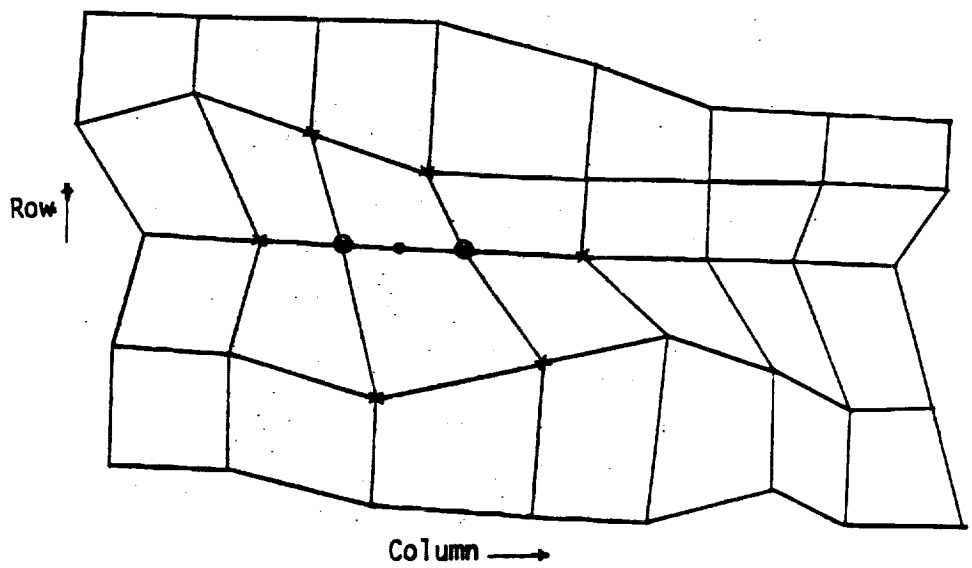


● Point where spline vector is required

⊗ Point fit exactly

× Point fit in least squares sense

Figure 4-K.17 Surrounding Point Locations for Row Edge Midpoint Splines for Doublet Design I Networks



● Point where spline vector is required

⊙ Point fit exactly

× Point fit in least squares sense

Figure 4-K.18 Surrounding Point Locations for Column Edge Midpoint Splines for Doublet Design I Networks

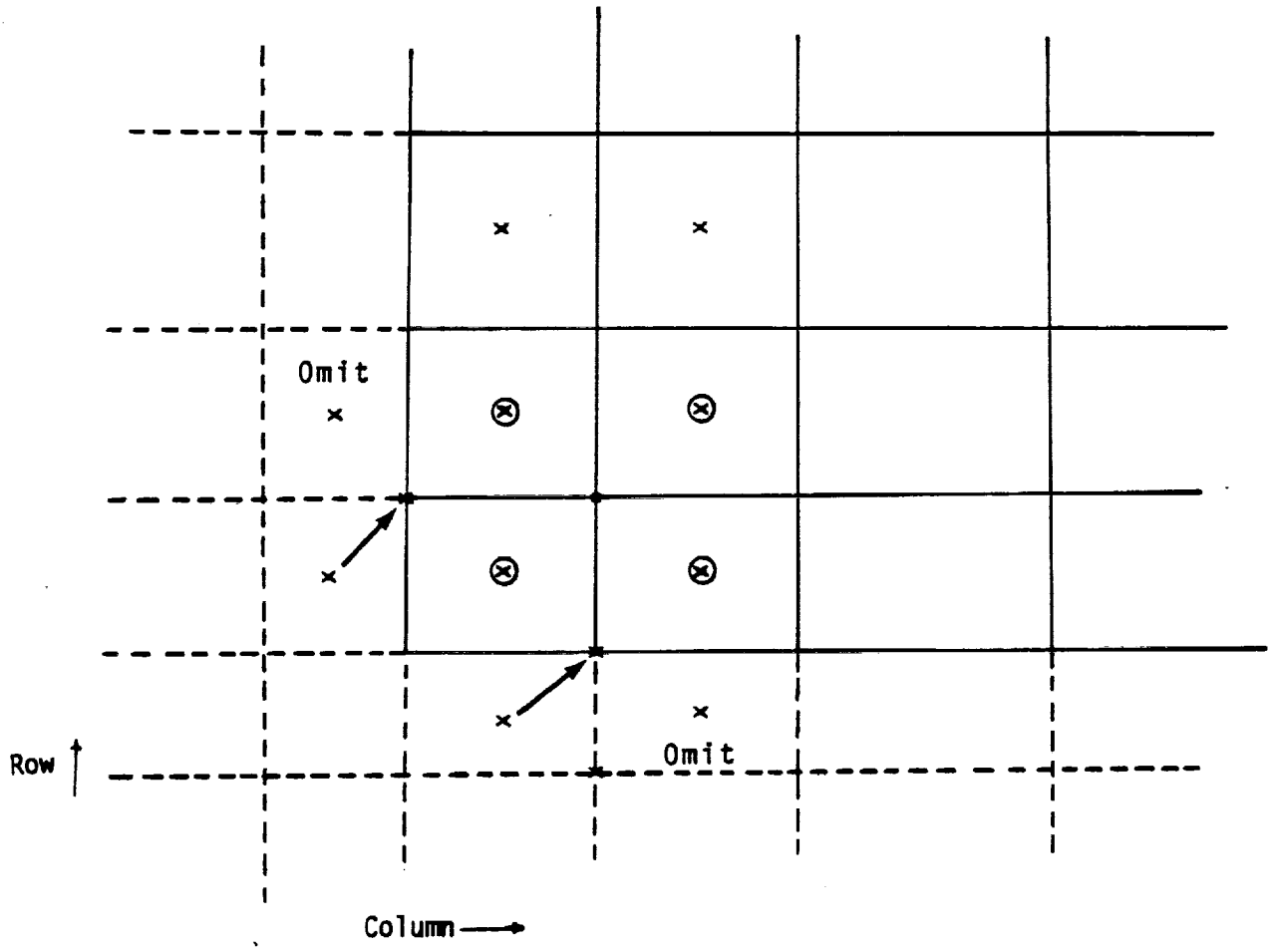


Figure 4-K.19 Point Selection for Corner Point Near Edge, Analysis Network Omit

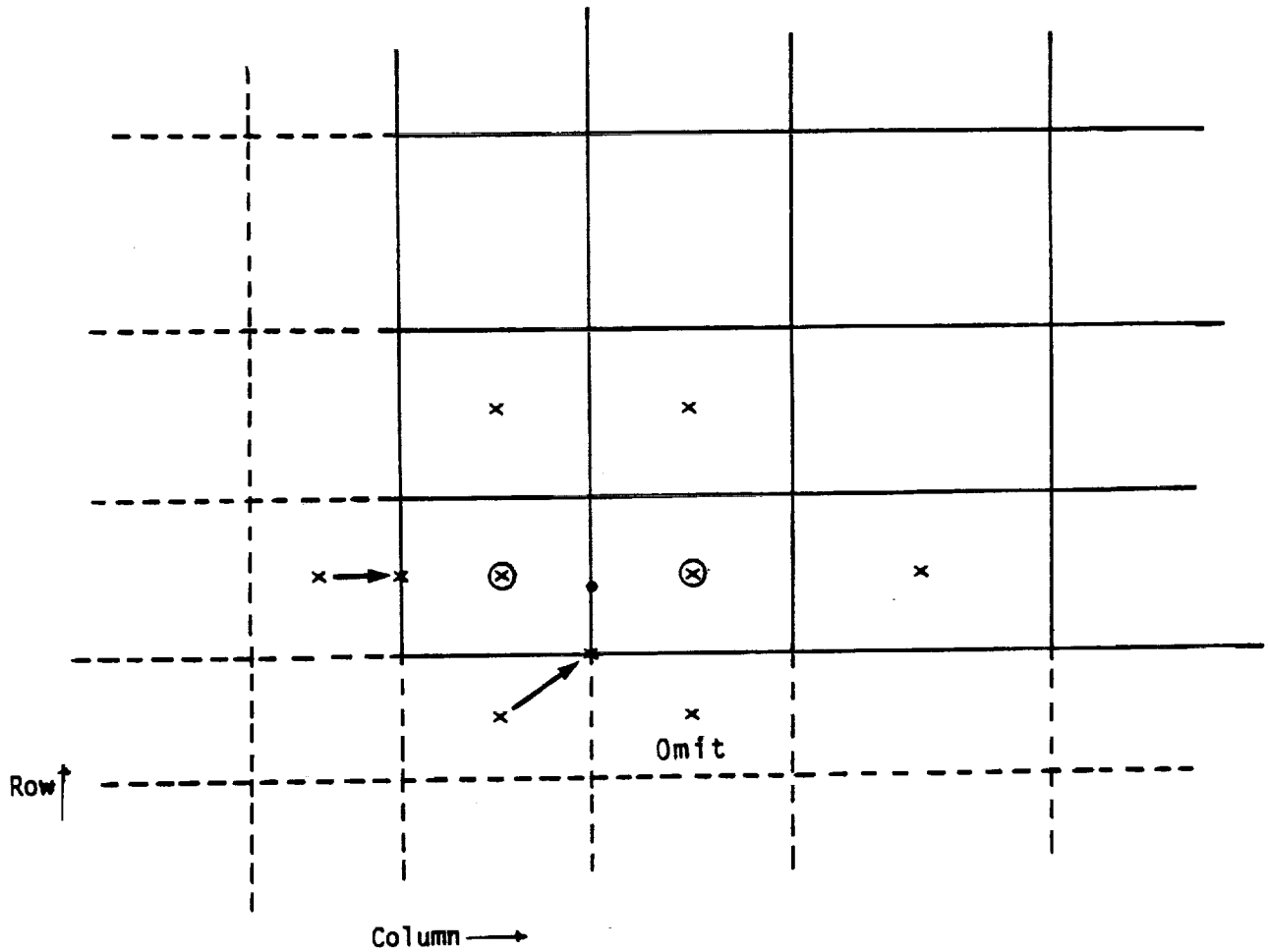


Figure 4-K.20 Point Selection for Row Edge Midpoint Near Edge, Analysis Network

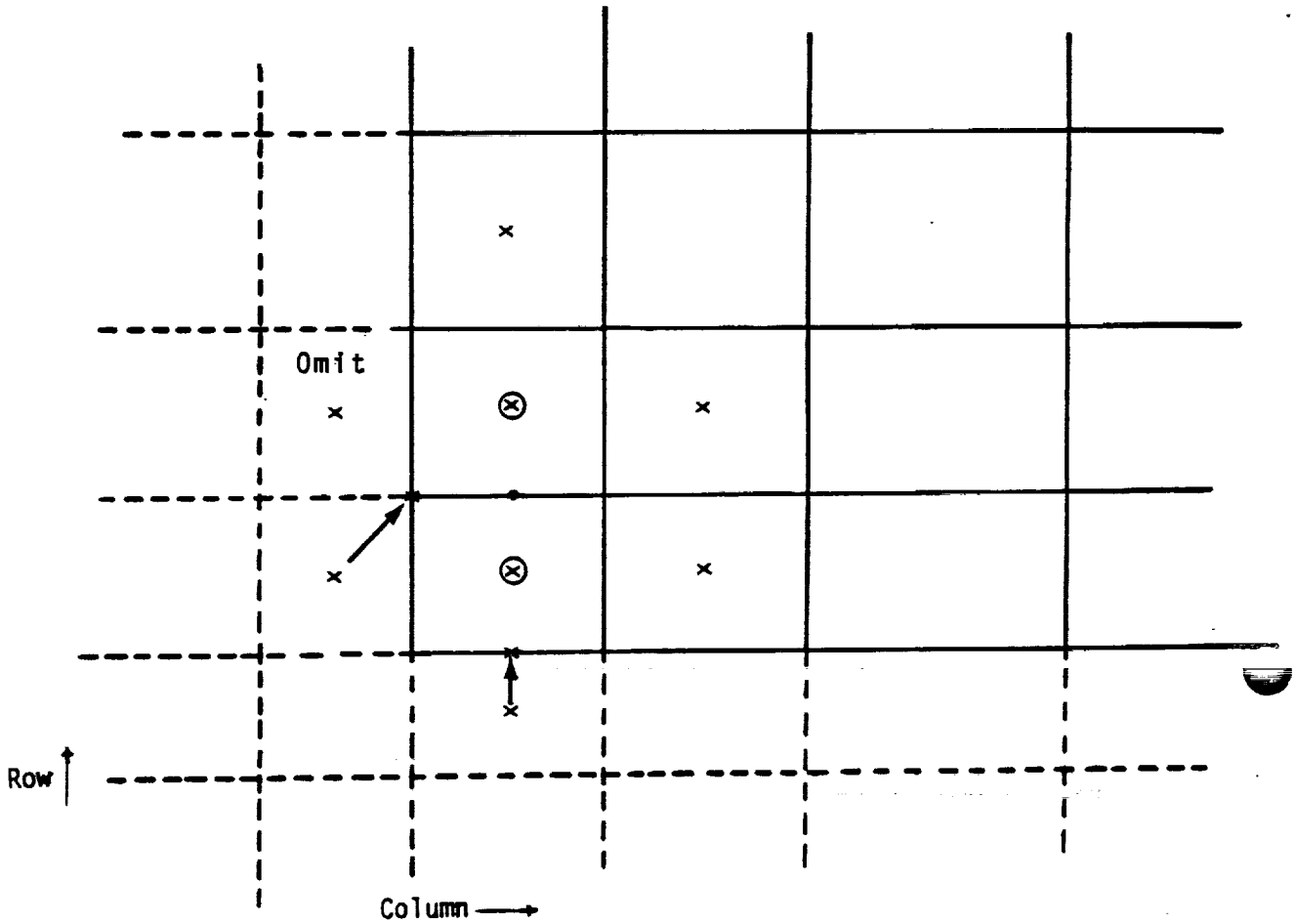


Figure 4-K.21 Point Selection for Column Edge Midpoint Near Edge, Analysis Network

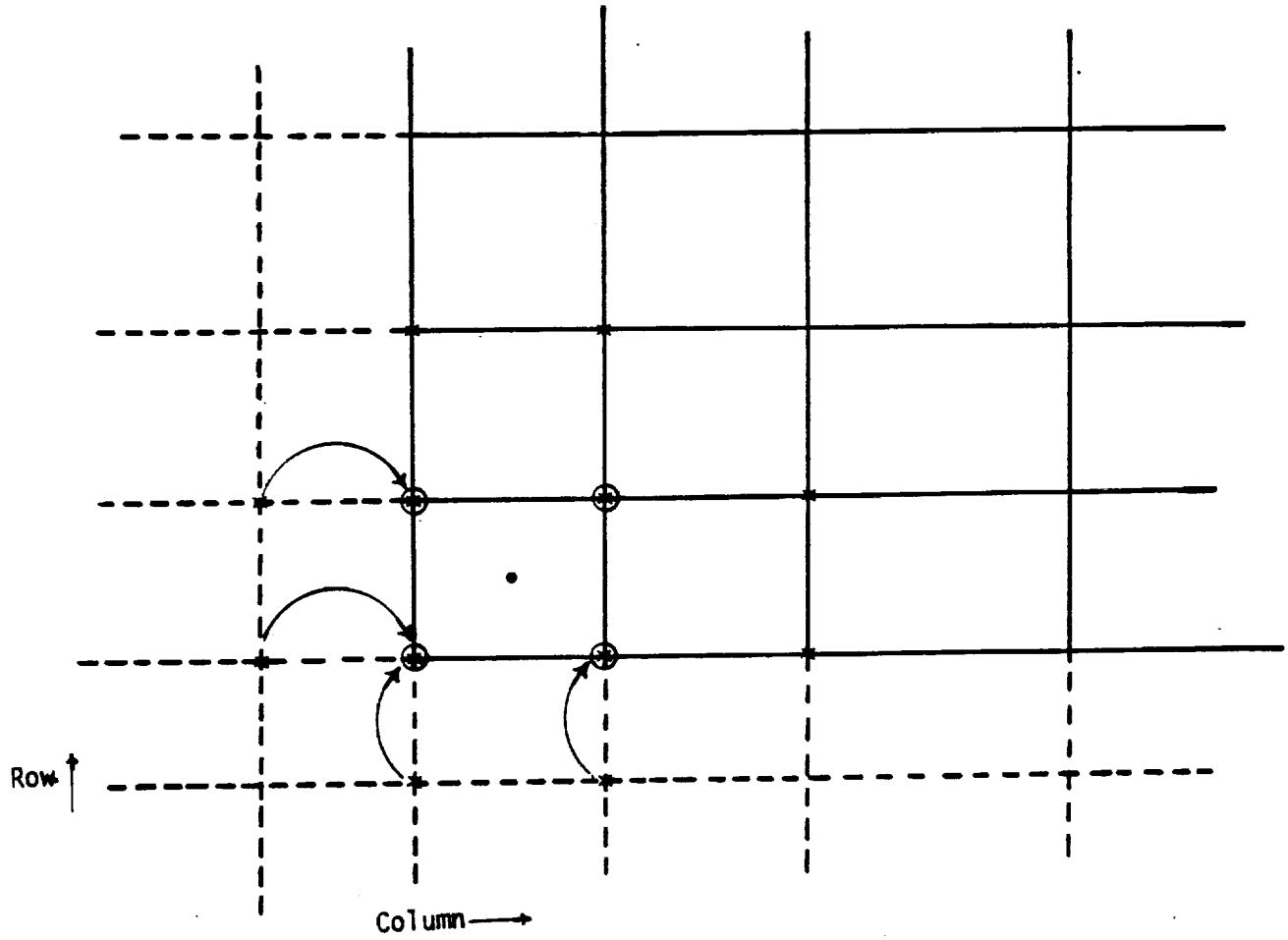


Figure 4-K.22 Point Selection for Center Point Near Edge, Design Network

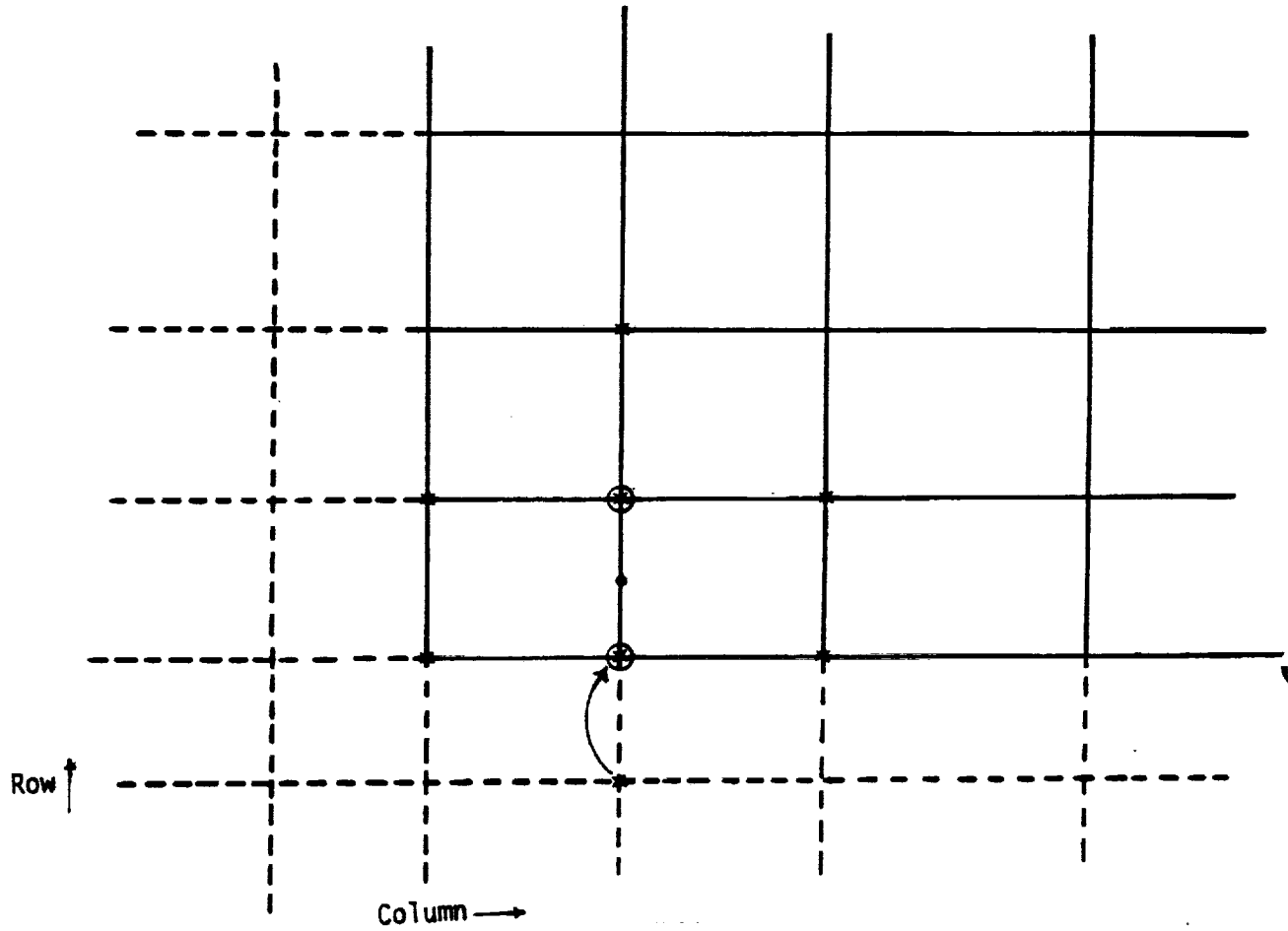


Figure 4-K.23 Point Selection for Row Edge Midpoint Near Edge, Design Network

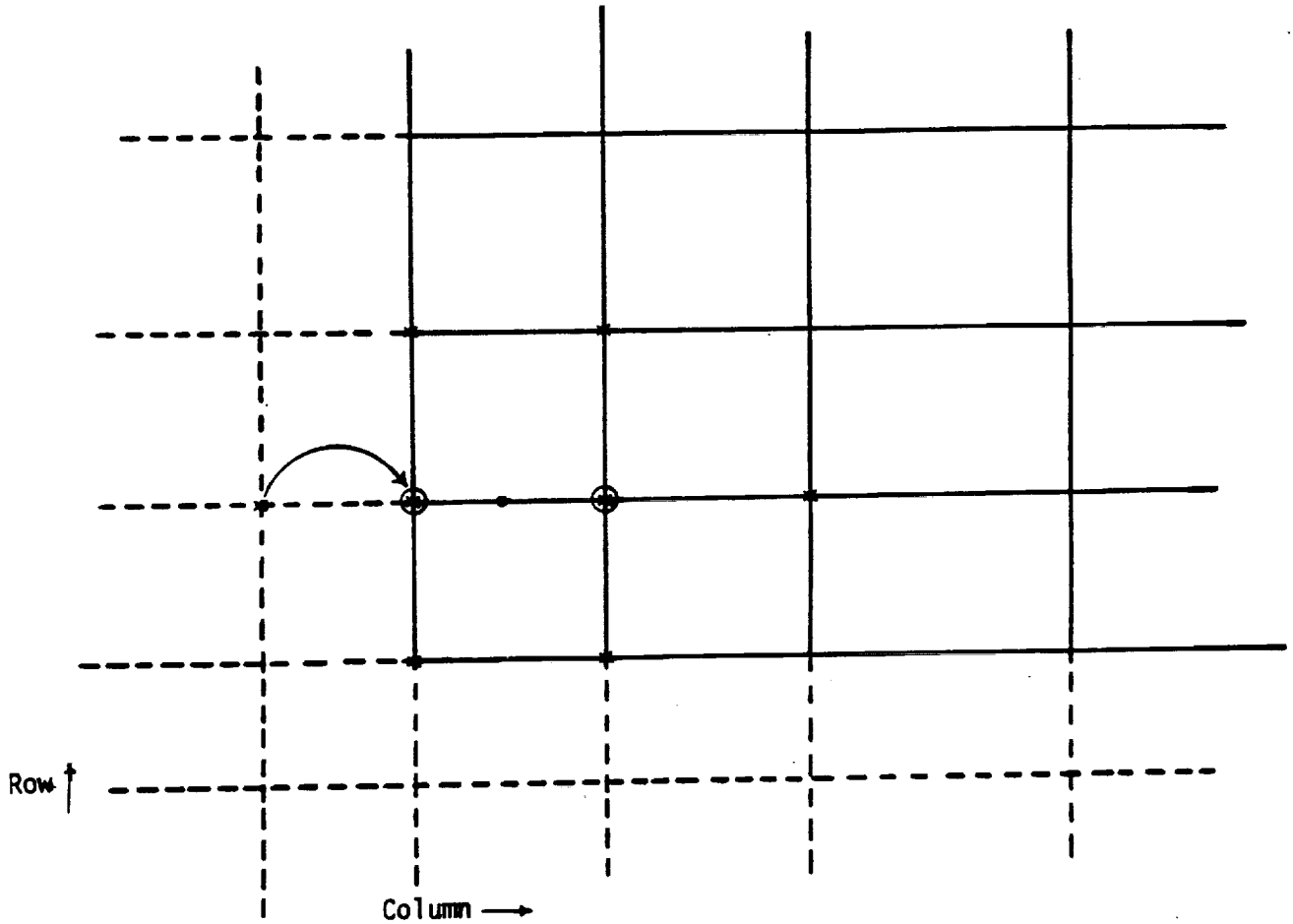


Figure 4-K.24 Point Selection for Column Edge Midpoint Near Edge, Design Network

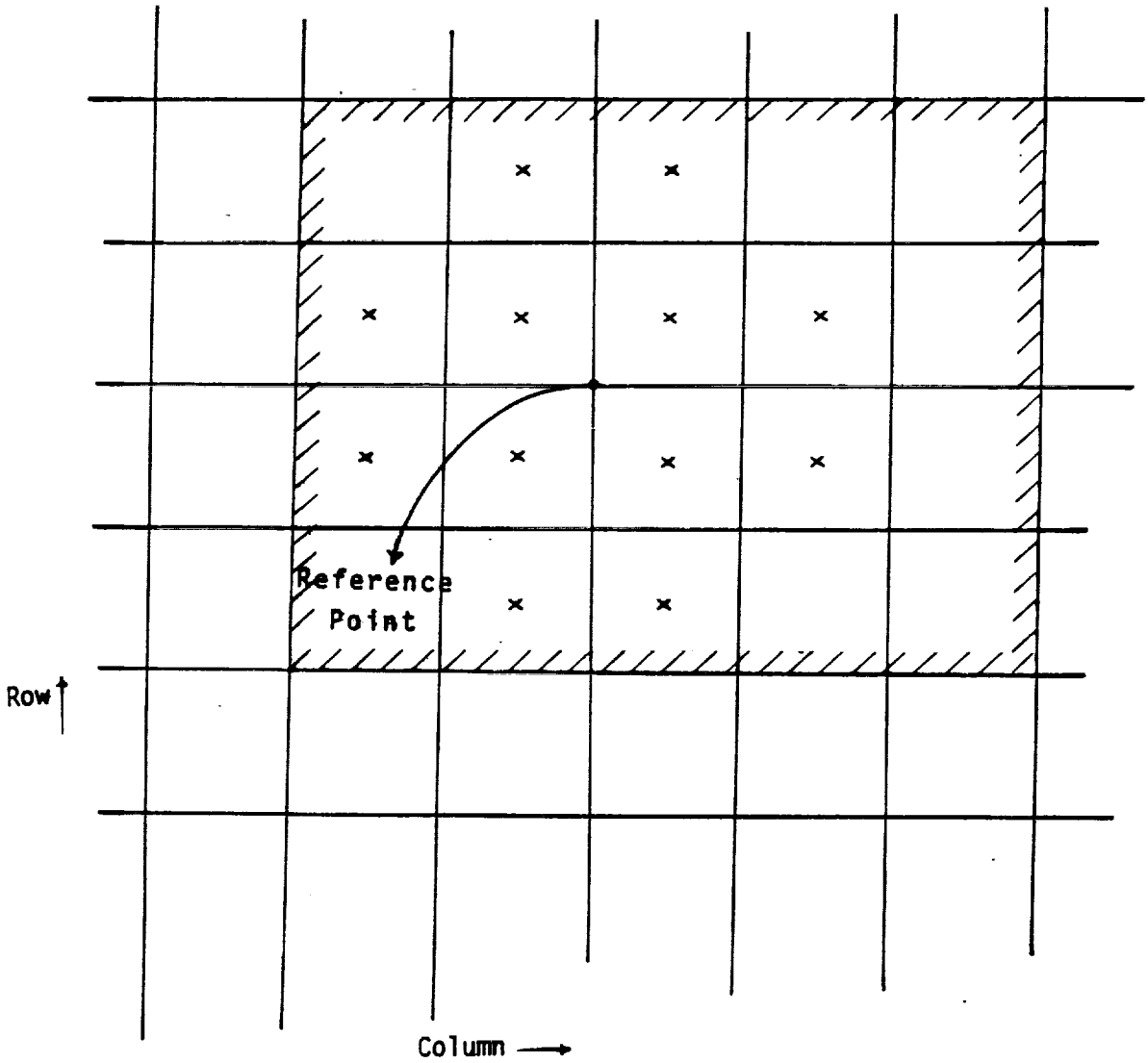
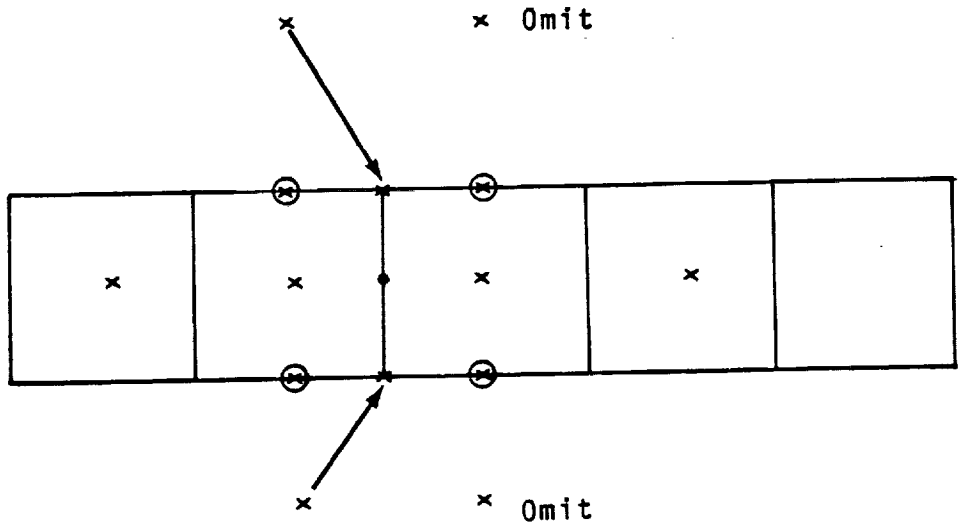


Figure 4-K.25 Illustration of Operation of Algorithm which Selects Surrounding Points for Spline Computations



⊗ Extra Points for One Column/One Row Network Spline

Figure 4-K.26 Surrounding Point Locations for One Row Network

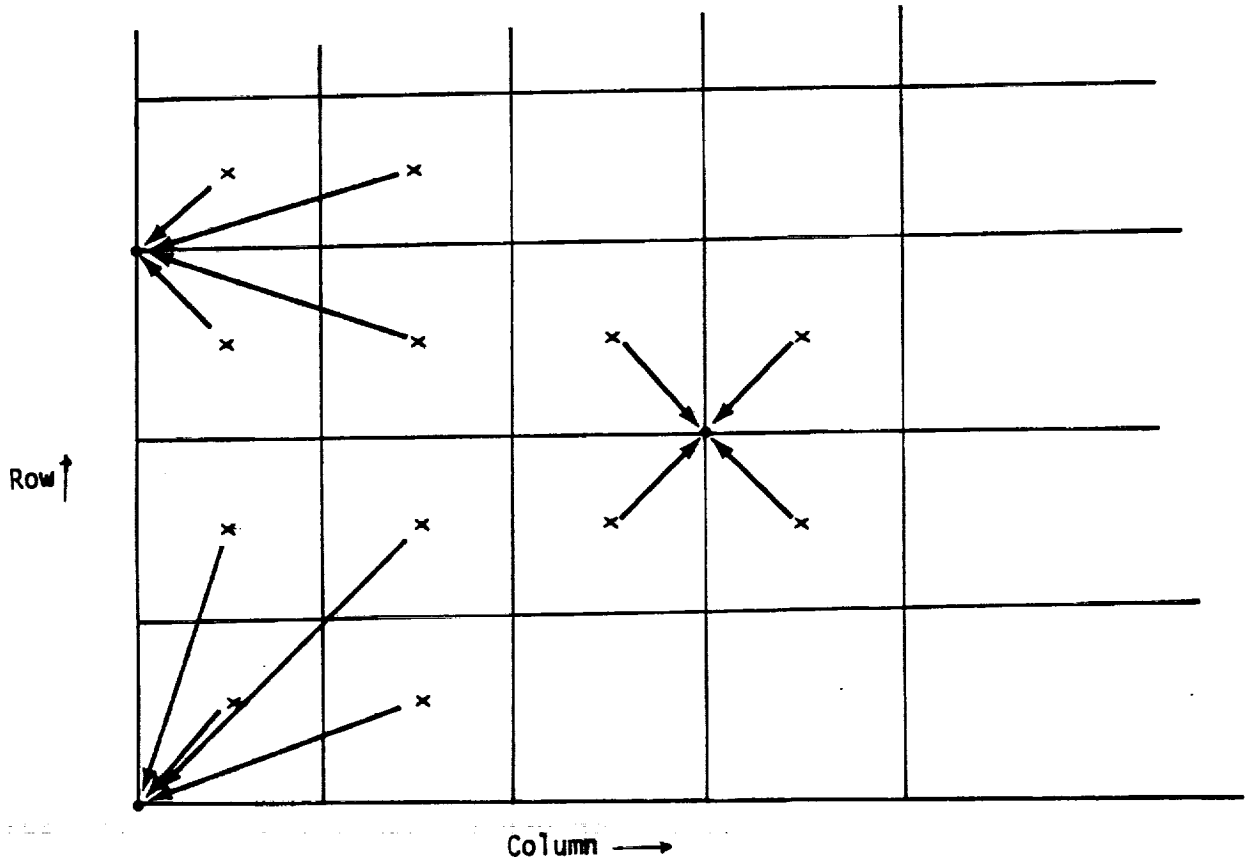


Figure 4-K.27 Surrounding Points for Source Analysis Spline Computation

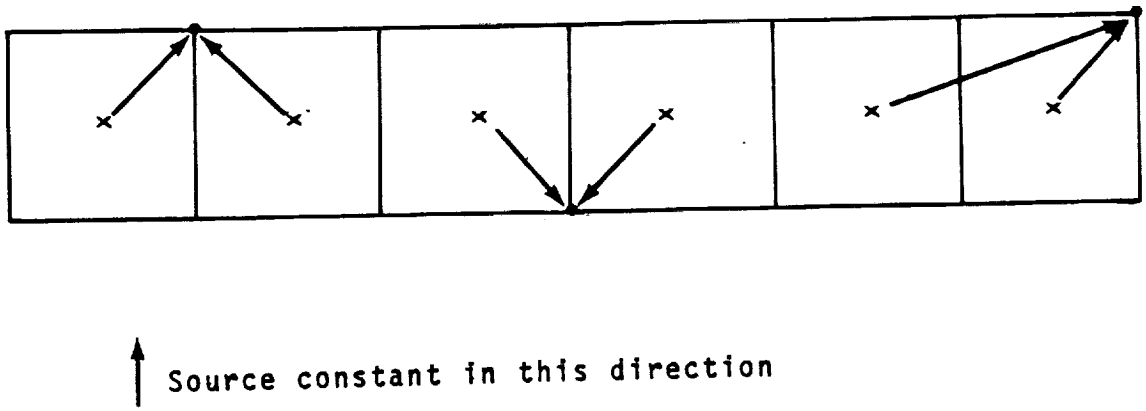


Figure 4-K.28 Source Spline Point Selected for One Column/Row Networks

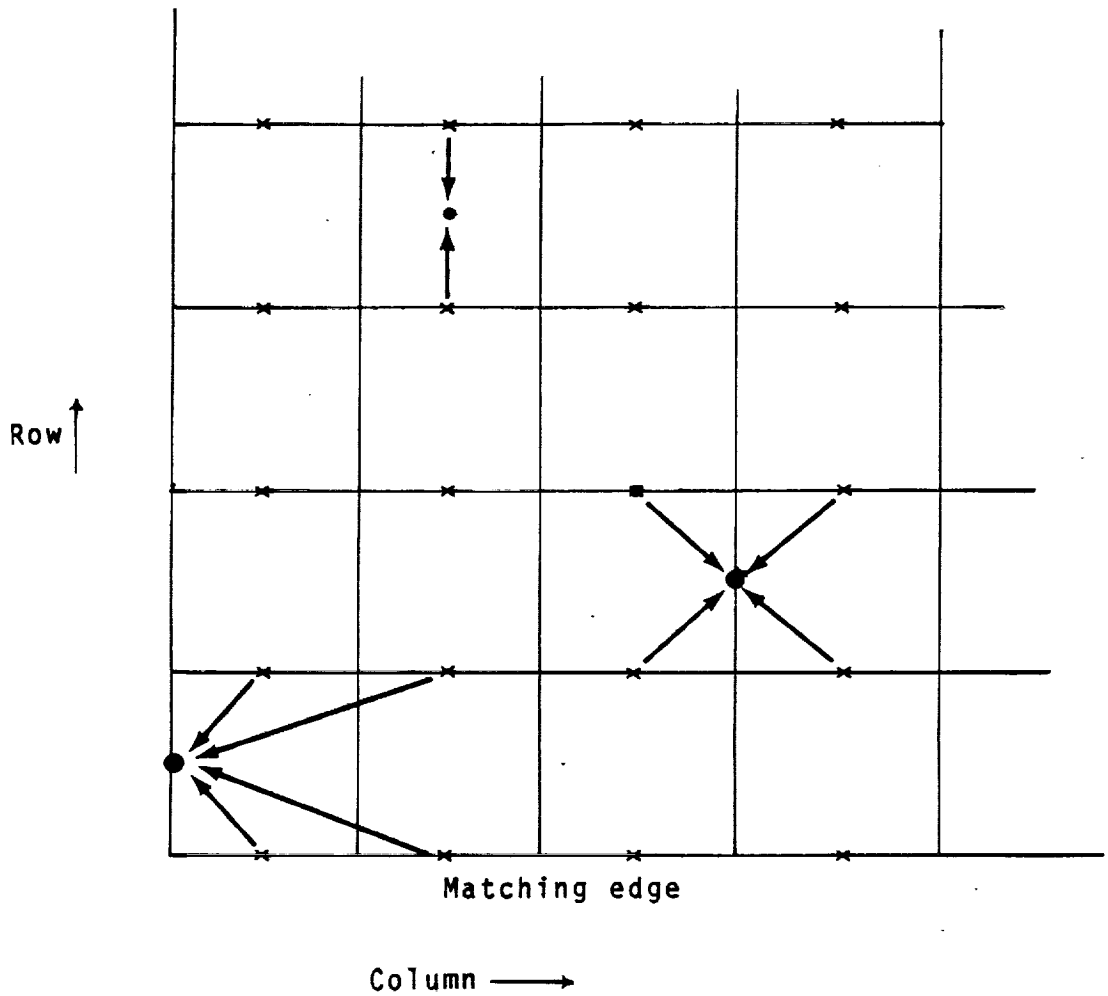


Figure 4-K.29 Surrounding Points for Source Design II Spline Computation

APPENDIX 4-L

GAP FILLING PANELS

4-L.0 Introduction

Gap filling panels are automatically added at any abutment where the distance between network edges in the abutment exceeds the user specified tolerance distance (unless the user indicates that they are not to be used as discussed in the PAN AIR User's Manual, Section 7 (Reference 2)). Figure 4-L.1 illustrates the situation. A theoretical discussion of gap filling panels occurs in the PAN AIR Theory Document, Appendix F, Section F.6 (Reference 1).

4-L.1 Gap Panel Construction

Construction of gap filling panels begins with the identification of one network edge as the most densely paneled. This is the reference edge. This edge is then parameterized (see PAN AIR Theory Document, Appendix F, Section F.6 (Reference 1)). Next a loop is set up over all other network edges in the abutment. The other edge is parameterized and the two parameterizations are merged and sequenced in ascending order.

For example, one sequence of parameterizations might be

0, 0.1, 0.2, 0.3, 0.4, 0.8, 1.0

and the other might be

0., 0.15, 0.23, 0.3, 0.41, 0.8, 0.9, 1.0

After merging and sequencing the new sequence becomes

0., 0., 0.1, 0.15, 0.2, 0.23, 0.3, 0.3, 0.4, 0.41, 0.8, 0.8, 0.9, 1.0, 1.0

The merged sequenced array of parameterizations is used to generate the gap filling panels. Each successive pair of entries in the array are used to define two coordinates on each of the two edges.

For example, for parameterizations 0.4 and 0.41 in the example above, the coordinates of a point 0.4 along the length of the first segment is computed and another 0.41 along the way of the first segment. This is also done for the other network edge. These four points become the four corner points of the gap filling panel.

The edge lengths of the panel edges are computed. If all edges are larger than the tolerance distance, the remaining gap panel data is computed (see below). If only one edge is short, the corner points of the short edge are redefined so that they are exactly the same point and a flag is set indicating the panel is triangular.

If more than one edge is short, the panel is omitted from the problem: no "GAP-PANEL" data set is written to the DQG data base.

A gap filling panel is assigned to a panel on each network edge other than the reference edge if the network edge contains two corner points of the gap filling panel.

Some care must be taken in the definition of gap filling panel to assure that its normal points in the same general direction as the panels along the network edge.

The data provided for gap filling panels are the corner points, edge midpoints, and center point coordinates of the gap filling panel, the network and panel to which it is assigned, the parameterizations of the two corner points and edge midpoints of both the gap filling panel and the network panel to which the gap panel belongs, and a flag indicating whether a panel is triangular.

For abutments with a plane of symmetry, a different construction procedure is used. The corner points of the gap filling panel are defined as the two corner points of the panel on the network edge and the orthogonal projection of the two points to the plane of symmetry.

The spline vectors for the gap filling panel are defined from the spline vectors at the corner point and edge midpoint of the panel to which the gap filling panel is assigned. A quadratic function along the panel edge is defined which fits the parameterizations of its corner points and edge midpoints. This function is evaluated at the parameterizations of three gap filling panel points (that is, the two corner points and the edge midpoint between them). The coefficients from this evaluation are used to accumulate a spline vector (see Section 4-K.1) for each of the three gap filling panel points on the edge. The same spline vectors are written for the two additional points perpendicular to the edge of the panel to which the gap panel belongs. See Figure 4-L.2.

The gap filling panel data set is keyed by a cumulative index counting the number of gap filling panels. The spline vectors are keyed by a dummy network index which is the cumulative gap filling panel index, and by dummy lattice indices (see Figure 4-L.2). The effect is as though the gap panel were a one panel network. When panel data is computed for the gap filling panels, the data is written with a keyset of zero for the network index, and the cumulative gap panel index and the number one for the panel lattice indices. A dummy "NETWORK-SPEC" data set is also written for gap filling panels. It treats them as a single 1 X N panel network where N is the number of gap filling panels.

One special feature concerning triangular gap filling panels should be noted. If the collapsed edge of a gap filling panel is on the edge opposite the network to which the gap panel belongs, the spline vector generation will in general produce a multiple valued doublet strength at the collapsed edge. To avoid this disastrous situation, if any gap filling panels have this property, they are ignored, that is they are discarded as if they had more than two short edges. The effect on the doublet strength of this action has not been investigated. See Figure 4-L.3.

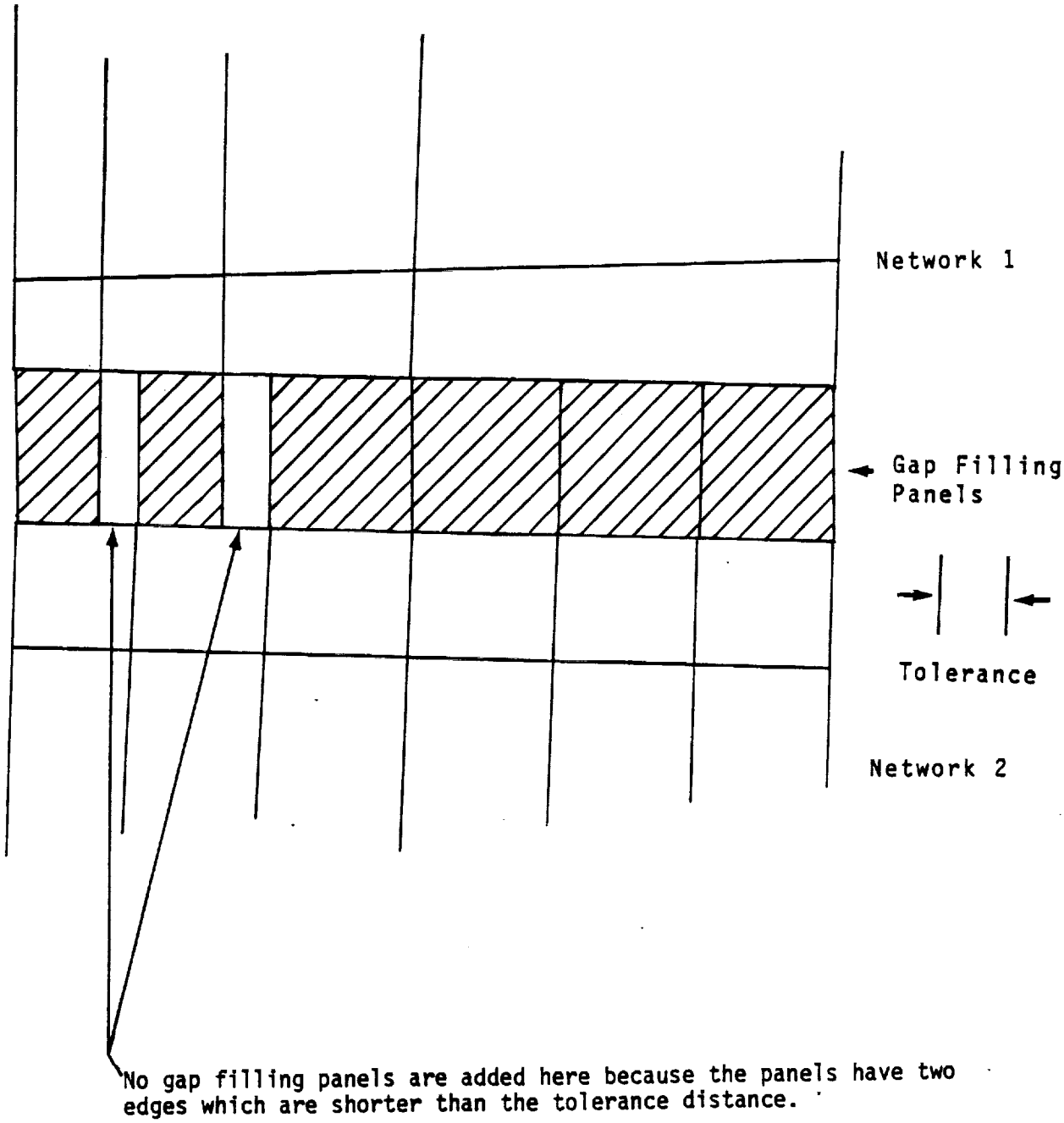


Figure 4-L.1 Addition of Gap Filling Panels to an Abutment

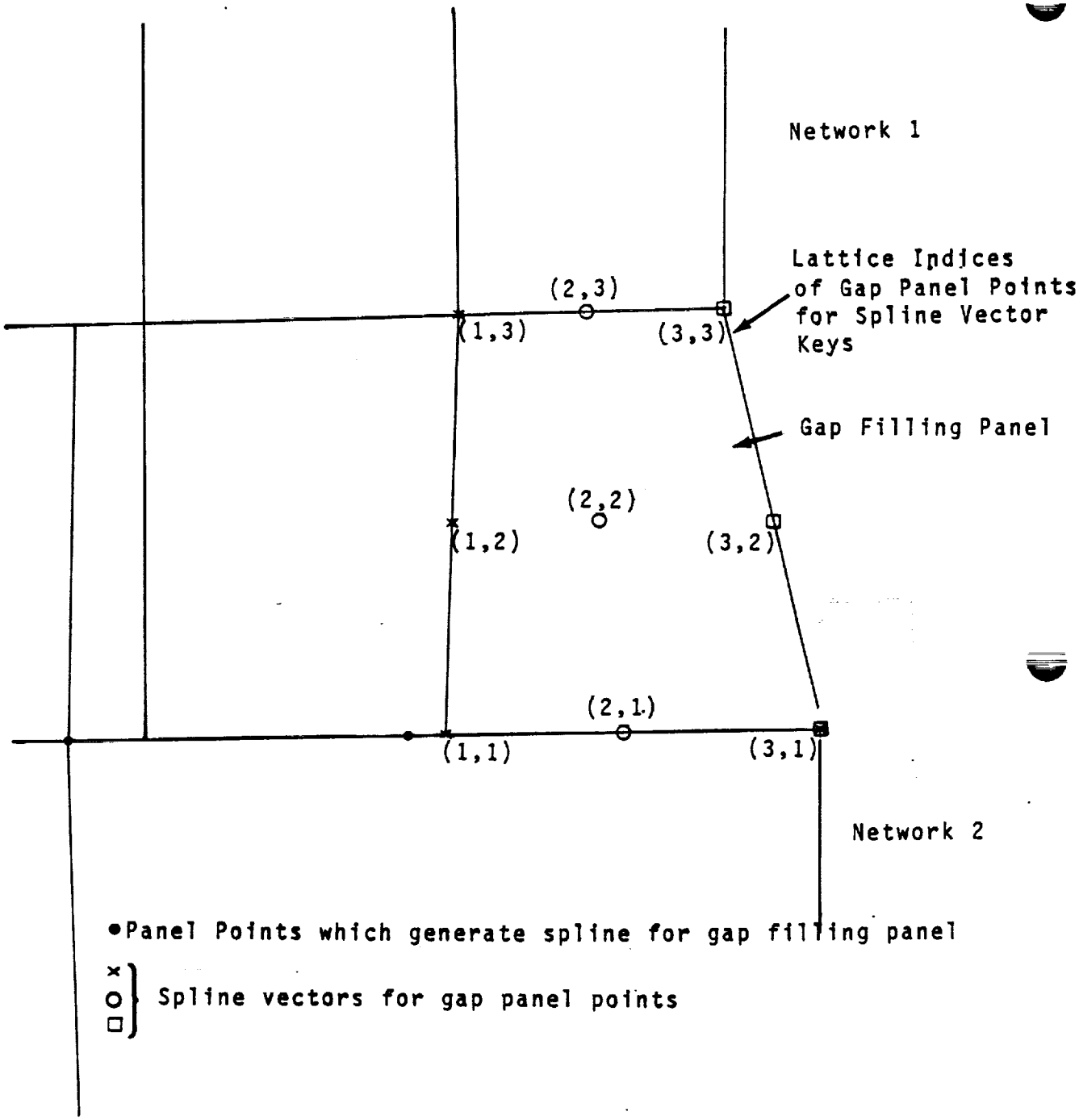


Figure 4-L.2 Indexing of points in a gap filling panel for use defining gap filling panel spline vectors

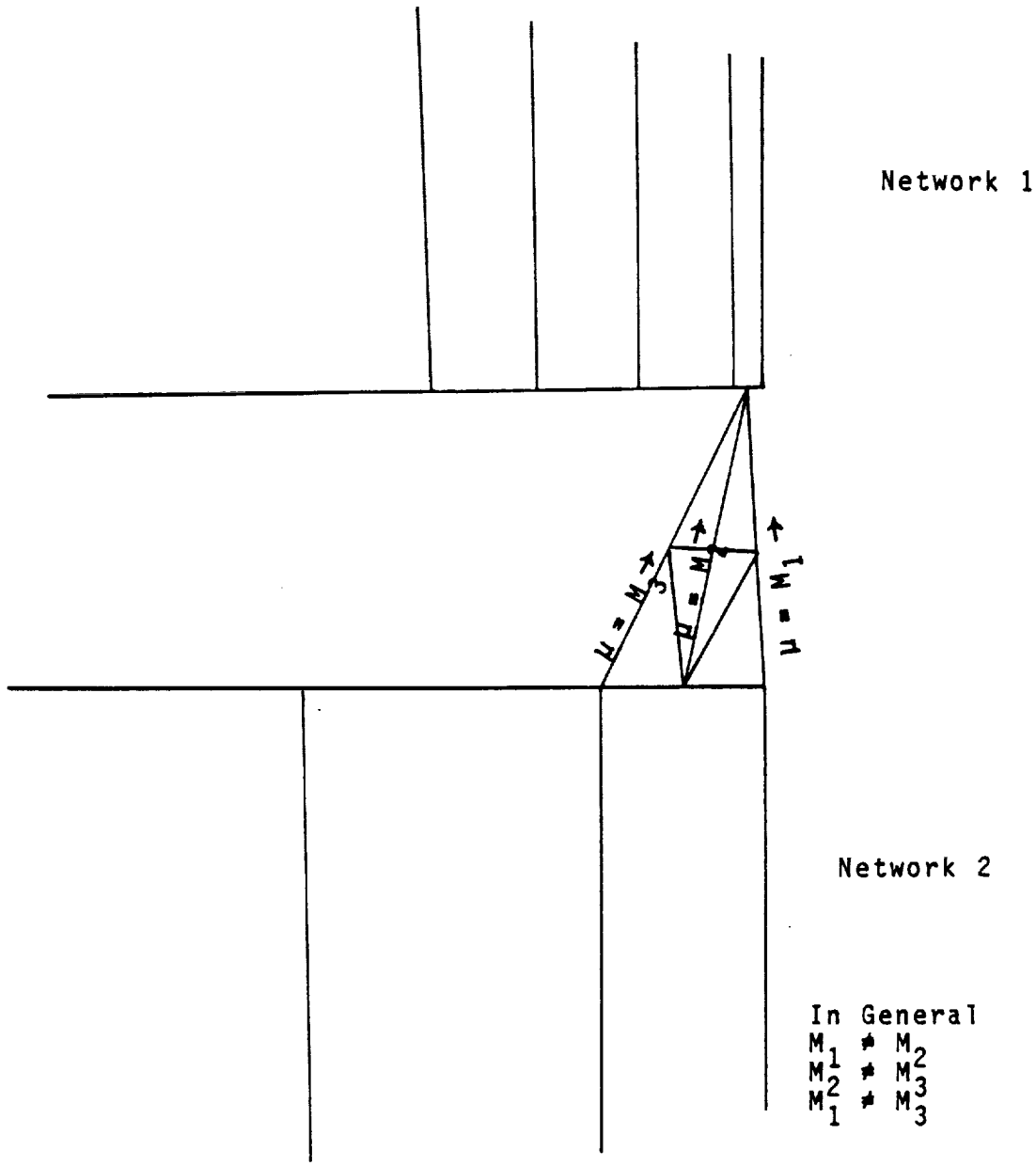


Figure 4-L.3 Excluded Special Case of Multiple Valued Doublet Strength for Gap Filling Panel

Doublet strength is constant along lines between networks. Thus if doublet doublet strength at the three points on network 2 are not equal, they give rise to an infinite derivative of doublet strength at the intersection point on network 1. In general the strength at the points on network 2 are equal.

APPENDIX 4-M

SELECTION OF BOUNDARY CONDITIONS

4-M.0 Introduction

The fourth overlay of DQG selects the appropriate boundary conditions to be imposed at each control point. The control points fall into three categories. Center control points are control points which lie at panel centers. Edge interior control points (refered to as edge control points) are those network edge control points which also lie on panel edge midpoints. The remaining control points are refered to as corner control points. They include network corner control points and the additional control points added by DQG to impose doublet matching at certain abutment intersections.

The algorithm used to assign boundary conditions depends on the control point type. Sections 4-M.1, 4-M.2 and 4-M.3 discuss the boundary condition assignments for center, edge and corner control points respectively. The boundary conditions may be subdivided into three groups: user specified, degenerate and DQG assigned. It is assumed here that the reader is familiar with the discussion of the formulation of the different boundary conditions found in Appendix H of the PAN AIR Theory Document (Reference 1). The assignment of user specified boundary conditions depends on a hierarchy. The implementation of the user specified boundary condition hierarchy will be discussed in section 4-M.4.

Before any boundary condition is assigned the number of boundary conditions that need to be imposed at each type of control point is determined. Program BNDYDF first counts the number of null boundary conditions at a control point. Boundary conditions will need to be imposed to bring the total number (including null boundary conditions) to at least two.

4-M.1 Selection of Boundary Conditions at Center Control Points

The selection of boundary conditions of center control points is made in a straight-forward manner in subroutine CNCPBC. User specified boundary conditions are assigned for each control point first. If the network lies in a plane of symmetry then degenerate boundary conditions replace the user specified boundary conditions. An antisymmetric boundary condition is replaced by a degenerate doublet boundary condition and a degenerate source replaces a symmetric boundary condition.

4-M.2 Selection of Boundary Conditions at Edge Interior Control Points

The boundary condition selection at edge control points is considerably more complicated than at center control points. The complications arise because a network edge may be in a plane of symmetry, on a plane of symmetry, on both planes of symmetry, or on neither planes of symmetry. The various possibilites are explained in Appendix H of the PAN AIR Theory Document (Reference 1). A sequence of subroutines is used to assign boundary conditions at edge points. Subroutine EDGECP reads the input boundary condition data, obtains the geometric data, initializes the output arrays and then calls subroutine ASGNBC. This subroutine prepares the user specified, matching and degenerate boundary conditions. DEGOUT determines if the assigned boundary conditions in the output array has the appropriate symmetry type. If it is not appropriate then it is replaced by a degenerate boundary condition.

For network edge and corner control points only one boundary conditon is normally required. The symmetry type of this boundary condition is determined by the network singularity type. It may happen that the first boundary condition specified by the user has a different symmetry type. In this case DEGOUT swaps the order of the user specified boundary condition in order to assign the boundary condition with the same symmetry type as the needed boundary condition.

4-M.3 Processing of Boundary Conditions at Corner Control Points

The most complex aspect of the corner point boundary condition assignment is the assignment of boundary conditions to insure doublet matching. The matching assignments are made in subroutines ASSIGN and ABTINT in the third overlay of DQG. At the end of the third overlay the matching information is written to the SPECIAL-POINTS data set in subroutine MTCHPT. The fourth overlay reads SPECIAL-POINTS data and interprets the matching data, assigns matching conditions and then writes out BNDRY-CONDN-SPEC data. This section describes how the matching data in the SPECIAL-POINTS data set is used to generate matching boundary conditions.

Arrays MCHCRP(20:4:2) and KSPPOS(20) in the SPECIAL-POINTS data set contain the information needed to assign matching conditions. The array KSPPOS(I) contains a value between 0 and 3. The value 0 indicates that the Ith special point is not on any plane of symmetry. Values of 1, 2 and 3 indicates that the special point is on the first, second and both plane(s) of symmetry respectively. The first index of MCHCRP refers to the special point. The second index refers to the symmetry condition. The third index refers to singularity type, 1 implying doublet and 2 implying source, although source matching cannot occur at corner points in PAN AIR. (Vorticity matching is also not imposed at corner points in PAN AIR.) Pointer spaces are provided to facilitate future modifications. If MCHCRP(I,J,1) contains 1 it means that the Ith special point of an edge E is to be used for doublet matching across the abutment in which edge E takes part. Furthermore if KSPPOS(I) is nonzero then the matching condition is assigned only for the Jth symmetry condition. However if KSPPOS(I) were zero (the Ith corner point is not on any plane of symmetry) then the assignment of the boundary conbdition would be independant of symmetry condition and if MCHCRP(I,1,1) contains 1 then the matching condition would be assigned to all symmetry conditions. The SPECIAL-POINTS dataset is keyed on network edges. A network corner point is defined by two edges. Therefore the matching condition at a corner point may be used to match doublet strength across either edge. In the fourth overlay of DQG subroutine EDGECP inspects the SPECIAL-POINTS data from the two adjacent edges and combines the matching flags to define a single matching flag, IMTCH. Then whenever IMTCH is not zero a matching boundary condition is assigned instead of a user specified boundary condition.

4-M.4 User Specified Boundary Conditions

The general form of the boundary condition equation as stated in the PAN AIR Theory Document, Appendix H (Reference 1) includes perturbation terms of average normal mass flux, average potential, tangential average velocity, difference normal mass flux (source), difference potential (doublet), and tangential difference velocity (doublet gradient). The coefficients of most

of these terms vanish for the typical case.

For most problems the user specifies two boundary conditions for every center control point. They may be specified in any order. Points on the network edge usually receive only one boundary condition. For simplicity, the user may specify the same two boundary conditions for the edge points as for the center points. DQG must select one of the boundary conditions to impose at the edge points.

Although the boundary conditions can be specified in any order, DQG must assign them according to a particular hierarchy. This hierarchy is:

1. difference potential (doublet)
2. average potential
3. average normal mass flux
4. tangential average velocity
5. tangential difference velocity (doublet gradient)
6. difference normal mass flux (source)
7. anything else

The procedure DQG uses to determine where the boundary conditions fit in the hierarchy merits some discussion. There are ten possible coefficients in the general boundary condition equation. A vector of length ten is defined (selection vector, SELVEC). Each component corresponds to one of the coefficients in the general BC equation. The vector is initialized to zero. For each coefficient which does not vanish the corresponding component of the vector is set to unity. The correspondence between the components of SELVEC and the boundary condition coefficient is listed below:

<u>Component</u>

1	coefficient of average normal mass flux
2	coefficient of average potential
3	x component of tangent vector for average velocity
4	y component of tangent vector for average velocity
5	z component of tangent vector for average velocity
6	coefficient of difference normal mass flux (source)
7	coefficient of difference potential (doublet)
8	x component of tangent vector for difference velocity (doublet gradient)
9	y component of tangent vector for difference velocity (doublet gradient)
10	z component of tangent vector for difference velocity (doublet gradient)

An array of projection vectors SVP(10,2,6) is defined. The first index varies over the components of the selection vector. The last index is over the six hierarchical categories. The middle index defines two complementary vectors. For example, SVP(I,1,1) = (0,0,0,0,0,0,1,0,0,0)
SVP(I,2,1) = (1,1,1,1,1,1,0,1,1,1)

This indicates (SVP(I,1,1)) that the seventh coefficient of difference potential (doublet) does not vanish and (SVP(I,2,1)) all other coefficients vanish.

Note that a boundary condition specifying doublet gives

$$SVP(I,1,1) \cdot SELVEC(I) = 1$$

and

$$SVP(I,2,1) \cdot SELVEC(I) = 0$$

The other five vectors SVP are defined in a similar fashion to allow specification of the hierarchical position of the boundary condition.

If the user boundary conditions do not fit in the six categories, they are used in the order the user has defined.

5.0 MAG MODULE

The primary function of the MAG module is to generate the MAK database which contains in particular the following three datasets.

(1) The AIC-MATRIX dataset giving the aerodynamic influence coefficient matrix [AIC] for each symmetry condition required by the user's problem formulation.

(2) The IC-MATRICES dataset containing the influence coefficients that describe the dependence of ϕ, $\vec{v}$ or $\vec{w}\cdot\hat{n}$ at selected control points upon the configuration's global singularity parameters.

(3) The MAG-PANEL-DATA dataset containing the essential panel data needed by the FDP module of PAN AIR.

These three datasets constitute the principal outputs from MAG. We begin this maintenance document for MAG by presenting a complete table of contents, listing the topics discussed.

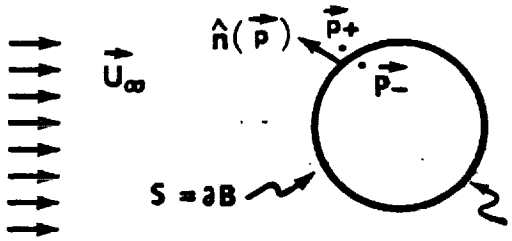
TABLE OF CONTENTS

TABLE OF CONTENTS (cont.)

5.1 INTRODUCTION

Within the general context of numerical methods for the solution of partial differential equations, the method which PAN AIR uses to solve the potential flow equation is the method of collocation applied to a boundary integral equation. In order to understand how MAG fits in to this general scheme, it is useful to consider the formulation of a specific problem.

Consider the figure below which summarizes the essential aspects of the problem of incompressible potential flow about a sphere, B. We will consider



Total Velocity $= \vec{V} = \vec{U}_\infty + \vec{v} = \vec{U}_\infty + \nabla\phi$

Perturbation Velocity $= \vec{v} = \nabla\phi$

Continuity: $\nabla\cdot\vec{v} = \nabla^2\phi = 0$

Impermeable Surface: $\vec{V}\cdot\hat{n} = (\vec{U}_\infty + \vec{v})\cdot\hat{n} = 0$

two different methods of formulating this problem as a boundary integral equation. The principle difference between these two formulations is the way in which they define ϕ interior to the sphere. Since ϕ in the interior of the sphere is of no interest anyway, each formulation can be regarded as being equally legitimate, at least from a theoretical point of view.

5.1.1 Formulation 1, Morino's Method [$\phi = 0$ inside B]

Here we set $\phi(\vec{p}) = 0$ for $\vec{p}\epsilon B$. Consequently, by the uniqueness theorem for the solution of Laplace's equation, $\nabla\phi = 0$ for all $\vec{p}\epsilon B$, and in particular, $(\partial\phi/\partial n)_{\vec{p}_-} = 0$. ($\vec{p}_-$ is a point on the boundary of B, just inside B). It is also true, obviously, that $\phi(\vec{p}_-) = 0$. In view of the definitions of source σ and doublet μ,

$$\sigma = (\partial\phi/\partial n)_{\vec{p}_+} - (\partial\phi/\partial n)_{\vec{p}_-} \tag{5.1.1}$$

$$\mu = (\phi)_{\vec{p}_+} - (\phi)_{\vec{p}_-} \tag{5.1.2}$$

we find that

$$\sigma = (\partial\phi/\partial n)_{\vec{p}_+} \tag{5.1.3}$$

$$\mu = (\phi)_{\vec{p}_+} \tag{5.1.4}$$

The value of $(\partial\phi/\partial n)_{\vec{p}_+}$ can be obtained directly from the impermeable surface boundary condition, $(\vec{V}\cdot\hat{n})_{\vec{p}_+} = 0$:

$$0 = (\vec{V}.\hat{n})_{\vec{p}_+} = \vec{U}_\infty .\hat{n} + (\vec{v}.\hat{n})_{\vec{p}_+} = \vec{U}_\infty .\hat{n} + (\partial\phi/\partial n)_{\vec{p}_+}$$

Thus,

$$\sigma = (\partial\phi/\partial n)_{\vec{p}_+} = -\vec{U}_\infty .\hat{n} \tag{5.1.5}$$

and σ is determined.

At this point we need to invoke the integral representation formula for ϕ, Green's third identity (cf. theory document, eqn. (3.2.7) with $\psi = 1/R$). Evaluating it at $\vec{p}_+$ and $\vec{p}_-$, we get

$$\phi(\vec{p}_\pm) = \left[-\frac{1}{4\pi} \iint_{\partial B} \sigma(\vec{q}) \ \psi(\vec{p},\vec{q}) \ dS_q + \frac{1}{4\pi} \iint_{\partial B} \mu(\vec{q}) \ (\partial \ \psi/\partial n_q) \ dS_q \right]_{\vec{p}_\pm} \tag{5.1.6}$$

Averaging these two expressions, we obtain a representation formula for ϕ_{avg}:

$$\frac{1}{2} [\phi(\vec{p}_+) + \phi(\vec{p}_-)] = \left[-\frac{1}{4\pi} \iint_{\partial B} \sigma \ \psi \ dS_q + \frac{1}{4\pi} \iint_{\partial B} \mu \ \ \psi_n \ dS_q \right]_{avg} \tag{5.1.7}$$

Now $\phi(\vec{p}_-) = 0$ and $\phi(\vec{p}_+) = \mu$. Using these facts in (5.1.7) and rearranging, we obtain the <u>Fredholm integral equation of the second kind</u> for μ:

$$\frac{1}{2} \mu(\vec{p}) - \left(\frac{1}{4\pi} \iint_{\partial B} \mu \ \ \psi_n \ \ dS_q \right)_{avg} = \left(-\frac{1}{4\pi} \iint_{\partial B} \sigma \ \ \psi \, dS_q \right)_{avg} \tag{5.1.8}$$

We put the integral involving σ on the right hand side because σ is known from the boundary condition data (cf. eqn. (5.1.5)).

The numerical solution of (5.1.8) by the method of collocation requires the evaluation of the source potential ϕ_S and the doublet potential ϕ_D,

$$\phi_S(\vec{p}) = \left(-\frac{1}{4\pi} \iint_{\partial B} \sigma(\vec{q}) \ \ \psi(\vec{p},\vec{q}) \ dS_q \right)_{avg} \tag{5.1.9}$$

$$\phi_D(\vec{p}) = \left(\frac{1}{4\pi} \iint_{\partial B} \mu(\vec{q}) \ (\partial \psi \ /\partial n_q) \ dS_q \right)_{avg} \tag{5.1.10}$$

where the singularity distributions σ and μ are assumed to have finite dimensional representations of the form

$$\sigma(\vec{q}) = \sum_{I=1}^{N} s_I \ (\vec{q}) \ \lambda_I \tag{5.1.11a}$$

$$\mu(\vec{q}) = \sum_{I=1}^{N} m_I \ (\vec{q}) \ \lambda_I \tag{5.1.11b}$$

The basis functions s_I, m_I are independent of the solution and, in PAN AIR, are defined by the module DQG. The global singularity parameters λ_I are either of "source type" or "doublet type". This fact is expressed formally by the implications:

$$s_I \text{ is not the zero function} \rightarrow m_I = 0 \tag{5.1.12a}$$

$$m_I \text{ is not the zero function} \rightarrow s_I = 0 \tag{5.1.12b}$$

In the current problem, the source parameters will all be known (since σ is known) and the doublet parameters will all be unknown until they are determined by the application of the collocation conditions.

If we agree that the N_μ unknown doublet parameters appear first in $\{\lambda_I\}$, followed by the N_σ known source parameters, then we observe that N_μ equations in the N_μ unknown values of λ_I can be obtained by evaluating equation (5.1.8) at N_μ points, called control points and denoted by $\vec{p}_I$, $I = 1,\ldots,N_\mu$. Performing this evaluation, one obtains the collocation conditions,

$$\sum_{J=1}^{N_\mu} A_{IJ}\,\lambda_J = b_I \tag{5.1.13}$$

where

$$A_{IJ} = \frac{1}{2}\, m_J(\vec{p}_I) - \left(\frac{1}{4\pi} \iint_{\partial B} m_J(\vec{q})\, \frac{\partial\, \psi\,(\vec{q},\, \vec{p}_I)}{\partial n_q}\, dS_q \right)_{avg} \tag{5.1.14}$$

$$b_I = \sum_{J=N_\mu+1}^{N_\mu+N_\sigma} \left(-\frac{1}{4\pi} \iint_{\partial B} s_J(\vec{q})\;\; \psi(\vec{q},\, \vec{p}_I)\, dS_q \right)_{avg} \lambda_J \tag{5.1.15}$$

The evaluation of the integrals appearing in (5.1.14-15) is discussed in appendix J of the Theory Document. In particular, the procedures necessary to obtain the average value integrals are discussed in appendix J.8.

We note in passing that the matrix elements A_{IJ} are called aerodynamic influence coefficients, or AIC's for short. In actual practice, some of the equations in (5.1.13) are replaced by doublet matching conditions of the form:

$$\sum_{k=1}^{n} s_k \left[\mu(\vec{p}) \;\middle|\; \text{network } k \right] = 0 \tag{5.1.16}$$

where $\vec{p}$ is a doublet matching point and the numbers s_k take on the values ± 1. For a detailed discussion of doublet matching along network abutments and at abutment intersections, see appendices B.3 and F of the theory document. In any event, we eventually obtain a system of equations of the form (5.1.13) which we solve to obtain the previously unknown singularity parameters,

λ_J, J=1, ... N_μ. Once all of the singularity parameters are known, σ and μ are completely known from the representation formulas (5.1.11). From these, ϕ is determined by equation (5.1.6) so that the problem is effectively solved.

5.1.2 Formulation 2, Hess' Method [$\mu = 0$ on ∂B]

In this method, we do not actually specify what ϕ is inside B, but rather define it indirectly by choosing the doublet strength to be identically zero. Thus, we assume that ϕ can be represented by a source distribution by the equation

$$\phi(\vec{p}) = -\frac{1}{4\pi}\iint_{\partial B} \sigma(\vec{q})\ \psi(\vec{p},\vec{q})\ dS_q \tag{5.1.17}$$

This expression may be differentiated to give values of (ϕ_n) at $\vec{p}_+$ and $\vec{p}_-$:

$$(\partial\phi/\partial n)_{\vec{p}_\pm} = \frac{1}{4\pi}\ \hat{n}(\vec{p}).\iint_{\partial B} \sigma(\vec{q})\ \nabla_q\ \psi(\vec{p}_\pm,\vec{q})\ dS_q \tag{5.1.18}$$

Evaluating the average and difference of $(\partial\phi/\partial n)_{\vec{p}_\pm}$, we find

$$(\partial\phi/\partial n)_{\vec{p}_+} - (\partial\phi/\partial n)_{\vec{p}_-} = \sigma(\vec{p})$$

$$\frac{1}{2}\left[(\partial\phi/\partial n)_{\vec{p}_+} + (\partial\phi/\partial n)_{\vec{p}_-}\right] = \frac{1}{4\pi}\ \hat{n}(\vec{p}).\left[\iint_{\partial B} \sigma(\vec{q})\ \nabla_q\ \psi(\vec{p},\vec{q})\ dS_q\right]_{avg}$$

Combining these, we may eliminate $(\partial\phi/\partial n)_{\vec{p}_-}$ to obtain

$$(\partial\phi/\partial n)_{\vec{p}_+} = \frac{1}{2}\ \sigma(\vec{p}) + \frac{1}{4\pi}\ \hat{n}(\vec{p}).\left[\iint_{\partial B} \sigma(\vec{q})\ \nabla_q\ \psi(\vec{p},\ \vec{q})\ dS_q\right]_{avg} \tag{5.1.19}$$

The boundary condition $(\vec{V}.\hat{n})_{\vec{p}_+} = 0$ may now be imposed to obtain

$$0 = (\vec{V}.\hat{n})_{\vec{p}_+} = \vec{U}_\infty.\hat{n} + (\vec{v}.\hat{n})_{\vec{p}_+} = \vec{U}_\infty\ .\hat{n} + (\partial\phi/\partial n)_{\vec{p}_+}$$

Substituting this into (5.1.19), we again obtain a Fredholm integral equation of the second kind,

$$\frac{1}{2}\ \sigma(\vec{p}) + \frac{1}{4\pi}\ \hat{n}(\vec{p}).\left(\iint_{\partial B} \sigma(\vec{q})\ \nabla_q\ \psi(\vec{p},\ \vec{q})\ dS_q\right)_{avg} = -\vec{U}_\infty\ .\hat{n} \tag{5.1.20}$$

In this formulation, the N_σ source parameters are the basic unknowns and are determined by imposing the integral equation at N_σ control points $\vec{p}_I$ and solving the resultant system of linear equations. Using the usual representation formula (cf. 5.1.11a), we obtain the linear system

$$\sum_{J=1}^{N_\sigma} A_{IJ}\ \lambda_J = b_I \tag{5.1.21}$$

where

$$A_{IJ} = \frac{1}{2} s_J(\vec{p}_I) + \frac{1}{4\pi} \hat{n}(\vec{p}_I).\left(\iint_{\partial B} s_J(\vec{q}) \nabla_q \psi(\vec{p}_I, \vec{q}) dS_q\right)_{avg} \tag{5.1.22}$$

$$b_I = -\vec{U}_\infty .\hat{n}(\vec{p}_I) \tag{5.1.23}$$

As before, the determination of all the singularity parameters provides us with a complete representation for σ and, by way of equation (5.1.17), a representation for ϕ.

Before moving on to the definitions of influence coefficients, we point out that the two formulations given here do not exhaust the possible formulations of the problem. In Appendix H, we discuss two other formulations:

(1) A modified Morino formulation is presented in which total Φ is set equal to zero in the sphere interior ($\Phi_L = 0$). This formulation has the advantage that no source IC's are required at all ($\sigma = 0$).

(2) A formulation in which both boundary conditions:

(a) $\phi_L = 0$ (lower surface perturbation stagnation)

(b) $\vec{V}.\hat{n} = 0$ (direct velocity boundary condition)

are imposed explicitly, so that both source and doublet parameters are unknowns in the problem. While this formulation is more costly, it produces better answers near the stagnation point on the surface of B.

In addition to discussing these other two formulations, appendix H also explicitly states and compares the integral equations corresponding to each of the four formulations.

The main purpose of studying these various formulations of the same potential flow problem is to emphasize the fact that in PAN AIR, problem formulation is the task of the program user. In particular we wish to point out that the fundamental integral representation formula (cf. eqn. (5.1.6)):

$$\phi(\vec{p}) = \frac{1}{\kappa} \iint_{S \cap D_p} [-\sigma/R + \mu \hat{n}.\nabla(1/R)] dS_q$$

is <u>not</u> an integral equation. Rather, the PAN AIR user implicitly uses this representation formula to help describe to the program the integral equation that he needs to solve. While it is true that Class I boundary conditions provide a convenient means of obtaining a standard formulation (essentially Morino's formulation), it should not be forgotten that many other formulations are possible and can be employed with PAN AIR. In particular, all four formulations of potential flow around a sphere described here and in Appendix H can be employed using PAN AIR. The function of PAN AIR is to perform the many and complex tasks of problem analysis and numerical solution once the user has formulated his problem.

5.1.3 Definitions of Influence Coefficients

Having described these two formulations of the problem of flow about a sphere, we have provided the background needed to motivate the definitions of the potential and velocity influence coefficients that MAG computes. The potential influence coefficients at a control point $\vec{p}$, motivated by the integral expressions appearing in (5.1.14) and (5.1.15), are defined by the formulas

$$\Phi IC_J^S(\vec{p}) = -\left(\frac{1}{\kappa} \iint_{\partial B} s_J(\vec{q})\ \psi(\vec{q}, \vec{p})\ dS_q\right)_{avg} \tag{5.1.24}$$

$$\Phi IC_J^D\ (\vec{p}) = \left(\frac{1}{\kappa} \iint_{\partial B} m_J(q)\ \hat{n}(\vec{q}).\ B\ \ \nabla_q\ \psi(\vec{q}, \vec{p})\ dS_q\right)_{avg} \tag{5.1.25}$$

where ψ is the general kernel function,

$$\psi = \begin{cases} 1/[\vec{q} - \vec{p}, \vec{q} - \vec{p}]^{1/2} & \vec{q}\varepsilon D_p \\ 0 & \vec{q}\varepsilon D_p \end{cases} \tag{5.1.26}$$

and κ is given by

$$\kappa = \begin{cases} 4\pi \text{ subsonic flow} \\ 2\pi \text{ supersonic flow} \end{cases} \tag{5.1.27}$$

and B is the dual metric which, in the compressibility axis coordinate system, takes the form

$$B = \begin{bmatrix} s\beta^2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad s\beta^2 = 1 - M_\infty^2 \tag{5.1.28}$$

Because each global singularity parameter λ_J is either a source or a doublet parameter, only one of ΦIC_J^S and ΦIC_J^D can be nonzero. Thus, no information is lost by defining ΦIC_J by

$$\Phi IC_J = \Phi IC_J^S + \Phi IC_J^D. \tag{5.1.29}$$

With these definitions, the value of $[\phi(\vec{p})]_{avg}$ is given by

$$[\phi(\vec{p})]_{avg} = \sum_{J=1}^{N} \Phi IC_J\ (\vec{p})\ \lambda_J \tag{5.1.30}$$

The coefficients A_{IJ} and b_I of eqns. (5.1.14-15) (Morino's formulation) are given in terms of these potential influence coefficients by

$$A_{IJ} = \frac{1}{2} m_J(\vec{p}_I) - \Phi IC_J^D (\vec{p}_I) \qquad (5.1.14)'$$

$$b_I = \sum_{J=N_\mu+1}^{N_\mu+N_\sigma} \Phi IC_J^S (\vec{p}_I) \lambda_J \qquad (5.1.15)'$$

By substituting these expressions into (5.1.13) and using (5.1.30), we can recover the original condition, (5.1.8). Doing the substitution, one has

$$\sum_{J=1}^{N_\mu} (\frac{1}{2} m_J(\vec{p}_I) - \Phi IC_J^D (\vec{p}_I)) \lambda_J = \sum_{J=N_\mu+1}^{N_\mu+N_\sigma} \Phi IC_J^S (\vec{p}_I) \lambda_J$$

Rearranging,

$$\frac{1}{2} \sum_{J=1}^{N_\mu} m_J(\vec{p}_I) \lambda_J = \sum_{J=1}^{N_\mu} \Phi IC_J^D (\vec{p}_I) \lambda_J + \sum_{J=N_\mu+1}^{N_\mu+N_\sigma} \Phi IC_J^S (\vec{p}_I) \lambda_J$$

$$= \sum_{J=1}^{N_\mu+N_\sigma} (\Phi IC_J^D (\vec{p}_I) + \Phi IC_J^S(\vec{p}_I)) \lambda_J$$

$$= \sum_{J=1}^{N} \Phi IC_J (\vec{p}_I) \lambda_J$$

Invoking (5.1.30) and the representation formula for μ (5.1.11b), we find

$$\frac{1}{2} \mu(\vec{p}_I) = (\phi(\vec{p}_I))_{avg} \qquad (5.1.31)$$

which is clearly equivalent to (5.1.8) evaluated at $\vec{p}_I$.

The integral expression appearing in (5.1.22) motivates the definition of source velocity influence coefficients (VIC's), viz.,

$$\overrightarrow{VIC}_J^S (p) = (\frac{1}{\kappa} \iint s_J (\vec{q}) \nabla_q \psi(\vec{q},\vec{p}) dS_q)_{avg} \qquad (5.1.32)$$

It is clear that $\overrightarrow{VIC}_J^S (\vec{p})$ is the average, above and below the singularity surface, of the expression*:

---

*The calculation goes as follows:

$$\nabla_p [- \frac{1}{\kappa} \iint s_J (\vec{q}) \nabla (\vec{q},\vec{p}) dS_q] = - \frac{1}{\kappa} \iint s_J(\vec{q}) (\nabla_p \psi(\vec{q},\vec{p})) dS_q$$

$$= (- \frac{1}{\kappa}) \iint_{\partial B} s_J (\vec{q}) [(-1) \nabla_q \psi(\vec{q},\vec{p})] dS_q$$

where we use the identity $\nabla_q \psi = - \nabla_p \psi$.

$$\nabla_p \left[-\frac{1}{\kappa} \iint_{\partial B} s_J(\vec{q}) \; \psi(\vec{q},\vec{p}) \, dS_q \right]$$

where the expression in square brackets bears an obvious relationship to the definition (5.1.24) of source potential influence coefficients. Given this fact, one is motivated to define doublet velocity influence coefficients by averaging the expression

$$\nabla_p \left[\frac{1}{\kappa} \iint_{\partial B} m_J(\vec{q}) \; \hat{n}(\vec{q}) \cdot \; B \; \nabla_q \; (\vec{q},\vec{p}) \, dS_q \right]$$
$$= -\frac{1}{\kappa} \iint m_J(\vec{q}) \; \hat{n}(\vec{q}) \cdot \; B \; \nabla_q \; \nabla_q \; \psi(\vec{q},\vec{p}) \, dS_q$$

above and below the singularity surface.

This leads to the definition,

$$\overrightarrow{VIC}_J^D(\vec{p}) = \left(-\frac{1}{\kappa} \iint_{\partial B} m_J(q) \; \hat{n}(\vec{q}) \cdot \; B \; \nabla_q \; \nabla_q \; \psi(\vec{q},\vec{p}) \, dS_q \right)_{avg} \tag{5.1.33}$$

When the doublet basis functions m_J are continuous interior to a network of panels, the line vortex integration by parts may be performed on the integral appearing in (5.1.33) to yield (cf. Appendix B.3, theory document)

$$\overrightarrow{VIC}_J^D(\vec{p}) = \left(\frac{1}{\kappa} \iint_{\partial B} (\hat{n} \times \nabla_q m_J) \times (B \; \nabla_q \; \psi) \, dS_q \right)_{avg}$$
$$+ \frac{1}{\kappa} \int_{\substack{\text{network} \\ \text{boundaries}}} m_J \; (B \; \nabla_q \; \psi) \times d\vec{r} \tag{5.1.34}$$

The right hand side of this expression consists of two parts, a surface integral called the "regular part" and a line integral called the "line vortex" part. When doublet matching is imposed along the network boundaries, the individual network contributions to the line vortex terms all cancel and may thus be discarded. In the instance of supersonic flow this is essential, since the line vortex integrals diverge at any point on the Mach cone which emanates from a kink in a network boundary. The only situation in which PAN AIR is designed to include the line vortex terms in the evaluation of $\overrightarrow{VIC}_J^D$ is when the following conditions are satisfied*:

(i) the flow is subsonic ($M_\infty < 1$)
(ii) the user has specified the particular network edge to be a "no doublet edge matching" edge.

The last type of influence coefficients, the normal mass flux IC's are defined in terms of the velocity influence coefficients and the panel normal at the control point, $\hat{n}(\vec{p})$, by the expressions

---

*Note: No line vortex terms have been implemented in version 3.0, although the needed interfaces for them have been included in MAG.

$$WIC_J^S(\vec{p}) = \hat{n}(\vec{p}) \cdot B\, \overrightarrow{VIC}_J^S(\vec{p})$$

$$WIC_J^D(\vec{p}) = \hat{n}(\vec{p}) \cdot B\, \overrightarrow{VIC}_J^D(\vec{p})$$

Here, B denotes the usual dual metric matrix (cf. Appendix E, PAN AIR Theory Document).

5.2 MAG Overview

5.2.1 Purpose of MAG

The module MAG (Matrix Generator) performs three primary tasks in the PAN AIR system. It generates

(i) the AIC-MATRIX dataset of the MAK database, (this dataset contains an AIC matrix for each required symmetry condition),

(ii) the IC-MATRICES dataset of the MAK database, (this dataset contains potential, velocity or normal mass flux influence coefficients as requested by the program user),

(iii) the MAG-PANEL-DATA dataset of the MAK database that is subsequently transcribed by MDG to the MDG database for subsequent use by FDP.

In addition to these three principal datasets, MAG generates a number of other datasets that aid in the use of these three principal datasets. These other datasets include

DATA-BASE-HEADER SYMMETRY	Contain global information.
COLMAP COLMAP-INVERSE COLMAP-BULK	Contain the connection between DQG and MAG singularity parameter indices.
ROWMAP ROWMAP-INVERSE ROWMAP-BULK	Contain control point information, the connection between DQG and MAG control point indices.

Three other small datasets are also written to the MAK database to help in the analysis of program errors, should they occur. These are

SOURCE-MATCHING DOUBLET-MATCHING VORTICITY-MATCHING	Contain a thorough analysis of all matching conditions.
PANEL-GROUP	Contains panel grouping information.

These datasets are also used by MAG in the actual generation of AIC's and IC's.

In the short discussions of these datasets that follow, we will first treat the principal datasets and following this, the auxiliary datasets.

5.2.1.1 The Principal Datasets, AIC-MATRIX and IC-MATRICES

The first of the two principle datasets generated by MAG contains AIC's, that is, aerodynamic influence coefficients. Each row of an AIC matrix describes the dependency of an imposed boundary condition upon the known and unknown singularity parameters, $\vec{\lambda}_K$ and $\vec{\lambda}_U$ that together represent σ and μ on the configuration's singularity surface. Thus the ith imposed boundary condition generates the AIC equation

$$\llcorner AIC_{U,i} \lrcorner \vec{\lambda}_U + \llcorner AIC_{K,i} \lrcorner \vec{\lambda}_K = b_i \tag{5.2.1}$$

The precise way by which the various types of boundary conditions generate AIC rows is described in detail in appendix K of the theory document. Aggregating all of the AIC conditions into a single matrix equation, we obtain

$$[AIC_U] \vec{\lambda}_U + [AIC_K] \vec{\lambda}_K = \vec{b} \tag{5.2.2}$$

Here, AIC_U will be square provided the number of imposed boundary conditions equals the dimension of $\vec{\lambda}_U$. When this happens (it is a fatal program error if it does not occur) we obtain the standard linear system

$$[AIC_U] \vec{\lambda}_U = \vec{b} - [AIC_K] \vec{\lambda}_K \tag{5.2.3}$$

In PAN AIR, RMS performs the L-U factorization of AIC_U while RHS computes $\vec{\lambda}_K$, forms the right hand side of (5.2.3) and performs forward and backward substitution to solve for $\vec{\lambda}_U$.

The second principle dataset generated by MAG, IC-MATRICES, gives the $\vec{\lambda}$ dependency of ϕ_A (average potential), $\vec{v}_A$ (average velocity) or $(w_n)_A$ (average normal mass flux) at selected control points.

These dependencies are expressed in the equation,

$$\begin{bmatrix} \phi_A \\ \hline \vec{v}_A \\ \hline (w_n)_A \end{bmatrix} = \begin{bmatrix} \llcorner \Phi IC \lrcorner \\ \hline [VIC] \\ \hline \llcorner WIC \lrcorner \end{bmatrix} \vec{\lambda} \qquad \vec{\lambda} = \begin{bmatrix} \vec{\lambda}_U \\ \hline \vec{\lambda}_K \end{bmatrix} \tag{5.2.4}$$

The coefficient matrix on the right is called the integral IC matrix for the control point and is denoted $[IC_A]$. (Here, the subscript "A" is used to connote "average".) The computations for $\llcorner \Phi IC \lrcorner$ and [VIC] are described in detail in appendices J and K of the theory document, appendix J giving the details of the panel influence coefficient (PIC) computations and appendix K describing how PIC's are assembled to get $\llcorner \Phi IC \lrcorner$ and [VIC]. The scalar normal mass flux IC's are defined by and can be computed from the formula:

$$\llcorner WIC \lrcorner = \hat{n}_o^T B_o [VIC] \tag{5.2.5}$$

where $\hat{n}_o$ denotes the surface normal at the control point (reference coordinates) and B_o denotes the standard dual metric in reference coordinates (cf. theory document, eqn. E.3.9).

An AIC row associated with a general boundary condition of the form

$$\begin{aligned} & c_A \phi_A + \vec{t}_A \cdot \vec{v}_A + a_A (\vec{w} \cdot \hat{n})_A \\ & + c_D \mu + \vec{t}_D \cdot \nabla \mu + a_D \sigma = b \end{aligned} \tag{5.2.6}$$

is readily computed from a control point's integral IC matrix $[IC_A]$ and its

singularity IC matrix $[IC_D]$, defined implicitly by the relation

$$\begin{bmatrix} \mu(\vec{p}) \\ \hline \nabla\mu(\vec{p}) \\ \hline \sigma(\vec{p}) \end{bmatrix} = [IC_D]\,\vec{\lambda} \tag{5.2.7}$$

(The subscript "D" in IC_D is used to connote "difference".) Using these, one finds that an AIC row is defined by

$$\llcorner AIC \lrcorner = \llcorner c_A, \vec{t}_A^T, a_A \lrcorner\ [IC_A] + \llcorner c_D, \vec{t}_D^T, a_D \lrcorner\ [IC_D] \tag{5.2.8}$$

Although eqn. (5.2.8) expresses correctly the evaluation of $\llcorner AIC \lrcorner$ in a formal sense, in actual practice MAG proceeds slightly differently so as to reduce processing costs. First note that not all rows of $[IC_A]$ may be required to evaluate equation (5.2.8). Thus if c_A, $\vec{t}_A$ or a_A is identically zero, the evaluation of the corresponding rows of $[IC_A]$ is not needed and may be suppressed. Another way cost can be reduced is by noting that if $\vec{t}_A = 0$, then

$$\llcorner \vec{t}_A^T, a_A \lrcorner \begin{bmatrix} VIC \\ \hline WIC \end{bmatrix} = (\vec{t}_A^T + a_A \hat{n}_o^T B_o)\ [VIC]$$

eliminating the need to explicitly compute $\llcorner WIC \lrcorner$. Finally we note that because $\mu(\vec{p})$ and $\nabla\mu(\vec{p})$ can depend on at most 25 doublet parameters, and $\sigma(\vec{p})$ can depend on at most 10 source parameters, there is no need to form $[IC_D]$ explicitly. Rather, its contribution is simply added in to the AIC row for the few entries of $\llcorner AIC \lrcorner$ that are actually affected.

It is appropriate to point out in this discussion of the principal datasets computed by MAG, that not all of the AIC rows are of the form (5.2.8). In particular, three types of matching conditions (source, doublet and velocity jump) and a closure condition can arise. The matching conditions, which are imposed along abutments, have the forms:

source matching: $$\sum_{\substack{\text{edges } E_k \\ \text{of abutment } A}} s_k\ \sigma(\vec{p}_k) = 0 \tag{5.2.9}$$

doublet matching: $$\sum_{\substack{\text{edges } E_k \\ \text{of abutment } A}} s_k\ \mu(\vec{p}_k) = 0 \tag{5.2.10}$$

velocity jump matching: (vorticity matching) $$\sum_{\substack{\text{edges } E_k \\ \text{of abutment } A}} s_k \vec{t}_k \cdot \left[\frac{\sigma(\vec{p}_k)\ \hat{n}_k}{(\hat{n}_k, \vec{\nu}_k)} + \nabla\mu(\vec{p}_k)\right] = 0 \tag{5.2.11}$$

In these equations (more fully developed in Appendices B.3, F.4, F.5, H.2, K.1 and K.6 of the theory document), we have used the following notation:

E_k denotes an edge of some network N_k participating in abutment A.

$\vec{p}_k$ denotes a point on edge E_k in network N_k at which the matching condition is being imposed. Note that all of the points $\vec{p}_k$ are essentially coincident.

s_k denotes the orientation of edge E_k relative to the intrinsic orientation of the abutment A. $s_k = \pm 1$.

$\vec{t}$ in the velocity jump matching condition, describes the direction of the component of velocity to be matched. The vector $\vec{t}$ points downstream along a wake surface. In the normal case that edge 1 is the wake's matching edge, $\vec{t}$ is calculated as the local column direction at the control point.

$\hat{n}_k$, $\vec{\nu}_k$ denote the normal and conormal on network N_k at $\vec{p}_k$.

When the singularity distributions are expressed in terms of the problem's singularity parameters, as in equations (5.1.11), the matching conditions (5.2.9) (5.2.10) and (5.2.11) define AIC rows in the usual way.

A closure condition can be expressed formally by the equation:

$$\sum_{\substack{\text{panel center} \\ \text{control points}}} A_k \left[a_{A,k} \hat{n}_k \cdot \vec{w}_A(\vec{p}_k) + a_{D,k}\, \sigma(\vec{p}_k)\right] = b \qquad (5.2.12)$$

Here, the sum extends over panel center control points $\vec{p}_k$ in some row or column of panels in a network. A_k denotes the area of the panel in which $\vec{p}_k$ lies and $a_{A,k}$, $a_{D,k}$ and b are coefficients provided by the user. Again, this equation generates a row in an AIC matrix when $\sigma(\vec{p}_k)$ and $\hat{n}_k \cdot \vec{w}_A(\vec{p}_k)$ are expressed in terms of $\vec{\lambda}$. Note that from equation (5.2.4)

$$\hat{n}_k \cdot \vec{w}_A(\vec{p}_k) = (w_n)_A(\vec{p}_k) = \lfloor WIC \rfloor \Big|_{\vec{p}_k} \vec{\lambda}$$

As a consequence, when a network has associated with it closure boundary conditions, the control points of that network are processed in such an order that the effect of the normal mass flux influence coefficients $\lfloor WIC \rfloor$ can be included into a "closure AIC buffer" as the different $\lfloor WIC \rfloor$ vectors are computed. The handling of closure thus induces a significant amount of complexity in the structure of MAG. For more details see Appendix 5-G.

The careful reader of this document will notice that nothing has been said up to this point about the impact of configuration symmetry upon the operation of MAG. In fact, as a consequence of the extensive analysis worked out in Appendices F.5, H and K of the PAN AIR theory document, the actual mechanics of handling symmetry are fairly simple. The main points to remember are the following:

(1) The AIC equation is formulated separately for each symmetry condition. Thus, for each of the four possible symmetrized potentials $\hat{\phi}^{ij}$, MAG has the job of computing the matrix AIC^{ij} in the symmetrized AIC equations

$$[AIC^{ij}]\,\hat{\lambda}^{ij} = [AIC_U^{ij},\ AIC_K^{ij}]\begin{Bmatrix}\hat{\lambda}_U^{ij}\\ \hat{\lambda}_K^{ij}\end{Bmatrix} = \hat{b}^{ij} \quad \text{(no summation over i and j)} \qquad (5.2.13)$$

Because the problem is separately formulated for each symmetry condition, we find that subroutines SDMTCH and MATCH must treat each symmetry condition separately when they analyze a matching condition imposed at a corner or extra control point (ITYPCP = 3 or 4) lying on a plane of symmetry (KSYMCP $\neq$ 0). (See Appendix F.5 of the theory document for a full discussion of the impact of symmetry upon doublet matching at an abutment intersection.)

(2) When the various images of a control point ($\vec{p}^{ij}$) are calculated in DINFLU, special care is required if the control point lies <u>in</u> either the first or the second plane of symmetry (ISYMCP $\neq$ 0). In accordance with the rules worked out in Appendix K of the theory document (cf. pg. K.5-5, algorithm A_1), one does the following

$\vec{p}$ lies <u>in</u> 1st P-O-S: set $\vec{p}^{-j} = \vec{p}^{+j}$

$\vec{p}$ lies <u>in</u> 2nd P-O-S: set $\vec{p}^{i-} = \vec{p}^{i+}$

Thus, if a control point is recognized as lying in a plane of symmetry, the reflection process is suppressed when we compute the control point's reflection with respect to the P-O-S in which it lies.

(3) When in subroutines QNFCAL and PIFCAL, the program determines whether or not the control point image under consideration lies directly on a particular panel or subpanel, care is taken that whenever the c.p. lies in the first (second) plane of symmetry, the same determination is made for $\vec{p}^{-j}$ (resp. $\vec{p}^{i-}$) as for $\vec{p}^{+j}$ (resp. $\vec{p}^{i-}$). As in (2) above, this is done in accordance with the rules developed in the theory document, Appendix K.5, algorithm A_1.

(4) Whenever an AIC row is constructed for a control point lying in a plane of symmetry, all integral IC's (ΦIC, VIC or WIC) are effectively multiplied by (1/2) before their contribution is added in to the AIC row. (See GENAIC for such closure AIC rows and GENBC for ordinary AIC rows.) This factor is included in accordance with the rules worked out in Appendices K.3 and K.6 of the theory document.

Aside from the complications required to implement these four points, the handling of cases with symmetry is pretty much the same as the handling of cases without symmetry.

5.2.1.2 The Principal Dataset MAG-PANEL-DATA

Because the streamline and off-body point processor FDP uses an influence coefficient subassembly that is a slight modification of the PIVC subassembly in MAG, most of the minimal panel defining quantities (cf. the PANDTA random file) are saved on the MAK database on the MAG-PANEL-DATA dataset. These data are subsequently transcribed to the MDG database for later use by FDP. The only items of the PANDTA minimal panel defining quantities that are not saved are the panel group singularity parameter index vectors IISF, IIDF.

5.2.1.3 The Auxiliary Datasets DATA-BASE-HEADER and SYMMETRY

The dataset DATA-BASE-HEADER contains the usual information identifying the run and indicating the final condition of the database, "COMPLETE" or not.

The dataset SYMMETRY contains global information concerning the symmetry conditions that were treated and various counts of control points, AIC rows and singularity parameters.

5.2.1.4 The Auxiliary Datasets COLMAP, COLMAP-INVERSE and COLMAP-BULK

Because it is necessary for MAG to suppress null singularity parameters and otherwise reorder the singularities in the following order (the range of the MAG indices is given in parentheses),

Known, nonupdatable	(1-10000)
Known, updatable	(10001-20000)
Unknown, nonupdatable	(20001-30000)
Unknown, updatable	(30001-40000)

various column map datasets are constructed to provide downstream modules with the information correlating the MAG and DQG singularity indexing schemes. The functions of these maps are illustrated below:

$$\text{MAG singularity index} \xrightarrow{\text{COLMAP}} \text{DQG singularity index}$$

$$\text{DQG singularity index} \xrightarrow[\text{INVERSE}]{\text{COLMAP}} \text{MAG singularity index}$$

$$\text{DQG singularity index} \xrightarrow[\text{BULK}]{\text{COLMAP}} \text{MAG singularity index}$$

The dataset COLMAP (respectively COLMAP-INVERSE) contains a dataset entry for each MAG singularity parameter (respectively non-null DQG singularity parameter). In addition, these datasets give singularity type (σ or μ), location (network, panel and point on panel), updatability type and whether or

not it is "known". The last dataset, COLMAP-BULK, provides a single array JSPDQG(1:NSPDQG) giving, for each DQG singularity index, the corresponding MAG singularity index. If the DQG singularity numbered IDQG is null, then JSPDQG(IDQG) = 0.

5.2.1.5 The Auxiliary Datasets ROWMAP, ROWMAP-INVERSE and ROWMAP-BULK

These datasets provide the relationships between the DQG control point indexing scheme and the MAG control point indexing scheme. The basic idea of the MAG control point indexing scheme is to put nonupdatable control points first (indexed 1-10000) followed by updatable control points (indexed 10001-20000). Control points that have neither an AIC row nor an IC matrix associated with them are excluded from the list of MAG control points. Within each major group of control points (updatable/nonupdatable), the control points are ordered by network with extra control points included at the end of each network's set. If a network has closure boundary conditions associated with it, special care is taken (in CONBLK) with the ordering of that network's control points so as to minimize the I-O activity in GENAIC associated with managing the AIC/closure buffer.

The various functions of the control point maps are illustrated below:

$$\text{MAG control point index} \xrightarrow{\text{ROWMAP}} \text{DQG control point index}$$

$$\text{DQG control point index} \xrightarrow[\text{INVERSE}]{\text{ROWMAP}} \text{MAG control point index}$$

$$\text{DQG control point index} \xrightarrow[\text{BULK}]{\text{ROWMAP}} \text{MAG control point index}$$

The dataset ROWMAP (resp. ROWMAP-INVERSE) contains a dataset entry for each MAG control point index (resp., for each DQG control point that is also a MAG control point: see remarks above). In addition these datasets give the control point's network location and position in space, information describing the effect of a control point's IC's on any closure boundary condition, the control point's "control point block index" and "row partition index" (used in organizing the computation of influence coefficients), as well as a variety of other information used in computing and saving the AIC's associated with a control point.

The last of these control point map datasets, ROWMAP-BULK contains a single array JCPMAP(1:NCPDQG) giving, for each DQG control point index, the corresponding MAG control point index. If no such MAG control point index exists, then JCPMAP(ICPDQG) = 0.

5.2.1.6 The Matching Condition Datasets

If a control point has any matching AIC conditions associated with it, this fact is recorded by MAG in both the ROWMAP and ROWMAP-INVERSE datasets. These conditions have the general form (compare with equations (5.2.9), (5.2.10), (5.2.11)):

$$\sum_{\substack{\text{edges } E_k \\ \text{of abutment } A}} s_k \, \theta(\vec{p}_k) = 0 \qquad (5.2.14)$$

where the generic function $\theta(\vec{p})$ may involve $\sigma(\vec{p})$, $\mu(\vec{p})$ or $(\vec{t}.\Delta\vec{v})(\vec{p}_k)$. The evaluation of $\theta(\vec{p}_k)$ can be expressed in terms of the global singularity parameters vector $\vec{\lambda}$ by the equation

$$\theta(\vec{p}_k) = \llcorner \theta IC_k \lrcorner \, \vec{\lambda}$$

For example if we are dealing with doublet matching, the entries of θIC are given by

$$\llcorner \theta IC_k \lrcorner \, J = m_J(\vec{p}_k)$$

Because the source and doublet distribution on a panel depends upon at most 35 global λ_J's, the row vectors θIC_k are quite simple and can be efficiently stored in a packed format giving the nonzeroes of θIC_k plus the corresponding indexing information. In subroutine SDMTCH, each matching condition associated with a control point is analyzed and the nonzero entries of each θIC_k are evaluated along with the associated MAG singularity parameter indices. This information is then saved on the appropriate "-MATCHING" dataset along with the numbers s_k and some indexing information. These data are then accessed by subroutine MATCH to construct full AIC rows during AIC generation.

5.2.1.7 The PANEL-GROUP Dataset

The PANEL-GROUP dataset is used mainly for local purposes in MAG to help organize the computation of influence coefficients. It is included on the MAK database principally as a debugging tool to help the maintenance programmer, should any problems occur during processing.

In MAG, all of the panels of the configuration are aggregated into panel groups, updatable and nonupdatable panels being placed in separate panel groups. A panel group is any collection of panels such that the outer splines for all of these panels depend upon $\leq$ MXING(=160) global singularity parameters λ_J. Some care is exercised to minimize the number of panel groups generated (cf. subroutine PANIJ, called by PANGRP) so that I-O costs associated with activity on the ICTP database can be minimized.

The actual information contained on the PANEL-GROUP database includes the following

- the number of global singularity parameters associated with the group ($\leq$ MXING = 160)
- a list of the MAG singularity parameter indices of these global singularity parameters

- o the number of panels in the panel group and a list of these panels, including the network identifier (i.e. the key to many DQG datasets: NETID = NETORD(K) where K is the network index) and the row and column indices of each panel

5.2.2 MAG Input/Output Data

The overall organization of data input to MAG and output from MAG is illustrated by Fig. 5.1. This figure also shows the internal communication of temporary databases PANDTA, FPDQ and ICTP used internally by MAG*.

Most of the input to MAG comes from DQG with DIP providing only the print flags and MEC providing the IC update flag and various information relating to system communication. The information provided by DQG falls into roughly three classes: global data, boundary condition and control point data, and panel data. The actual map names and DQG datasets used are listed below.

Class	Dataset	Map Name	Notes
Global	GLOBAL	GLOBAL	
	NETWK-SPEC	NETWK	
	SINGULARITY-SPEC	SNGSPC	
B.C. and C.P. data	BNDRY-CONDN-SPEC	BNDRY	Principally b.c. coefficients
	BNDRY-CONDN-SPEC	CNTRQ	Control point description
	CLOSURE	CLOSE	Closure coefficients and c.p. lists
	EXTRA-HYPO-LOC	EXHYLO	Main source of matching information
	SPECIAL-POINTS	SPCPT	Used to help find extra control points
Panel data	MAG-PANEL-SPEC	PANSPEC	DQG writes a panel dataset especially for MAG

As noted in section 5.2.1, the principal output from MAG consists of the AIC-MATRIX, the IC-MATRICES and the MAG-PANEL-DATA datasets. That part of AIC-MATRIX corresponding to unknown singularities (AIC_U) is read by RMS and factored. The portion of AIC-MATRIX corresponding to known singularities (AIC_K) is used by RHS to form a right hand side (cf. equation (5.2.3)). RHS then uses the factored AIC_U matrix as provided by RMS to solve for $\vec{\lambda}_U$.

---

*These temporary "databases" (they are actually Fortran files) contain the following data

PANDTA - A random file for storage of minimal panel defining quantities

FPDQ - Sequential files for storage of minimal panel defining quantities for nonupdatable (FPDQNU) and updatable (FPDQUP) panels

ICTP - A set of 12 sequential files used for temporary storage of influence coefficients, $[IC_A]$. File names are of the form ICTP01, ICTP02, . . . ICTP12.

The IC-MATRICES dataset is used by MDG to evaluate ϕ, $\vec{v}$ and $\vec{w}.\hat{n}$ at various points on the network surfaces for subsequent use by the post-processors, PDP, CDP and PPP. The MAG-PANEL-DATA dataset is transcribed by MDG to the MDG database for subsequent use by FDP, the streamline and off-body point processor.

In addition to these principal output datasets, the COLMAP and COLMAP-INVERSE datasets are used by RMS, RHS and MDG to establish the connection between MAG and DQG singularity parameter indices. Similarly the ROWMAP and ROWMAP-INVERSE datasets provide the relationship between MAG and DQG control point indices. Finally the SYMMETRY dataset provides global information relating to the execution of MAG that includes the following:

- Indicators describing which of the various symmetry conditions of ϕ (i.e. $\hat{\phi}^{SS}$, $\hat{\phi}^{AS}$, $\hat{\phi}^{AA}$, $\hat{\phi}^{SA}$) are nonzero
- Flow symmetry flags
- Counts of MAG control points, updatable and nonupdatable
- Counts of MAG singularity parameters, known and unknown, updatable and nonupdatable
- Counts of AIC rows, updatable and nonupdatable

5.2.3 Data Base Interfaces

As noted in Section (5.2.2) above, MAG's communication with external components of the PAN AIR system by way of the various databases is summarized in Fig. 5.1. Note that during an IC update run, MAG updates an existing MAK database, a process that requires some reading and rewriting of the MAK database.

MAG's communication with the three internal temporary databases, summarized in Fig. 5.1, is discussed in some detail in Section (5.3.3) which gives additional information concerning the internal databases.

5.3 MODULE DESCRIPTIONS

The main overlays and their structure are described in this section along with a description of MAG's internal databases.

5.3.1 Overall Structure

Fig. 5.2 illustrates the overall structure of the MAG module. The execution of the program proceeds roughly from top to bottom and from left to right, following the usual order for the traversal of a tree. The first overlay of the program (MAG10) performs problem setup while the second overlay (MAG20) performs actual IC and AIC generation. In the second overlay, the three components CBSET, ICTEMP and GENAIC are executed repeatedly in that order, once for each control point block. In Fig. 5.3, various internal library routines are classified and listed. Assuming that the appropriate common block environment has been established, these library routines can be called from any place in MAG.

5.3.2 Overlay Descriptions

The main overlay of MAG performs some initialization of some common blocks (via LOCKDATA), opens the random file PANDTA for minimal panel defining quantities, invokes MAG10 to perform some setup functions, invokes MAG20 to perform the actual IC and AIC generation and terminates execution with a call to MAGFIN.

5.3.2.1 MAG10, Overlay (1.0)

MAG10, the setup overlay of MAG, invokes the following routines in the order given to perform their required functions.

OPENDB: OPENDB opens the MEC, DIP, DQG and MAK databases, establishing all of the SDMS maps used throughout the remainder of the program. OPENDB then reads DQG's GLOBAL and NETWK-SPEC datasets, establishing global information concerning the whole flow configuration. (OPENDB also opens and maps the MAGX and MAGY temporary databases. Although these are not currently implemented, MAGX would be a replacement for the ICTPxx set of files for temporary storage of IC's and MAGY would be a replacement for the collection of files PANDTA, FPDQNU, FPDQUP used for storage of minimal panel data.)

BLOCK: BLOCK performs some checking of input data, establishes global constants (e.g. π) and then defines some global parameters and arrays relating to Mach number, compressibility axis orientation and configuration symmetry conditions.

COLMAP: COLMAP generates the MAG indexing system of all of the non-null DQG singularity parameters. If the user has requested it via input print options, COLMAP generates schematic maps of MAG and DQG singularity parameter indices.

PANGRP: PANGRP defines the panel groups according to the following considerations:

(i) Updatable and nonupdatable panels go into different groups.

(ii) No panel group may have more than MXING singularity parameters associated with it (cf. the parameter MXING = 160).

As each panel is included in some panel group, its panel data is read from the MAG-PANEL-SPEC.DQG dataset, cleaned up by MGPAND, transformed into a minimal packet of data by PAKPQF, and saved on various files. During this process, DQG singularity parameter indices are transformed into MAG singularity parameter indices, and these in turn are turned into panel group singularity parameter indices, numbers between 1 and 160 (MXING). The map INDGRP from panel group s.p. indices to MAG s.p. indices is also generated and saved on the PANEL-GROUP dataset, along with a list of all the panels in each panel group. Finally it should be noted that PANGRP generates the MAG-PANEL-DATA dataset on the MAK database.

CONBLK: CONBLK defines the control point block data structure that describes the organization for the processing of control points. The rules for construction of control point blocks are as follows:

(i) Let NCNSYM denote the number of symmetry conditions under consideration in the current PAN-AIR run.

(ii) Let MXRCPB denote the maximum number of rows per control point block. This number, nominally set equal to 100, is related to the amount of scratch memory available to subroutine ICTEMP in the second overlay. (A buffer of size MXRCPB*MXING = 100*160 = 16,000 is used there.)

(iii) Let MXRP denote the maximum number of IC rows that may be associated with a row partition. MXRP is defined by subroutine BLOCK and is related to the amount of scratch memory available in subroutine GENAIC.

(iv) Let MXCPBK denote the maximum number of control points per control point block. This is set to 150 by LOCKDATA, and must be consistent with the memory allocation for the array WCB in /CPBLK/. The value of 150, somewhat larger than MXRCPB = 100, is used because control points that require matching AIC rows to be computed may actually have no IC rows associated with them, (e.g., matching control points).

(v) Given these definitions for the important parameters, CONBLK calls PROCP to include a control point into a control point block while observing the following rules:

(a) Updatable and nonupdatable control points are not mixed in the same block.

(b) The total number of IC rows per control point block = NCNSYM * (Sum of the IC rows for each control point) must be $\leq$ MXRCPB.

(c) No more then MXCPBK control points in a control point block.

A row partition is a subset of a control point block that contains the IC's for a set of control points but only one symmetry condition. Additional conditions associated with control point row partitions include:

(d) The number of IC rows associated with a row partition must be less than or equal to MXRP.

(e) NCNSYM*(The number of row partitions in a control point block) must be less than or equal to MXICTP (= 12), the number of sequential files associated with the ICTPxx database.

In the process of defining the control point blocks and row partitions, PROCP also defines the ROWMAP and ROWMAP-INVERSE data structures and saves crucial information needed for the complete processing of closure control points by CLSROW. In this way, when all of the control points of a network have been processed by PROCP, subroutine CLSROW updates the ROWMAP and ROWMAP-INVERSE datasets to include closure information for any control points whose IC's affect a closure condition.

5.3.2.2 MAG20, Overlay (2,0)

MAG20, the IC and AIC generation overlay of MAG, performs dynamic allocation of memory for the second overlay and, for each control point block, invokes the following routines in the order given to perform their required functions.

CBSET: For each control point in the control point block, CBSET generates an 11 word packet of data containing the information about the control point needed by ICTEMP.

ICTEMP: ICTEMP computes the total influence of the configuration on the current control point block and writes this data to the ICTP database. The organization of ICTEMP is as follows:

```
For each panel group in the configuration [updatable groups only on an
update run if the control point block is nonupdatable]

     For each panel in the panel group

          Read the panel data from a sequential file (FPDQNU or FPDQUP),
          unpack it and extend it

          For each control point in the current control point block

               Unpack the control point's data packet and determine if any
               IC's are needed

               If IC's are needed, invoke PIVC to compute the panel on
               control point influences for all symmetry conditions and
               include them in the RIC influence coefficient buffer for
               the panel group on control point block influence

          end, loop on control points

     end, loop on panels
```

Using WRICT, write the panel group on control point block influences to the ICTP temporary database, the data for each control point row partition going to a separate file.

end loop on panel groups

GENAIC: For each control point block, GENAIC generates the appropriate contributions to the AIC-MATRIX and IC-MATRICES datasets. The organization of GENAIC is as follows:

For each symmetry condition of interest

For each row partition

Compute the number of rows in the row partition (NICPRT)

Read in from the ICTP database and aggregate the influence of all panel groups on the current row partition/symmetry condition combination.

For each control point in the row partition

Get the panel data for the panel on which the control point lies

Using the aggregate influence coefficients, $[IC_A]$, generate any nonclosure AIC rows associated with the control point and write them to the AIC-MATRIX dataset

Include the effect of the current control point's IC's on any closure AIC row, taking care that the closure AIC buffer is properly managed.

end loop on control points

end loop on row partitions

end loop on symmetry conditions

If the closure AIC buffer is nonempty, write it out to the AIC-MATRIX dataset.

5.3.3 MAG Databases

The principal output of MAG, the MAK database, has already been described in some detail in section (5.2.1). The SDMS master definition of this database is included on the PAN AIR delivery tape. A short program (MDPRNT) that generates an indexed listing of a master definition is listed in Appendix 5-A.

In the process of constructing MAG, it has been found convenient to define and implement three temporary "databases", each of which is nothing more than a specific set of files designated for a specific purpose. The main characteristics of these databases are outlined below.

Database	File Names	File Organization	Data
PANDTA	6LPANDTA	random (READMS/WRITMS)	Minimal panel data
FPDQ	6LFPDQNU, 6LFPDQUP	seq. (BUFFERIN/OUT)	Minimal panel data
ICTP	6LICTP01 6LICTP02 . . . . . . 6LICTP12	seq. (BUFFERIN/OUT)	Panel group on control point block influence coefficients

In the subsections that follow we will discuss separately each of these "databases".

Before going into this discussion, one final remark is appropriate concerning temporary databases in MAG. Since it is possible that at some future time the computing centers will reduce the high cost penalty associated with the use of random I-O as compared with sequential I-O, master definitions have been created for two temporary SDMS databases, MAGX and MAGY that could be used to replace the temporary databases discussed here. The master definitions for these databases may be found on the PAN AIR delivery tape

MAGX (IC-TEMP)	would replace ICTP
MAGY (PANEL-DATA)	would replace both PANDTA and FPDQ

5.3.3.1 PANDTA Database: Random Access to Minimal Panel Data

The random file PANDTA, opened in program MAG, is defined in subroutine PANGRP and used in those parts of the code where it is necessary to have random access to the panel data. These places include SDMTCH where the various matching conditions are analyzed and GENAIC where boundary conditions are imposed.

The file PANDTA is a standard random file accessed using the READMS/WRITMS package (available on both CDC and CRAY). A specific record of PANDTA, containing the minimal panel defining quantities for a specific panel, is accessed using the global panel index IPNDEX associated with the panel. The computation of this index in two cases of interest is illustrated below.

Case 1: network index = K, panel row index = IPAN, panel column index = JPAN
IPNDEX = IPAN + (JPAN-1)*NROWNT(K) + NPNCUM(K)

Case 2: network id = KNET, panel row index = IPAN, panel column index = JPAN
set K = NETINV(KNET) and proceed as in Case 1
note: If K = NETINV(KNET), then KNET = NETORD(K)

The actual data stored on PANDTA for each panel includes all those panel defining quantities that would be expensive or impossible to recompute "on the fly". The resulting data packet of 256 words is about 8 times smaller than the corresponding panel defining quantities dataset record of version 1.0 of PANAIR. The decision to include a particular data item in the data packet was based on a "5 microsecond tradeoff". That is, after an analysis of a number

of computing center charging algorithms, and taking into consideration the program's size and file buffer sizes, it was determined that it is cost effective to regenerate any data items for which the cost is less than 5 μs per word on a CDC 7600 or Cyber 760 computer.

We conclude our discussion of PANDTA by listing all of the items contained in a data packet

Panel location	KNETNR, ICOLNR, IROWNR
Source outer splines	ASTS
Source MAG s.p. indices	IISMAG, INS
Source panel group s.p. indices	IISF
Doublet outer splines	ASTD
Doublet MAG s.p. indices	IIDMAG, IND
Doublet panel group s.p. indices	IIDF
Panel geometry	CP, EN, NCONVX, AREAQ
Skew transformation and parameters	AQ, C1, C2, C3, CTEST
Line vortex flags	LVTERM
Data for farfield test	RFMIN, QDLTF, PWF, PXF, DIAMF
Panel σ/μ type	ITS
Index of collapsed side	ICS
Index describing half panel cut	ISQN
(s,t) parameters for regeneration of half-panel μ splines	STRC
panel diameter	DIAMF, DIAM
$\mathrm{sgn}(\hat{n}_5 \cdot \hat{c}_0)$, used in supersonic influence coefficients	SGXF
subinclined/superinclined indices	IIN

5.3.3.2 FPDQ Database: Sequential Access to Minimal Panel Data

When it is reasonable to do so, as in the panel influence coefficient generation in subroutine ICTEMP, it is more efficient to access the panel data from a sequential file than from a random file. For this reason, the FPDQ database was created. This database consists of two sequential files, FPDQNU containing nonupdatable panel data and FPDQUP containing updatable panel data. The actual data associated with each panel is exactly the same as the data contained on PANDTA. The organization of the data is different, however, in that FPDQNU and FPDQUP are buffered sequential files, each buffer containing 256 word panel data packets for 8 panels. The panel data is stored on these files in precisely the order it is required by ICTEMP as ICTEMP computes the influence of all the panel groups on a particular control point block.

5.3.3.3 ICTP Database: Sequential File Storage of the Influence Coefficients for a Control Point Block

The ICTP database is a set of 12 sequential files used for the storage of the influence coefficients for a control point block. In this section we will discuss both the structure and the construction of this database.

The structure of the database: To illustrate the structure of the database, we will consider the construction of ICTP for a run in which there are two active symmetry conditions ($\hat{\phi}^S$ and $\hat{\phi}^A$, say) and for a control

point block having 5 row partitions for each symmetry condition. Thus, there are 10 (which is ≤ 12) total row partitions for the control point block. The number of IC rows for each of these row partitions is given by the vector:

$$[R_K] = [9, 12, 9, 11, 9, 9, 12, 9, 11, 9] \tag{5.3.1}$$

Note that the last five row partitions (corresponding to $\hat{\phi}^A$) have the same number of rows as the first five (corresponding to $\hat{\phi}^S$). Note also that the total number of rows in the control point block is given by:

$$9 + 12 + 9 + \ldots + 9 + 11 + 9 = 100 \tag{5.3.2}$$

satisfying the restriction that a control point block have (≤ 100) IC rows associated with it.

The construction of the ICTP database for a c.p. block is performed in overlay 2 as follows:

Rewind all units ICTP01, . . . ICTP12 (MAG20)

For each panel group in the configuration (ICTEMP)

Compute the influence of the panel group on the control point block (ICTEMP/PIVC)

Place the current panel group's influence on the ICTP database as follows (ICTEMP/WRICT):

For each row partition number K, with R_K rows do:

Write the R_K rows of the IC buffer associated with row partition K to unit ICTP_K, being careful to include global MAG singularity parameter indices for the current panel group

end, loop on row partitions

end loop on panel groups

In order to be even more explicit about the structure of the ICTP database, we consider a control point block having two IC row partitions as follows

$$[R_K] = [2, 2] \tag{5.3.3}$$

The panel group will be assumed to have global singularity parameters associated with it with indices given by

$$[\text{INDGRP}] = [20004, 20001, 20005, 2, 20002, 20006, 20003, 1] \tag{5.3.4}$$

To facilitate the disk ouput of data in WRICT, the panel group on control point influences are stored transposed in memory, and are given by the array

$$[RIC] = \begin{bmatrix} 3.12 & 3.56 & 5.98 & 6.91 \\ 2.14 & 7.23 & 7.95 & 8.73 \\ 2.52 & 6.13 & 7.21 & 4.31 \\ 1.31 & 4.35 & 6.31 & 6.14 \\ 7.23 & 6.39 & 7.12 & 5.14 \\ 6.48 & 9.25 & 8.64 & 6.15 \\ 7.23 & 4.27 & 6.31 & 5.21 \\ 6.31 & 3.26 & 5.76 & 6.36 \end{bmatrix}, \quad \begin{bmatrix} 20004 \\ 20001 \\ 20005 \\ 2 \\ 20002 \\ 20006 \\ 20003 \\ 1 \end{bmatrix} \qquad (5.3.5)$$

row partition No. 1 No. 2 Associated MAG s.p. indices

These influence coefficients for the panel group and the control point block are written to the ICTP databases by performing the following writes to the two sequential files, ICTP01 and ICTP02.

$$\begin{bmatrix} 20004 & 3.12 & 3.56 \\ 20001 & 2.14 & 7.23 \\ 20005 & 2.52 & 6.13 \\ 2 & 1.31 & 4.35 \\ 20002 & 7.23 & 6.39 \\ 20006 & 6.48 & 9.25 \\ 20003 & 7.23 & 4.27 \\ 1 & 6.31 & 3.26 \end{bmatrix} \longrightarrow \text{ICTP01} \qquad (5.3.6)$$

$$\begin{bmatrix} 20004 & 5.98 & 6.91 \\ 20001 & 7.95 & 8.73 \\ 20005 & 7.21 & 4.31 \\ 2 & 6.31 & 6.14 \\ 20002 & 7.12 & 5.14 \\ 20006 & 8.64 & 6.15 \\ 20003 & 6.31 & 5.21 \\ 1 & 5.76 & 6.36 \end{bmatrix} \longrightarrow \text{ICTP02} \qquad (5.3.7)$$

To continue this example and further illuminate the nature of ICTP, suppose further that the problem under consideration has exactly two panel groups associated with it. We will assume that the record described by (5.3.6) above is the first record written to ICTP01 and that the second is given by

$$\begin{bmatrix} 20008 & 2.42 & 8.71 \\ 20004 & 4.31 & 6.54 \\ 3 & 7.25 & 4.74 \\ 20007 & 6.41 & 5.92 \\ 20002 & 1.72 & 6.81 \\ 20006 & 6.37 & 5.41 \\ 2 & 9.21 & 3.69 \end{bmatrix} \longrightarrow \text{ICTP01} \qquad (5.3.8)$$

To get the full influence on row partition No. 1, one then aggregates the arrays appearing in (5.3.6) and (5.3.8) to obtain

$$
\begin{bmatrix}
1 & 6.31 & 3.26 \\
2 & 10.52 & 8.04 \\
3 & 7.25 & 4.74 \\
20001 & 2.14 & 7.23 \\
20002 & 8.95 & 13.20 \\
20003 & 7.23 & 4.27 \\
20004 & 7.43 & 10.10 \\
20005 & 2.52 & 6.13 \\
20006 & 12.85 & 14.66 \\
20007 & 6.41 & 5.92 \\
20008 & 2.42 & 8.71
\end{bmatrix}
= \text{influence of whole configuration on first control point row partition} \qquad (5.3.9)
$$

This aggregation task corresponds to what is done in subroutine GENAIC during the IC aggregation phase.

5.3.4 Data Flow

The internal data communication in MAG is summarized in Fig. 5.4. This diagram, which is a truncated version of the overall program structure diagram Fig. 5.2, describes all database activity by map name (for SDMS datasets) or file name (for temporary internal databases). The correlation between the map names noted in Fig. 5.4 and the actual datasets is given by Fig. 5.5. The actual nature of the data flow activity is indicated in Fig. 5.4 by an I (for input), O (for output), S (for setup: open or rewind), C (for close) or R (for release) preceeding the map name. For example, I:MECHED indicates that the map MECHED is used to input data from the MEC dataset DATA-BASE-HEADER.

5.4 LOWER LEVEL FUNCTIONS

5.4.1 Function Decomposition

In this section we present a functional decomposition summary that describes roughly the activities of the routines appearing in Fig. 5.2, MAG's overall program structure diagram. This functional decomposition summary will consist of two parts, one part for the overall program organization and a second part for the PIVC subassembly used to evaluate panel influence coefficients. In Appendix 5-A we provide a listing of a simple program FDPRNT that generates a more detailed printout of subroutine functional decompositions extracted from the code itself.

Functional Decomposition for MAG, upper level routines MAG [0,0]

A. Perform standard program initialization (including BLOCKDATA), initialize SDMS, and open the random file PANDTA. [ISDMS] [OPENMS]

B. Invoke overlay [1,0], program MAG10, to perform all preliminary analysis needed to achieve efficient computation of influence coefficients. [MAG10]

BA Open data bases, define all SDMS maps and initialize the following global data. [OPENDB]

- Run identifiers
- IC update flag
- MAG print flags
- network counts, network identifiers, network dimensions, plus other basic network data
- singularity parameter and control point counts
- basic symmetry information
- gap filling panel count
- basic compressibility axis data and Mach number
- counts for singularity parameters and control points

BB Double check consistency of compressibility axis and symmetry information, expand upon compressibility axis and symmetry data and initialize global constants. Compute the maximum row partition size based upon the available scratch memory in GENAIC. [BLOCK]

BC Define MAG singularity parameter indices for all non-null DQG singularity parameters, generating the datasets COLMAP, COLMAP-INVERSE and COLMAP-BULK. The DQG s.p.'s are assigned MAG s.p. indices in accordance with the numbering scheme: [COLMAP]

Partition No.	Type	MAG s.p. index range
1	known, nonupdatable	1 - 10000
2	known, updatable	10001 - 20000
3	unknown, nonupdatable	20001 - 30000
4	unknown, updatable	30001 - 40000

On an update run, care is taken to ensure consistency of MAG s.p. indices. Printed singularity parameter maps will be produced if they have been requested.

BD Define the panel groups to be used during the PIC and AIC generation. A panel group consists exclusively of either updatable or nonupdatable panels. Taken together, all of the outer splines for the panels in a group depend upon $\leq$ MXING global singularity parameters. The panel grouping is performed by processing each network of the configuration (including the gap-filling "network") in such a way as to minimize the number of groups (cf. PANIJ). For each network, do the following: [PANGRP]

BDA Proceeding in an appropriate order through the panels of a network, include each panel in a panel group of appropriate type, creating new panel groups as needed. [PANIJ]

BDB Update the list of singularity parameters for the current panel group, keeping track of each global s.p. index and the corresponding group s.p. index. Generate the group s.p. indices (IISF, IIDF) corresponding to the global s.p. indices (IISMAG, IIDMAG) associated with the panel's outer splines.

BDC Generate for each panel a full set of panel defining quantities, expanding upon the panel data transmitted from DQG. [MGPAND]

BDCA Generate the essential near field data that was not transmitted from DQG. This includes skewness parameters and the associated coordinate transformation along with the quasi near field panel cut strategy (ISQN) and the associated (s*, t*) values for computing half panel splines (cf. Appendix I.3, theory document). By extending all near field panel data via a call to PSDDQG, ensure that nothing unexpected happens in Overlay 2. [NEARDT]

BDCB Generate the essential far field data that was not transmitted from DQG. This includes especially the parameters (PXF, PWF, SGXF and QDLTF) used in the rapid far field test. By extending all far field data with calls to FFDQGX, RACOF and XCOF, ensure that nothing unexpected happens in Overlay 2. [FARDT]

BDCC Save panel data on MAG-PANEL-DATA for subsequent use by MDG and FDP.

BDCD Pack up a very minimal set of panel defining quantities (256 words per panel) saving them on the random file PANDTA and buffering them out to either FPDQNU (non-updatable panel) or FPDQUP (updatable panel). [PAKPQF] [PAKAST]

BE Define the control point blocks and row partitions to be used during PIC and AIC generation. As one proceeds, MAG c.p. indices are assigned to each control point that is recognized as a MAG c.p. During this processing the ROWMAP, ROWMAP-INVERSE and ROWMAP-BULK datasets are defined and/or updated. Processing is performed on a network by network basis. [CONBLK]

BEA For each control point in a network (processed in an order designed to minimize I-O activity on the closure AIC buffer), read the control point's defining quantities and determine if it should be a MAG control point. If so, do the following: [PROCP]

BEAA Analyze any matching conditions associated with the control point, handling each symmetry condition separately for corner and extra control points on a plane of symmetry. [SDMTCH]

BEAAA Evaluate the dependencies of σ, μ and $\vec{t}.\Delta\vec{v}$ for each c.p. in a matching condition, producing an entry on one of the datasets: SOURCE-MATCHING, DOUBLET-MATCHING or VORTICITY-MATCHING. [GRAD]

BEAB Save closure information if appropriate.

BEAC Perform data checking of c.p. data and then write out information to the ROWMAP and ROWMAP-INVERSE datasets.

BEB If appropriate, place closure condition coefficients (a_A, a_D, AIC row index and updatability flag, symmetry conditions indicator) in appropriate places on the ROWMAP and ROWMAP-INVERSE datasets. [CLSROW]

BEC For each network, generate maps of MAG c.p. indices and AIC row indices.

BED If requested, generate reports summarizing control point and boundary condition information. [DRWMAP]

C. Invoke overlay [2,0], program MAG20, to perform evaluation of influence coefficients. The processing is performed by control point blocks. After packing into WCB some minimal control point defining information for the current c.p. block, MAG20 generates the IC's and AIC's for the current block as follows. [MAG20]

CA For each panel group, the influence of all the panels upon the current c.p. block is calculated and written to the ICTPxx database, the information for each row partition/symmetry combination being written to a different file. This is accomplished as follows. [ICTEMP]

CAB for each panel in the current panel group, the minimal panel data is obtained from the appropriate sequential file (FPDQNU or FPDQUP), unpacked by UPKPQF and extended by FFDQGX. Then, for each control point in the current c.p. block, the influence of the current panel on each control point is calculated by PIVC and included in the panel group/c.p. block IC buffer. [PIVC]

CAC When the influences of all the panels in the current group have been calculated, the panel group/c.p. block IC buffer is written out to the ICTPxx database. Care is taken to include corresponding global singularity parameter information. Note that each file in the ICTPxx database contains all the panel group influences for precisely one row partition/symmetry condition combination. [WRICT]

CB Once the influence of the whole configuration upon the current c.p. block has been calculated and saved on ICTPxx, the program proceeds to generate the IC's and AIC's for all of the control points in the current block. This is done as follows. For each row partition/symmetry condition combination, all of the influence coefficients are read from the appropriate file on the ICTPxx database and then aggregated. Then, for each control point in the row partition, the following tasks are performed. [GENAIC]

CBA The control point's boundary condition information and other defining quantities are placed in the appropriate common regions. [CBMOVE]

CBB The dependencies of σ, μ and $\nabla\mu$ are evaluated for the control point's hypothetical location. [GRAD]

CBC The AIC rows (for the current c.p. and symmetry condition) corresponding to general [GENBC], singularity specification [GENBC] and matching [MATCH] boundary conditions are calculated and written to the AIC-MATRIX dataset. [GENBC], [MATCH]

CBD Place any user requested influence coefficients on the IC-MATRICES dataset.

CBE Include the effect of the current c.p.'s influence coefficients on any closure condition. Much care must be taken with the management of the closure AIC buffer both here and when the processing of the current c.p. block is completed.

5.4.2 Functional Decomposition for the PIVC Subassembly

The purpose of PIVC is to compute and add in to the panel group/c.p. block IC buffer the influence of a given panel upon all symmetry conditions of a particular control point.

A. Calculate all required images of the basic c.p., taking special care when the c.p. lies in a plane of symmetry, and determine the method of PIC calculation for each c.p. image. [DINFLU]

B. If necessary, extend the panel defining quantities to permit quasi-near field evaluation (i.e. type 5, using QNFCAL). [PSDDQ5]

C. Invoke IC to organize the calculation of the panel influences. IC achieves its purpose as follows. [IC]

CA For each image of the control point, select the appropriate method of calculation and use it. The choices are:

CAA Monopole far field evaluation. (type 1) [FFPIC]

CAB Dipole far field evaluation. (type 2) [FFPIC]

CAC Quadruple far field evaluation. (type 3) [FFPIC]

CAD Quasi far field evaluation, involving one call to NFTPIC. (type 4) [QFFCAL]

CAE Quasi near field evaluation, involving two calls to NFTPIC. Note that this method is always considered if the influence test has indicated a near field (PIFCAL) should be used. Note further that this method may fail to give adequate answers so that PIFCAL is then required. (type 5) [QNFCAL]

CAF True 8 subpanel near field evaluation involving 8 calls to NFTPIC. (type 6) [PIFCAL]

Once the panel influences have been calculated for the current c.p. image, line vortex terms could be included. At present, line vortex terms are not implemented in PAN AIR. However allowance has been included for their future inclusion by providing the following control logic:

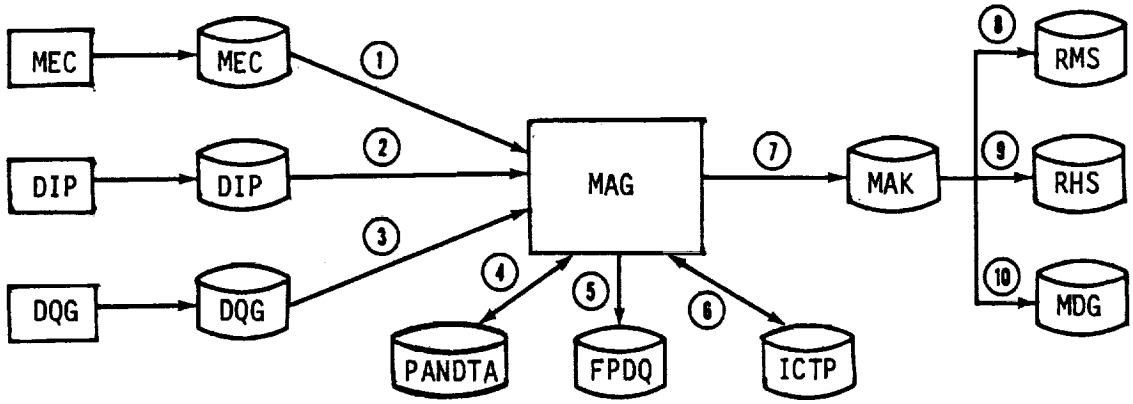
CAG Subroutine LINVOR would select a near or a far field evaluation procedure for line vortex terms. The near field procedure performed by NFLVT would be essentially the procedure described in Appendix J.10 of the theory document. The far field procedure performed by FFLVT would implement a combination of the ideas in Appendix J.9 and J.10 of the theory document.

CAH Once the full influence of a panel on a c.p. image has been calculated, the influence is added (with appropriate sign) into the various panel on c.p. symmetry condition accumulators.

CAI When accumulators containing the influence of the panel on the c.p. symmetry conditions are complete, the outer spline matrices are applied and the results are aggregated in to the appropriate places in the panel group/c.p. block influence buffer.

5.4.3 Subroutine Descriptions

In Appendix 5-A we provide a listing of a simple program SDPRNT that generates a printout of subroutine descriptions extracted from the code itself.



1. Run identification, IC update flag and data base directory information (this last item via CHPADB).

2. Print options.

3. Boundary condition and control point data, including closure information.
   Global information.
   Data for matching conditions.
   Panel defining data.
   Singularity parameter information.

4. Random access to a minimal set of panel defining quantities (created in PANGRP, accessed in SDMTCH and GENAIC).

5. Sequential access to a minimal set of panel defining quantities, file FPDQNU for nonupdatable panels, file FPDQUP for updatable panels (created by PANGRP, accessed by ICTEMP).

6. A set of twelve sequential files is used for efficient temporary storage of panel group on control point block influence information. File names are ICTP01, ICTP02, ... ICTP12.

7. Principal data written are the [AIC] matrix (AIC-MATRIX) and the control point ϕ, $\vec{v}$ and $\vec{w}.\hat{n}$ influence coefficients (IC-MATRICES). Also, the dataset MAG-PANEL-DATA.

8. That part of the AIC-MATRIX dataset corresponding to unknown singularities, $\vec{x}_U$. ($[AIC_U]$, cf. equation (5.2.2), maintenance doc.).

9. That part of the AIC-MATRIX dataset corresponding to known singularities, $\vec{x}_K$, ($[AIC_K]$, cf. equation (5.2.2), maintenance doc.).

10. The IC-MATRICES, $[IC_A]$, (cf. equation (5.2.4), maintenance doc.). Also, MAG-PANEL-DATA for subsequent usage by FDP.

Figure 5.1 - Data Base Relationships

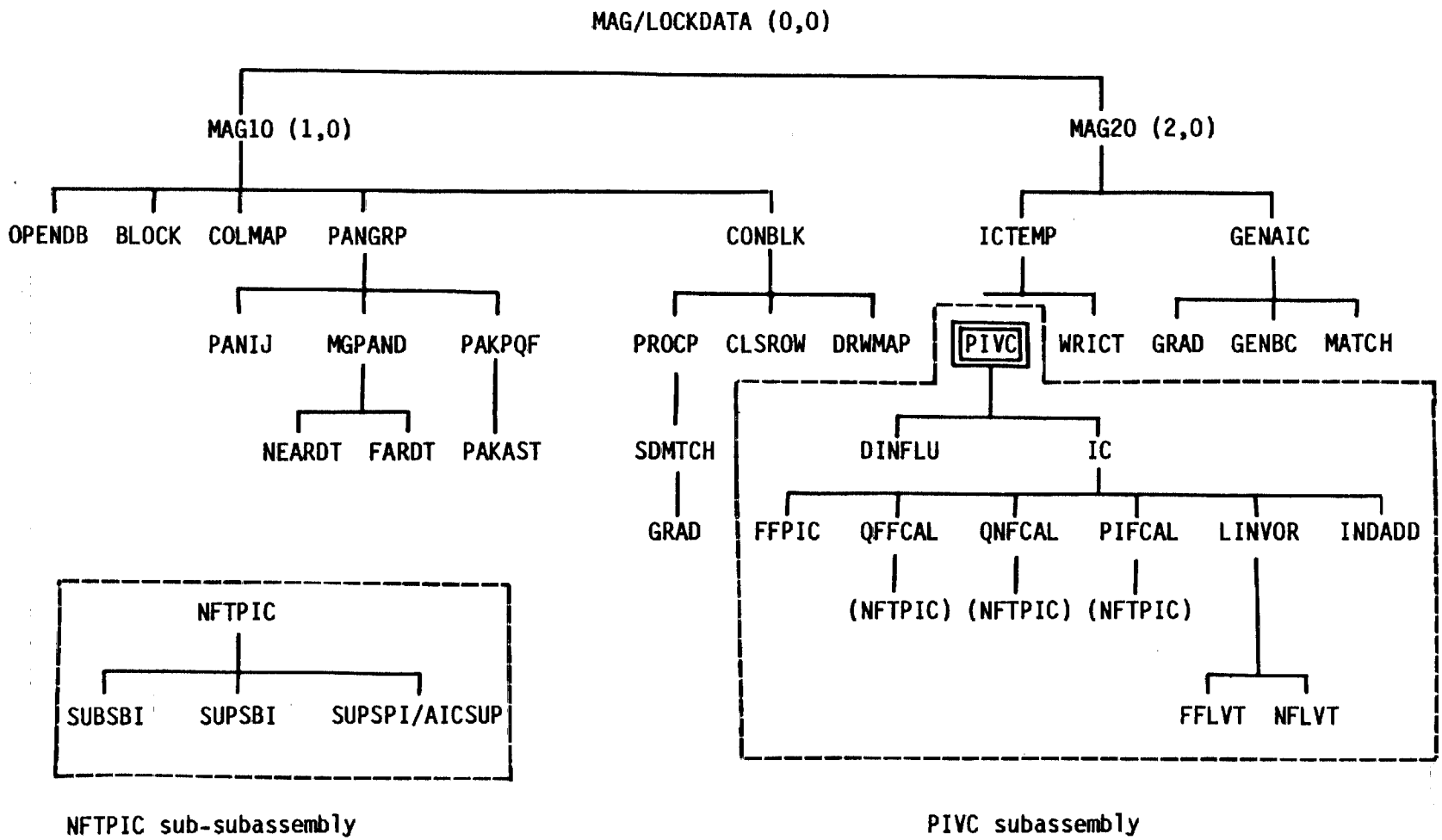


Figure 5.2 - Overall Program Structure Diagram,
Including PIVC Subassembly

MAG.CMPAXS

(Inner products and compressibility scaling)

ACIP $\vec{x}^T |C_o| \vec{y}$

ADIP $\vec{x}^T |B_o| \vec{y}$

CPIP $\vec{x}^T C_o \vec{y} = [\vec{x},\vec{y}]$

DCIP $\vec{x}^T B_o \vec{y} = \{\vec{x},\vec{y}\}$

CSCAL1 $\vec{y}_j \leftarrow C_o \vec{y}_j$

CSCAL2 $\vec{y}_j \leftarrow B_o \vec{y}_j$

MAG.PDRGEN

(Panel defining quantities regeneration)

PAKPQF Pack panel data
- PAKAST Pack splines

UPKPQF Upack panel data
- UPKAST Unpack splines
- XBPOSH (PALIB) project in $\bar{X}$
- RCSLOC (PALIB) $\hat{n} \rightarrow A$
- CCALN (PALIB) C_{ij}
- RACOF (PALIB) RA(3,5)
- XCOF (PALIB) QA(6,9)

PSDDQG

PSDDQ5 Quasi-near field data
- RCSLOC (PALIB) $\hat{n} \rightarrow A$
- BIQUAD (PALIB) $(\phi_\alpha, \frac{\partial}{\partial s}\phi_\alpha, \frac{\partial}{\partial t}\phi_\alpha)$
- TCOF (PALIB) coeff (B,B',C)

PSDDQ6 Near field data
- GTALAM (PALIB) κ parmameters
- PDQSUB (PALIB) RR, QQ, PP

MAG.JOBTRM

(Job termination and error handling)
- DBABT SDMS recovery
- MAGERR fatal error
- MAGFIN terminate job
- MAGMSG print msg, post fatal flag

MAG.IOPACK

(Sequential and AIC I-O)
- RDAIC Read and Write
- WRAIC from AIC ds
- REBUF Read and Write
- WRBUF from local seq. file

MAG.CPDATA

(Packing and unpacking of c.p. data for a c.p. block)
- CBMOVE extract c.p. data
- CBSET pack c.p. data
- CPINFO Calc. NE= $\sum$ IECP

MAG.PIVC

(PIVC Subassembly)

PIVC Panel on c.p. images influence
- DINFLU Influence Test
- IC Organize PIC calculation
  - FFPIC monopole, dipole, quadrupole
  - QFFCAL (NFTPIC) quasi far field
  - QNFCAL (NFTPIC) quasi near-field
  - PIFCAL (NFTPIC) near field
  - LINVOR linear vortex organizer
    - FFLVT far field stub
    - NFLVT near field stub
  - INDADD accumulate PIC's in group on block buffer

MAG.PIVC.NFTPIC

(NFTPIC sub-subassembly)

NFTPIC Organize flat panel near field computation
- SUBSBI $M_\infty < 1$
- SUPSBI $M_\infty > 1$, subinclined
- SUBSPI $M_\infty > 1$, superinclined
- AICSUP

PALIB.PDRGEN

(Panel Defining quantities regeneration, PALIB)

RCSLOC $\hat{n} \rightarrow A$

XBPOSH project in $\bar{X}$

CCALN $C_{ij} = \iint \xi^{i-1} \eta^{j-1} d\xi\, d\eta$

RACOF RA (3,5)

XCOF QA (6,9)

BIQUAD $(\phi_\alpha, \phi_{\alpha,s}, \phi_{\alpha,t})$

TCOF coeff $(B_i\ B_i'\ C)$

PDQSUB RR, QQ, PP

GTALAM κ parameters

PALIB.GENERAL

XFERA copy an array

MUL3X3 (3x3)(3xn) matrix mult.

ZERO zero an array

PIW4, UNPIW4 pack, unpack 4/word

OUTLIN, OUTVEC, OUTMAT, OUTMXV free field output pkg

SRCHOL search ordered list

SORTAK sort IA keeping KEY in synch

VMUL $\vec{y}\ \ \alpha\vec{x}$

CROSS $\vec{c} = \vec{a} \times \vec{b}$

VADD $\vec{z} = \vec{x} + \alpha\vec{y}$

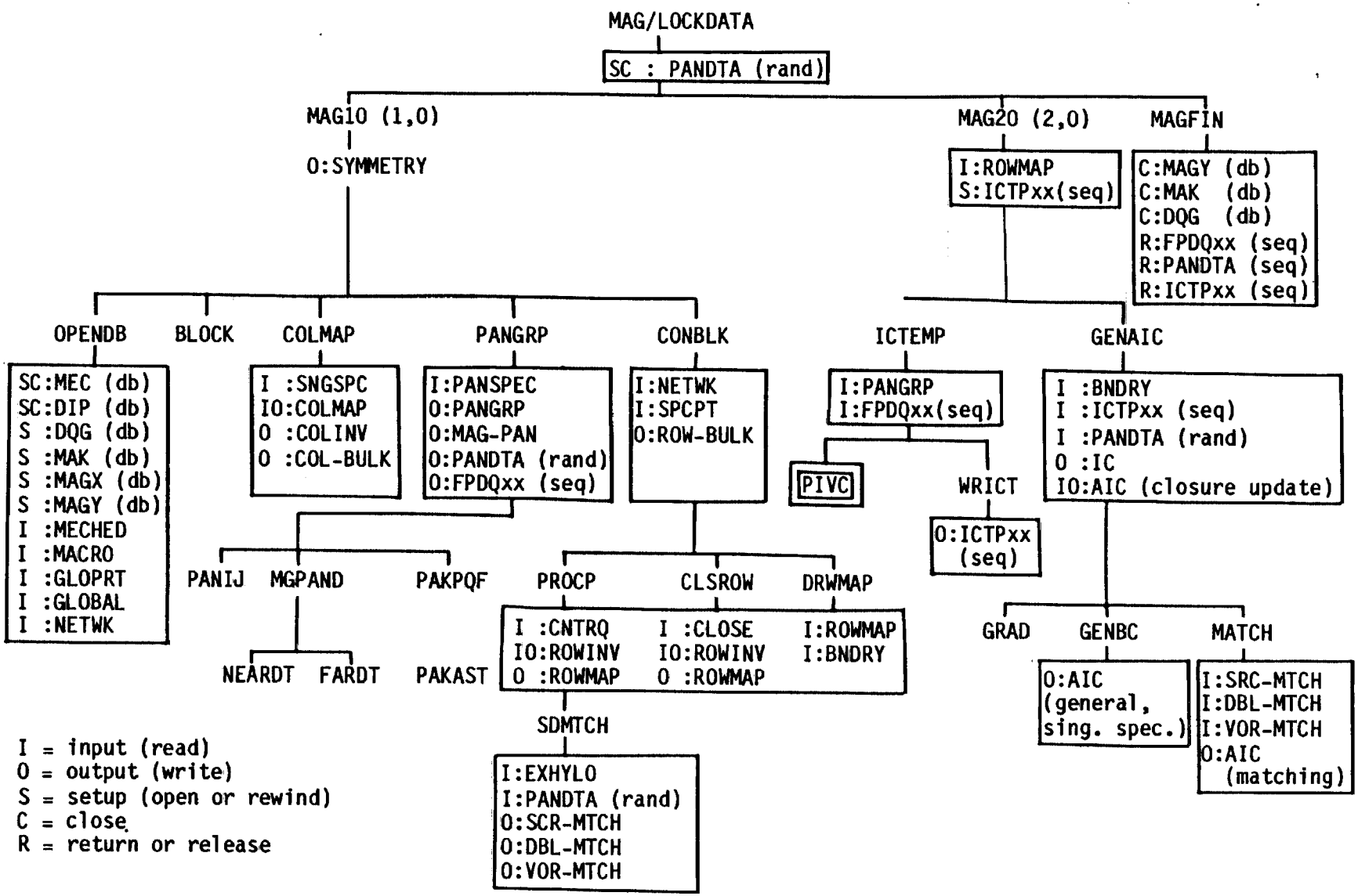
SHLSRT sort integer array

NRPTHP $\min\{\|\vec{Q}-\vec{Q}_o\|: \vec{Q} \in H\}$

RRAAX $\vec{y} = A\ \vec{x}+\vec{y}$

MXMACA $C \leftarrow C + A*B$

Figure 5.3 Sublibraries used by MAG



- o The panel data is placed on the ordinary FORTRAN files PANDTA (random), and FPDQNU, FPDQUP (sequential)
- o ICTPxx denotes a set of 12 sequential files used for temporary storage of IC's
- o The correlation between map name and dataset name is given in the following figure
- o File names are followed by (rand), (seq) indicating random or sequential organization
- o Database names are followed by (db), all other items are map names

Figure 5.4 - Data Flow Diagram for MAG Giving Data Activity by Map Name

Map or file name	Data-base	Dataset name	Remarks and notes
AIC	MAK	AIC-MATRIX	Written in GENBC, MATCH; written and updated in GENAIC (closure)
BNDRY	DQG	BNDRY-CONDN-SPEC	Printed in DRWMAP, used in GENAIC and below (see CNTRQ)
CLOSE	DQG	CLOSURE	Read in CLSROW to help include closure info on ROWMAP, ROWINV
CNTRQ	DQG	BNDRY-CONDN-SPEC	Read in PROCP to help create ROWMAP, ROWINV, ROW-BULK
COLINV	MAK	COLMAP-INVERSE	Written by COLMAP
COLMAP	MAK	COLMAP	Read by COLMAP for data checking during update, written by COLMAP
COL-BULK	MAK	COLMAP-BULK	Written by COLMAP, data used by PANGRP
DBL-MTCH	MAK	DOUBLET-MATCHING	Written by SDMTCH, used by MATCH
EXHYLO	DQG	EXTRA-HYPO-LOC	Read by SDMTCH to help create DBL-MTCH, SRC-MTCH, VOR-MTCH
FPDQNU (seq)	FPDQ	(minimal defining quantities)	Nonupdatable panels: Buffered and written by PANGRP, read by ICTEMP
FPDQUP (seq)	FPDQ	(minimal defining quantities)	Updatable panels: Buffered and written by PANGRP, read by ICTEMP
GLOBAL	DQG	GLOBAL	Global data, read in OPENDB, used everywhere
GLOPRT	DIP	GLOBAL-PRINTS	MAG print flags, read in OPENDB
IC	MAK	IC-MATRICES	Written by GENAIC
ICTPxx (seq)	ICTP	(IC temp. storage)	For each c.p. block, written by ICTEMP/WRICT, read by GENAIC
MACRO	MEC	MACRO-OPTIONS	IC update flag read in OPENDB
MAG-PAN	MAK	MAG-PANEL-DATA	Written by PANGRP for transcription by MDG, used in FDP
MAKHED	MAK	DATA-BASE-HEADER	Written by MAGFIN as part of program termination
MECHED	MEC	DATA-BASE-HEADER	Run identifiers read by OPENDB
NETWK	DQG	NETWK-SPEC	Read in OPENDB to create nw table, in CONBLK for extra nw info
PANDTA (rand)	PANDTA	(random access, min. panel data)	Written by PANGRP, read by SDMTCH and GENAIC
PANGRP	MAK	PANEL-GROUP	Written by PANGRP, read by ICTEMP
PANSPEC	DQG	MAG-PANEL-SPEC	Read by PANGRP
ROWINV	MAK	ROWMAP-INVERSE	Read (for data checking) by PROCP, written by PROCP, updated by CLSROW
ROWMAP	MAK	ROWMAP	Written by PROCP, updated by CLSROW, printed in DRWMAP, read by MAG20
ROW-BULK	MAK	ROWMAP-BULK	Written by CONBLK
SNGSPC	DQG	SINGULARITY SPEC	Read by COLMAP
SPCPT	DQG	SPECIAL-POINTS	Read by CONBLK to find extra control points
SRC-MTCH	MAK	SOURCE-MATCHING	Written by SDMTCH, used by MATCH
SYMTRY	MAK	SYMMETRY	Written by MAG10 at the end of problem setup
VOR-MTCH	MAK	VORTICITY-MATCHING	Written by SDMTCH, used by MATCH

Figure 5.5 - List of all Map and File Names

APPENDIX 5-A

PROGRAMMING AIDS: EXTRACTION PROGRAMS FOR MAG

As noted in the main text of this section, we list here three standalone programs that are used to generate useful printfiles for the maintenance programmer. The listings provided include both the code and the short job control record that has been used on the BCS Cyber system. The job control record will have to be modified in order for these job decks to be executed at an alternative site. The decks provided here perform the following functions.

FDPRNT: This program reads a full SOURCE file for the MAG program and extracts the functional decompositions for each subroutine. When all functional decompositions have been extracted, a sequential index and an alphabetical index are generated and written to the output file, FDLIST.

SDPRNT: This program reads a full SOURCE file for the MAG program and extracts the subroutine descriptions for each subroutine. This listing, which is significantly shorter than the listing generated by FDPRNT, has generated for it both a sequential and an alphabetical index.

MDPRNT: This program generates an indexed listing for an SDMS master definition file. This program may be used to generate a useful listing for any master definition file. It is particularly recommended for working with long master definitions such as those of the DQG and MAK databases.

All of the source code listed in this section has been included on the version 3.0 delivery tape.

```
FDPRNT,T50,P02,CM130000.
USER,*.  MIKE EPTON/773-9405/3N-P1
ATTACH,OLDPL=MAGPL21/UN=QPSMDSE.
UPDATE,P,D,C,F,S.
REWIND,SOURCE.
FTN,OPT=2,R=3.
LGO,SOURCE.
REPLACE,TAPE2=FDLIST.
EXIT,U.
REPLACE,OUTPUT=FDOUT.
DAYFILE,FDDAY.
REPLACE,FDDAY.
===  EOR  ===
*ID GETEM
*C MAG.FFDQGX
===  EOR  ===
      PROGRAM FDPRNT (TAPE1,TAPE2,INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
      INTEGER BUF(80)
     X        , DECK(80), DCKSUM(80,200), DCKPAG(200)
     X        , NAMEDK(200), KEY(200)
      LOGICAL WRITE, NEWPAG
C
      NCARD = 0
      NWRIT = 0
      NPAGE = 0
      NLINE = 0
      NDECK = 0
      JDECK = 0
  100 CONTINUE
      READ (1,120) BUF
  120 FORMAT (80A1)
      IF ( EOF(1) ) 1000,150
  150 CONTINUE
      NCARD = NCARD + 1
      WRITE  =  .FALSE.
C
      IDECK = 0
      IF ( BUF(1).EQ.1H*  .AND.
     X     BUF(2).EQ.1HD  .AND.
     X     BUF(3).EQ.1HE  .AND.
     X     BUF(4).EQ.1HC  .AND.
     X     BUF(5).EQ.1HK
     X                          ) IDECK = 1
C                                 DON'T PROCESS UNTIL FIRST  *DECK  IS
C                                 ENCOUNTERED
      IF ( IDECK.NE.0 ) JDECK = 1
      IF ( JDECK.EQ.0 ) GO TO 100
C
      IF ( IDECK.NE.0 ) WRITE = .TRUE.
C
```

```
      IF ( BUF(1).EQ.1HC  .AND.
     X     BUF(2).EQ.1HN
     X                            ) WRITE = .TRUE.
C
      IF ( BUF(1).EQ.1HC  .AND.
     X     BUF(2).EQ.1HF
     X                            ) WRITE = .TRUE.
C
      IF ( BUF(1).EQ.1HC  .AND.
     X     BUF(2).EQ.1HD
     X                            ) WRITE = .TRUE.
C
      IF ( BUF(1).EQ.1HC  .AND.
     X     BUF(2).EQ.1HP
     X                            ) WRITE = .TRUE.
C
      IF ( BUF(1).EQ.1HC  .AND.
     X     BUF(2).EQ.1HM
     X                            ) WRITE = .TRUE.
C
      IF ( BUF(1).EQ.1HC  .AND.
     X     BUF(2).EQ.1HR
     X                            ) WRITE = .TRUE.
C
      IF ( BUF(1).EQ.1HC  .AND.
     X     BUF(2).EQ.1HF  .AND.
     X     BUF(3).EQ.1HH
     X                            ) WRITE = .TRUE.
C
      IF ( BUF(1).EQ.1HC  .AND.
     X     BUF(2).EQ.1HF  .AND.
     X     BUF(3).EQ.1HT
     X                            ) WRITE = .TRUE.
C
      IF ( .NOT. WRITE ) GO TO 100
      NWRIT = NWRIT + 1
      NEWPAG= .FALSE.
      IF ( IDECK.NE.0 .OR. MOD(NLINE,58).EQ.0 ) NEWPAG = .TRUE.
      IF ( IDECK.EQ.0 ) GO TO 200
          NDECK   =  NDECK + 1
          CALL XFERA (BUF,DECK,80)
          CALL XFERA (BUF,DCKSUM(1,NDECK),80)
          DCKPAG(NDECK) = NPAGE + 1
  200 CONTINUE
C
      IF ( .NOT. NEWPAG ) GO TO 300
          NPAGE  =  NPAGE + 1
          NLINE  =  0
          WRITE (2,6001) DECK, NPAGE
 6001 FORMAT (1H1,80A1,10X,"PAGE",I5  ,/)
```

```
  300 CONTINUE
      NLINE = NLINE + 1
C
      WRITE (2,6000) BUF
 6000 FORMAT (1X,80A1)
      GO TO 100
C
C
C
 1000 CONTINUE
      WRITE (2,6003)
 6003 FORMAT (1H1,"        INDEX OF SUBROUTINE FUNCTIONAL DECOMPOSITIONS
     X (IN THE ORDER THEY APPEAR) "   ,  // )
 6004 FORMAT (1H1,"        ALPHABETICAL INDEX OF SUBROUTINE FUNCTIONAL DE
     XCOMPOSITIONS   "   ,  // )
      DO 1100 IDECK = 1,NDECK
          WRITE (2,6002) (DCKSUM(I,IDECK),I=7,20), DCKPAG(IDECK)
 6002 FORMAT (8X,14A1,"...   ...   ...   ...   ...   ...   ...   ..."
     X ,I4)
 1100 CONTINUE
C
C
C
      WRITE (2,6004)
      DO 1120 IDECK = 1,NDECK
          CALL PACKNM ( DCKSUM(7,IDECK), NAMEDK(IDECK) )
 1120 CONTINUE
      CALL ISHELL (NDECK, NAMEDK, KEY)
      DO 1150 J = 1,NDECK
          IDECK   =  KEY(J)
          WRITE (2,6002) (DCKSUM(I,IDECK),I=7,20), DCKPAG(IDECK)
 1150 CONTINUE
C
C
C
      WRITE (6,1200) NCARD,NWRIT
 1200 FORMAT ("1  TOTAL NUMBER OF CARDS:",I6,"   CARDS EXTRACTED:",I6)
      STOP
      END
      SUBROUTINE XFERA (A,B,N)
      DIMENSION A(1),B(1)
      DO 10 I = 1,N
   10 B(I) = A(I)
      RETURN
      END
      SUBROUTINE PACKNM ( ICH, JCH)
      DIMENSION ICH(10)
      DIMENSION IOUT(8)
C
      ENCODE (8,6000,IOUT) (ICH(I),I=1,8)
```

```
 6000 FORMAT (8A1)
      JCH     =  IOUT(1)
      RETURN
      END
      SUBROUTINE ISHELL (N,A,KEY)
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                                                             *
CC    *                    * * * P U R P O S E * * *                *
C     *                          - - - - - - -                      *
C     *                                                             *
C     *   ISHELL SORTS AN INTEGER ARRAY  A(N)  USING THE SHELL SORT *
C     *   ALGORITHM.(CF. JULY 1958, CACM, AN ARTICLE BY DONALD M.   *
C     *   SHELL).                                                   *
C     *                                                             *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C
C
      INTEGER  A(N), KEY(N), ASV
CC    *                                                             *
C     *   INITIALIZE THE ARRAY KEY CONTAINING ORIGINAL ADDRESSES    *
C     *                                                             *
      IF ( N.LE.0 ) RETURN
      DO 10 I = 1,N
   10 KEY(I)  =  I
      M       =  N
  100 CONTINUE
      M       =  M/2
      IF ( M.LE.0 ) RETURN
      JMAX    =  N - M
      DO 200 J = 1,JMAX
      IA      =  J
      IAP     =  IA + M
  150 IF ( A(IA) .LE. A(IAP) ) GO TO 200
          ASV     =  A(IA)
          A(IA)   =  A(IAP)
          A(IAP)  =  ASV
C
          ASV     =  KEY(IA)
          KEY(IA) =  KEY(IAP)
          KEY(IAP)=  ASV
C
          IAP     =  IA
          IA      =  IA - M
          IF ( IA.GT.0 ) GO TO 150
  200 CONTINUE
      GO TO 100
      END
```

```
SDPRNT,T50,P02,CM130000.
USER,*.  MIKE EPTON/773-9405/3N-P1
ATTACH,OLDPL=MAGPL21/UN=QPSMDSE.
UPDATE,P,D,C,F,S.
REWIND,SOURCE.
FTN,OPT=2,R=3.
LGO,SOURCE.
REPLACE,TAPE2=SDLIST.
EXIT,U.
REPLACE,OUTPUT=SDOUT.
DAYFILE,SDDAY.
REPLACE,SDDAY.
===  EOR  ===
*ID GETEM
*C MAG.FFDQGX
===  EOR  ===
      PROGRAM SDPRNT (TAPE1,TAPE2,INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
      INTEGER BUF(80)
     X        , DECK(80), DCKSUM(80,200), DCKPAG(200)
     X        , NAMEDK(200), KEY(200)
      LOGICAL WRITE, NEWPAG
C
      NCARD = 0
      NWRIT = 0
      NPAGE = 0
      NLINE = 0
      NDECK = 0
  100 CONTINUE
      READ (1,120) BUF
  120 FORMAT (80A1)
      IF ( EOF(1) ) 1000,150
  150 CONTINUE
      NCARD = NCARD + 1
      WRITE  =  .FALSE.
C
      IDECK = 0
      IF ( BUF(1).EQ.1H*  .AND.
     X     BUF(2).EQ.1HD  .AND.
     X     BUF(3).EQ.1HE  .AND.
     X     BUF(4).EQ.1HC  .AND.
     X     BUF(5).EQ.1HK
     X                          ) IDECK = 1
      IF ( IDECK.NE.0 ) WRITE = .TRUE.
C
      IF ( BUF(1).EQ.1HC  .AND.
     X     BUF(2).EQ.1HN
     X                          ) WRITE = .TRUE.
C
      IF ( BUF(1).EQ.1HC  .AND.
     X     BUF(2).EQ.1HP
```

```
     X.AND.BUF(3).NE.1HE
     X                                ) WRITE = .TRUE.
C
      IF ( .NOT. WRITE ) GO TO 100
      NWRIT = NWRIT + 1
      NEWPAG= .FALSE.
      IF ( MOD(NLINE,58).EQ.0 ) NEWPAG = .TRUE.
      IF ( MOD(NLINE,58).GT.48 .AND. IDECK.NE.0 ) NEWPAG = .TRUE.
      IF ( IDECK.EQ.0 ) GO TO 200
          NDECK   =  NDECK + 1
          CALL XFERA (BUF,DECK,80)
          CALL XFERA (BUF,DCKSUM(1,NDECK),80)
          DCKPAG(NDECK) = NPAGE + 1
  200 CONTINUE
C
      IF ( .NOT. NEWPAG ) GO TO 300
          NPAGE  =  NPAGE + 1
          NLINE  =  0
          WRITE (2,6001)  NPAGE
 6001 FORMAT (1H1,60X,10X,"PAGE",I5   ,/)
  300 CONTINUE
      NLINE = NLINE + 1
C
      IF ( IDECK.NE.0 ) WRITE (2,6006)
 6006 FORMAT ( "     " )
      IF ( IDECK.NE.0 ) GO TO 100
      WRITE (2,6000) BUF
 6000 FORMAT (1X,80A1)
      GO TO 100
C
C
C
 1000 CONTINUE
      WRITE (2,6003)
 6003 FORMAT (1H1,"        INDEX OF SUBROUTINE DESCRIPTIONS "
     X " (IN THE ORDER THEY APPEAR) "   ,   // )
 6004 FORMAT (1H1,"        ALPHABETICAL INDEX OF SUBROUTINE DESCRIPTIONS"
     X      ,  // )
      DO 1100 IDECK = 1,NDECK
          WRITE (2,6002) (DCKSUM(I,IDECK),I=7,20), DCKPAG(IDECK)
 6002 FORMAT (8X,14A1,"...   ...   ...   ...   ...   ...   ...   ..."
     X ,I4)
 1100 CONTINUE
C
C
C
      WRITE (2,6004)
      DO 1120 IDECK = 1,NDECK
          CALL PACKNM ( DCKSUM(7,IDECK), NAMEDK(IDECK) )
 1120 CONTINUE
```

```
      CALL ISHELL (NDECK, NAMEDK, KEY)
      DO 1150 J = 1,NDECK
          IDECK   =  KEY(J)
          WRITE (2,6002) (DCKSUM(I,IDECK),I=7,20), DCKPAG(IDECK)
 1150 CONTINUE
C
C
C
      WRITE (6,1200) NCARD,NWRIT
 1200 FORMAT ("1  TOTAL NUMBER OF CARDS:",I6,"   CARDS EXTRACTED:",I6)
      STOP
      END
      SUBROUTINE XFERA (A,B,N)
      DIMENSION A(1),B(1)
      DO 10 I = 1,N
   10 B(I) = A(I)
      RETURN
      END
      SUBROUTINE PACKNM ( ICH, JCH)
      DIMENSION ICH(10)
      DIMENSION IOUT(8)
C
      ENCODE (8,6000,IOUT) (ICH(I),I=1,8)
 6000 FORMAT (8A1)
      JCH     =  IOUT(1)
      RETURN
      END
      SUBROUTINE ISHELL (N,A,KEY)
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                                                               *
CC    *                  * * * P U R P O S E * * *                  *
C     *                        - - - - - - -                          *
C     *                                                               *
C     *   ISHELL SORTS AN INTEGER ARRAY  A(N)  USING THE SHELL SORT   *
C     *   ALGORITHM.(CF. JULY 1958, CACM, AN ARTICLE BY DONALD M.     *
C     *   SHELL).                                                     *
C     *                                                               *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C
C
      INTEGER  A(N), KEY(N), ASV
CC    *                                                               *
C     *   INITIALIZE THE ARRAY KEY CONTAINING ORIGINAL ADDRESSES      *
C     *                                                               *
      IF ( N.LE.0 ) RETURN
      DO 10 I = 1,N
   10 KEY(I)  =  I
      M       =  N
```

```
  100 CONTINUE
      M       =  M/2
      IF ( M.LE.0 ) RETURN
      JMAX    =  N - M
      DO 200 J = 1,JMAX
      IA      =  J
      IAP     =  IA + M
  150 IF ( A(IA) .LE. A(IAP) ) GO TO 200
          ASV     =  A(IA)
          A(IA)   =  A(IAP)
          A(IAP)  =  ASV
C
          ASV     =  KEY(IA)
          KEY(IA) =  KEY(IAP)
          KEY(IAP)=  ASV
C
          IAP     =  IA
          IA      =  IA - M
          IF ( IA.GT.0 ) GO TO 150
  200 CONTINUE
      GO TO 100
      END
```

```
MDPRNT,T10,P04,CM150000.
USER,*.  MIKE EPTON/773-9405/3N-P1
FTN,R=3,OPT=2.
GET,A=MDMAK.
LGO,A.
REPLACE,OUTPUT=MDOUT.
EXIT,U.
COST,LO=F.
DAYFILE,MDDAY.
REPLACE,MDDAY.
===  EOR  ===
      PROGRAM MDPRNT (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
      INTEGER BUF(90), CHAR, DOL, DUF(90), DUFBLK(90)
      INTEGER NPAGE(200), TITLE(8,200)
      INTEGER NAMEDS(200), KEY(200)
      DATA DOL/1H$/
      DATA NL/57/
      DO  7000 J = 1,90
 7000 DUF(J) = 1H
C
      LINE    =  0
      IPG     =  0
      NDS     =  0
C
    1 CONTINUE
      READ (5,2) BUF
    2 FORMAT (90A1)
      IF ( EOF(5) ) 950,3
    3 CONTINUE
C                                   LOOK FOR WORD "DATASET"
      DO 10 J = 1,84
      JSV     =  J
      IF ( BUF(J).EQ.DOL ) GO TO 30
C
      IF ( BUF(J  ) .EQ. 1HE
     N.AND.BUF(J+1) .EQ. 1HN
     D.AND.BUF(J+2) .EQ. 1HD  ) GO TO 30
C
      JSAVE   =  J
      IF ( BUF(J  ) .EQ. 1HD
     A.AND.BUF(J+1) .EQ. 1HA
     T.AND.BUF(J+2) .EQ. 1HT
     A.AND.BUF(J+3) .EQ. 1HA
     S.AND.BUF(J+4) .EQ. 1HS
     E.AND.BUF(J+5) .EQ. 1HE
     T.AND.BUF(J+6) .EQ. 1HT  ) GO TO 20
   10 CONTINUE
      GO TO 30
C                                   FOUND IT. SET LINE = 52, SET DUF
   20 CONTINUE
```

```
      LINE     =  NL
      DO 15 J = 1,90
      DUFBLK(J) = BUF(J)
      IF ( BUF(J).EQ. 1H  ) DUFBLK(J) = 1HA
   15 DUF(J) = BUF(J)
C
      INDNAM = JSAVE + 7
      JBEG   = INDNAM
      JFIN   = JBEG + 23
      DO 17 J = JBEG,JFIN
          IF ( DUF(J) .EQ. 1H  ) GO TO 17
               INDNAM  =  J
               GO TO 18
   17 CONTINUE
   18 CONTINUE
      INDLST  =  INDNAM + 23
      NDS     =  NDS + 1
      ENCODE (24,16,TITLE(1,NDS))  (DUF(J),J=INDNAM,INDLST)
   16 FORMAT (24A1)
      ENCODE ( 8,19,NAMEDS(NDS)) (DUFBLK(J),J=INDNAM,INDLST)
   19 FORMAT (8A1)
      NPAGE(NDS) = IPG + 1
C
   30 CONTINUE
      LINE     =  LINE + 1
      IF ( LINE.GT.NL ) LINE = 1
      IF ( LINE.EQ.1 ) IPG = IPG + 1
      IF ( LINE.EQ.1 ) WRITE (6,6001) (DUF(K),K=1,72), IPG
      WRITE (6,6002) BUF
      GO TO 1
C
  950 CONTINUE
      WRITE (6,6003)
      DO 960 K = 1,NDS
  960 WRITE (6,6004) (TITLE(I,K),I=1,3), NPAGE(K)
C
C
C
      CALL ISHELL (NDS, NAMEDS, KEY)
      WRITE (6,6006)
      DO 980 IK = 1,NDS
          K        =  KEY(IK)
          WRITE (6,6005) (TITLE(I,K),I=1,3), NPAGE(K)
  980 CONTINUE
      STOP
C
 6001 FORMAT (1H1,4X,72A1,23X,"PAGE",I5,/)
 6002 FORMAT (5X,90A1)
 6003 FORMAT (1H1,"          DATASETS IN THE ORDER THEY ARE DECLARED "
     X  , // )
```

```
 6004 FORMAT (5X,3A10,"  ...   ...   ...   ...   ...   ...",I5)
 6005 FORMAT (5X,3A10,"  ...   ...   ...   ...   ...   ...",I5)
 6006 FORMAT (1H1,"          DATASETS IN ALPHABETICAL ORDER        "
     X   , // )
      END
      SUBROUTINE ISHELL (N,A,KEY)
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                                                             *
CC    *                    * * * P U R P O S E * * *                *
C     *                          - - - - - - -                      *
C     *                                                             *
C     *   ISHELL SORTS AN INTEGER ARRAY  A(N)  USING THE SHELL SORT *
C     *   ALGORITHM.(CF. JULY 1958, CACM, AN ARTICLE BY DONALD M.   *
C     *   SHELL).                                                   *
C     *                                                             *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C
C
      INTEGER  A(N), KEY(N), ASV
CC    *                                                             *
C     *   INITIALIZE THE ARRAY KEY CONTAINING ORIGINAL ADDRESSES    *
C     *                                                             *
      IF ( N.LE.0 ) RETURN
      DO 10 I = 1,N
   10 KEY(I)   =  I
      M        =  N
  100 CONTINUE
      M        =  M/2
      IF ( M.LE.0 ) RETURN
      JMAX     =  N - M
      DO 200 J = 1,JMAX
      IA       =  J
      IAP      =  IA + M
  150 IF ( A(IA) .LE. A(IAP) ) GO TO 200
          ASV      =  A(IA)
          A(IA)    =  A(IAP)
          A(IAP)   =  ASV
C
          ASV      =  KEY(IA)
          KEY(IA)  =  KEY(IAP)
          KEY(IAP) =  ASV
C
          IAP      =  IA
          IA       =  IA - M
          IF ( IA.GT.0 ) GO TO 150
  200 CONTINUE
      GO TO 100
      END
```

APPENDIX 5-B

DATA BASE COMMUNICATIONS CHART

The Data Base Communications Chart for the SDMS databases is presented in three forms. Each form is alphabetized by columns, from left to right. The first form has a column order of Data Base, Dataset Name, Map Name, Common Block, and Program/Subroutine. The second form has a column order of Data Base, Map Name, Dataset Name, Common Block, and Program/Subroutine. The third form has a column order of Common Block, Data Base, Map Name, Dataset Name, and Program/Subroutine. Thus a person can get a cross reference on a data element by knowing either the Dataset name, Map Name or Common Block name. Note that the map "AIC" which is accessed directly only by the I-O interface routines RDAIC and WRAIC, is listed as being referenced by GENAIC, GENBC and MATCH, which are the routines that use these I-O interface routines.

FIRST FORM

DATA BASE	DATASET NAME	MAP NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
DIP	GLOBAL-PRINTS	GLOPRT	Dynamic	OPENDB
DQG	BNDRY-CONDN-SPEC	BNDRY	Dynamic [/BCDATA/] [/CNTRQ/]	DRWMAP, GENAIC
DQG	BNDRY-CONDN-SPEC	CNTRQ	Dynamic [/CNTRQ/]	PROCP
DQG	CLOSURE	CLOSE	Dynamic	CLSROW
DQG	EXTRA-HYPO-LOC	EXHYLO	Dynamic	SDMTCH
DQG	GLOBAL	GLOBAL	/MAGNUM/ /MAGGLO/ /SYMTRY/ Dynamic	OPENDB
DQG	MAG-PANEL-SPEC	PANSPEC	Dynamic [/DQGPAN/]	PANGRP
DQG	NETWK-SPEC	NETWK	Dynamic [/MAGNUM/]	OPENDB CONBLK
DQG	SINGULARITY-SPEC	SNGSPC	Dynamic	COLMAP
DQG	SPECIAL-POINTS	SPCPT	Dynamic	CONBLK
MAK	AIC-MATRIX	AIC	Dynamic	GENAIC GENBC MATCH
MAK	COLMAP	COLMAP	Dynamic	COLMAP
MAK	COLMAP-BULK	COL-BULK	Dynamic	COLMAP
MAK	COLMAP-INVERSE	COLINV	Dynamic	COLMAP
MAK	DATA-BASE-HEADER	MAKHED	Dynamic [/RUNIDS/]	MAGFIN
MAK	DOUBLET-MATCHING	DBL-MTCH	Dynamic	SDMTCH, MATCH
MAK	IC-MATRICES	IC	Dynamic	GENAIC

FIRST FORM (CONT.)

DATA BASE	DATASET NAME	MAP NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
MAK	MAG-PANEL-DATA	MAG-PAN	/PANDF/ /PANDQ/	PANGRP
MAK	PANEL-GROUP	PANGRP	Dynamic	PANGRP ICTEMP
MAK	ROWMAP	ROWMAP	Dynamic [/CNTRQ/]	PROCP CLSROW DRWMAP MAG20
MAK	ROWMAP-BULK	ROW-BULK	Dynamic	CONBLK
MAK	ROWMAP-INVERSE	ROWINV	Dynamic [/CNTRQ/]	PROCP CLSROW
MAK	SOURCE-MATCHING	SRC-MTCH	Dynamic	SDMTCH MATCH
MAK	SYMMETRY	SYMTRY	/SYMTRY/ /MAGNUM/	MAG10
MAK	VORTICITY-MATCHING	VOR-MTCH	Dynamic	SDMTCH MATCH
MEC	DATA-BASE-HEADER	MECHED	/RUNIDS/	OPENDB
MEC	MACRO-OPTIONS	MACRO	/MAGNUM/	OPENDB

SECOND FORM

DATA BASE	MAP NAME	DATASET NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
DIP	GLOPRT	GLOBAL-PRINTS	Dynamic	OPENDB
DQG	BNDRY	BNDRY-CONDN-SPEC	Dynamic [/BCDATA/] [/CNTRQ/]	DRWMAP, GENAIC
DQG	CLOSE	CLOSURE	Dynamic	CLSROW
DQG	CNTRQ	BNDRY-CONDN-SPEC	Dynamic [/CNTRQ/]	PROCP
DQG	EXHYLO	EXTRA-HYPO-LOC	Dynamic	SDMTCH
DQG	GLOBAL	GLOBAL	/MAGNUM/ /MAGGLO/ /SYMTRY/ Dynamic	OPENDB
DQG	PANSPEC	MAG-PANEL-SPEC	Dynamic [/DQGPAN/]	PANGRP
DQG	NETWK	NETWK-SPEC	Dynamic [/MAGNUM/]	OPENDB CONBLK
DQG	SNGSPC	SINGULARITY-SPEC	Dynamic	COLMAP
DQG	SPCPT	SPECIAL-POINTS	Dynamic	CONBLK
MAK	AIC	AIC-MATRIX	Dynamic	GENAIC GENBC MATCH
MAK	COL-BULK	COLMAP-BULK	Dynamic	COLMAP
MAK	COLINV	COLMAP-INVERSE	Dynamic	COLMAP
MAK	COLMAP	COLMAP	Dynamic	COLMAP
MAK	DBL-MTCH	DOUBLET-MATCHING	Dynamic	SDMTCH, MATCH
MAK	IC	IC-MATRICES	Dynamic	GENAIC
MAK	MAG-PAN	MAG-PANEL-DATA	/PANDF/ /PANDQ/	PANGRP

SECOND FORM (CONT.)

DATA BASE	MAP NAME	DATASET NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
MAK	MAKHED	DATA-BASE-HEADER	Dynamic [/RUNIDS/]	MAGFIN
MAK	PANGRP	PANEL-GROUP	Dynamic	PANGRP ICTEMP
MAK	ROW-BULK	ROWMAP-BULK	Dynamic	CONBLK
MAK	ROWINV	ROWMAP-INVERSE	Dynamic [/CNTRQ/]	PROCP CLSROW
MAK	ROWMAP	ROWMAP	Dynamic [/CNTRQ/]	PROCP CLSROW DRWMAP MAG20
MAK	SRC-MTCH	SOURCE-MATCHING	Dynamic	SDMTCH MATCH
MAK	SYMTRY	SYMMETRY	/SYMTRY/ /MAGNUM/	MAG10
MAK	VOR-MTCH	VORTICITY-MATCHING	Dynamic	SDMTCH MATCH
MEC	MACRO	MACRO-OPTIONS	/MAGNUM/	OPENDB
MEC	MECHED	DATA-BASE-HEADER	/RUNIDS/	OPENDB

THIRD FORM

COMMON BLOCK	DATA BASE	MAP NAME	DATASET NAME	PROGRAM/ SUBROUTINE
Dynamic [/BCDATA/] [/CNTRQ/]	DQG	BNDRY	BNDRY-CONDN-SPEC	DRWMAP, GENAIC
Dynamic [/CNTRQ/]	DQG	CNTRQ	BNDRY-CONDN-SPEC	PROCP
Dynamic [/CNTRQ/]	MAK	ROWINV	ROWMAP-INVERSE	PROCP CLSROW
Dynamic [/CNTRQ/]	MAK	ROWMAP	ROWMAP	PROCP CLSROW DRWMAP MAG20
Dynamic [/DQGPAN/]	DQG	PANSPEC	MAG-PANEL-SPEC	PANGRP
/MAGNUM/ /MAGGLO/ /SYMTRY/ Dynamic	DQG	GLOBAL	GLOBAL	OPENDB
Dynamic [/MAGNUM/]	DQG	NETWK	NETWK-SPEC	OPENDB CONBLK
/MAGNUM/ /SYMTRY/	MAK	SYMTRY	SYMMETRY	MAG10
/MAGNUM/	MEC	MACRO	MACRO-OPTIONS	OPENDB
/PANDF/ /PANDQ/	MAK	MAG-PAN	MAG-PANEL-DATA	PANGRP
Dynamic [/RUNIDS/]	MAK	MAKHED	DATA-BASE-HEADER	MAGFIN
/RUNIDS/	MEC	MECHED	DATA-BASE-HEADER	OPENDB
Dynamic	DIP	GLOPRT	GLOBAL-PRINTS	OPENDB
Dynamic	DQG	CLOSE	CLOSURE	CLSROW
Dynamic	DQG	EXHYLO	EXTRA-HYPO-LOC	SDMTCH
Dynamic	DQG	SNGSPC	SINGULARITY-SPEC	COLMAP
Dynamic	DQG	SPCPT	SPECIAL-POINTS	CONBLK

THIRD FORM (CONT.)

COMMON BLOCK	DATA BASE	MAP NAME	DATASET NAME	PROGRAM/ SUBROUTINE
Dynamic	MAK	AIC	AIC-MATRIX	GENAIC GENBC MATCH
Dynamic	MAK	COL-BULK	COLMAP-BULK	COLMAP
Dynamic	MAK	COLINV	COLMAP-INVERSE	COLMAP
Dynamic	MAK	COLMAP	COLMAP	COLMAP
Dynamic	MAK	DBL-MTCH	DOUBLET-MATCHING	SDMTCH, MATCH
Dynamic	MAK	IC	IC-MATRICES	GENAIC
Dynamic	MAK	PANGRP	PANEL-GROUP	PANGRP ICTEMP
Dynamic	MAK	ROW-BULK	ROWMAP-BULK	CONBLK
Dynamic	MAK	SRC-MTCH	SOURCE-MATCHING	SDMTCH MATCH
Dynamic	MAK	VOR-MTCH	VORTICITY-MATCHING	SDMTCH MATCH

APPENDIX 5-C

DYNAMIC MEMORY MANAGEMENT AND PROGRAM LIMIT PARAMETERS

Some care was taken in the design and construction of MAG to allow certain critical parameters to be increased as larger computers become available. On the following two pages (5-C.2 and 5-C.3), these parameters are listed together with their values (sometimes a formula), their description, where they are defined and the common block where they reside. In addition, notes are given describing the parts of the program that need to be changed if any particular parameter is altered. Notice that in preparing these charts, (and occasionally in other parts of this document as well), we have used the abbreviations "c.p." for control point, "s.p." for singularity parameter, and "nw" for network.

At the present writing (October 1984), the program is being run on various models of CRAY computers for which the following increases in critical parameters would be desirable and suitable.

MXING	384
MXCB	500
MXRCPB	150
MXCPBK	200
MXDY10	50,000
MXDY21	75,000
MXDY22	75,000
PANMAX	4,801

On the page following the summary of critical parameters, we present figures describing the allocation of scratch memory by programs MAG10, which organizes phases (1,A), (1,B) and (1,C) of the execution (problem setup) and MAG20, which organizes phases (2,A) and (2,B) of the execution (IC calculation and aggregation, AIC generation). Notice that the scratch common block /DYNAM/, containing the scratch work array W, is separately declared each time it is needed and not generated by a COMDECK call. In the memory maps presented on page 5-C.4, the various scratch array lengths appear inside the boxes, the array addresses (in W, /DYNAM/) appear to the right of the corresponding box and the array name for the given phase of processing is given in brackets below the array address.

Critical Parameters	Value	Description	Where Defined
NSING	NMBRSP(3)	Total no. of DQG s.p. parameters	OPENDB
MXING	160	Max # of s.p.'s associated w panel group. Also max # of panels in a group.	LOCKDATA
MXFGRD	$\max_k [(2 M_k-1)(2N_k-1)]$	Max # of pts in any nw's fine grid	OPENDB
NPDQBF	2048	Sequential file buffer size for panel data	LOCKDATA
MXRWCL	200	Maximum no. of rows or columns in any network	LOCKDATA
MXCB	250	Maximum # of c.p. blocks	LOCKDATA
NCPDQG	NMBRCP(4)	Total no. of DQG control points	OPENDB
MXRCPB	100	Max # of IC rows associated w a c.p. block	LOCKDATA
MXRP	[(MXDY22-MXING-2*NSING)/(NSING+MXING)]	Maximum number of IC rows allowed in a control point row partition	BLOCK
MXCPBK	150	Maximum # of c.p.'s in a c.p. block	LOCKDATA
MXPSRC	10	Max # of source parameter [B^S] may depend upon	LOCKDATA
MXPDBL	25	Max # of doublet parameters [B^D] may depend upon	LOCKDATA
MXDY10	40,000	Scratch memory for execution of MAG10	LOCKDATA
MXDY21	42,000	Scratch memory for execution of ICTEMP	LOCKDATA
MXDY22	42,000	Scratch memory for execution of GENAIC	LOCKDATA
MXPGP	100	Max # of panel groups	LOCKDATA
MXHYLO	10	Max # of points involved in any given matching condition	LOCKDATA
MXICTP	12	Maximum # of files on the ICTPxx db	LOCKDATA
MXNET	100	Maximum # of networks	---
MXNWID	200	Maximum value for a network id	---
NPANBF	8	Number of panels' data stored in a sequential file buffer for FPDQNU or FPDQUP	LOCKDATA
PANMAX	3001	Maximum number of panels + 1, the size of the index array for file PANDTA	LOCKDATA
PANWPR	256	Number of words per panel for the minimal panel data packet	LOCKDATA
NWCB	11	Number of words per c.p. for the c.p. data packet	LOCKDATA

Common Block	Notes
/MAGNUM/	Arrays of this length are dynamically allocated throughout MAG
/MAGPRM/	Arrays of this length are dynamically allocated throughout MAG
/MAGNUM/	Used to allocate space for fine grid maps in COLMAP and CONBLK
/SQFPDQ/	All sequential file buffers are dynamically allocated (2 in PANGRP, 1 in ICTEMP)
/MAGPRM/	Used to allocate the NCL closure data array in CONBLK
/MAGPRM/	cf. NCBSZ, /MAGNUM/
/MAGNUM/	Used to allocate JCPMAP, calculated by CONBLK
/MAGPRM/	Main impact of this parameter is on the size of RIC, phase 2,A
/MAGPRM/	Determined by the amount of memory in phase 2,B. Things should be arranged, if possible, so that MXRP*MXICTP$\geq$MXRCPB
/MAGPRM/	cf. WCB,/CPBLK/. Exceeds MXRCPB because some c.p.'s have no IC rows
/MAGPRM/	cf. PANGRP: arrays IISD, LOCSD, IISF, IISMAG, ASTS, ASTSF, BS. Also, see PAKPQF, UPKPQF. Arrays SG (SDMTCH); SG, SGH (GENAIC)
/MAGPRM/	cf. PANGRP: arrays IISD, LOCSD, IIDF, IIDMAG, ASTD, ASTDF, BD. Also, see PAKPQF, UPKPQF. Arrays AMU, DMU (SDMTCH); AMU, DMU, AMUH, DMUH (GENAIC)
/MAGPRM/	cf. W, /DYNAM/, MAG10.
/MAGPRM/	cf. W, /DYNAM/, MAG20.
/MAGPRM/	cf. W, /DYNAM/, MAG20.
/MAGPRM/	cf. NGRPSP, NGRPPA, /MAGNUM/.
/MAGPRM/	see arrays SIGNX, SPBIAS (MATCH); SPBIAS, SIGNX, XHYLO, KNETX, ICOLX, IROWX, MSUBPX (SDMTCH)
/MAGPRM/	cf. the list of file names ICTPSQ(1:MXICTP) in LOCKDATA. Should be a multiple of 4, $\geq$ 12
---	cf. NETORD, NROWNT, NCOLNT, NPNCUM, /MAGNUM/
---	cf. NETINZ, NETINV(1:200),/MAGNUM/
/SQFPDQ/	Always make sure that NPANBF*PANWPR = NPDQBF
/PINDEX/	cf. PANDEX, /PINDEX/.
/PINDEX/	cf. PANGRP, array PQF; cf. SDMTCH, array PQF.
/MAGPRM/	cf. WCB, /CPBLK/

MAG10 Dynamic Allocation

1,A [COLMAP]

NSING	← LCOLMP [ICOLMP]
NSING	← LSNGPK [ISNGPK]
MXFGRD	← LMAPFN [MAPFN]

1,B [PANGRP]

NSING	← LCOLMP [ICOLMP,IMAGSP]
2*MXING	← LNDGRP [INDGRP]
2*MXING	← LNDSRT [INDSRT]
2*MXING	← LNDLOC [INDLOC]
2*MXING	← LKNTGP [KNETGP]
2*MXING	← LIRWGP [IROWGP]
2*MXING	← LICLGP [ICOLGP]
2*NPDQBF	← LPDQ [WPDQ]

1,C [CONBLK]

NSING	← LCOLMP [ICOLMP]
5*MXRWCL	← LNCL [NCL]
2*(MXCB+1)	← LMAGCB [MAGCB]
NCPDQG	← LJCPMP [JCPMAP]
MXRWCL	← LICLCP [ICLCP]
MXRWCL	← LAA [AA]
MXRWCL	← LAD [AD]
3*MXFGRD	← LCPAIC [KCPAIC]

MAG20 Dynamic Allocation

2,A [ICTEMP]

(MXRCPB+1)*MXING	← LRIC [RIC]
MXING	← LKNETG [KNETGP]
MXING	← LROWGP [IROWGP]
MXING	← LCOLGP [ICOLGP]
NPDQBF	← LPDQ [WPDQ]

2,B [GENAIC]

MXRP*NSING	← LSIC [RIC]
(MXRP+1)*(MXING)	← LBIC [IBIC,BIC]
NSING	← LAIC [AIC]
NSING	← LAICCL [AICK]

APPENDIX 5-D

THE PIVC SUBASSEMBLY

The PIVC subassembly, which adds the influence of a given panel upon a particular control point into a group on block IC buffer, lies at the very heart of MAG. This package of subroutines performs the evaluation of panel influence coefficients described in detail in appendix J of the theory document. In this appendix we describe the operation and overall structure of the package with a special emphasis on the more arcane attributes of its operation.

The basic tree structure for the PIVC subassembly has been outlined in figure 5.2 (see the lower right hand corner). On the last page of this appendix (5-D.7) we have reproduced this structure and have added remarks describing the function of each element.

The basic data that are input to the PIVC subassembly are:

(i) The principal image of the control point (ZCP) plus its associated data (/CNTRQ/).

(ii) The panel defining quantities read from the FPDQxx database and unpacked by UPKPQF. These include all data required for far field and quasi far field computations (INFLU= 1,2,3,4), but exclude some of the data required for quasi-near field or near field computations (INFLU=5,6). The "data flags" INDQNF and INDQRP were set to .FALSE. by UPKPQF when the current panel's data were read in, and are subsequently set to .TRUE. only when the corresponding data are required and regenerated with a call to PSDDQ5 or PSDDQ6.

(iii) A partially completed group-on-block IC buffer, AIC, together with pointers (LAIC) that give the starting column indices in the buffer associated with each symmetry condition of the control point.

Given these data PIVC begins its task by calling DINFLU to evaluate all geometric images of the control point* and PIC computation indices for each image (N.B. 1=monopole, 2=dipole, 3=quadrupole, 4=quasi-far, 5=quasi-near, 6=near field). If any PIC computation indices have the values 5 or 6 (quasi-near or near field values), subroutine PSDDQ5 is called to generate quasi-near field data and that fact is recorded in INDQNF. (Note that type 6 data for a true near field computation are not immediately generated. This is because, even when DINFLU recommends a type 6 PIC computation, subroutine IC always first attempts a type 5 computation before deciding that a type 6 is necessary.) Having regenerated any required type 5 data, PIVC then zeroes out PIC buffers for all required symmetry conditions and proceeds to call subroutine IC.

---

*Remark: For control points lying in a plane of symmetry, special care is taken in DINFLU not to reflect those points in their plane of symmetry. (cf. algorithm A_1, appendix K.5, Theory Document.)

Subroutine IC invokes lower level PIC routines to perform the actual computation of the influence of a panel upon a control point image. In this process there are four main areas where there is some subtlety in the code. We describe each of them.

(1) The velocity influence coefficients required from subroutine FFPIC are, in the language of appendix K of the theory document, (cf. p. K.6-2),

$$R^{ij}\,\vec{V}^{Q}_{\sigma}(R^{ij}\vec{p},\ s_I) \qquad R^{ij}\,\vec{V}^{Q}_{\mu}(R^{ij}\vec{p},\ m_I)$$

$$\vec{\nu}^{T}_{p}\,R^{ij}\,\vec{V}^{Q}_{\sigma}(R^{ij}\vec{p},\ s_I) \qquad \vec{\nu}^{T}_{p}\,R^{ij}\,\vec{V}^{Q}_{\mu}(R^{ij}\vec{p},\ m_I) \tag{5.D.1}$$

Because FFPIC works mainly in the mean panel's local coordinate system, the velocity influence coefficients most readily computed would be denoted

$$J\,\vec{V}'^{Q}_{\sigma}(R^{ij}\vec{p},\ \phi_\alpha) \qquad \vec{V}'^{Q}_{\mu}(R^{ij}\vec{p},\ \phi_\beta) \tag{5.D.2}$$

where the prime (') indicates that these VIC's are given in the local coordinate system. The basis functions $\{\phi_\alpha,\ \alpha=1,2,3\}$ and $\{\phi_\beta,\ \beta=1,\ldots,6\}$ are the standard polynomial basis functions $\lfloor 1,\ \xi,\ \eta,\ \ldots\ \eta^2/2 \rfloor$ used for the representation of the approximate source and doublet distributions in the far field. The way in which ϕ_α are used to construct approximations to s_I and m_I is easily described and is given, in the language of appendix I.3 of the theory document by,

$$\lfloor \phi_\alpha \rfloor^{1\times 3}\ [PSPL^S]^{3\times 5}\ [B^S]^{5\times N_S} \approx \lfloor s_I \rfloor^{1\times N_S} \tag{5.D.3}$$

$$\lfloor \phi_\beta \rfloor^{1\times 6}\ [PSPL^D]^{6\times 9}\ [B^D]^{9\times N_D} \approx \lfloor m_I \rfloor^{1\times N_D} \tag{5.D.4}$$

with the following matrix-FORTRAN variable connections

$$[PSPL^S] = RA, \quad [B^S] = ASTS, \quad [PSPL^S]\,[B^S] = ASTSF \tag{5.D.5}$$

$$[PSPL^D] = QA, \quad [B^D] = ASTD, \quad [PSPL^D]\,[B^D] = ASTDF \tag{5.D.6}$$

The primed VIC's are transformed to reference coordinates using the reference to local transformation A_5 (cf. appendix E.3 of the Theory Document) for subpanels 5 through 8:

$$\vec{V}^{Q}_{\sigma}(R^{ij}\vec{p},\ \phi_\alpha) = A_5^T(J\,\vec{V}'^{Q}_{\sigma}(R^{ij}\vec{p},\ \phi_\alpha) \tag{5.D.7}$$

$$\vec{V}^{Q}_{\mu}(R^{ij}\vec{p},\ \phi_\beta) = A_5^T(\vec{V}'^{Q}_{\mu}(R^{ij}\vec{p},\ \phi_\beta) \tag{5.D.8}$$

The required results may thus be obtained by forming the following quantities, as indicated.

$$(R^{ij}\ A_5^T)\ [J\ \vec{V}_\sigma^Q(R^{ij}\vec{p},\ \phi_\alpha)] \tag{5.D.9}$$

$$(R^{ij}\ A_5^T)\ [\vec{V}_\mu^Q(R^{ij}\vec{p},\ \phi_\beta)] \tag{5.D.10}$$

$$(\vec{v}_p^T\ R^{ij}\ A_5^T)\ [J\ \vec{V}_\sigma^Q(R^{ij}\vec{p},\ \phi_\alpha)] \tag{5.D.11}$$

$$(\vec{v}_p^T\ R^{ij}\ A_5^T)\ [\vec{V}_\mu^Q(R^{ij}\vec{p},\ \phi_\beta)] \tag{5.D.12}$$

Thus, when transforming velocity PIC's out of local coordinates, FFPIC applies the matrix $R^{ij}\ A_5^T$ (=RATF(*,i,j), see FFDQGX). Further, when generating normal mass flux PIC's, FFPIC multiplies on the left by the vector:

$$\llcorner ARNU \lrcorner \quad = \quad \vec{v}_p^T\ [R^{ij}\ A_5^T] = [A_5\ R^{ij}\ \vec{v}_p]^T \tag{5.D.13}$$

(2) The quasi far field evaluation of PIC's (QFFCAL) is very similar to the far field evaluation except that NFTPIC (the near field PIC routine) is used to evaluate the PIC's in local coordinates. Having computed the PIC's in local coordinates, NFTPIC then returns the following quantities, (where the vector $\llcorner ZNU \lrcorner$ and the 3x3 array [ART] are passed through the calling sequence)

$$[ART]\ [J\ \vec{V}_\sigma^Q(R^{ij}\vec{p},\ \phi_\alpha)],\ \text{etc.} \tag{5.D.14}$$

$$\llcorner ZNU \lrcorner \quad [ART]\ [J\ \vec{V}_\sigma^Q(R^{ij}\vec{p},\ \phi_\alpha)],\ \text{etc.} \tag{5.D.15}$$

By comparing these with the expressions appearing in (5.D.9-12) above, we see that the correct choices for $\llcorner ZNU \lrcorner$ and [ART] are:

$$[ART] = R^{ij}\ A_5^T \qquad (= \text{RATF}(*,\ i,j)) \tag{5.D.16}$$

$$\llcorner ZNU \lrcorner = \vec{v}_p^T \qquad (= \text{ZNUCP, see /CNTRQ/}) \tag{5.D.17}$$

(3) The calculation of quasi-near and near field PIC's by QNFCAL and PIFCAL requires evaluation of expressions of the form,

$$R^{ij} \sum_{k,\alpha} A_k^T\ [J_k\ \vec{V}_\sigma^{Q_k}(R^{ij}\vec{p},\ \phi_\alpha^k)]\ \text{SPSPL}_{\alpha\gamma}^S \tag{5.D.18}$$

$$\vec{v}_p^T\ R^{ij} \sum_{k,\alpha} A_k^T\ [J_k\ \vec{V}_\sigma^{Q_k}(R^{ij}\vec{p},\ \phi_\alpha^k)]\ \text{SPSPL}_{\alpha\gamma}^S \tag{5.D.19}$$

with similar expressions for doublet influence coefficients. Here, the sum with respect to k extends over subpanels k for which the reference to local transformation is A_k and having subpanel basis functions $\{\phi_\alpha^k\}$. The

symbol $SPSPL^S$ denotes the source subpanel (or half-panel) spline matrix. In the actual operation of the code, we loop over the subpanels, calculating for each and accumulating into an array, the quantities

$$\sum_k \left\{ \sum_\alpha \left(A_k^T \, [J_k \, \vec{V}_\sigma^{Q_k}(R^{ij}\vec{p}, \phi_\alpha^k)] \;\; SPSPL_{\alpha\gamma}^S \right) \right\} \tag{5.D.20}$$

$$\sum_k \left\{ \sum_\alpha \left([\vec{v}_p^T \, R^{ij}] \, A_k^T \, [J_k \, \vec{V}_\sigma^{Q_k}(R^{ij}\vec{p}, \phi_\alpha^k)] \;\; SPSPL_{\alpha\gamma}^S \right) \right\} \tag{5.D.21}$$

Thus the k-th call to NFTPIC is made with ⌊ZNU⌋ and [ART] given by

$$\lfloor ZNU \rfloor = \vec{v}_p^T \, R^{ij} \quad \text{(cf. calculation of RNU in IC)} \tag{5.D.22}$$

$$[ART] = A_k^T \tag{5.D.23}$$

Further, when the loop over subpanels is complete, R^{ij} must still be applied to the velocity influence coefficients (as distinguished from the normal mass flux IC's), the quantities expressed by (5.D.20).

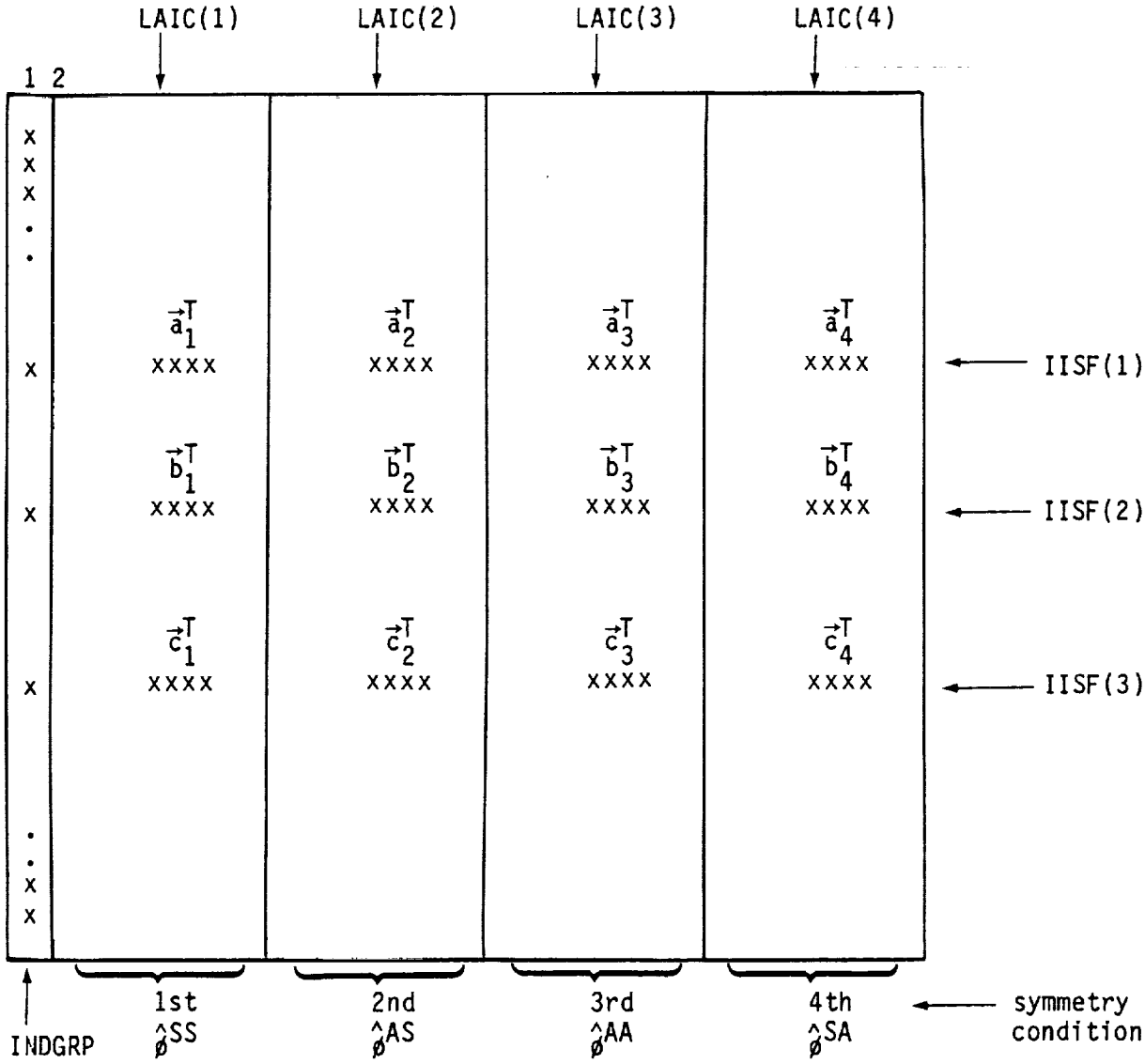
(4) Whenever some of the PIC computations are types 1 through 4 for some control images and types 5 or 6 for other control point images, the far field or quasi far field PIC's must be re-expressed in terms of the 5 panel source and 9 panel doublet parameters before they are accumulated. This transformation, accomplished with the help of the panel splines PSPL (=RA) and $PSPL^D$ (=QA), is necessary because ASTS and ASTD (rather than ASTSF and ASTDF) will be used to express the symmetrized PIC's in terms of global singularity parameters. (The code that applies $PSPL^S$ and $PSPL^D$ to panel influence coefficients directly follows the calls to FFPIC and QFFCAL in IC.)

The preceeding discussion summarizes the main fine points of subroutine IC. Once the PIC's for a given control point image have been calculated, they are added/subtracted into the accumulators for the required symmetry conditions.

When all control point images have been processed, the influence of the panel on each control point symmetry condition is complete. Subroutine IC is now prepared to add these PIC's into the panel group on control point block IC buffer, applying an outer spline matrix as it proceeds. The figure below demonstrates the inclusion of the source PIC's for the 1st symmetry condition into the IC buffer. To simplify the example we assume that ASTS depends on only 3 global source parameters. Defining

$$[\vec{a}_1 \quad \vec{b}_1 \quad \vec{c}_1] = \underbrace{[PIC^S]^{4\times5}}_{\text{1st symmetry condition}} \quad [ASTS]^{5\times3} \tag{5.D.24}$$

we see in figure 5-D.1 (see p. 5-D.6) that $\vec{a}_1$, $\vec{b}_1$ and $\vec{c}_1$ are added into rows IISF(1), IISF(2) and IISF(3) of AIC, starting in column LAIC(1). Similarly, the second symmetry condition PIC's are added into the same set of rows starting with column LAIC(2), and so on.



Global s.p. indices associated with this panel group.

IC's for various symmetry conditions

AIC = Panel Group on Control Point Block IC Buffer

Figure 5-D.1 Inclusion of Panel Influence Coefficients in the Panel Group on Control Point Block IC Buffer

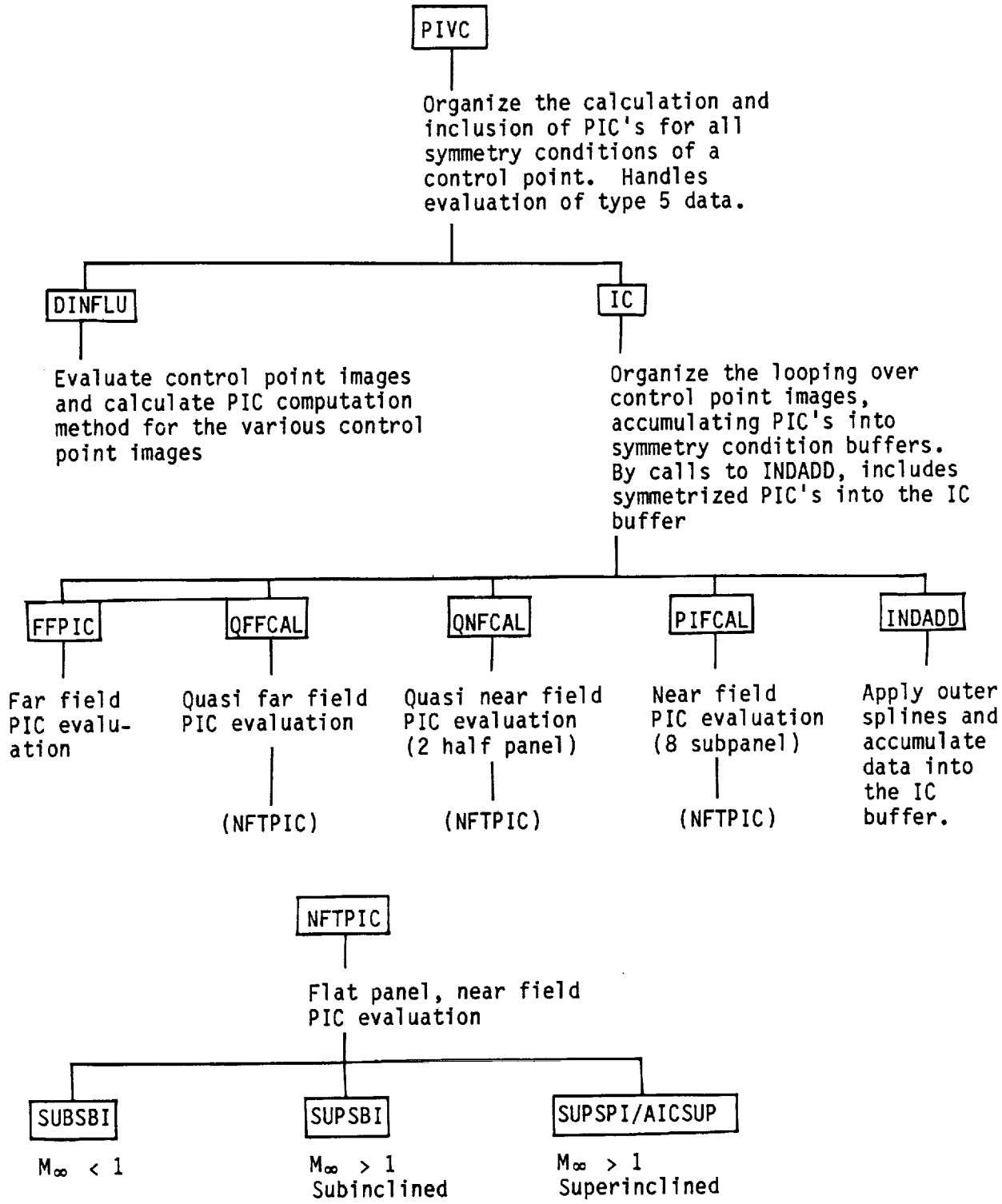


Figure 5-D.2 Tree Diagram for the PIVC Subassembly

APPENDIX 5-E

PANEL DEFINING QUANTITIES IN MAG

Much care is taken in MAG to minimize run cost (both CPU and I-O) by means of careful handling of the panel defining quantities. In this appendix we discuss where the various panel defining quantities come from and how the concept of data regeneration is implemented in MAG.

The basic observations which motivated the way in which panel defining quantities were handled were the following:

(i) Disk I-O, especially random I-O, tends to be quite expensive. Version 1.0 of MAG was unacceptably inefficient with respect to its use of I-O, mainly because DQG's PANEL-SPEC dataset was far larger than necessary.

(ii) Many of the panel defining quantities used in PAN AIR are "soft" quantities in the sense that they can be quickly regenerated from other panel defining quantities much more cheaply than they can be read from disk. On the other hand, some of the quantities are very "hard", in the sense that they can be regenerated only with a large amount of work. During the design and coding of MAG, an element or an array of panel defining quantities was judged to be "soft" if it could be regenerated at an average cost of under 5 μsec per word on a CDC 7600. This tradeoff criterion was developed from a careful examination of various data center charge algorithms combined with a good understanding of the execution environment of the program.

As a consequence of these observations a set of hard panel defining quantities was identified that was as small as possible, consistent with the 5 μsec tradeoff. When this had been done, the volume of hard quantities was about 270 words/panel. Now since a CDC 7600 handles random I-O much more efficiently when record sizes are a multiple of 64, the "hard" quantities were reduced to just below 256 by identifying as soft an array (CF) that slightly violated the tradeoff criterion. Having defined the "hard" panel data, a set of routines PAKPQF and UPKPQF were constructed that respectively create and subsequently unravel a 256 word panel data packet. Once this data packet has been created (by PANGRP's call to PAKPQF), it is written out to the random file PANDTA (for random access by SDMTCH and GENAIC) and also to the FPDQxx set of buffered sequential files (for sequential access by ICTEMP).

The common block definition charts at the end of this appendix describe the provenance of each entry in the four panel defining quantity common blocks, (/PANDQ/, /PANDF/, /PANDQX/, /PANDFX/). For the hard panel data in /PANDQ/ and /PANDF/, either the source in /DQGPAN/ (read from DQG's MAG-PANEL-SPEC dataset) is indicated or else the routine in MAG that generates the data is listed. The basic organizational idea for these common blocks is that /PANDQ/ and /PANDF/ contain the hard panel data (exceptions: ASTSF, ASTDF) while /PANDFX/ contains soft far field data regenerated by calls from UPKPQF to CCALN and FFDQGX, and /PANDQX/ contains soft near field data regenerated with calls to XCOF, RACOF, PSDDQ5 and PSDDQ6.

/PANDQ/ definition

PANDTA	Variable	Definition in PANGRP	Notes w.r.t. PANGRP def., related value in DQGPAN	Subsequent Definition	Notes w.r.t. subsequent definition
X	KNETNR	MGPAND	/DQGPAN/:NWKID	UPKPQF	
X	ICOLNR	MGPAND	/DQGPAN/:ICLPAN	UPKPQF	
X	IROWNR	MGPAND	/DQGPAN/:IRWPAN	UPKDQF	
X	ITS	MGPAND	ITS=1 (if σ)+2 (if μ)	UPKPQF	
X	ICS	MGPAND	index of collapsed side. Compared to ICSPAN in /DQGPAN/.	UPKPQF	
X	ISQN	NEARDT	defines 1/2 panel cut	UPKPQF	
X	INS	MGPAND	/DQGPAN/: (IDS/6)	UPKPQF	
X	IND	MGPAND	/DQGPAN/: (IDD/10)	UPKPQF	
X	IIN	MGPAND	sgn $\{\hat{n}_k, \hat{n}_k\}$	UPKPQF	
X	IISMAG	PANGRP	/DQGPAN/:BS(6,*) plus IMAGSP	UPKPQF	UPKPQF reformats slightly the PANDTA data
X	IIDMAG	PANGRP	/DQGPAN/:BD(10,*) plus IMAGSP	UPKPQF	
X	CP	MGPAND	/DQGPAN/:PC	UPKPQF	
X	EN	MGPAND	/DQGPAN/:RN	UPKPQF	
X	AQ	NEARDT	Skewed coord. transform	UPKPQF	
X	C1	NEARDT	Skewness parameters	UPKPQF	
X	C2	NEARDT		UPKPQF	
X	C3	NEARDT		UPKPQF	
X	CTEST	NEARDT	used in QNF test, QNFCAL	UPKPQF	
X	DIAM	MGPAND	/DQGPAN/:PDIAM	UPKPQF	
X	AREAQ	MGPAND	/DQGPAN/:SBAREA(9)	UPKPQF	
X	ASTS	MGPAND	/DQGPAN/:BS(1:5,1:INS)	UPKPQF	unpacked by UPKAST (sparse storage)
X	ASTD	MGPAND	/DQGPAN/:BD(1:9,1:IND)	UPKPQF	
X	STALAM	NEARDT	not used	---	not used
X	STRC	NEARDT	(s*,t*) for QNF splines	UPKPQF	

X : This variable is saved on the PANDTA dataset.

/PANDF/ definition

PANDTA	Variable	Definition in PANGRP	Notes w.r.t PANGRP def., related value in DQGPAN	Subsequent Definition	Notes w.r.t. subsequent definition
*	KNETF	MGPAND	/PANDQ/:KNETNR	UPKPQF	/PANDQ/:KNETNR
*	ICOLF	MGPAND	/PANDQ/:ICOLNR	UPKPQF	/PANDQ/:ICOLNR
*	IROWF	MGPAND	/PANDQ/:IROWNR	UPKPQF	/PANDQ/:IROWNR
*	INSF	MGPAND	/PANDQ/:INS	UPKPQF	/PANDQ/:INS
*	INDF	MGPAND	/PANDQ/:IND	UPKPQF	/PANDQ/:IND
*	ITSF	MGPAND	/PANDQ/:ITS	UPKPQF	/PANDQ/:ITS
*	ICSF	MGPAND	/PANDQ/:ICS	UPKPQF	/PANDQ/:ICS
*	CPFZ	MGPAND	/PANDQ/:CP(1:3,9)	UPKPQF	/PANDQ/:CP(1:3,9)
*	CPF	MGPAND	/PANDQ/:CP(1:3,1:4)	UPKPQF	/PANDQ/:CP(1:3,1:4)
*	ENCF	MGPAND	/PANDQ/:EN(1:3,5)	UPKPQF	/PANDQ/:EN(1:3,5)
X	RFMIN	MGPAND	min (IIN(K) IIN(K)≠0)	UPKPQF	
X	QDLTF	FARDT	used for rapid influence	UPKPQF	
X	PWF	FARDT	test, supersonic flows	UPKPQF	
X	PXF	FARDT	($M_\infty > 1$)	UPKPQF	
X	DIAMF	MGPAND	/PANDQ/:DIAM	UPKPQF	
X	SGXF	FARDT	$\mathrm{sgn}(\hat{n}_5 \cdot \hat{c}_0)$	UPKPQF	
	ASTSF	MGPAND	[RA][ASTS]	UPKPQF	calculated: [RA][ASTS]
	ASTDF	MGPAND	[QA][ASTD]	UPKPQF	calculated: [QA][ASTD]
X	IISF	PANGRP	panel group indices, σ	UPKPQF	
X	IIDF	PANGRP	panel group indices, μ	UPKPQF	
X	NCONVX	MGPAND	/DQGPAN/:NOTCVX	UPKPQF	
X	LVTERM	MGPAND	/DQGPAN/:LVFLAG	UPKPQF	

X : This variable is saved on the PANDTA dataset.

* : This variable is saved under a different name on the PANDTA dataset.

5-E.3

/PANDQX/ definition

Data Type	Variable	Regeneration	Notes w.r.t. regeneration	Theory Document Reference
Far-field	QA	XCOF	doublet panel spline	I.3.1
	RA	RACOF	source panel spline	I.3.1
Quasi-near	QK	PSDDQ5	doublet half panel spline	I.3.2
	RK	PSDDQ5	source half panel spline	I.3.2
	PK	PSDDQ5	half panel local coordinates	---
	SGX	PSDDQ5	sgn $(\hat{n}_i, \hat{c}_o)$	---
	AJ	PSDDQ5	J_k area jacobian	E
	AR	PSDDQ5	reference to local transformations, A_k } RCSLOC	E
	ART	PSDDQ5	A_k^T	---
Near	QQ	PSDDQ6	(C1, C2) → (ALAM) via GTALAM, (ALAM) → (QQ) via PDQSUB	I.2.3
	RR	PSDDQ6	Source subpanel splines	I.2.2
	PP	PSDDQ6	Subpanel local coordinates	---

Remarks: (1) XCOF [QA = $PSPL^D$], RACOF [RA=$PSPL^S$] are called from UPKPQF
(2) PSDDQ5 is called from PIVC
(3) PSDDQ6 is called from IC

/PANDFX/ definition

Data Type	Variable	Regeneration	Notes w.r.t. regeneration	Theory Document Reference
Far-field	HM	FFDQGX	calculated from CF	I.4.1, J.9
	HBM	FFDQGX	calculated from CF	I.4.1, J.9
	RQFF	FFDQGX	mean panel, projected in $\bar{X}$, expressed in X (compressibility coordinates)	---
	AF	FFDQGX	(ENCF) → (AF) via RCSLOC	E
	AJF	FFDQGX	(ENCF) → (AJF) via RCSLOC	E
	RF	FFDQGX	(ENCF) → (RF) via RCSLOC	E
	RADF	FFDQGX	.5*DIAMF	---
	PF	FFDQGX	mean panel, projected in $\bar{X}$, expressed in local coordinates	---
	CF	CCALN	$C_{ij} = \iint \xi^{i-1} \eta^{j-1} d\xi \, d\eta$, always calculated by CCALN	I.4.3
	RATF	FFDQGX	$R^{ij} \cdot A^T$ (n.b. RATF(*,1,1) ≡ AFT)	K.2
	LVORTX	FFDQGX	line vortex flag for whole panel	---
	QLVT	FFDQGX	line vortex inner splines	---

Remark: CCALN [CF = C_{ij}] and FFDQGX are called from UPKPQF

5-E.5

APPENDIX 5-F

PRINTED OUTPUT AND PROGRAMMING AIDS

In this appendix we describe the printout and print control features included in the MAG program. Most of the printout from MAG is principally of interest to the maintenance programmer trying to track down the source of a problem. In addition to providing a fairly comprehensive picture of the analysis performed by MAG, the printout from MAG has proved to be quite helpful in tracking down errors in the user's problem specification and in DQG's analysis of that specification.

In the subsections that follow, we will describe the printout from MAG according to the following classification scheme:

(1) Printout controlled by MAG's internal print flags. (cf. common block /WFLAGS/ defined by OPENDB using dataset GLOBAL-PRINTS from the DIP database)

(2) A summary of the error conditions detected by MAG

5.F.1 Print Flag Controlled Output

Each of the descriptions that follow is headed with the following information

- o the name of the internal print flag (in /WFLAGS/) that controls the particular printout
- o the index of the print flag in the array MAG-PRINTS (DIP database) that turns on each internal print flag, enclosed in brackets: [2]
- o the program elements in MAG that refer to the internal print flag
- o a one-line description of the printout

WDFALT, [1], (MAG, COLMAP, MAG20) High level problem and processing statistics, timing information

MAG prints the following information:

(1) Source and doublet PIC counts for each type of PIC computation method (e.g. near field, far field, etc.)

(2) A summary of the volume of I-O for each of nearly 30 categories of I-O. This information includes estimates of the volume (in words) of I-O for each category along with the apparent number of I-O requests. Because the operating system buffers most I-O operations, the actual number of I-O requests is generally somewhat lower than the apparent number, which is computed by counting the number of calls to I-O routines (e.g. READ, WRITE, REBUF, WRBUF, ESGET, ESPOR). In figure 5-F.1 we have produced an

index describing for each category of I-O its general nature, its map or file name and the name of the routines requesting the I-O.

(3) CPU totals for PIC evaluation and AIC assembly

(4) Counts for general, matching and closure boundary conditions for each symmetry condition.

COLMAP prints the number of (MAG) singularity parameters in the four categories: (1) known, nonupdatable, (2) known, updatable, (3) unknown, nonupdatable, (4) unknown, updatable.

MAG20 prints a running summary of the c.p. block processing together with the corresponding CPU time.

WCMMAP, [1], (MAG10, MAG20) Absolute octal addresses of dynamic memory

As a supplement to the program load map, MAG prints the absolute octal addresses of all of the arrays allocated from the dynamic memory buffer (common block /DYNAM/).

WCPLST, [2], (DRWMAP) Summary of ROWMAP's information about each MAG control point

For each MAG control point, the following information is printed. The label in brackets is the label appearing at the top of each page on the printout.

[IUPDCP]	Control point updatability type (0=nonupdatable, 1=updatable)	
[IMAGCP]	MAG control point index	
[IDQGCP]	DQG control point index	
[IBLKCP]	c.p.'s block index	
[IRPTCP]	c.p.'s row partition index	
[KNETCP]	network identifier	describe control point location
[IROWCP]	panel row index	describe control point location
[ICOLCP]	panel column index	describe control point location
[MSUBCP]	subpanel index	describe control point location
[IROW]	fine grid row index	describe control point location
[ICOL]	fine grid column index	describe control point location
[IN-POS-FLAG]	1 → in first POS, 2 → in second POS, otherwise 0	
[ON-POS-FLAG]	1 → $\overline{\text{on}}$ first POS only, $\overline{2}$ → on second POS only, 3 → on both	
[NULLIN]	0 → no AIC rows for this c.p., 1 → at least 1 AIC row	
[IPCP]	IC row counts for required ø, $\vec{v}$ and $\vec{w}.\hat{n}$ IC's	
[NTCHAR]	Number of AIC rows for this c.p.	
[IDRWCP]	AIC row indices for this c.p.	
[ONST]	Control point type (1=center), (2=edge), (3=corner), (4=extra)	
[IUPDCL]	Updatability type of affected closure c.p.	
[IDRWCL]	AIC row index of affected closure boundary condition	

[LSYMCL]	Index of symmetry conditions for affected closure conditions (bit vector)
[ZCP]	Control point location, reference coordinates
[ICHAR]	Characteristics for b.c.'s, all symmetry conditions, 1st and 2nd b.c.
[IPDPBC]	PDP IC row counts for ϕ, $\vec{v}$ and $\vec{w}.\hat{n}$ IC's required in post processing

WBCLST, [2], (DRWMAP)
Summary of boundary condition information for each AIC row

For each AIC row, the following information is printed. The full printout is repeated for each symmetry condition of interest. The data headers are listed in brackets.

[ROW]	AIC row index
[BC TYPE]	Boundary condition type (e.g. GENERAL, SNG-SPEC etc.)
[CH]	Boundary condition character index (1 → GENERAL, etc.)
[CPI:MAG]	MAG c.p. index
[CPI:DQG]	DQG c.p. index
[U]	Updatability type [0=nonupdatable, 1=updatable]
[R]	1 → first b.c., 2 → second b.c., (for this c.p.)
[A/A:WN]	a_A, coefficient of $(\vec{w}.\hat{n})_A$
[C/A:PHI]	c_A, coefficient of $(\phi)_A$
[T/A:V/A]	$\vec{t}_A$, coefficient of $(\vec{v})_A$
[A/D:SG]	a_D, coefficient of σ
[C/D:MU]	c_D, coefficient of μ
[T/D:DMU]	$\vec{t}_D$, coefficient of $\nabla\mu$
[CLS/ROW]	AIC row index of affected closure condition (negative if first)
[A/A:WN]	$a_{A,k}$, coefficient of $A_k(\vec{w}.\hat{n})_k$ in affected closure condition
[A/D:SG]	$a_{D,k}$, coefficient of $A_k(\sigma)_k$ in affected closure condition

WSPMAP, [3], COLMAP
Singularity Parameter Maps

Singularity parameter maps are printed giving a schematic description of both the MAG and the DQG singularity parameter indices. For each point on the fine grid matrix for each network, the MAG and the DQG source and doublet indices are printed.

WBCMAP, [3], (CONBLK) Control point and AIC maps

For each network a fine grid matrix map is printed giving the following information:

- o DQG c.p. index
- o MAG c.p. index
- o first boundary condition AIC row number
- o second boundary condition AIC row number

WGENRL, [4], (OPENDB, BLOCK) Generally useful information

OPENDB prints some global information about the whole configuration that is useful, but poorly formatted. BLOCK prints information relevant to the calculation of MXRP of /MAGPRM/.

WCLMAP, [4], (COLMAP) COLMAP processing summary

Includes the bulk column mapping array ICOLMP/IMAGSP, giving the DQG → MAG singularity parameter index mapping.

WPNGRP, [4], (PANGRP) PANGRP processing summary

Includes all information written to the PANEL-GROUP dataset.

WCNBLK, [4], (CONBLK) CONBLK processing summary

Includes the bulk control point mapping array JCPMAP giving the DQG → MAG c.p. index mapping.

5.F.2 Error Conditions Detected by MAG

A summary of all error conditions detected by MAG has been generated, and presented in figure 5-F.2, by extracting from the code all calls to subroutines MAGERR and MAGMSG. The format of the calls to each of these is identical: the name of the subroutine generating the error message followed by the error message itself (up to 40 characters). Calls to MAGERR are

immediately fatal while calls to MAGMSG are temporarily ignored, causing fatal error termination at the end of the execution of MAG10.

Occassionally some extra information is printed out along with the error message to help explain the cause of the problem. If the particular error is extremely uncommon, it will probably be necessary to refer to the code to interpret this printout.

	MAP or filename	Activity R=read W=write	Subroutines	Nature(*) of data
1.	ICTPxx	W	WRICT	IC
2.	ICTPxx	R	GENAIC	IC
3.	IC [1]	W	GENAIC	IC (ϕ)
4.	IC [2-4]	W	GENAIC	IC ($\vec{v}$)
5.	IC [5]	W	GENAIC	IC ($\vec{w}.\hat{n}$)
6.	AIC	W	GENBC, MATCH	AIC
7.	AIC	R	GENAIC	AIC (closure)
8.	AIC	W	GENAIC	AIC (closure)
9.				
10.	FPDQxx	R	ICTEMP	PDQ
11.	PANDTA (random)	R	GENAIC	PDQ
12.				
13.				
14.	PANDTA (random)	R	SDMTCH	PDQ
15.	ROWMAP, ROWINV	W	CLSROW, PROCP	CP
16.	ROWMAP, ROWINV	R	CLSROW, PROCP, DRWMAP	CP
17.	ROWMAP	R	MAG20	CP
18.	PANSPEC	R	PANGRP	PDQ
19.	PANDTA (random)	W	PANGRP	PDQ
20.	FPDQxx	W	PANGRP	PDQ
21.	BNDRY, CNTRQ	R	DRWMAP, PROCP	CP
22.	BNDRY	R	GENAIC	CP
23.	EXHYLO	R	SDMTCH	MATCH
24.				
25.	PANGRP	R	ICTEMP	---
26.	SRC/DBL/VOR-MTCH	W	SDMTCH	MATCH
27.	SRC/DBL/VOR-MTCH	R	MATCH	MATCH
28.				
29.				
30.				

* Codes describing nature of data

IC	Influence coefficients (ΦIC, $\overrightarrow{\text{VIC}}$ or WIC)
AIC	Entries in the [AIC] matrix
PDQ	Panel defining quantities
CP	Control point and/or boundary condition data
MATCH	Information relating to matching conditions

Figure 5-F.1 Index to Summary of Substantial I-O

```
**************  43      MAGERR
          MAG          CALL MAGERR ("MAG","TERMINATED DUE TO ACCUMULATED ERRORS    ")        MAG     175

        DBABT       CALL MAGERR ("DBABT","FATAL SDMS ERROR DETECTED                 ")        DBABT    54

       UPKPQF       IF ( L-1 .GT. PANWPR ) CALL MAGERR ("UPKPQF"                              UPKPQF 140
                   X                      ,"PANEL DATA OVERFLOW                    ")        UPKPQF 141
       UPKPQF       IF ( L.NE.2 ) CALL MAGERR ('UPKPQF'                                       841024   1
                   X                      ,'L.NE.2, PACKED PANEL DATA OUT OF SYNCH  ')        841024   2
        MAG10       IF ( NERR .NE. 0 ) CALL MAGERR ("MAG10 "                                  MAG10  247
                   1                      ,"SDMS WRITE ERROR, DB = MAG, MAP = SYMTRY")        MAG10  248
        MAG10       CALL MAGERR ("MAG10","SCRATCH WORKING ARRAY TOO SMALL         ")          MAG10  251

        BLOCK       IF ( MXRP .LT. 5 ) CALL MAGERR ("BLOCK"                                   BLOCK  273
                   X                      ,"MAX ROW PARTITION IS LESS THAN 5        ")        BLOCK  274
       CLSROW           IF ( NERR .NE. 0 ) CALL MAGERR ("CLSROW"                              CLSROW 170
                   1                      "SDMS READ ERROR, DB = DQG, MAP = CLOSE  ")        CLSROW 171
       CLSROW               CALL MAGERR ('CLSROW'                                             841024  41
                   X                      ,'MIN. MAG CP INDEX .GT. 20000, SEE JCPMAP')        841024  42
       CLSROW               IF ( NERR .NE. 0 ) CALL MAGERR ("CLSROW"                          CLSROW 206
                   1                      ,"SDMS READ ERROR, DB = MAG, MAP = ROWINV ")        CLSROW 207
       CLSROW               IF ( NERR .NE. 0 ) CALL MAGERR ("CLSROW"                          CLSROW 238
                   1                      ,"SDMS WRITE ERROR, DB = MAG, MAP = ROWINV")        CLSROW 239
       CLSROW               IF ( NERR .NE. 0 ) CALL MAGERR ("CLSROW"                          CLSROW 249
                   1                      ,"SDMS WRITE ERROR, DB = MAG, MAP = ROWMAP")        CLSROW 250
       COLMAP           IF ( NERR .NE. 0 ) CALL MAGERR ("COLMAP"                              COLMAP 181
                   1                      ,"SDMS READ ERROR, DB = DQG, MAP = SNGSPC ")        COLMAP 182
       COLMAP               IF ( NERR .NE. 0 ) CALL MAGERR ("COLMAP"                          COLMAP 215
                   1                      ,"SDMS WRITE ERROR, DB = MAG, MAP = COLMAP")        COLMAP 216
       COLMAP               IF ( NERR .NE. 0 ) CALL MAGERR ("COLMAP"                          COLMAP 219
                   1                      ,"SDMS WRITE ERROR, DB = MAG, MAP = COLINV")        COLMAP 220
       COLMAP               IF ( NERR .NE. 0 ) CALL MAGERR ("COLMAP"                          COLMAP 226
                   1                      ,"SDMS READ ERROR, DB = MAG, MAP = COLMAP ")        COLMAP 227
       COLMAP       IF ( NERR .NE. 0 ) CALL MAGERR ("COLMAP"                                  COLMAP 248
                   1                      ,"SDMS WRITE ERROR, DB = MAG, MAP=COL-BULK")        COLMAP 249
       CONBLK       IF ( NERR .NE. 0 ) CALL MAGERR ("CONBLK"                                  CONBLK 253
                   1                      ,"SDMS READ ERROR, DB = DQG, MAP = NETWK  ")        CONBLK 254
       CONBLK       IF ( NERR .NE. 0 ) CALL MAGERR ("CONBLK"                                  CONBLK 309
                   1                      ,"SDMS READ ERROR, DB = DQG, MAP = SPCPT  ")        CONBLK 310
       CONBLK       IF ( NERR .NE. 0 ) CALL MAGERR ("CONBLK"                                  CONBLK 395
                   1                      ,"SDMS WRITE ERROR, DB = MAG, MAP=ROW-BULK")        CONBLK 396
       DRWMAP       IF ( NERR .NE. 0 ) CALL MAGERR ("DRWMAP"                                  DRWMAP 185
                   1                      ,"SDMS READ ERROR, DB = MAG, MAP = ROWMAP ")        DRWMAP 186
       DRWMAP       IF ( NERR .NE. 0 ) CALL MAGERR ("DRWMAP"                                  DRWMAP 194
                   1                      ,"SDMS READ ERROR, DB = DQG, MAP = BNDRY  ")        DRWMAP 195
       DRWMAP       IF ( NERR .NE. 0 ) CALL MAGERR ("DRWMAP"                                  DRWMAP 248
                   1                      ,"SDMS READ ERROR, DB = MAG, MAP = ROWMAP ")        DRWMAP 249
       DRWMAP       IF ( NERR .NE. 0 ) CALL MAGERR ("DRWMAP"                                  DRWMAP 255
                   1                      ,"SDMS READ ERROR, DB = DQG, MAP = BNDRY  ")        DRWMAP 256
       OPENDB       IF ( IRO.NE.0 ) CALL MAGERR ("OPENDB"                                     820428  35
```

Figure 5-F.2 Error Conditions Detected by MAG (page 1 of 3)

```
         X                        ,"DIP  DATABASE IS DEFECTIVE                 ")   820428  36
OPENDB    IF ( IRO .NE. 0 ) CALL MAGERR ("OPENDB"                                  OPENDB 187
         1                        ,"DQG  DATABASE IS DEFECTIVE                 ")   OPENDB 188
OPENDB    IF ( IRO .NE. 0 ) CALL MAGERR ("OPENDB"                                  OPENDB 194
         1                        ,"MAG  DATABASE IS DEFECTIVE                 ")   OPENDB 195
OPENDB    IF ( IRO .NE. 0 ) CALL MAGERR ("OPENDB"                                  OPENDB 199
         1                        ,"MAGX DATABASE IS DEFECTIVE                 ")   OPENDB 200
OPENDB    IF ( IRO .NE. 0 ) CALL MAGERR ("OPENDB"                                  OPENDB 207
         1                        ,"MAGY DATABASE IS DEFECTIVE                 ")   OPENDB 208
OPENDB    IF ( NERR .NE. 0 ) CALL MAGERR ("OPENDB"                                 OPENDB 561
         1                        ,"SDMS READ ERROR, DB = DQG, MAP = GLOBAL ")     OPENDB 562
OPENDB    IF ( NERR .NE. 0 ) CALL MAGERR ("OPENDB"                                 OPENDB 585
         1                        ,"SDMS WRITE ERROR, DB = DQG, MAP = NETWK ")     OPENDB 586
OPENDB    IF ( KNET.LT.1  .OR.  KNET.GT.200 ) CALL MAGERR ("OPENDB"                OPENDB 587
         1                        ,"NETWORK-ORDER VALUE IS OUT OF RANGE        ")  OPENDB 588
OPENDB    IF ( KNET.LT.1  .OR.  KNET.GT.200 ) CALL MAGERR ('OPENDB'                841024  78
         X                        ,'NETWORK ID OUTSIDE RANGE OF (1,200)     ')     841024  79
PAKPQF    IF ( NPQF .GT. PANWPR ) CALL MAGERR ("PAKPQF"                            PAKPQF 141
         X                        ,"PANEL DATA BUFFER EXCEEDS LIMIT            ")  PAKPQF 142
PANGRP              IF ( NERR .NE. 0 ) CALL MAGERR ("PANGRP"                      PANGRP 248
         1                        ,"SDMS READ ERROR, DB = DQG, MAP = PANSPEC")     PANGRP 249
PANGRP              IF ( INS.GT.10  .OR.  IND.GT.25 ) CALL MAGERR ('PANGRP'       841024  82
         X                        ,'INS.GT.10 OR IND.GT.25: TOO MANY SP-S   ')     841024  83
PANGRP                 IF ( NERR .NE. 0 ) CALL MAGERR ("PANGRP"                   PANGRP 334
         1                        ,"SDMS WRITE ERROR, DB = MAG, MAP = PANGRP")     PANGRP 335
PANGRP              IF ( NPQF.GT.NPAKFF ) CALL MAGERR ('PANGRP'                   841024  84
         X                        ,'PANEL DATA PACKET (NPQF) EXCEEDS NPAKFF ')     841024  85
PANGRP              IF ( LWPDQ+NPQF-1 .GT. NPDQ ) CALL MAGERR ('PANGRP'           841024  86
         X                        ,'PANEL DATA BUFFER EXCEEDS ALLOCATION    ')     841024  87
PANGRP              IF ( IPNDEX+1 .GT. PANMAX ) CALL MAGERR ("PANGRP"             PANGRP 413
         X                        ,"CONFIGURATION HAS MORE THAN 3000 PANELS")      CRAY   128
PANGRP         IF ( NERR .NE. 0 ) CALL MAGERR ("PANGRP"                           PANGRP 457
         1                        ,"SDMS WRITE ERROR, DB = MAG, MAP = PANGRP")     PANGRP 458
 PROCP     IF ( NERR .NE. 0 ) CALL MAGERR ("PROCP "                                 PROCP  290
         1                        ,"SDMS READ ERROR, DB = DQG, MAP = CNTRQ  ")     PROCP  291
 PROCP     IF ( NERR .NE. 0 ) CALL MAGERR ("PROCP "                                 PROCP  473
         1                        ,"SDMS READ ERROR, DB = MAG, MAP = ROWINV ")     PROCP  474
 PROCP     IF ( NERR .NE. 0 ) CALL MAGERR ("PROCP "                                 PROCP  505
         1                        ,"SDMS WRITE ERROR, DB = MAG, MAP = ROWMAP")     PROCP  506
 PROCP     IF ( NERR .NE. 0 ) CALL MAGERR ("PROCP "                                 PROCP  516
         1                        ,"SDMS WRITE ERROR, DB = MAG, MAP = ROWINV")     PROCP  517
SDMTCH         IF ( NERR .NE. 0 ) CALL MAGERR ("SDMTCH"                           SDMTCH 245
         1                        ,"SDMS WRITE ERROR, DB=MAG, MAP=SRC-MTCH  ")     SDMTCH 246
SDMTCH         IF ( NERR .NE. 0 ) CALL MAGERR ("SDMTCH"                           SDMTCH 312
         1                        ,"SDMS WRITE ERROR, DB=MAG, MAP=DBL-MTCH  ")     SDMTCH 313
 MAG20   X    CALL MAGERR ('MAG20'                                                841024  91
         X                        ,'DYNAMIC CM NEEDED(L21,L22) EXCEEDS LIMIT')     841024  92
 MAG20     IF ( NERR .NE. 0 ) CALL MAGERR ("MAG20"                                  MAG20  267
         1                        ,"SDMS READ ERROR, DB = MAG, MAP = ROWMAP .")    MAG20  268
 CBSET      CALL MAGERR ('CBSET'                                                   841024  95
```

Figure 5-F.2 Error Conditions Detected by MAG (page 2 of 3)

```
                X                        ,'PACK+UNPACK DOES NOT GIVE ORIGINAL DATA')        841024  96
GENAIC           IF ( JTP.GT.MXICTP ) CALL MAGERR ('GENAIC'                                 841024  97
                X                        ,'ATTEMPT TO USE MORE THAN MXICTP FILES   ')       841024  98
GENAIC           IF ( ICP2.LT.ICP1 ) CALL MAGERR ('GENAIC'                                  841024  99
                X                        ,'EMPTY ROW PARTITION DETECTED            ')       841024 100
GENAIC           IF ( NERR .NE. 0 ) CALL MAGERR ("GENAIC"                                   GENAIC 377
                1                        ,"SDMS READ ERROR, DB = DQG, MAP = BNDRY  ")       GENAIC 378
GENAIC           IF ( NERR .NE. 0 ) CALL MAGERR ("GENAIC"                                   GENAIC 463
                1                        ,"SDMS WRITE ERROR, DB = MAG, MAP = IC    ")       GENAIC 464
GENAIC                IF ( IPCP(2).EQ.0 ) CALL MAGERR ('GENAIC'                             841024 101
                X                        ,'CLOSURE BC: NEITHER V NOR W.N IC-S THERE')       841024 102
GENBC                 IF ( IPCP(2).EQ.0 ) CALL MAGERR ('GENBC'                              841024 103
                X                        ,'GENERAL BC: V IC-S NOT HERE FOR T.V TERM')       841024 104
GENBC                 IF ( IPCP(2).EQ.0 ) CALL MAGERR ('GENBC'                              841024 105
                X                        ,'GENERAL BC: V AND WN IC-S ABSENT FOR W.N')       841024 106
GENBC                 IF ( IPCP(1).EQ.0 ) CALL MAGERR ('GENBC'                              841024 107
                X                        ,'GENERAL BC: PHI IC-S ABSENT PHI TERM    ')       841024 108
ICTEMP           IF ( NERR .NE. 0 ) CALL MAGERR ("ICTEMP"                                   ICTEMP 217
                1                        ,"SDMS READ ERROR, DB = MAG, MAP = PANGRP ")       ICTEMP 218
MATCH            IF ( NERR .NE. 0 ) CALL MAGERR ("MATCH "                                   MATCH  126
                1                        ,"SDMS READ ERROR, DB=MAG MAP=SRC/DBL-MTCH")       MATCH  127
RDAIC                 CALL MAGERR ("RDAIC"                                                  RDAIC   98
                X                        ,"AIC PARTITION WAS THE WRONG SIZE        ")       RDAIC   99
RDAIC            IF ( NERR .NE. 0 ) CALL MAGERR ("RDAIC "                                   RDAIC  101
                1                        ,"SDMS READ ERROR, DB = MAG, MAP = AIC    ")       RDAIC  102
WRAIC            IF ( NERR .NE. 0 ) CALL MAGERR ("WRAIC "                                   WRAIC   88
                1                        ,"SDMS WRITE ERROR, DB = MAG, MAP = AIC   ")       WRAIC   89
WRICT            IF ( JTP.GT.MXICTP ) CALL MAGERR ('WRICT'                                  841024 109
                X                        ,'ATTEMPT TO USE MORE THAN MXICTP FILES   ')       841024 110

**************  45      MAGMSG
MAG10                 CALL MAGMSG ('MAG10'                                                  841024  15
                X                        ,'IMBALANCE BETWEEN S.P. AND B.C. COUNTS  ')       841024  16
COLMAP                      CALL MAGMSG ("COLMAP"                                           COLMAP 234
                1                        ,"MAG UPDATABLE S.P. INDEX INCONSISTENCY  ")       COLMAP 235
COLMAP                CALL MAGMSG ('COLMAP'                                                 841024  66
                X                        ,'BAD NETWORK ID DETECTED                 ')       841024  67
PAKAST                CALL MAGMSG ("PAKAST"                                                 PAKAST 112
                1                        ,"BUFFER OVERFLOW                         ")       PAKAST 113
SDMTCH                CALL MAGMSG ("SDMTCH/S"                                               SDMTCH 205
                1                        ,"SDMS READ ERROR, DB = DQG, MAP = EXHYLO ")       SDMTCH 206
SDMTCH                CALL MAGMSG ("SDMTCH/D"                                               SDMTCH 272
                1                        ,"SDMS READ ERROR, DB = DQG, MAP = EXHYLO ")       SDMTCH 273
SDMTCH           CALL MAGMSG ("SDMTCH"                                                      SDMTCH 321
                1                        ,"TOO MANY S.P.-S INVOLVED IN MATCHING    ")       SDMTCH 322
```

Figure 5-F.2 Error Conditions Detected by MAG (page 3 of 3)

APPENDIX 5-G

HANDLING CLOSURE B.C.'s IN MAG

The design capability in PAN AIR needs to be able to impose a special class of non-local boundary condition, called the closure condition. The precise form of the AIC constraint induced by this boundary condition is given in the PAN AIR theory document by equation (K.1.25) when there is no symmetry in the problem, and by equations (K.6.45-46), (K.6.48-49) and (K.6.55-58) when symmetry is present. For the purpose of the present discussion, we reproduce here equation (K.1.25) for the case of no symmetry.

$$\sum_{\substack{\text{panels} \\ P_k}} A_k \left\{ a_{A,k}\, \hat{n}_k^T [B_o] [VIC_k] + a_{D,k} \lfloor 1, \xi_k', \eta_k' \rfloor [SPSPL_k^S] [B_k^S] \right\} \vec{\lambda} = b \qquad (5.G.1)$$

Here the sum extends over the panels P_k in a row or column of a network. The other symbols in this equation have the meanings:

A_k	=	area of panel P_k (AREAQ)
$a_{A,k}$	=	the coefficient of $(\vec{w}.\hat{n})_k$ in the closure condition (AACLCP)
$\hat{n}_k$	=	the panel normal at the panel center control point $\vec{p}_k$ on panel P_k ($[B_o]\hat{n}_k$ = ZNUCP)
$[B_o]$	=	the dual metric matrix in reference coordinates (cf. app. E, theory document)
$[VIC_k]$	=	the 3xN velocity influence matrix for control point $\vec{p}_k$ (stored in RIC)
$a_{D,k}$	=	the coefficient of $(\sigma)_k$ in the closure condition (ADCLCP)
ξ_k', η_k'	=	the local coordinates of $\vec{p}_k$'s hypothetical location (not recessed)
$[SPSPL_k^S]$	=	the source subpanel spline matrix for the subpanel on which $\vec{p}_k$ lies
$[B_k^S]$	=	the 5xN (extended) source outer spline matrix for panel P_k
$\vec{\lambda}$	=	the vector of global singularity parameters
b	=	a user specified value

Note that by virtue of equation (5.2.5) of this section that the combination $\hat{n}_k^T [B_o] [VIC_k]$ could be replaced by $\lfloor WIC_k \rfloor$ when only normal mass flux IC's are computed for $\vec{p}_k$.

The main difficulty that must be dealt with to handle this equation is the fact that the velocity influence coefficients $[VIC_k]$ are not necessarily saved

on any file after they have been generated (in GENAIC) and used to compute AIC rows for $\vec{p}_k$. Thus, unless some special care is taken, the information required to generate a closure AIC row will not all be available when it is needed.

To deal with this situation, subroutine GENAIC has some very special logic to perform asynchronous management of a closure-AIC buffer, AICK. The basic idea is to add in to AICK the individual contributions from each control point $\vec{p}_k$ as the various [VIC_k] matrices become available. In order to implement this idea, we must know, for each control point $\vec{p}_k$ the following information that MAG stores on the ROWMAP dataset:

(i) The AIC row number of the closure condition that is affected by $\vec{p}_k$ [|IDRWCL|]

(ii) The symmetry conditions for which $\vec{p}_k$ makes a contribution to a closure condition [LSYMCL]

(iii) The updatability type for any affected closure condition [IUPDCL]

(iv) Whether or not control point $\vec{p}_k$ has the smallest MAG control point index of all the MAG control points influencing a particular closure condition. (IDRWCL<0 if $\vec{p}_k$ has the smallest MAG c.p. index.)

Associated with the AICK buffer, subroutine GENAIC keeps track of the following "AICK identifiers"

(a) the closure AIC row number [INDRWK]

(b) the closure AIC row's symmetry condition [ISYMK]

(c) the closure AIC row's updatability type [IUPCPK]

These three items are the keys to the record in the AIC-MATRIX dataset to which the closure AIC row will eventually be written. In fact, during the actual construction of the AIC closure row, partial sums of equation (5.G.1) may be written to and read from this record several times.

The actual management of the closure AIC buffer is then handled as follows. As GENAIC is called to process each control point block, it begins by setting the "AICK identifiers" to impossible values (e.g. symmetry condition 0) to indicate that AICK is empty. Next, whenever a matrix [VIC_k] is generated for a control point $\vec{p}_k$ and that control point influences a closure condition [IDRWCL $\neq$ 0], GENAIC prepares to include the effect of [VIC_k] into the AIC closure condition. This is done as follows

(1) If the "AICK identifiers" disagree with those associated with $\vec{p}_k$ and the current symmetry condition, the AICK buffer must be written to the AIC-MATRIX dataset (provided AICK is not empty) and the AICK buffer reset. Next, the AICK identifiers are set equal to (|IDRWCL| , ISYM, IUPDCL). Then, if $\vec{p}_k$ is the first c.p. to influence AICK, we set AICK=0. Otherwise AICK is read from the AIC-MATRIX dataset using the AICK identifiers.

(2) We are now ready to include $\vec{p}_k$'s contribution into AICK. First the [VIC] contribution is included as follows

$$\text{AICK} \longleftarrow \text{AICK} + (f.A_k.a_{A,k})\ \text{WIC}_k \qquad (5.G.2)$$

(WIC's available)

or

$$\text{AICK} \longleftarrow \text{AICK} + (f.A_k.a_{A,k})\ \vec{v}_k^T\ [\text{VIC}_k] \qquad (5.G.3)$$

(VIC's available)

The factor f is included here to account for the effect of networks that lie in a plane of symmetry. We have

$$f = \begin{cases} 1/2 & \text{if the network lies in a P-O-S} \\ 1 & \text{if the network does not lie in a P-O-S} \end{cases} \qquad (5.G.4)$$

(3) Next we include the source term in AICK. This is done by using the arrays SGH and IISMAG which give the nonzero entries and the corresponding locations of the expression

$$\lfloor 1,\ \xi_k',\ \eta_k' \rfloor \quad [\text{SPSPL}_k^S]\ [B_k^S] \qquad (5.G.5)$$

The inclusion is performed in a fairly obvious fashion. No special handling of control points lying in a plane of symmetry is required.

(4) When GENAIC is finished with the processing of a control point block, it writes the AICK buffer out to the AIC-MATRIX dataset using as keys the AICK identifiers.

Some final remarks are in order concerning the treatment of closure.

First it should be clear that the volume of I-O on the AIC-MATRIX dataset could be quite high if the "AICK identifiers" changed each time a control point contribution was added in to AICK. In order to minimize the I-O activity, subroutine CONBLK sees to it that the control points are processed in the correct order. If a network's closure conditions involve sums over network columns, then the control points are processed by columns; if the sums are over network rows, the control points are processed by rows.

The second set of remarks we wish to make concerns the generation of closure information for the ROWMAP dataset. This is done as follows. As CONBLK processes the control points of each network, PROCP records the following information relating to each control point that has a closure condition

(1) network identifier
(2) DQG control point index

} keys to the CLOSURE dataset

(3) Updatability type

(4) MAG's AIC row index for the closure condition

(5) The symmetry conditions having a closure condition

Next, when all of the control points in the network have been processed by PROCP, subroutine CONBLK invokes CLSROW to process each closure condition, updating the ROWMAP and ROWMAP-INVERSE datasets by adding the following information:

(1) The coefficients $a_{A,k}$ and $a_{D,k}$ (cf. eqn. 5.G.1) obtained from DQG's CLOSURE dataset

(2) The updatability type of the closure condition

(3) MAG's AIC row index for the closure condition, multiplied by (-1) if ROWMAP's control point has the smallest MAG c.p. index of all c.p.'s affecting the closure condition

(4) The symmetry conditions having a closure condition

APPENDIX 5-H

ALTERNATE PROBLEM FORMULATIONS

As promised at the end of section (5.1.2), we now take up the discussion of alternate integral equation formulations for the problem of incompressible potential flow about a sphere. This appendix will consist of three parts:

(i) a discussion of the modified Morino, doublet alone formulation,

(ii) a discussion of the direct velocity formulation,

(iii) a summary of all four formulations treated in this document, comparing and contrasting the relative merits of each.

5-H.1 Formulation 3, The Modified Morino Method

[$\Phi = 0$ inside B]

This formulation, which leads to an identically zero source strength, chooses the flow interior to B to be total stagnation. That is, we assume that interior to B the total velocity $\vec{V} = 0$:

$$\vec{V} = \vec{U}_\infty + \nabla \phi = 0 \quad \text{for points} \quad \vec{p} \in \text{int}(B) \tag{5.H.1}$$

Now since $\vec{U}_\infty$ satisfies the relation,

$$\vec{U}_\infty = \nabla_p \Phi_\infty , \tag{5.H.2}$$

$$\Phi_\infty = \vec{U}_\infty \cdot \vec{p} \tag{5.H.3}$$

we find that the total potential Φ defined by

$$\Phi = \Phi_\infty + \phi \tag{5.H.4}$$

must satisfy

$$\nabla \Phi = \vec{V} = 0 \quad \text{for points} \quad \vec{p} \in \text{int}(B) \tag{5.H.5}$$

As a trivial consequence of equation (5.H.5), we find that Φ must be constant interior to B. The actual value of the constant is immaterial so that we take it to be zero:

$$\Phi \equiv 0 \quad \text{for points} \quad \vec{p} \in \text{int}(B) \tag{5.H.6}$$

Having determined that Φ is identically zero inside B, we can now obtain some relations for the source and doublet strength on ∂B, the boundary of B. We have for σ:

$$\sigma = (\partial\phi/\partial n)_{\vec{p}_+} - (\partial\phi/\partial n)_{\vec{p}_-}$$

$$= (\partial\phi/\partial n)_{\vec{p}_+} + (\partial \Phi_\infty /\partial n)_{\vec{p}_+} - (\partial\phi/\partial n)_{\vec{p}_-} - (\partial \Phi_\infty/\partial n)_{\vec{p}_-}$$

since Φ_∞ is continuous across ∂B. Using equation (5.H.4) this becomes,

$$\sigma = (\partial \Phi/\partial n)_{\vec{p}_+} - (\partial \Phi/\partial n)_{\vec{p}_-} .$$

Now, since $\Phi \equiv 0$ interior to B, $\nabla \Phi \equiv 0$ there as well and we obtain

$$\sigma = (\partial \Phi/\partial n)_{p_+} = (\hat{n}.\ \nabla \Phi)_{\vec{p}_+} = (\hat{n}.V)_{\vec{p}_+} = 0 \qquad (5.H.7)$$

Here we have used equation (5.H.5) together with the boundary condition $(\hat{n}.\vec{V})_{\vec{p}_+} = 0$ (see the figure on p. 5.4). The relation for the doublet strength that we require is derived:

$$\mu = (\phi)_{\vec{p}_+} - (\phi)_{\vec{p}_-} \qquad \text{(by definition)}$$

$$= (\Phi_\infty)_{\vec{p}_+} + (\phi)_{\vec{p}_+} - (\Phi_\infty)_{\vec{p}_-} - (\phi)_{\vec{p}_-} \qquad \text{(since } \Phi_\infty \text{ is continuous)}$$

$$= (\Phi)_{\vec{p}_+} - (\Phi)_{\vec{p}_-} \qquad \text{(by 5.H.4)}$$

$$= (\Phi)_{\vec{p}_+} + (\Phi)_{\vec{p}_-} \qquad \text{(since } (\Phi)_{\vec{p}_-} \equiv 0)$$

$$= 2\,\Phi_\infty + 2(\phi(\vec{p}))_{avg} \qquad (5.H.8)$$

We now invoke the representation theorem (cf. eqn. (5.1.6)) with σ set equal to zero to obtain

$$\phi(\vec{p}) = \frac{1}{4\pi} \iint_{\partial B} \mu\ (\hat{n}.\nabla_q \psi)dS_q \qquad (5.H.9)$$

Denoting the average value, above and below ∂B, of the integral appearing on the right with the subscript "avg," we write

$$(\phi(\vec{p}))_{avg} = \frac{1}{2}(\phi(\vec{p}+) + \phi(\vec{p}-)) = \left(\frac{1}{4\pi} \iint_{\partial B} \mu(\hat{n}.\nabla_q \psi)dS_q\right)_{avg} \qquad (5.H.10)$$

Substituting this result into (5.H.8) and rearranging slightly we obtain the required integral equation,

$$\frac{1}{2}\mu(\vec{p}) - \left(\frac{1}{4\pi} \iint_{\partial B} \mu(\hat{n}.\nabla_q \psi)dS_q\right)_{avg} = \Phi_\infty \qquad (5.H.11)$$

The integral equation (5.H.11) is readily transformed into an AIC equation of the usual form (cf. equation (5.1.13)) by assuming a finite dimensional representation for μ (cf. eqn. (5.1.11b))

$$\mu(\vec{q}) = \sum_{J=1}^{N} m_J(\vec{q})\ \lambda_J$$

and then evaluating equation (5.H.11) at N collocation points (i.e. control points), $\vec{p}_I$. One obtains

$$\sum_{J=1}^{N} A_{IJ}\,\lambda_J = b_I \tag{5.H.12}$$

where

$$A_{IJ} = \frac{1}{2}\, m_J(\vec{p}_I) - \frac{1}{4\pi}\left\{\iint_{\partial B} m_J(\vec{q})\ (n.\nabla_q \psi)\, dS_q\right\}_{\vec{p}_{I,avg}} \tag{5.H.13}$$

$$b_I = \Phi_\infty(\vec{p}_I) \tag{5.H.14}$$

5-H.2 Formulation 4, The Direct Velocity Formulation

Here, as in the Morino formulation, we assume that the perturbation potential is identically zero inside B. In contrast with the Morino formulation, however, we leave the source distribution as an unknown and explicitly impose the impermeable surface boundary condition, $(\vec{V}.\hat{n})_{\vec{p}+} = 0$.

To see how this works, observe that the condition

$$\phi = 0 \quad \text{inside B} \tag{5.H.15}$$

implies as well that μ satisfies

$$\mu = (\phi)_+ - (\phi)_- = (\phi)_+ + (\phi)_- \tag{5.H.16}$$

Writing the representation formula for ϕ in the form

$$(\phi(\vec{p}))_{avg} = -\frac{1}{4\pi}\left(\iint_{\partial B} \sigma\ \psi\ dS_q\right)_{\vec{p},avg} + \frac{1}{4\pi}\left(\iint_{\partial B} \mu\ \hat{n}.\nabla_q\ dS_q\right)_{\vec{p},avg} \tag{5.H.17}$$

We obtain from equations (5.H.16) and (5.H.17) the result

$$\frac{1}{2}\,\mu(\vec{p}) = (\phi(\vec{p}))_{avg} = -\frac{1}{4\pi}\left(\iint_{\partial B} \sigma\ \psi\ dS_q\right)_{\vec{p},avg} + \frac{1}{4\pi}\left(\iint_{\partial B} \mu\ \hat{n}.\nabla_q \psi\, dS_q\right)_{\vec{p},avg} \tag{5.H.18}$$

Note that this integral equation is essentially identical to the integral equation (5.1.8) obtained for the Morino formulation.

Our next task will be to obtain a second integral equation by combining the impermeable surface condition with the fundamental representation formula for $\vec{v}(\vec{p})$, equation (B.3.9) of the theory document. Assuming that doublet

matching is performed, we may discard the line vortex term and rewrite equation (B.3.9) in the following form, appropriate to the present context:

$$(\vec{v}(\vec{p}))_{avg} = \left[\frac{1}{4\pi} \iint_{\partial B} \sigma(\vec{q})\ \nabla_q \psi\ dS_q + \frac{1}{4\pi} \iint_{\partial B} (\hat{n} \times \nabla\mu) \times \nabla_q \psi\ dS_q \right]_{\vec{p},avg} \tag{5.H.19}$$

Next, we write the impermeable surface boundary condition in the form

$$\begin{aligned} 0 = (\vec{V}.\hat{n})_{\vec{p}_+} &= \vec{U}_\infty \cdot \hat{n}(\vec{p}) + \hat{n} \cdot \vec{v}(\vec{p}_+) \\ &= \vec{U}_\infty \cdot \hat{n}(\vec{p}) + \frac{1}{2}\hat{n}(\vec{p}).[(\vec{v}(\vec{p}_+) - \vec{v}(\vec{p}_-)] + \frac{1}{2}\hat{n}(\vec{p}).[\vec{v}(\vec{p}_+) + \vec{v}(\vec{p}_-)] \\ &= \vec{U}_\infty .\hat{n} + \frac{1}{2}\sigma + \hat{n}(\vec{p}).(\vec{v}(\vec{p}))_{avg} \end{aligned} \tag{5.H.20}$$

Substituting (5.H.19) into this then yields our second integral equation:

$$\frac{1}{2}\sigma(\vec{p}) + \hat{n}(\vec{p}) \cdot \left[\frac{1}{4\pi} \iint_{\partial B} \sigma\ \nabla_q\ dS_q + \frac{1}{4\pi} \iint_{\partial B} (\hat{n} \times \nabla\mu) \times \nabla_q \psi\ dS_q \right]_{\vec{p},avg} = -\vec{U}_\infty \cdot \hat{n}(\vec{p}) \tag{5.H.21}$$

This integral equation* in combination with (5.H.18) constitutes a pair of dual integral equations for the unknown functions $\sigma(\vec{q})$ and $\mu(\vec{q})$.

We write out pair of equations (5.H.18) and (5.H.21) in the form:

$$\frac{1}{2}\mu + \frac{1}{4\pi} \iint_{\partial B} \sigma\ \psi\ dS_q - \frac{1}{4\pi} \iint_{\partial B} \mu(\vec{q})\ \hat{n}(\vec{q}).\ \nabla_q\ \psi\ dS_q = 0$$

$$\frac{1}{2}\sigma + \frac{1}{4\pi} \iint_{\partial B} \sigma(\vec{q})\ \hat{n}(\vec{p}) \cdot \nabla_q \psi\ dS_q + \frac{1}{4\pi} \iint_{\partial B} (\hat{n} \times \nabla\mu) \times \nabla\ \psi\ dS_q \cdot \hat{n}(\vec{p}) = -\vec{U}_\infty \cdot \hat{n}(\vec{p}) \tag{5.H.22}$$

---

* It is worthwhile pointing out that equation (5.H.21), which can be written in shorthand form as

$$\frac{1}{2}\sigma + \hat{n}(\vec{p}) \cdot (\vec{v}(\vec{p}))_{avg} = -\vec{U}_\infty \cdot \hat{n}(\vec{p}) \qquad \text{(A)}$$

could be replaced by

$$2\hat{n}(\vec{p}) \cdot (\vec{v}(\vec{p}))_{avg} = -U_\infty \cdot \hat{n}(\vec{p}) \qquad \text{(B)}$$

the difference between these two equations being the quantity

$$\hat{n} \cdot (\vec{v}(\vec{p}))_{avg} - \frac{1}{2}\sigma = \hat{n} \cdot \vec{v}(\vec{p}_-)$$

Equation (A) is preferred over equation (B), however, because it has better numerical properties.

In writing down equation (5.H.22) in this form we have adopted the convention that all integrals are to be interpreted as average value integrals, above and below ∂B. It can be shown that this constitutes an Hermitian system of integral equations having the form

$$\sum_{\beta=1}^{2} M_{\alpha\beta}\, u_\beta(\vec{p}) + \sum_{\beta=1}^{2} \iint_{\partial B} K_{\alpha\beta}(\vec{p},\vec{q})\, u_\beta(\vec{q})\, dS_q = b_\alpha(p) \tag{5.H.23}$$

with

$$M_{\alpha\beta} = M_{\beta\alpha}\,, \qquad K_{\alpha\beta}(\vec{p},\vec{q}) = K_{\beta\alpha}(\vec{q},\vec{p}) \tag{5.H.24}$$

Of the four formulations studied in this section, only this one leads to an Hermitian integral equation or system of integral equations.

5-H.3 Summary of Four Formulations

In this section we write down in one place the integral equations on ∂B obtained from each of the four methods of formulation that have been studied. In what follows, all integral expressions are to be interpretted as being averaged, above and below ∂B

[1] Morino's formulation

$$\frac{1}{2}\mu(\vec{p}) - \frac{1}{4\pi}\iint_{\partial B} \mu\, \hat{n}(\vec{q}) \cdot \nabla_q \psi\, dS_q = -\frac{1}{4\pi}\iint_{\partial B} \sigma\, \psi\, dS_q \tag{5.H.25}$$

where $\sigma(\vec{q}) = -\vec{U}_\infty \cdot \hat{n}(\vec{q})$

[2] Hess' formulation ($\mu \equiv 0$)

$$\frac{1}{2}\sigma(\vec{p}) + \frac{1}{4\pi}\hat{n}(\vec{p}) \cdot \iint_{\partial B} \sigma(\vec{q})\, \nabla_q \psi\, dS_q = -\vec{U}_\infty \cdot \hat{n}(\vec{p}) \tag{5.H.26}$$

[3] Modified Morino formulation ($\sigma \equiv 0$)

$$\frac{1}{2}\mu(\vec{p}) - \frac{1}{4\pi}\iint_{\partial B} \mu\, \hat{n}(\vec{q}) \cdot \nabla_q \psi\, dS_q = \Phi_\infty(\vec{p}) \tag{5.H.27}$$

[4] Direct Velocity formulation

$$\frac{1}{2}\mu + \frac{1}{4\pi}\iint_{\partial B} \sigma\, \psi\, dS_q - \frac{1}{4\pi}\iint_{\partial B} \mu\, \hat{n}(\vec{q}) \cdot \nabla_q \psi\, dS_q = 0 \tag{5.H.28a}$$

$$\frac{1}{2}\sigma + \frac{1}{4\pi}\iint_{\partial B} \sigma(\vec{q})\, \hat{n}(\vec{p}) \cdot \nabla_q \psi\, dS_q + \frac{1}{4\pi}\iint_{\partial B} (\hat{n} \times \nabla\mu) \times \nabla\psi\, dS_q \cdot \hat{n}(\vec{p}) = -\vec{U}_\infty \cdot \hat{n}(\vec{p}) \tag{5.H.28b}$$

First we must emphasize that σ (resp. μ) for one formulation is not the same as σ (resp. μ) for another formulation. Other appropriate remarks are:

(1) Both of the Morino formulations [1] and [3] lead to the same AIC matrix. (Compare equation (5.1.14) to (5.H.13) to observe this.) However, the modified Morino formulation is less expensive because it requires no source influence coefficients.

(2) The general folklore relating to these various formulations suggests that in terms of solution quality, [4] is best, [1] is second best, [3] third best and [2] is the least accurate. In particular, Hess' formulation, because it does not allow doublet wake sheets, cannot be used for lifting configurations.

(3) In terms of cost, method [2] is least expensive followed by [3], [1] and [4] in that order.

(4) All of these integral equations are Fredholm integral equations of the second kind. This is all to the good, since Fredholm integral equations of the first kind are notorious for leading to poorly conditioned matrix equations. Second kind integral equations on the other hand usually lead to fairly well conditioned matrix equations. This is especially true when, as in PAN AIR, the integral kernels are singular.

(5) The direct velocity formulation [4] leads to an Hermitian system of integral equations. This fact helps explain the excellent numerical properties of this method of formulation.

6.0 REAL MATRIX SOLVER (RMS) MODULE

6.1 INTRODUCTION

The RMS module solves large systems of linear equations. Most of the subroutines used by RMS reside on the PAN AIR library (PALIB).

(1) Decompose the square matrix [AIC] into [L] and [R] plus pivoting terms [P].

$$[AIC] = [L]\ ([P][R])$$

where,

[AIC] is the Aerodynamic Influence Coefficient matrix
[L] is the lower-triangle matrix
[R] is the upper-triangle matrix
[P] represents pivot terms

(2) Perform the forward and backward substitutions with [L] and [R] plus the right-hand-side constraints matrix, [RHS], to find the lambda solutions [λ].

The matrix [AIC] is generated in the PAN AIR module MAG and the matrix [RHS] is generated in the module RHS. The primary function of the RMS module is to perform the decomposition of the AIC matrices. The forward and backward substitution will be done in the latter stages of RHS. The subroutine that RMS uses to decompose [AIC] is called RMSD and resides on PALIB. For efficiency's sake, the subroutines operate on matrices in blocked (partitioned) format. To interface with the other modules of PAN AIR, the RMS module performs the following steps:

(1) Block the partition of AIC matrix corresponding to the unknown singularity parameters (it was generated in rows by the MAG module).

(2) Decompose the "unknown" AIC matrix into [L], [R], and [P].

(3) RMS, during an update run, uses a restart capability of RMSD by setting the restart row and column submatrix indices to point to the first submatrix not previously decomposed. If [AIC] consists of k by k submatrices, then the restart index must be set to k+1.

The module also generates a permanent RMS data base for the storage of the decomposed matrix. The [L] and [R] matrices replace the [AIC] matrix while the pivot terms [P] are stored on the RMS data base.

A temporary data base, RMST, is used for internal data storage.

6.2 RMS OVERVIEW

6.2.1 Purpose of RMS

The Real Matrix Solver (RMS) is a module of the PAN AIR System. RMS will block and decompose the AIC matrices that were generated in the MAG module.

6.2.2 RMS Output Data

RMS prints only error diagnostics information when SDMS errors and matrix singularity errors occur. Appendix 6-E lists the possible RMS error diagnostics.

6.2.3 Data Base Interfaces

RMS reads input data from data bases created by MEC and MAG. The MEC data base provides data base names, account numbers, data base status, date of creation, and other similar information. The MAK data base created by the MAG module contains the unknown AIC matrices. RMS creates a single data base during its execution. The decomposed AIC matrices are always stored on the RMS data base and later used in the RHS module. Also, the RMS data base is used as input during a RMS update run. In addition, RMS uses a temporary data base RMST. The RMS and RMST data base master definition is described in Appendix 6-D.

6.3 MODULE DESCRIPTION

The main overlays and their subroutines are briefly summarized in this section. The RMS functional decomposition is shown in Appendix 6-B and a chart of the subroutine tree structure is presented in Appendix 6-A.

6.3.1 Overall Structure

Figure 6.1 illustrates the top level structure of the module RMS. The functional decomposition of all overlays and their subprograms is given in Appendix 6-B.

6.3.2 Overlay Descriptions

A summary description of each overlay of the module RMS is given in the following paragraphs.

6.3.2.1 RMS Overlay (0,0)

The top level overlay (0,0) of RMS initializes the data bases and controls access to the 3 primary overlays.

6.3.2.2 RMSINT Overlay (1,0)

Checks the status of data bases and initializes the COMMON BLOCK variables used in the matrix process.

6.3.1.3 BLOCKA Overlay (2,0)

The RMS subroutines from RMSLIB operate on the AIC matrices in blocked partitioned format and all matrices must be stored as a direct data set on RMS data base (Figure 6.2). The BLOCKA module will block, by rows, the AIC matrices which were generated in the MAG module. In order to accomplish this, there must be a scratch array buffer of length at least long enough to hold any 3 submatrices required for the RMS decomposition process. The size of the buffer is preset to 30,000. If larger blocks are desired, the user must redimension blank common and data load the new length into labeled common /BCLEN/.

The blocking of the AIC matrices is accomplished by calling subroutine BLOCK for each non-updatable and/or updatable partition.

6.3.2.4 DCOMPO Overlay (3,0)

The subroutine RMSD from RMSLIB is called to decompose the non-updatable and/or updatable partitions of the AIC matrices which were generated in BLOCKA. In order to accomplish this there must be in core, a scratch array at least long enough to hold any 3 submatrices produced from BLOCKA. The size of the buffer area is preset to 30,000 [decimal], the same as in BLOCKA.

Subroutine RMSD is called to decompose the AIC matrix into the product [L], [R], and generates the pivoting information matrix [P]. The [L] and [R] matrices are the lower-triangular and the upper-triangular matrices which replace the blocked AIC submatrices on the RMS data base after the decomposition process.

The following process takes place during an AIC update run after the decomposed AIC matrices are saved on the data base in an original run through the PAN AIR system.

Original Run

- o Decomposition of the AIC blocked submatrices.
- o The [L], [R], and [P] submatrices are saved on the RMS data base.

Update Run

- o Retrieve the [L], [R], and [P] submatrices from the original run from the RMS data base.
- o Restart the initial problem by setting the index key to point to the first submatrix not previously decomposed.
- o The [L], [R], and [P] submatrices produced from the original run or update run will be saved on the RMS data base.

6.3.3 RMS Data Base

The Master Definitions of the data bases RMS and RMST are given in Appendix 6-D.

6.3.4 RMS Interfaces

Figure 6.3 summarizes the internal and external data interfaces between the RMS module and the RMS, MEC, and MAK data bases. The RMS data base is used by the module RHS. RMS uses the RMS data base as input of the decomposed AIC matrices created in a previous run during an Update Run.

6.3.5 Data Flow

Figures 6.4 through 6.6 illustrate the data flow for the major sections of RMS. Detailed data flow information can be found by consulting these figures, Appendix 6-C (Data Base Communication Chart), and the glossaries of the program/subroutines which are of interest.

6.4 LOWER LEVEL FUNCTIONS

6.4.1 Functional Decomposition

See Appendix 6-B for a description of the RMS decomposition.

6.4.2 Subroutine Descriptions

6.4.2.1 BLKGEN

Submodule BLKGEN generates RMS blocking information for the non-updatable and/or updatable partitions of the AIC matrices.

The core storage in Blank Common area available for use as scratch array for temporary data storage is calculated using REQFL from PALIB. The blocking information parameters are calculated for the updatable and/or non-updatable AIC submatrix block sizes and are saved on the RMS data base.

During an AIC update run, the submatrix block and decomposition restart location is calculated. The blocking information from the previous run stored on RMS data base is read and the blocking information is calculated for the additional row/column updates. Finally, all blocking information is saved on the RMS data base.

6.4.2.2 BLOCK

Submodule BLOCK converts a matrix stored as rows into a matrix stored by blocks. The basic processing steps in blocking matrices is done in a two step operation. In the first step, the matrix is read by rows, and written on a temporary data base by rectangular blocks in subroutine RECBLK. In the second step, enough rectangular blocks are read to form larger blocked submatrices and written out on the RMS data base for later processing. These steps are repeated until all rows of the AIC matrices are processed.

6.4.2.3 RECBLK

Submodule RECBLK produces enough rectangular blocks to produce larger blocks by reading (a row at a time) the AIC matrices from the MAK data base created in the MAG module. These rectangular blocks are merged to form a larger blocked rectangular matrix. Finally, the blocked rectangular matrix is sub-divided into smaller column blocks and written out onto the RMS temporary data base. This process is repeated until all rows of the AIC matrix are processed.

6.4.2.4 SQBLK

Submodule SQBLK reads enough rectangular blocks from the RMS temporary data base produced in RECBLK to form square block submatrices. The blocked submatrices are written onto the RMS data base for later processing in module DCOMPO. This process is repeated until all rectangular submatrices are read and processed.

6.4.2.5 RMSD

Decomposes the blocked AIC submatrices formed in SQBLK and stored on the RMS data base into [L], [R], and [P].

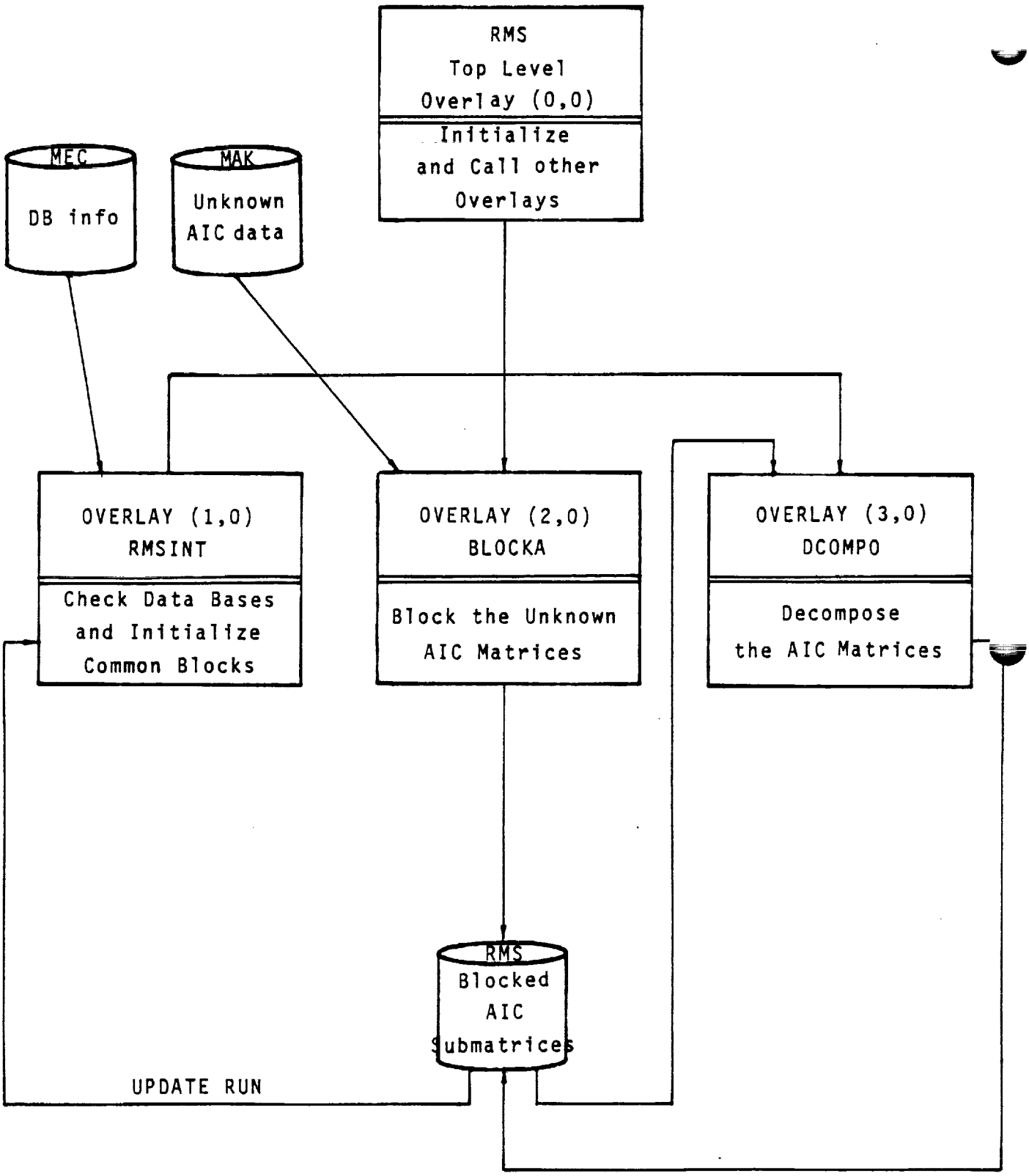


Figure 6.1 - Top Level Structure of RMS

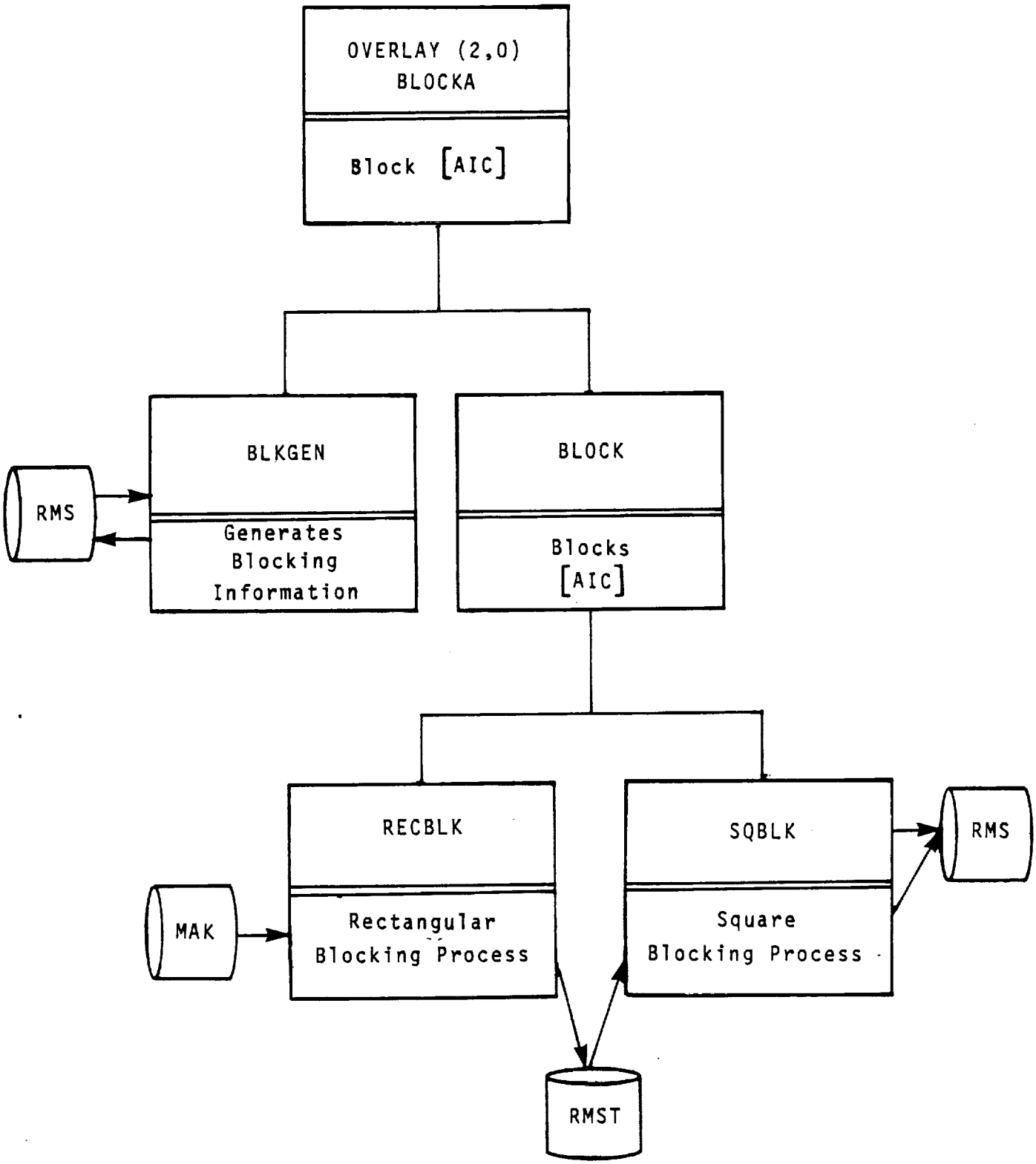
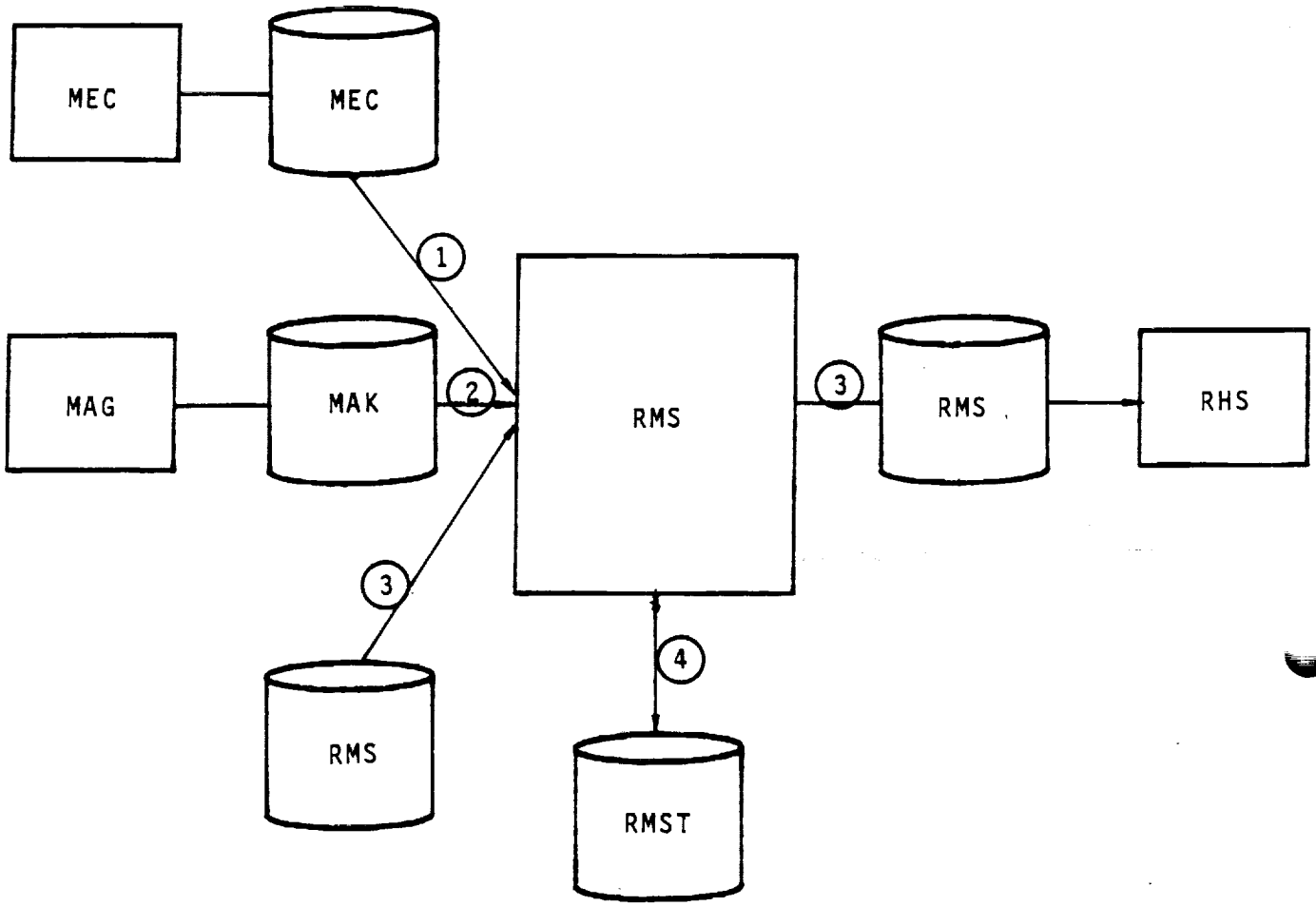


Figure 6.2 - Structure of (2,0) Overlay of RMS



1 Data Base Directory Information
2 Unknown AIC Matrix by Rows
3 Decomposed AIC Matrix
4 Temporary Storage for Blocks of Submatrices

Figure 6.3 - Data Base Relationships

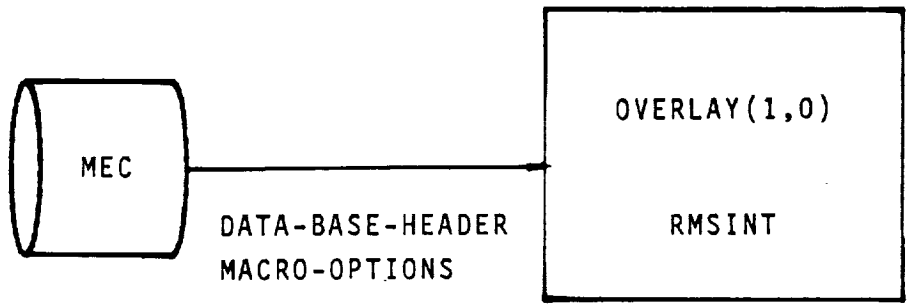


Figure 6.4 - Structure and Data Flow of (1,0) Overlay

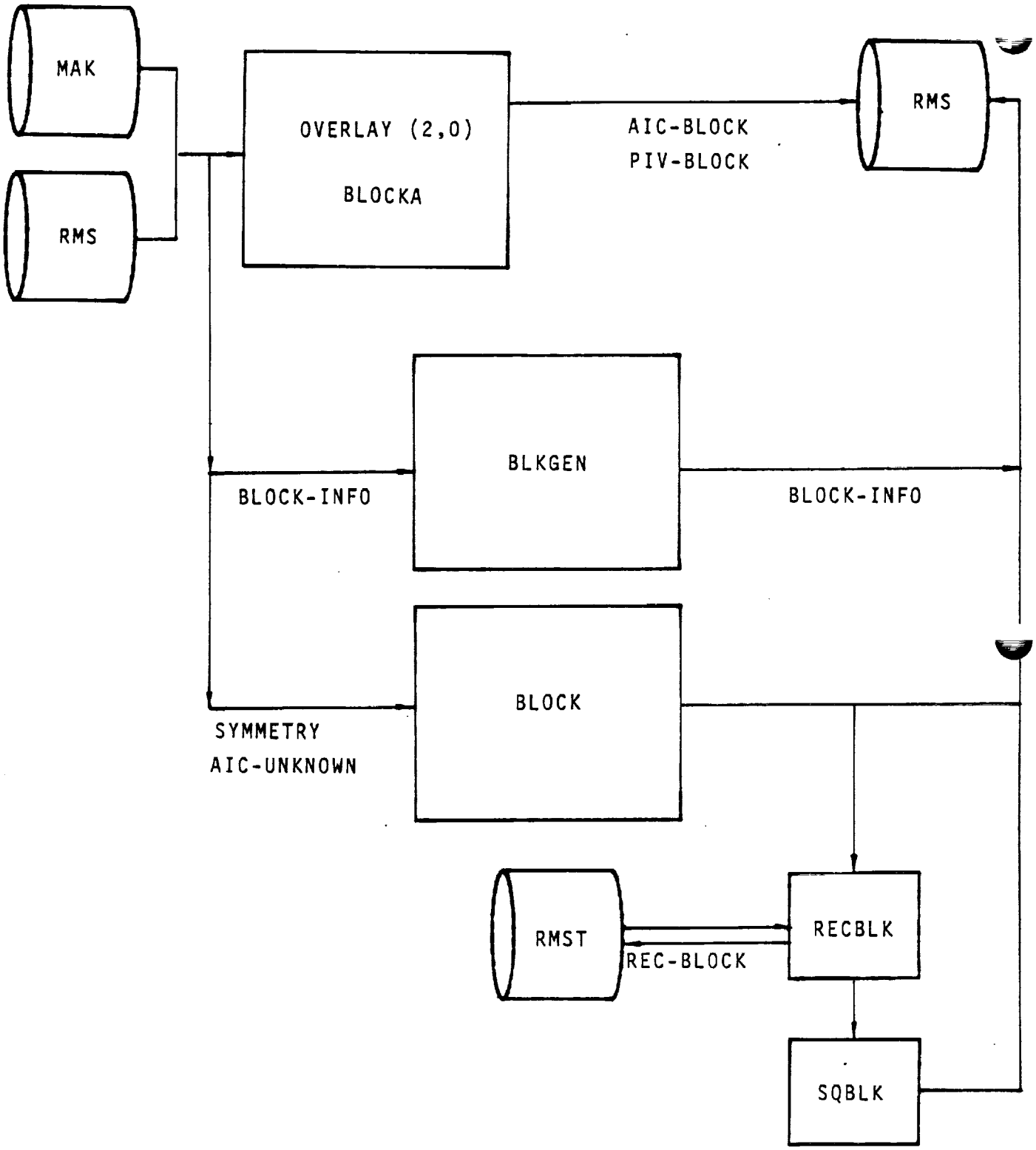


Figure 6.5 - Structure and Data Flow of (2,0) Overlay

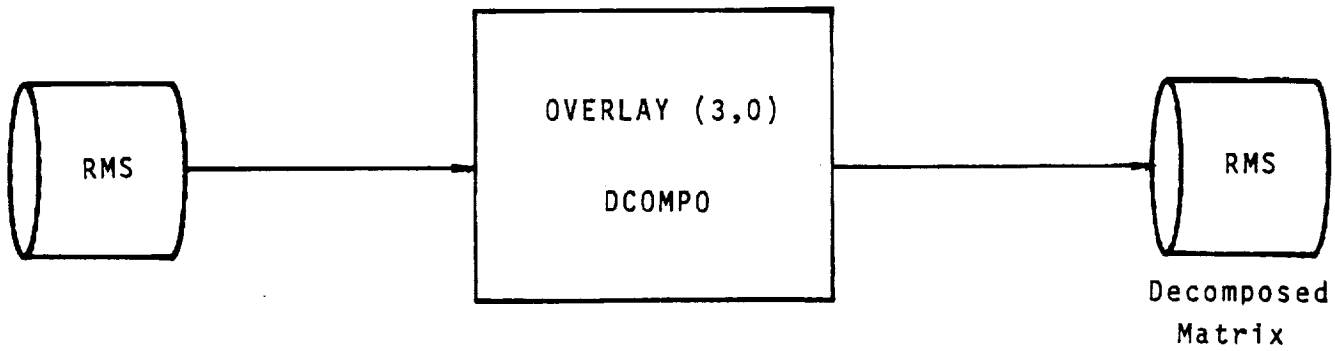


Figure 6.6 - Structure and Data Flow of (3,0) Overlay

APPENDIX 6-A TREE STRUCTURE

The tree structure diagram of the RMS module has been deleted from this document. It is, however, available on the installation tape.

APPENDIX 6-B
RMS FUNCTIONAL DECOMPOSITION

A Initialization of RMS Execution [Overlay (0,0)] (RMS)
  A Initialize Printout (PRGBEG)
  B Initialization of SDMS for Execution (ISDMS)

  C Check Data Base(s) and Initialize Common Blocks [Overlay (1,0)] (RMSINT)
    A Open Data Base MEC (DBOPEN)
    B Get Data-Base-Header (ESGET)
    C Check Data-Base-Location of MAK, RMS, and RMST (CHPADB)
    D Close Data Base MEC (DBCLOS)

B Blocks the Unknown AIC Matrices [Overlay (2,0)] (BLOCKA)
  A Open Data Bases MAK and RMS (PAOPEN)
  B Generates RMS Blocking Information for Non-Updatable and Updatable Partitions (BLKGEN)
    A Determines Core Storage Available (REQFL)
    B Get Previous AIC Blocking Information from the RMS Data Base (ESGET)
    C Determine AIC Block Sizes
      A Calculates Maximum Block Size for [AIC]
      B Determines Size of All [AIC] Non-Update Blocks
      C Determines Size of All [AIC] Update Blocks
    D Calculate Restart Location and Total Problem Size
    E Generate [AIC] Blocking Information on RMS Data Base
      A Place [AIC] Blocking Information on RMS Data Base (ESPUT)
      B Replace [AIC] Blocking Information on RMS Data Base (ESREP)
    F Gets Previous [AIC] Blocking Information From the RMS Data Base (ESGET)
    G Determines RHS Block Sizes (For RHS Module)
      A Calculates Maximum Block Size for RHS Blocks (For RHS Module)
      B Error Test (For RHS Module)
      C Determines Size of RHS Blocks (For RHS Module)
  C Performs Blocking of AIC Matrices
    A Blocks Non-Updatable (Upper-Left) Portion of AIC (BLOCK)
    B Blocks Updatable (Upper-Right) Portion of [AIC] (BLOCK)
    C Blocks Updatable (Lower-Left) Portion of [AIC] (BLOCK)
    D Blocks Updatable (Lower-Right Portion of [AIC] (BLOCK)
    A Open Temporary Data Base RMST (PAOPEN)
    B Form Enough Rectangular Blocks to Form Submatrices (RECBLK)
      A Computes Number of Rectangles Per Block
      B Distributes the Number of Rows in Current Block
      C Gets a Matrix Row (ESGET)
      D Writes Rectangular Row Partition onto Data Base RMST (ESPUTR)
    C Form Row of Square Blocks From the Rectangular Blocks (SQBLK)

A Read Blocked Column of Rectangular Matrices
B Write Submatrix onto Data Base
D Close and Return Data Base RMST (PACLOS)

D Close Data Base MAK and RMS (PACLOS)

C Decomposes the AIC Matrices [Overlay (3,0)] (DECOMPO)

A Open Data Base RMS (PAOPEN)
B Decomposes the Non-Updatable (Upper-Left) Portion of [AIC] (RMSD)
C Decomposes the Updatable Portions of [AIC] (RMSD)
D Write Data Base Header Data Set (ESPUT)
E Close Data Base RMS (PACLOS)

D Terminal Program Execution and Printout (PRGEND)

APPENDIX 6-C
DATA BASE COMMUNICATION CHARTS

The Data Base Communications Chart is presented in three forms. Each form is alphabetized by columns, from left to right. The first form has a column order of Data Base, Dataset Name, Map Name, Common Block, and Program/Subroutine. The second form has a column order of Data Base, Map Name, Dataset Name, Common Block, and Program/Subroutine. The third form has a column order of Common Block, Data Base, Map Name, Dataset Name, and Program/Subroutine. Thus a person can get a cross reference on a data element by knowing either the Dataset Name, Map Name or Common Block name.

FIRST FORM

DATA BASE	DATA SET-NAME	MAP NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
MEC	DATA-BASE-HEADER	MECHDR	/RUNIDS/	RMSINT
MEC	MACRO-OPTIONS	UPDATE	/BLKINF/	RMSINT
MAK	AIC-UNKNOWN	AICUNK	/BLKINF/	BLOCKA
MAK	SYMMETRY	SYM	/BLKINF/	BLOCKA
RMS	AIC-BLOCK	AICBLK	Dynamic	DCOMPO
RMS	BLOCK-INFO	BLIN	/BLKINF/	BLKGEN
RMS	PIV-MAT	PIVMAT	Dynamic*	DCOMPO
RMST	REC-BLOCK	RECMAT	Dynamic*	BLOCK
RMST	REC-BLOCK	RECMAT2	Dynamic*	BLOCK

SECOND-FORM

DATA BASE	MAP-NAME	DATA-SET-NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
MEC	MECHDR	DATA-BASE-HEADER	/RUNIDS/	RMSINT
MEC	UPDATE	MACRO-OPTIONS	/BLKINF/	RMSINT
MAK	AICUNK	AIC-UNKNOWN	/BLKINF/	BLOCKA
MAK	SYM	SYMMETRY	/BLKINF/	BLOCKA
RMS	AICBLK	AIC-BLOCK	Dynamic	DCOMPO
RMS	BLIN	BLOCK-INFO	/BLKINF/	BLKGEN
RMS	PIVMAT	PIV-MAT	Dynamic*	DCOMPO
RMST	RECMAT	REC-BLOCK	Dynamic*	BLOCK
RMST	RECMAT2	REC-BLOCK	Dynamic*	BLOCK

THIRD-FORM

COMMON BLOCK	DATA BASE	MAP-NAME	DATA SET NAME	PROGRAM/ SUBROUTINE
/RUNIDS/	MEC	MECHDR	DATA-BASE-HEADER	RMSINT
/BLKINF/	MEC	UPDATE	MACRO-OPTIONS	RMSINT
/BLKINF/	MAK	AICUNK	AIC-UNKNOWN	BLOCKA
/BLKINF/	MAL	SYM	SYMMETRY	BLOCKS
Dynamic	RMS	AICBLK	AIC-BLOCK	DCOMPO
/BLKINF/	RMS	BLIN	BLOCK-INFO	BLKGEN
Dynamic*	RMS	PIVMAT	PIV-MAT	DCOMPO
Dynamic*	RSMT	RECMAT	REC-BLOCK	BLOCK
Dynamic*	RMST	RECMAT2	REC-BLOCK	BLOCK

* Dynamic mapping of the dataset is used, thus requiring no common block storage. See Section 13 of this document.

APPENDIX 6-D MASTER DEFINITION

The data base master definition listing of the DIP module has been deleted from this document. It is produced from the PAN AIR tape during installation.

APPENDIX 6-E
RMS ERROR MESSAGES

The following is a list of the RMS error diagnostic messages by overlay.

- o RMS - OVERLAY (0,0)
  ERROR IN MODULE ERRMOD
  where, ERRMOD is RMSINT, BLOCKA, or DCOMPO.

- o RMSINT - OVERLAY (1,0)
  ******* NERR = ____ IS COUNT OF FATAL ERRORS FROM RMSINT.

- o BLOCKA - OVERLAY (2,0)
  ******** ERROR IN MODULE ERRMOD
  where, ERRMOD is BLKGEN, BLOCK(UL), BLOCK(UR), BLOCK(LL), or BLOCK(LR)

  Note: UL designates Upper Left partition of AIC,
  UR designates Upper Right partition of AIC,
  LL designates Lower Left partition of AIC,
  LR designates Lower Right partition of AIC.

- o BLKGEN
  ******** FATAL ERROR IN BLKGEN ********
  CORE NOT LARGE ENOUGH TO HOLD ONE COLUMN OF RIGHT-HAND-SIDE MATRIX.

- o DCOMPO - OVERLAY (3,0)
  1. ******** ABNORMAL ERROR EXIT FOR NON-UPDATE DECOMPOSITION RUN FOR AIC MATRIX IMAGE.
  2. ******** ABNORMAL ERROR EXIT FOR UPDATE DECOMPOSITION RUN FOR AIC MATRIX IMAGE.
  3. ERROR CODE IS IERR
     WHERE, IERR = -1 SCRATCH ARRAY IS NOT LONG ENOUGH TO PROCESS SOLUTION.
     = -2 IMPLIES FAILURE IN SDMS.
     = K IMPLIES SOLUTION APPEARS SINGULAR AT ROW K.

  All other RMS error diagnostics are written from the RMSLIB subprograms during execution.

7.0 RIGHT HAND SIDE (RHS) MODULE

7.1 INTRODUCTION

The primary function of the RHS module is to find the unknown singularity parameters λ_{un} of the linear system

$$[AIC] \begin{bmatrix} \lambda_n \\ \lambda_{un} \end{bmatrix} = [RHS]$$

from a decomposed AIC matrix, [AIC], the known singularity parameter, matrix, $[\lambda_n]$, and the right-hand-side constraints matrix, [RHS]. Related tasks include the generation of the right-hand-side matrix, [RHS], and the forward/backward substitition using the decomposed AIC matrix.

The module also generates a permanent data base, RHS, for the minimal data generator module MDG. This data base contains all the known and unknown singularity parameters λ.

Two temporary data bases, RHSX and RMST, are used for internal data storage.

7.2 RHS OVERVIEW

7.2.1 Purpose of RHS

The PAN AIR system was created to find the pressures, forces and velocities of a flow around an arbitrary body. The problem is reduced to finding the solution of a large linear system of equations:

$$[AIC]\ [\lambda] = [RHS]$$

where [AIC] is a matrix of aerodynamic influence coefficients, [RHS] is a matrix of right-hand-side constraints and $[\lambda]$ is the singularity vector which may contain known and unknown parameters.

The RHS module constructs the [RHS] matrix and uses forward and backward substitution in conjunction with a decomposed [AIC] matrix from the RHS module to find the unknown portion of the singularity vector $[\lambda]$. More detail of the process is given in Paragraph 7.3.

7.2.2 RHS Input/Output Data

7.2.2.1 Output

Very little printed output is given by the module RHS. What does occur is error diagnostic information if an error is encountered. A message is printed at the end of execution indicating the number of errors, if any.

7.2.3 Data Base Interfaces

The module RHS creates one data base, RHS, for MDG and two temporary data bases, RHSX and RMST, for internal storage. The data bases from DIP, DQG, MEC, RMS and MAG are used for required input data. Figure 7.1 illustrates the relationships between RHS and all used data bases.

The input data from MEC, DIP, DQG, MAG and RMS are used by RHS. The MEC data base furnishes names, account numbers, status and related information for all the data bases. The DIP data base gives user option and constraint data. The DQG data base furnished constraint and global data. Finally, the RMS data base has available the decomposed AIC matrix.

The two temporary data bases, RHSX and RMST, are used for internal storage. The data base RHSX stores the DIP constraint data while RMST contains the blocking information for the AIC matrix.

The RHS permanent data base contains several sets of data. The symmetry information, the RHS right-hand-side matrix, the on-set flow vector for each control point, the inverse value of the partition of the AIC matrix corresponding to the known singularities and the values of all singularities in row or column format.

7.3 MODULE DESCRIPTION

The linear system of the PAN AIR problem can be placed into the parition form

$$\begin{bmatrix} AIC_{nn} & 0 \\ AIC_{un} & AIC_{uu} \end{bmatrix} \begin{bmatrix} \lambda_n \\ \lambda_u \end{bmatrix} = \begin{bmatrix} RHS_n \\ RHS_u \end{bmatrix}$$

where $[AIC_{nn}]$ is a diagonal submatrix corresponding to the known singularities $[\lambda_n]$, $[AIC_{un}\ AIC_{uu}]$ is the partition of the aerodynamic influence coefficient matrix corresponding to the unknown singularites $[\lambda_{un}]$, $[RHS_n]$ is the submatrix of right-hand-side constraints corresponding to the known singularities and $[RHS_u]$ is the submatrix of the right-hand-side constraints corresponding to the unknown singularities. With minimal effort, the linear system can be put into the form:

$$[AIC_{nn}]\,[\lambda_n] = [RHS_n] \tag{7.3.1}$$

$$[AIC_{uu}]\,[\lambda_u] = [RHS_u] - [AIC_{un}]\,[\lambda_n] \tag{7.3.2}$$

It is this latter form with which RHS is concerned.

7.3.1 Overall Structure

Figure 7.2 illustrates the structure of the module RHS. The top level overlay (0,0) RHSPRG initializes the data bases and controls access to the other five overlays. The functional decomposition of all the overlays and their subprograms is given in Appendix 7-B. See also Appendix L of the Theory Document (Reference 1) for theoretical details.

7.3.2 Overlay Descriptions

7.3.2.1 OPENDB Overlay (1,0)

The program initializes the data bases used by RHS and creates the data base maps which set up correspondence between the data base variables and the program variables. Figure 7.3 illustrates the overall program execution of OPENDB.

The DIP constraints data is also read from its data base and converted to a form usable by RHS. The restructured data is placed on the temporary data base RHSX.

7.3.2.2 PBCAD Overlay (2,0)

The primary purpose of the second overlay is to compute the right-hand-side constraint matrix, $[RHS_u]$ (the right-hand-side of equation (5.7.26) of the Theory Document, Reference 1) corresponding to unknown singularities λu. In this process, the onset flows are calculated. The data is then saved on the RHS permanent data base. Figure 7.4 illustrates the overall data transfer.

7.3.2.3 RHSC Overlay (3,0)

The third overlay serves as a controller for three sub-overlays, KNOWN, TRANSF and KWNCTR. Figure 7.5 shows the overall execution process of the three sub-overlays.

7.3.2.4 KNOWN Overlay (3,1)

The calculation of the right-hand-side constraint matrix $[RHS_n]$ (which constitute the extreme right-most terms in equation (5.7.17) of the Theory Document (Reference 1)) for the known singularity parameters occurs in overlay (3,1). It also finds the known singularities from the known diagonal matrix, [DI] occurring in the same equation. The data is stored on the RHS data base.

7.3.2.5 TRANSF Overlay (3,2)

The second level overlay (3,2) symmetrizes and transforms the known singularities from row to column format. The data is stored on the RHS data base.

7.3.2.6 KWNCTR Overlay (3,3)

The second level overlay (3,3) finds the right-hand-side constraint matrix $[RHS_u]$, for the unknown singularities and then subtracts off the contribution $[AIC_{un}]$ due to the known singularities. That is, it computes the complete right-hand-side of equation (5.7.17) of the Theory Document (Reference 1).

7.3.2.7 RHSOLV Overlay (4,0)

The fourth overlay solves the system of equations (7.3.2) for the unknown singularities λn using forward and backward substitution on the decomposed matrix $[AIC_u]$. Figure 7.6 indicates the interaction between the program and the affected data bases.

7.3.2.8 RHSD Overlay (5,0)

The last overlay gives a report on execution, either successful or unsuccessful. Figure 7.7 indicates the interaction between the program and data base.

7.3.3 RHS Data Bases

The master definitions of the data bases RHS and RHSX are given in Appendix 7-D.

7.3.4 RHS Interfaces

Figure 7.1 summarizes the internal and external data interfaces between modules and their data bases.

The interrelationship (functional decomposition) between all overlays and subroutines is given in Appendix 7-B.

7.3.4.1 Internal Interfaces

The RHSX temporary data base is used as a scratch file for the right-hand-sides constraint and singularity data.

7.3.4.2 External Interfaces

The input data is obtained from the MEC, DIP, DQG, MAK and RMS data bases. The MEC data base furnishes data base names, accounts, status and other related information for all data bases. The DIP data base introduces the user options affecting RHS and the constraint data used for constructing the right-hand-side constraints. The DQG data base supplies control point data. The MAK data base provides the unknown singularity portion of the aerodynamic influence coefficient matrix. Finally, the RMS data base provides the decomposed known singularity portion of the aerodynamic influence coefficient matrix.

The output data, consisting of the right-hand-side constraints and singularity data, is stored on the RHS data base. This data is used by the MDG module.

7.3.5 Data Flow

The execution of RHS has already been discussed to some extent in Paragraph 7.1 and Figure 7.1. A somewhat different picture to the exact sequence is given in Figures 7.2 through 7.6. With the information given previously, the figures should be self explanatory.

7.4 LOWER LEVEL FUNCTIONS

7.4.1 Functional Decomposition

See Appendix 7-B for a description of the RHS decomposition.

7.4.2 Subroutine Descriptions

BLKGEN
Generates RHS blocking information for non-updateable and updateable partitions.

BLOCK
Converts a matrix stored by rows into a matrix stored by blocks.

CENTER
Arranges the looping sequence for processing center control points. See Appendix 7-E.

CNSTR
Initializes the constraints conditions of all the different RHS solutions for general boundary conditions.

CNSTRT
Initializes the constraints conditions of all the different RHS solutions for boundary conditions which define a known singularity parameter.

COLMNZ
Converts RHS solution matrix LAMBDAS from blocks to columns of non-updateable and updateable singularity parameters.

CPINFO
Initializes the control point data, including boundary conditions, image locations and normals.

DIPCTS
The constraints data in DIP is in the user input format. RHS requires the constraint conditions for each control point. This subroutine pre-processes the DIP data and stores the results on RHSX data base.

DIPFUL
Reads the constraint data, that is, the coefficients of equation (H.3.25) of the Theory Document (Reference 1), from the DIP data base into a holding array and orders them according to the control points. See Appendix 7-D.

EDGE
Arranges the looping sequence for processing edge control points. See Appendix 7-D.

MAPS
Defines all the required SDMS maps.

NETSP
Gets netwk-spec and special-points datasets from DQG.

NEWCST
Transfers new constraints data from a holding array to the output array. See Appendix 7-D.

PRNCST
Adds the most recently computed constraints data to the RHSX data set. See Appendix 7-D.

PREP
Defines constraints dataset, the closure boundary condition edge and the constraints that are required from DIP. See Appendix 7-D.

RECBLK
Reads a matrix stored by rows and writes it in rectangular blocks.

RHSVEC
Generates the onset flow vector and computes the right-hand-side from the user-input parameters. It is also used in MAG.C for generating right-hand-side vectors for known singularities. RHSVEC computes the left-hand-side of equation (H.3.25) of the Theory Document (Reference 1).

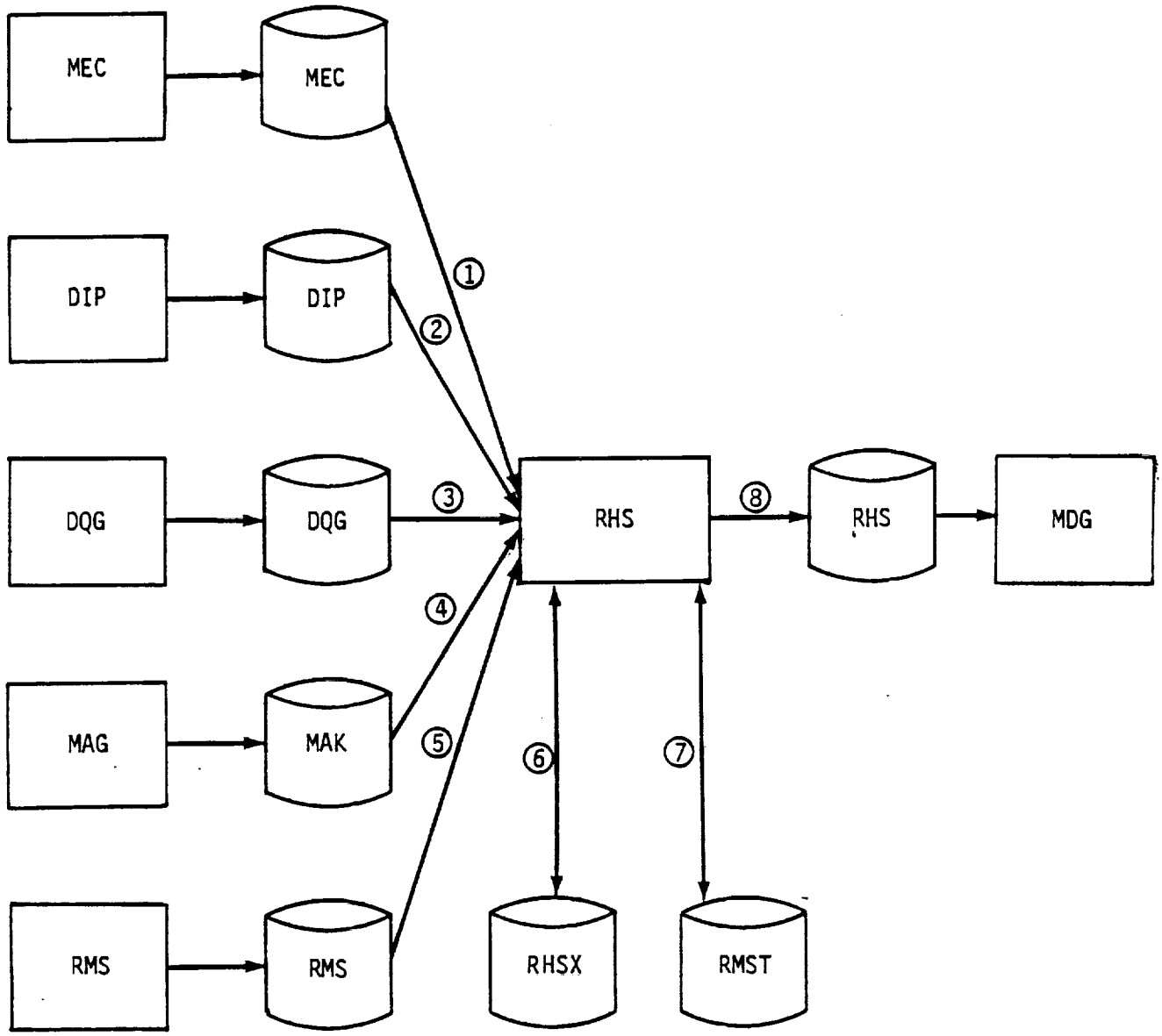
SLNDAT
Initializes /NUM/ and /SYM/ and generates solution-data.

SNGINF
Initializes the information on the known singualrity, its image locations and normals.

SPECIL
Arranges the looping sequence for processing special control points. See Appendix 7-D.

SQBLK
Reads a matrix stored by rectangular blocks and writes it by square blocks.

SYMKWN
Symmetrizes known singularities and forms partial columns of the symmetrized singularities.



① - Data Base Directory Information
② - Input Options and Constraints
③ - Global Data and Constraints
④ - Partition of AIC Matrix for Unknown Singularities
⑤ - Decomposed AIC Matrix
⑥ - DIP Constraints Data
⑦ - Blocking Information for the AIC Matrix
⑧ - The Known and Unknown Sigularities for the MDG Module

Figure 7.1 - Data Base Relationships

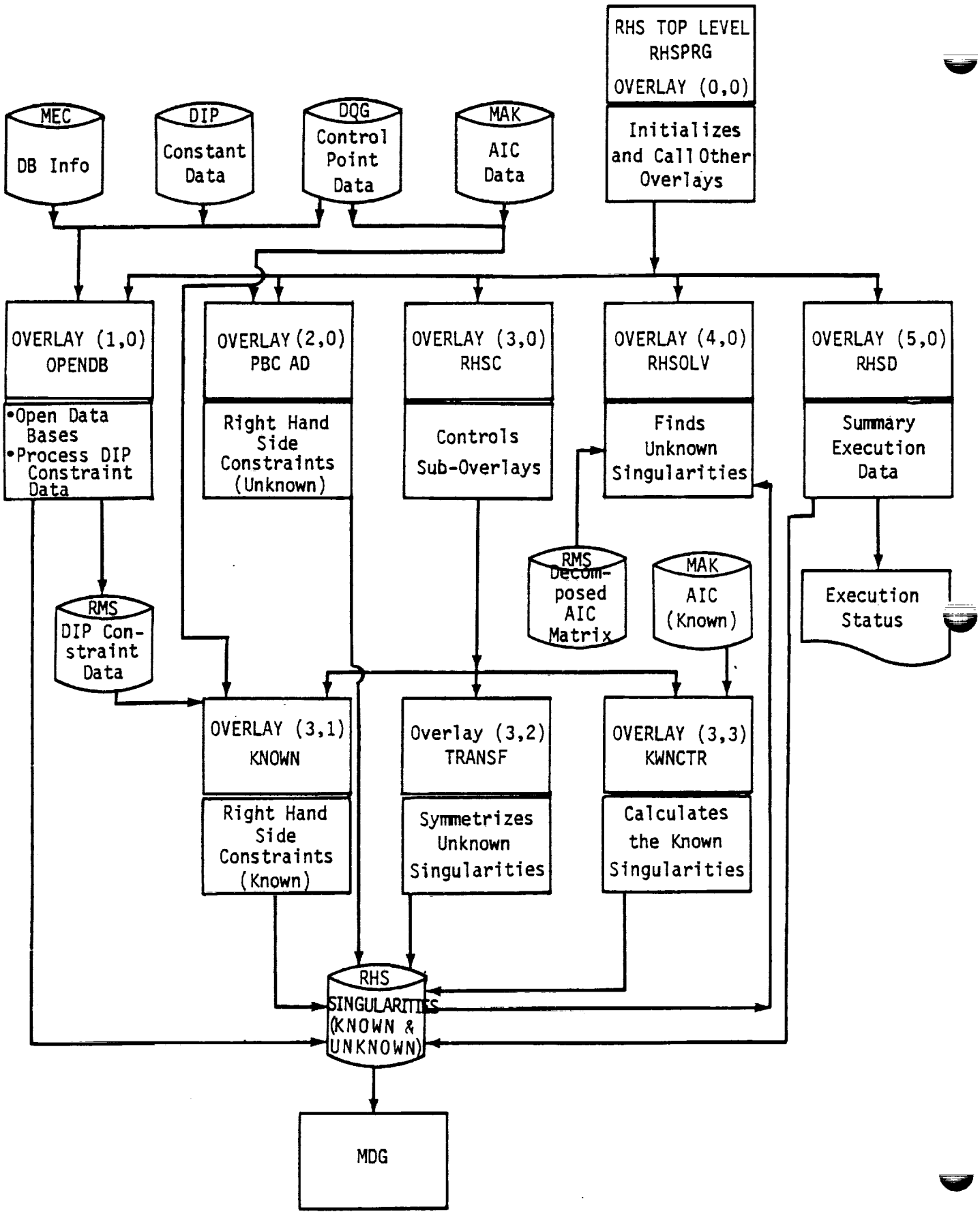


Figure 7.2 - Structure and Data Flow of RHS

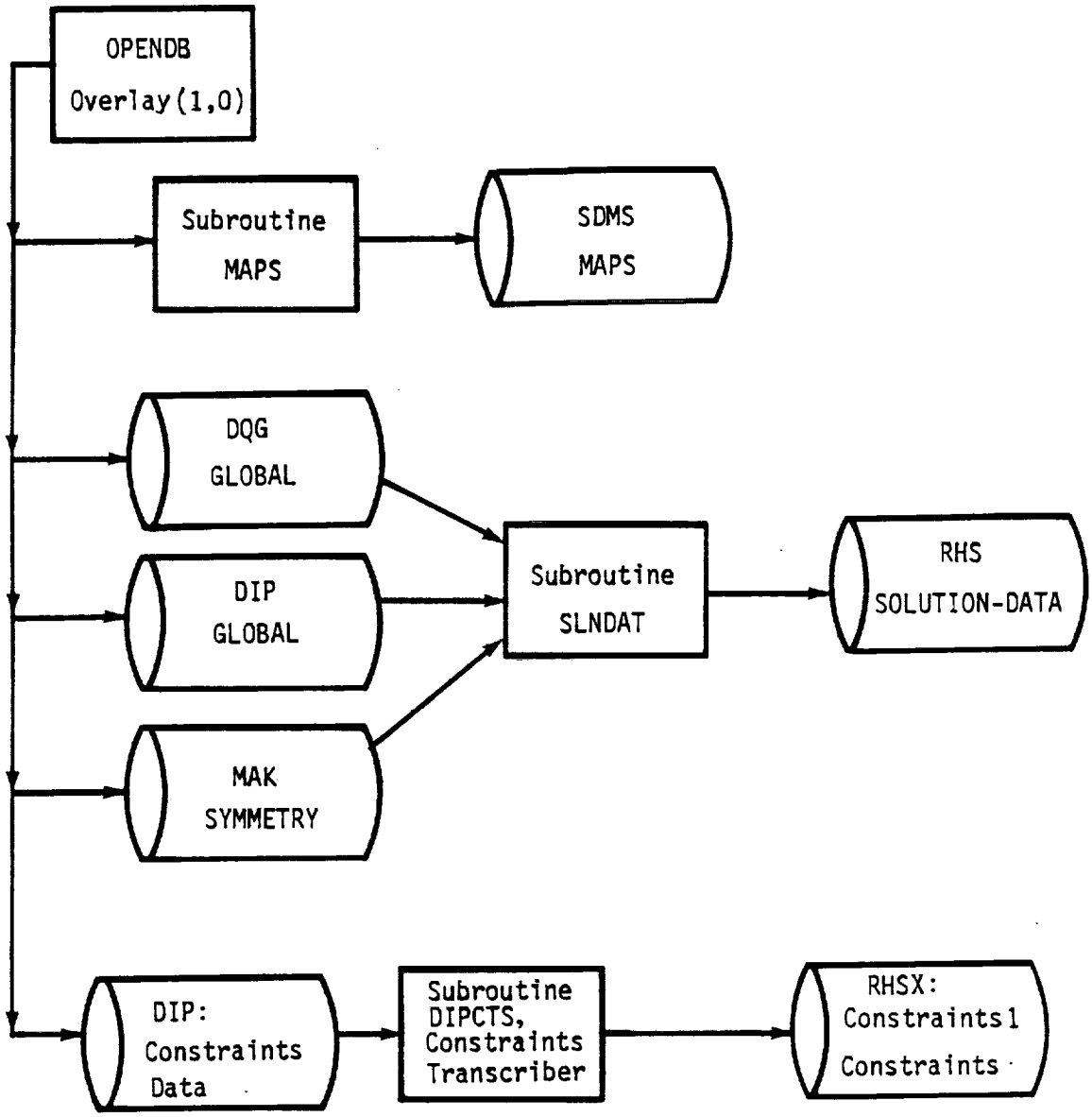


Figure 7.3 - Structure and Data Flow for Overlay (1,0)

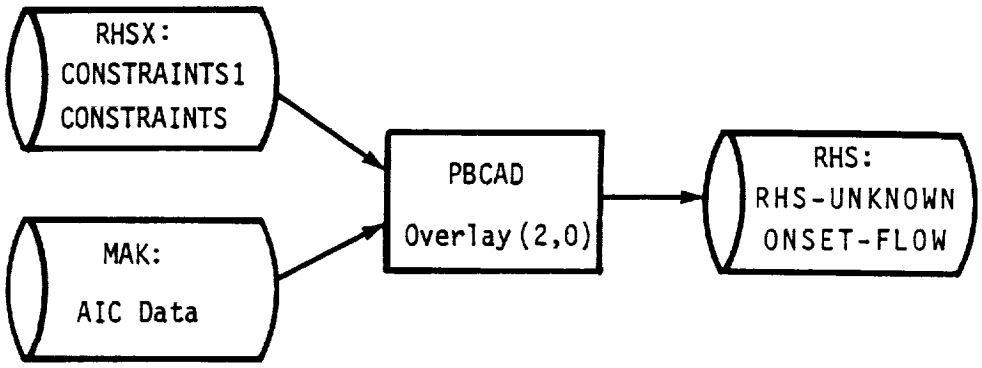


Figure 7.4 - Structure and Data Flow for Overlay (2,0)

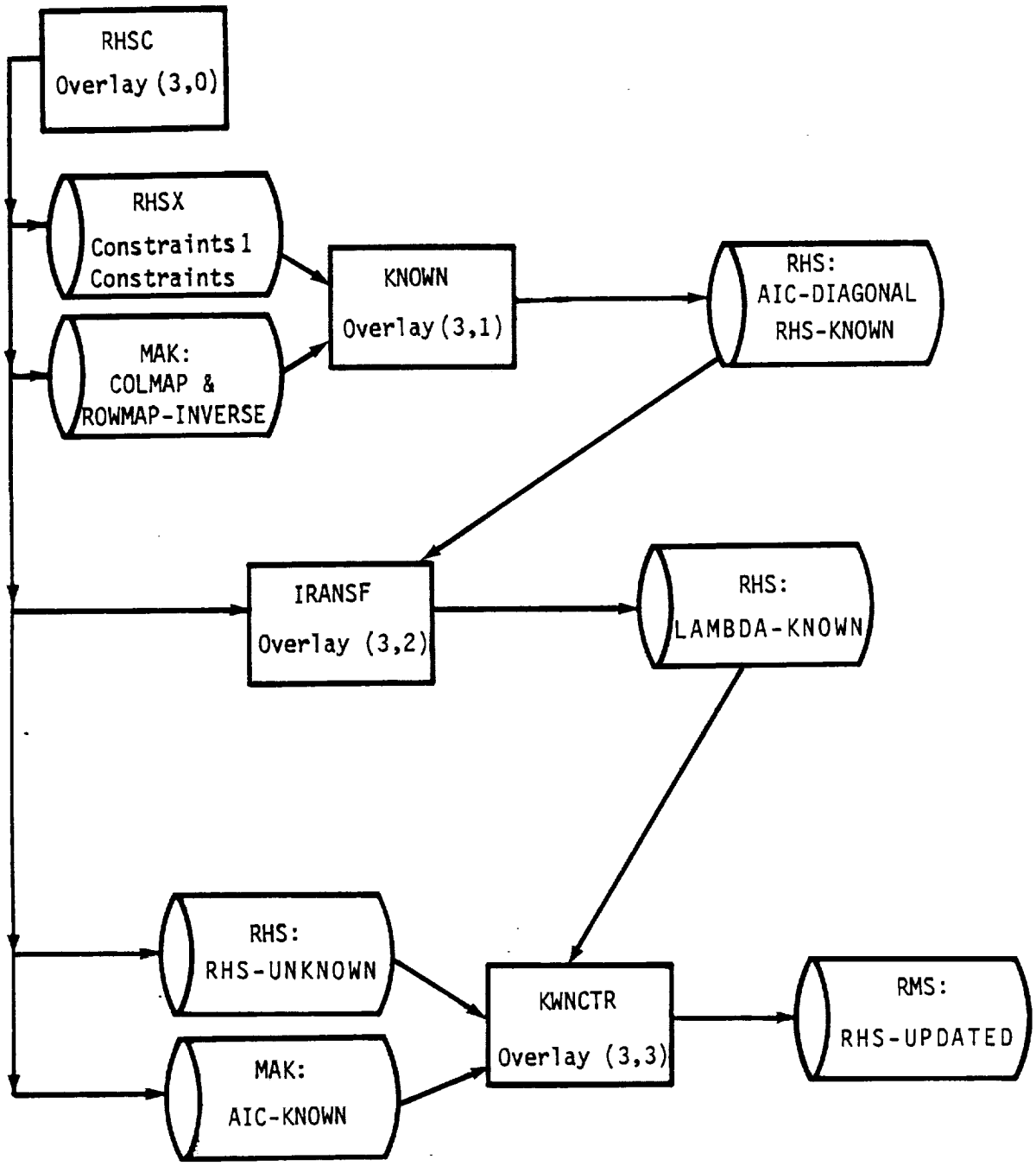


Figure 7.5 - Structure and Data Flow for Overlay (3,0)

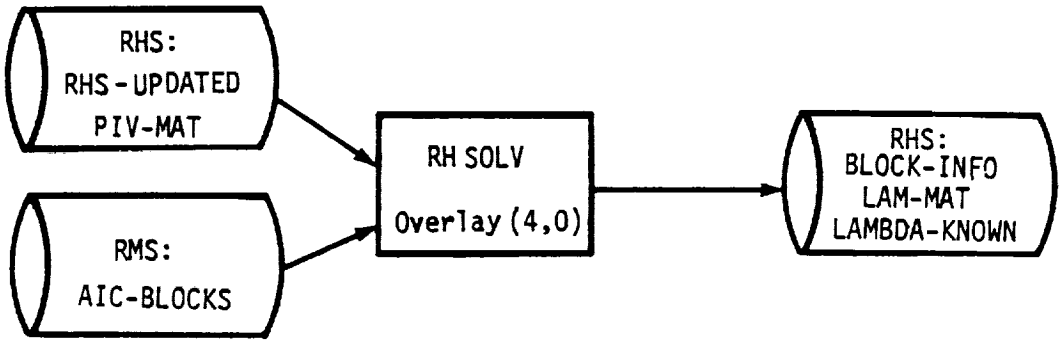


Figure 7.6 - Structure and Data Flow for Overlay (4,0)

Figure 7.7 - Structure and Data Flow for Overlay (5,0)

APPENDIX 7-A TREE STRUCTURE OF RHS

The tree structure diagram of the RMS module has been deleted from this document. It is, however, available on the installation tape.

APPENDIX 7-B FUNCTIONAL DECOMPOSITION

The functional decomposition of the RHS module is presented here. The decomposition labels given in the order of their execution and therefore may not be alphabetic.

Open data bases and initialize (OPENDB) [OVERLAY (1,0)]

A. Check the condition of the data bases and generate SDMS maps (MAPS)
   A. Open MEC data base and define SDMS maps for MEC (DBOPEN/DSMAP/SVMAP/ENDMAP)
   B. Initialize /RUNIDS/ (ESGET)
   C. Check data base conditions (CHPADB)
   D. Close MEC data base (DBCLOS)
   E. Open DQG data base and define SDMS maps (PAOPEN/DSMAP/SVMAP/DVMAP/ENDMAP)
   F. Open MAK data base and define SDMS maps (PAOPEN/DSMAP/SVMAP/DVMAP/ENDMAP)
   G. Open RHS and RHSX data bases and define SDMS maps (PAOPEN/DSMAP/SVMAP/DVMAP/ENDMAP)
   H. Open DIP data base and define SDMS maps (PAOPEN/DSMAP/SVMAP/DVMAP/ENDMAP)

B. Initialize global data and generate solution data set (SLNDAT)
   A. Get global data set (ESGET)
   B. Initialize /NUM/ and /SYM/ (ESGET/CAB)
   C. Get solution data set from DIP (ESGET)
   D. Generate entry to RHS solution data (ESPOR)

C. Pre-process constraints data (DIPCTS)
   A. Get network data from DIP (ESGET)
   B. Get network-spec data set from DIP (ESGET)
   C. Initialize constraints value
   D. Write to constraints -1 data set (ESPOR)
   E. Write to constraints -2 data set (ESPOR)
   F. Get DQG network-spec and special points data sets (ESGET)
   G. Get networks bulk data control data set from DIP (ESGET)
   H. Process full constraints transcriber (DIPFUL)
      A. Pre-process DIP constraints (PREP)
         A. Write to constraints -1 data set (ESPOR)
         B. Initialize edge condition (ZERO)
         C. Define a point on the edge
         D. Get DQG boundary condition data set (ESGET)
         E. Initialize edge condition to closure
         F. Initialize mapping information (ZERO)
         G. Define mapping information (ZERO)
      B. Define term
         C. Initialize smear condition
         D. Define column number
         E. Get DIP constraints data (ESGET)
         F. Process center control points (CENTER)
            A. Convert to five lattice index
            B. Define solution - ID
            C. Process constraints (PRCNST)
               A. Define boundary condition number
               B. Zero the constraints array (ZERO)
               C. Get the current constraints -2 data set (ESGET)

D. Get new constraints data (NEWCST)
    A. Define length of constraint term
    B. Define constraints value
    C. Set constraints array
    D. Get tangent vector from DQG boundary condition data set (ESGET)
E. Replace the current constraints -2 data set (ESPOR)

G. Process edge control points (EDGE)
    A. Define solution - ID
    B. Define the number of control points involved
    C. Define fine lattice indices
    D. Process constraints (PRCNST)

H. Process special control points (SPECIL)
    A. Define lattice indices
    B. Define solution - ID
    C. Process constraints (PRCNST)
        A. Define boundary condition number
        B. Zero the constraints ARRAY
        C. Get the current constraints -2 data set (ESGET)
        D. Get new constaints data (NEWCST)
            A. Define length of constraint term
            B. Define constraints value
            C. Set constraints array
            D. Get tangent vector from DQG boundary condition data set (ESGET)
        E. Replace the current constraints -2 data set (ESPOR)

G. Process edge control points (EDGE)
    A. Define solution - ID
    B. Define the number of control points involved
    C. Define fine lattice indices
    D. Process constraints (PRCNST)

H. Process special control points (SPECIL)
    A. Define lattice indices
    B. Define solution - ID
    C. Process constraints (PRCNST)

Produce boundary condition analysis data corresponding to unknown singularities (PBCAD) [Overlay (2,0)]

A. Get velocity vector information and number of right-hand-sides (ESGET)

B. Define partition type and start and end indices of control points in the partition

C. Initialize control point and boundary condition information (CPINFO)
   A. Get row map data set (ESGET)
   B. Get boundary condition spec data set (ESGET)
   C. Calculate image control point and normal vector (IMAGE)
   E. Get B-pointer data set

D. Zero the RHS solution (ZERO)

E. Initialize constraints (CNSTR)
   A. Get constraints 1 data set (ESGET)
   B. Define new lattice indices
   C. Set lattice indices as default
   D. Get constraints 2 data set (ESGET)

F. Initialize constriants

G. Compute right-hand-side vector (RHSVEC)
   A. Calculate onset flow
   B. Calculate right-hand-side
   C. Set right-hand-side to zero

H. Store local onset flow (ESPOR)

J. Write to RHS-unknown data set (ESPOR)

Solve for known singularities (KNOWN) [Overlay (3,1)]

A. Initialize
   A. Get velocity vector information and number of RHS (ESGET)
   B. Start VARDIM (STARTR)
   C. Set up blank common storage for known-AIC diagonal (INITIR)

B. Define partition type and the start and end indices of known singularities in the partition

C. Initialize singularity and boundary condition information (SNGINF)
   A. Get colmap data set (ESGET)
   B. Get singularity-spec data set and initialize B-pointer (ESGET)
   C. Convert panel indices to fine grid lattice indices
   D. Get boundary-condition-spec data set (ESGET)
   E. Calculate image normal and coordiates (IMAGE)

D. Initialize constraint information (CNSTRT)
   A. Get constraints -1 data set (ESGET)
   B. Define new lattice indices
   C. Set lattice indices as default
   D. Get constraints -2 data set (ESGET)

E. Initialize the constraint conditions (ZERO)

F. Compute right-hand-side vector (RHSVEC)
   A. Calculate onset flow
   B. Calculate right-hand-side
   C. Set right-hand-side to zero

H. Generate entry to RHS-KNOWN data set (ESPOR)

I. Solve for known singularities (CAD)

J. Generate entry to SING-KNOWN data set (ESPOR)

K. Accumulate to AIC-diagonal

L. Enter into AIC-diagonal data set (ESPOR)

M. Delete blank common storage (DELETR)

Transform rows of known singularities into columns and symmetrize (TRANSF) [Overlay (3,2)]

A. Start VARDIM (STARTR)

B. Find out the core memory available and the number of rows that can be fit in core (REQFL)

C. Set up blank common storage (INITIR)

D. Define partition type

E. Symmetrize the known singularities and form partial columns (SYMKWN)
   A. Initialize partition information
   B. Zero the matrix (ZERO)
   C. Read in a row of SING-KNOWN data set (ESGET)
   D. Increment-counter
   E. Transfer solution to the symmetrized matrix (XFERA)
   F. Symmetrize the known singularities (CAPDB)
   G. Form partial column of known singularities
   H. Generate entry to lambda-part data set (DESPUT)

F. Delete blank common storage (DELETR)

G. Set up blank common for a column of known singularities (INITIR)

H. Initialize partition information and counter

I. Read a partition of lambda-part data set (DESGET)

J. Increment counter

K. Generate entry to lambda-known data set (ESPOR)

L. Delete blank common storage (DELETR)

Update and generate RHS with contribution from known singularities (KWNCTR) [Overlay (3,3)]

A. Define SDMS maps for matrix multiplication (DSMAP/SVMAP/DVMAP/ENDMAP)

B. Start VARDIM (STARTR)

C. Define partition type

D. Define map information

E. Generate known singularitiy contribution (MULTI)

F. Define the row number in each partition

G. Get accumulated contribution to RHS(ESGET)

H. Form the row of known singularity contribution (ESGET)
I. Update RHS solution vector (CAMB)

J. Generate entry to data set (ESPOR)

K. Close RHSX data base (PACLOS)

Solve for right-hand-side (RHSOLV) [Overlay (4,0)]

A. Open RMS and RHS data bases and define maps (PAOPEN/DSMAP/DVMAP/ENDMAP)

B. Read the right-hand-side matrix information (ESGET)

C. Determine block sizes for RHS (BLKGEN)
   A. Determine core storage available (REQFL)
   B. Get previous AIC blocking info from RMS data base (ESGET)
   C. Determine AIC block size
   D. Calculate restart location and total problem size
   E. Place AIC blocking info on RMS data base (ESPUT/ESREP)
   G. Determine RHS block sizes

D. Write blocking information onto RHS data base (ESPOR)

E. Open temporary data base RHSX and define map for blocked RHS (PAOPEN/DSMAP/DVMAP/ENDMAP)

F. Perform the blocking of RHS (BLOCK)
   A. Open temporary data base RMST and define maps (PAOPEN/DSMAP/DVMAP/ENDMAP)
   B. Form rectangular subblocks (RECBLK)
      A. Compute number of rectangles per block
      B. Distribute the number of rows in current block
      C. Get a matrix row (ESGET)
      D. Write rectangular row partition onto data base RMST (ESPOR)
   C. Form row of blocks (SQBLK)
      A. Read blocked column of rectangular matrix (ESGET)
      B. Write submatrix onto data base (DESPUT/DESREP)

G. Solve the system of equations via forward and backward substitution (RMSFB)

H. Convert the blocked solution matrix to columns (COLMNZ)
   A. Read a block and append to array containing column of blocks (DESGET)
   B. Write a column of non-updateable lambdas (ESPUT)
   C. Read a block and append to array (DESGET)
   D. Write a column of updateable lambdas (ESPOR)

I. Close and return data base RHSX (PACLOS)

J. Close RMST data base (PACLOS)

Close data bases (RHSE) [Overlay (5,0)]

A. Set condition parameter of RHS data base to 'COMPLETE', if successful execution and to 'FATAL' otherwise

B. Write 'DATA-BASE-HEADER' dataset (ESPOR)

C. Close DIP, DQG, MAK and RHS data bases (PACLOS)

APPENDIX 7-C RHS DATA BASE COMMUNICATION CHART

The Data Base Communications Chart is presented here in three forms. Each form is alphabetized by columns, from left to right. The first form has a column order of Data Base, Dataset Name, Map Name, Common Block, and Program/Subroutine. The second form has a column order of Data Base, Map Name, Dataset Name, Common Block, and Program/Subroutine. The third form has a column order of Common Block, Data Base, Map Name, Dataset Name, and Program/Subroutine. Thus a person can get a cross reference on a data element by knowing either the Dataset Name, Map Name or Common Block name.

FIRST FORM

DATA BASE	DATASET-NAME	MAP-NAME	COMMON BLOCK*	PROGRAM/ SUBROUTINE
DIP	CLOS-COND	DIPCLS	Dynamic	MAPS
DIP	CLOS-COND	DIPCLS	/FILDIP/	MAPS
DIP	COEFF-GEN-BC	DIPGBC	Dynamic	MAPS
DIP	COEFF-GEN-BC	DIPGBC	/FILDIP/	MAPS
DIP	GLOBAL	NETDATA	Dynamic	MAPS
DIP	GLOBAL	SOLDATA	Dynamic	MAPS
DIP	LOCAL-FLOW	DIPLFW	Dynamic	MAPS
DIP	LOCAL-FLOW	DIPLFW	/FILDIP/	MAPS
DIP	NETWK-BDC	NETWKBD	Dynamic	MAPS
DIP	NETWK-SPEC	NETWKSP	Dynamic	MAPS
DIP	SPEC-FLOW	DIPSFW	Dynamic	MAPS
DIP	SPEC-FLOW	DIPSFW	/FILDIP/	MAPS
DQG	B-POINTER	BPOINT	Dynamic	MAPS
DQG	BNDRY-CONDN-SPEC	BNDRY	Dynamic	MAPS
DQG	BNDRY-CONDN-SPEC	DIPTNG	Dynamic	MAPS
DQG	GLOBAL	GLOBAL	/NUM/	MAPS
DQG	GLOBAL	GLOBAL	/SYM/	MAPS
DQG	GLOBAL	ROTATE	Dynamic	MAPS
DQG	NETWK-SPEC	NETWK	Dynamic	MAPS
DQG	SINGULARITY-SPEC	SNGSPC	Dynamic	MAPS
DQG	SPECIAL-POINTS	SPECPT	Dynamic	MAPS
MAK	AIC-KNOWN	AICKWN	Dynamic	MAPS
MAK	AIC-KNOWN	MATRXA	Dynamic	KWNCTR
MAK	AIC-KNOWN	MATRXA	ICPUP,ISNGUP,IPOS	KWNCTR
MAK	COLMAP	COL1	Dynamic	MAPS
MAK	ROWMAP	ROW1	Dynamic	MAPS
MAK	ROWMAP-INVERSE	ROWIN1	Dynamic	MAPS
MAK	SYMMETRY	SYM1	Dynamic	MAPS
MAK	SYMMETRY	SYM1	/NUM/	MAPS
MAK	SYMMETRY	SYM1	/SYM/	MAPS
MEC	DATA-BASE-HEADER	MECHED	/RUNIDS/	MAPS
RHS	AIC-DIAGONAL	AICDIA	Dynamic	MAPS
RHS	DATA-BASE-HEADER	RHSHED	Dynamic	MAPS
RHS	LAMBDA-KNOWN	LMBKWN	Dynamic	MAPS
RHS	LAMBDA-KNOWN	MATRXB	Dynamic	KWNCTR
RHS	LAMBDA-KNOWN	MATRXB	IROWUP,JPOS	KWNCTR
RHS	LAM-MAT	LAMBLIC	Dynamic	RHSOLV
RHS	LAM-MAT	LAMBLIC	IMAG	RHSOLV
RHS	LAMBDA-UNKNOWN	LAMUNK	IMAG	RHSOLV
RHS	ONSET-FLOW	ONSET	Dynamic	MAPS
RHS	PIV-MAT	PIVMAT	IPOS	RHSOLV
RHS	RHS-UNKNOWN	RHSUNK	Dynamic	MAPS
RHS	RHS-UPDATED	RHSUPT	Dynamic	MAPS
RHS	RHS-KNOWN	RHSKWN	Dynamic	MAPS
RHS	SINE-KNOWN	SNGKWN	Dynamic	MAPS
RHS	SOLUTION-DATA	SOLDAT	Dynamic	MAPS

DATA BASE	DATASET-NAME	MAP-NAME	COMMON BLOCK*	PROGRAM/ SUBROUTINE
RHS	BLOCK-INFO	BLRHS	/BLKINF/	RHSOLV
RHS	BLOCK-INFO	BLRHS	NAICS	RHSOLV
RHS	PIV-MAT	PIVMAT	Dynamic	RHSOLV
RHS	RHS-UNKNOWN	RHSUNKX	Dynamic	RHSOLV
RHS	RHS-UNKNOWN	RHSUNKX	IMAGE,IUP	RHSOLV
RHS	SOLUTION-DATA	RHSIZE	/CPOS/	RHSOLV
RHS	SOLUTION-DATA	RHSIZE	NRHS	RHSOLV
RHSX	CONSTRAINTS1	CNSTR1	Dynamic	MAPS
RHSX	CONSTRAINTS2	CNSTR1	Dynamic	MAPS
RHSX	LAMBDA-PART	LMDPRT	Dynamic	MAPS
RHSX	MATRIX-C	MATRXC	Dynamic	MAPS
RHSX	RHS-MATRIX	RHSBLIC	Dynamic	MAPS
RMS	AIC-BLOCKS	AICBLK	Dynamic	RHSOLV
RMS	AIC-BLOCKS	AICBLK	IPOS	RHSOLV
RMS	BLOCK-INFO	BLIN	F. P.	BLKGEN
RMS	REC-BLOCK	RECMAT	Dynamic	BLOCK
RMS	REC-BLOCK	RECMAT2	Dynamic	BLOCK

SECOND FORM

DATA BASE	MAP-NAME	DATASET-NAME	COMMON BLOCK*	PROGRAM/ SUBROUTINE
DIP	DIPCLS	CLOS-COND	Dynamic	MAPS
DIP	DIPCLS	CLOS-COND	/FILDIP/	MAPS
DIP	DIPGBC	COEFF-GEN-BC	Dynamic	MAPS
DIP	DIPGBC	COEFF-GEN-BC	/FILDIP/	MAPS
DIP	DIPLFW	LOCAL-FLOW	Dynamic	MAPS
DIP	DIPLFW	LOCAL-FLOW	/FILDIP/	MAPS
DIP	DIPSFW	SPEC-FLOW	Dynamic	MAPS
DIP	DIPSFW	SPEC-FLOW	/FILDIP/	MAPS
DIP	NETDATA	GLOBAL	Dynamic	MAPS
DIP	NETWKBD	NETWK-BDC	Dynamic	MAPS
DIP	NETWKSP	NETWK-SPEC	Dynamic	MAPS
DIP	SOLDATA	GLOBAL	Dynamic	MAPS
DQG	BNDRY	BNDRY-CONDN-SPEC	Dynamic	MAPS
DQG	BPOINT	B-POINTER	Dynamic	MAPS
DQG	DIPTNG	BNDRY-CONDN-SPEC	Dynamic	MAPS
DQG	GLOBAL	GLOBAL	/NUM/	MAPS
DQG	GLOBAL	GLOBAL	/SYM/	MAPS
DQG	NETWK	NETWK-SPEC	Dynamic	MAPS
DQG	ROTATE	GLOBAL	Dynamic	MAPS
DQG	SNGSPC	SINGULARITY-SPEC	Dynamic	MAPS
DQG	SPECPT	SPECIAL-POINTS	Dynamic	MAPS
MAK	AICKWN	AIC-KNOWN	Dynamic	MAPS
MAK	COL1	COLMAP	Dynamic	MAPS
MAK	MATRXA	AIC-KNOWN	Dynamic	KWNCTR
MAK	MATRXA	AIC-KNOWN	ICPUP,ISNGUP,IPOS	KWNCTR
MAK	ROWIN1	ROWMAP-INVERSE	Dynamic	MAPS
MAK	ROW1	ROWMAP	Dynamic	MAPS
MAK	SYM1	SYMMETRY	Dynamic	MAPS
MAK	SYM1	SYMMETRY	/NUM/	MAPS
MAK	SYM1	SYMMETRY	/SYM/	MAPS
MEC	MECHED	DATA-BASE-HEADER	/RUNIDS/	MAPS
RHS	AICDIA	AIC-DIAGONAL	Dynamic	MAPS
RHS	BLRHS	BLOCK-INFO	/BLKINF/	RHSOLV
RHS	BLRHS	BLOCK-INFO	NAICS	RHSOLV
RHS	LAMBLIC	LAM-MAT	Dynamic	RHSOLV
RHS	LAMBLIC	LAM-MAT	IMAG	RHSOLV
RHS	LAMUNK	LAMBDA-UNKNOWN	IMAG	RHSOLV
RHS	LMBKWN	LAMBDA-KNOWN	Dynamic	MAPS
RHS	MATRXB	LAMBDA-KNOWN	Dynamic	KWNCTR
RHS	MATRXB	LAMBDA-KNOWN	IROWUP,JPOS	KWNCTR
RHS	ONSET	ONSET-FLOW	Dynamic	MAPS
RHS	PIVMAT	PIV-MAT	Dynamic	RHSOLV
RHS	PIVMAT	PIV-MAT	IPOS	RHSOLV
RHS	RHSHED	DATA-BASE-HEADER	Dynamic	MAPS
RHS	RHSIZE	SOLUTION-DATA	/CPOS/	RHSOLV
RHS	RHSIZE	SOLUTION-DATA	NRHS	RHSOLV

DATA BASE	MAP-NAME	DATASET-NAME	COMMON BLOCK*	PROGRAM/ SUBROUTINE
RHS	RHSKWN	RHS-KNOWN	Dynamic	MAPS
RHS	RHSUNK	RHS-UNKNOWN	Dynamic	MAPS
RHS	RHSUNKX	RHS-UNKNOWN	Dynamic	RHSOLV
RHS	RHSUNKX	RHS-UNKNOWN	IMAGE,IUP	RHSOLV
RHS	RHSUPT	RHS-UPDATED	Dynamic	MAPS
RHS	SNGKWN	SINE-KNOWN	Dynamic	MAPS
RHS	SOLDAT	SOLUTION-DATA	Dynamic	MAPS
RHSX	CNSTR1	CONSTRAINTS1	Dynamic	MAPS
RHSX	CNSTR1	CONSTRAINTS2	Dynamic	MAPS
RHSX	LMDPRT	LAMBDA-PART	Dynamic	MAPS
RHSX	MATRXC	MATRIX-C	Dynamic	MAPS
RHSX	RHSBLIC	RHS-MATRIX	Dynamic	MAPS
RMS	AICBLK	AIC-BLOCKS	Dynamic	RHSOLV
RMS	AICBLK	AIC-BLOCKS	IPOS	RHSOLV
RMS	BLIN	BLOCK-INFO	F. P.	BLKGEN
RMS	RECMAT	REC-BLOCK	Dynamic	BLOCK
RMS	RECMAT2	REC-BLOCK	Dynamic	BLOCK

THIRD FORM

COMMON BLOCK*	DATA BASE	MAP-NAME	DATASET-NAME	PROGRAM/ SUBROUTINE
Dynamic	DIP	DIPCLS	CLOS-COND	MAPS
Dynamic	DIP	DIPGBC	COEFF-GEN-BC	MAPS
Dynamic	DIP	DIPLFW	LOCAL-FLOW	MAPS
DIPamic	DIP	DIPSFW	SPEC-FLOW	MAPS
Dynamic	DIP	NETDATA	GLOBAL	MAPS
Dynamic	DIP	NETWKBD	NETWK-BDC	MAPS
Dynamic	DIP	NETWKSP	NETWK-SPEC	MAPS
Dynamic	DIP	SOLDATA	GLOBAL	MAPS
/FILDIP/	DIP	DIPCLS	CLOS-COND	MAPS
/FILDIP/	DIP	DIPGBC	COEFF-GEN-BC	MAPS
/FILDIP/	DIP	DIPLFW	LOCAL-FLOW	MAPS
/FILDIP/	DIP	DIPSFW	SPEC-FLOW	MAPS
Dynamic	DQG	BNDRY	BNDRY-CONDN-SPEC	MAPS
Dynamic	DQG	BPOINT	B-POINTER	MAPS
Dynamic	DQG	DIPTNG	BNDRY-CONDN-SPEC	MAPS
Dynamic	DQG	NETWK	NETWK-SPEC	MAPS
Dynamic	DQG	ROTATE	GLOBAL	MAPS
Dynamic	DQG	SNGSPC	SINGULARITY-SPEC	MAPS
Dynamic	DQG	SPECPT	SPECIAL-POINTS	MAPS
/NUM/	DQG	GLOBAL	GLOBAL	MAPS
/SYM/	DQG	GLOBAL	GLOBAL	MAPS
Dynamic	MAK	AICKWN	AIC-KNOWN	MAPS
Dynamic	MAK	COL1	COLMAP	MAPS
Dynamic	MAK	MATRXA	AIC-KNOWN	KWNCTR
Dynamic	MAK	ROWIN1	ROWMAP-INVERSE	MAPS
Dynamic	MAK	ROW1	ROWMAP	MAPS
Dynamic	MAK	SYM1	SYMMETRY	MAPS
ICPUP,ISNGUP,IPOS	MAK	MATRXA	AIC-KNOWN	KWNCTR
/NUM/	MAK	SYM1	SYMMETRY	MAPS
/SYM/	MAK	SYM1	SYMMETRY	MAPS
/RUNIDS/	MEC	MECHED	DATA-BASE-HEADER	MAPS
Dynamic	RHS	AICDIA	AIC-DIAGONAL	MAPS
Dynamic	RHS	LAMBLIC	LAM-MAT	RHSOLV
Dynamic	RHS	LMBKWN	LAMBDA-KNOWN	MAPS
Dynamic	RHS	MATRXB	LAMBDA-KNOWN	KWNCTR
Dynamic	RHS	ONSET	ONSET-FLOW	MAPS
Dynamic	RHS	RHSHED	DATA-BASE-HEADER	MAPS
Dynamic	RHS	RHSKWN	RHS-KNOWN	MAPS
Dynamic	RHS	RHSUNK	RHS-UNKNOWN	MAPS
Dynamic	RHS	RHSUPT	RHS-UPDATED	MAPS
Dynamic	RHS	SNGKWN	SINE-KNOWN	MAPS
Dynamic	RHS	SOLDAT	SOLUTION-DATA	MAPS
IMAG	RHS	LAMBLIC	LAM-MAT	RHSOLV
IMAG	RHS	LAMUNK	LAMBDA-UNKNOWN	RHSOLV
IPOS	RHS	PIVMAT	PIV-MAT	RHSOLV

COMMON BLOCK*	DATA BASE	MAP-NAME	DATASET-NAME	PROGRAM/ SUBROUTINE
IROWUP,JPOS	RHS	MATRXB	LAMBDA-KNOWN	KWNCTR
/BLKINF/	RHS	BLRHS	BLOCK-INFO	RHSOLV
/CPOS/	RHS	RHSIZE	SOLUTION-DATA	RHSOLV
Dynamic	RHS	PIVMAT	PIV-MAT	RHSOLV
Dynamic	RHS	RHSUNKX	RHS-UNKNOWN	RHSOLV
IMAGE,IUP	RHS	RHSUNKX	RHS-UNKNOWN	RHSOLV
MAICS	RHS	BLRHS	BLOCK-INFO	RHSOLV
NRHS	RHS	RHSIZE	SOLUTION-DATA	RHSOLV
Dynamic	RHSX	CNSTR1	CONSTRAINTS1	MAPS
Dynamic	RHSX	CNSTR1	CONSTRAINTS2	MAPS
Dynamic	RHSX	LMDPRT	LAMBDA-PART	MAPS
Dynamic	RHSX	MATRXC	MATRIX-C	MAPS
Dynamic	RHSX	RHSBLIC	RHS-MATRIX	MAPS
Dynamic	RMS	AICBLK	AIC-BLOCKS	RHSOLV
Dynamic	RMS	RECMAT	REC-BLOCK	BLOCK
Dynamic	RMS	RECMAT2	REC-BLOCK	BLOCK
F. P.	RMS	BLIN	BLOCK-INFO	BLKGEN
IPOS	RMS	AICBLK	AIC-BLOCKS	RHSOLV

* If dynamic mapping is used for some or all of the elements of a dataset, thus requiring no common block storage, it is indicated as such. See section 13 of this document for details of dynamic mapping.

APPENDIX 7-D DATA BASE MASTER DEFINITIONS

The data base master definition listing of the DIP module has been deleted from this document. It is produced from the PAN AIR tape during installation.

APPENDIX 7-E THE DIP FULL CONSTRAINTS TRANSCRIBER

The full constraints transcriber takes the user-input boundary condition data on the DIP data base, where it is stored in a fashion suitable for the fairly arbitrary input format, and stores it on the RHSX data base in a form suitable for use by RHS. The controlling routine for the constraints transcription in general is DIPCST. If the boundary conditions for a network are "Class 1" (that is, internal stagnation on thin impermeable surface), DIPCTS computes the appropriate constraints and writes them to the CNSTR2 data set (as well as writing a flag to the CNSTR1 data set that the boundary condition is network-wide).

If the boundary conditions are not class 1, DIPCTS calls DIPFUL, which then controls the writing of the CNSTR data sets for that network. DIPFUL first calls PREP, which finds out from DQG and DIP precisely which constraint coefficients supplied by DIP are residing on the data base (and therefore not necessarily zero). Certain constraints ($b_\emptyset$, b_T, and b_N of equation (H.3.25) of the Theory Document) reside on the COEF-GEN-BC data set of the DIP data base. The constraint b_0 resides on the DIP SPEC-FLOW data set, (but on the CLOS-COND data set if a closure condition is involved) while t_T resides on the DQG BOUNDARY-COND-SPEC data set. The local onset flow ΔU_i (cf., equation (H.3.22) of the Theory Document (Reference 1) comes from the DIP LOCAL-FLOW data set.

Next, DIPFUL loops on each of the possible constraints. It reads the appropriate constraints data, which may be smeared by DIP over all control points on an edge, or over a column of panel center control points, into /FILDIP/. The appropriate subroutine (CENTER for panel center control points, EDGE for edge control points, and SPECIL for special control points) loops over all points in the column or edge, computing their fine grid lattice indices, and then calling PRCNST to process the constraints data for that particular control point.

PRCNST then reads the CNSTR2 data set for that control point into /CSTDIP/. At this point in the computation, that data set contains those constraint coefficients which have been processed so far, while the rest are zero (initially, all constraints are set to zero). PRCNST then calls NEWCST, which transfers the value of the constraint under consideration from the holding array in /FILDIP/ into /CSTDIP/. PRNCST then writes out /CSTDIP/, which has one non-zero entry fewer than before, onto CNSTR2 data set.

It may occur that boundary conditions are not class 1, yet no constraints are reported by DIP because the user-input contributions to the right-hand-side is zero. In that case, no CNSTR2 data set is written, and it's absence in the second or third overlay (subroutine CNSTR or CNSTRT) is assumed to mean that all constraint values are zero. A warning message to that effect is then printed out once for each network.

APPENDIX 7-F THE UPDATE CAPABILITY

In case of either an IC update or a solution update (terms defined in Section 7.2.3 of the User's Manual, Reference 2), the old RHS data base is not used, but rather an entirely new RHS data base is created. Two restrictions on the user result from the operation of RHS during update runs. The first is the rather obvious one that the RHS data base designated by the user must not exist before the execution of RHS. The second arises from the fact that DQG is generally not re-run in a solution update, while RHS obtains the right-hand-side tangent vectors t_T from the DQG data base. Thus, a solution update which involves changing the tangent vector t_T is not permissible unless DQG is re-run as well.

8.0 MINIMAL DATA GENERATOR (MDG) MODULE

8.1 INTRODUCTION

The function of the MDG module is to compute average singularity values and perturbation flow quantities at control and grid points. These are interpolated from the singularity values at singularity locations (panel center points and/or network edge midpoints). Singularity values include source and doublet values. The perturbation flow quantities include the average potential, average normal mass flux and average velocity. Data used by post processing modules CDP, FDP and PDP is stored on the MDG permanent data base. Three temporary data bases are used by MDG for intermediate data.

8.2 MDG OVERVIEW

8.2.1 Purpose of MDG

The MDG module copies global data from DIP, network, and panel geometry data from DQG, and solution data from RHS to the MDG data base. MDG multiplies the IC (Influence Coefficient) matrices by the singularity matrices and unsymmetrizes the results as flow quantities at control points. Separate singularities are formed at grid points and control points. Complete control point flow quantities are formed from the values produced in multiplying the IC matrices, if they exist, or from the boundary condition values if the IC matrix did not exist for the control point. Potential splines are computed, similiar to the doublet analysis splines. From values at the control points, the potential splines produce average potential, normal mass flux, and average velocity at grid points.

8.2.2 MDG Input/Output Data

8.2.2.1 Input

The input data required by the MDG module comes from MEC, DIP, DQG, MAK, and RHS data bases. The MEC data base furnishes names, account numbers, status, and related information for all data bases. The DIP data base provides global data. The DQG data base provides network, panel geometry, control point, boundary condition, and splining data to MDG. The MAK data base contains the IC matrices and directory to translate the MAG control point indices to DQG control point indices. The RHS data base contains the singularities stored in columns by solutions, and in rows or partial row blocks stored by singularity index, and a directory to translate RHS singularity indices to DQG global singularity indices. It also contains partitioning and symmetry information needed in performing the post multiplication of the IC's and singularities, unsymmetrizing, and for unblocking the unknown singularities.

8.2.2.2 Output

Three temporary data bases are created during execution of MDG. The MDGF data base is created and used throughout MDG processing. It contains the error data set, control and grid point information for each panel, blocked singularity data at grid points, blocked control point flow quantity data, and potential spline data. The MDGM data base is used to form temporary singularity products, then released when the unsymmetrized blocked product is

formed on the MDGC data base. The MDGM data base is used again to store the blocked grid flow quantity data produced from the control point values and the potential splines. The MDGC data base contains the intermediate IC values at control points in blocks and the singularities at control points both blocked by images and solutions. It is released after all permanent control point data has been created, and block control point data is on the MDGF data base.

The MDG permanent data base contains several data sets. These include global, network, solution data, and control point geometry, control point data, grid point geometry, and grid point data.

Very little printed output is given by the MDG module. CPU usage is reported at the end of each overlay of MDG. If any errors occur during execution of MDG, error messages are printed as encountered and a summary of these errors is listed at the completion of MDG. For an error free MDG run, a comment is printed stating that the MDG module was successfully completed.

8.2.3 Data Base Interface

The MDG module creates a permanent data base named MDG for use by the PDP (Point Data Processor), the CDP (Configuration Data Processor) and the FDP (Field Data Processor) modules. MDG also creates three temporary data bases named MDGC, MDGF, and MDGM which are used for storing intermediate data. The external input to MDG is from MEC, DIP, DQG, MAK and RHS data bases. Figure 8.1 illustrates the relationship between MDG and these data bases.

8.3. MODULE DESCRIPTION

The MDG module consists of seven main overlays. A short description of these overlays follows. A tree diagram (Appendix 8-A) relates all overlays and their subroutines.

The main function of MDG is to calculate the values of average potential, average normal mass flux and average velocity at control points and at grid points. As described in Appendix M of the PAN AIR Theory Document (Ref. 1), these flow quantities are related in a linear fashion to the singularity values coming from the module RHS. Hence, a matrix product of the form $[IC] \cdot \lambda$ is used to find each of these values. Here [IC] is the matrix expressing the linear relationship. It is computed by the MAG module. λ is a vector of singularity values computed by the RHS module.

The above matrix multiplication gives the potential (ϕ), normal mass flux ($w \cdot n$) and velocity (v_a) only at the center of each panel and/or at the midpoint along a network edge. Values at the grid points and the actual control points (displaced slightly by the DQG module) must also be found. The MDG module interpolates these values to find the corresponding value of potential, mass flux and velocity using quadratic splines at the actual control and grid point locations. The spline coefficients are calculated by various methods. Refer to Section I.2 of the PAN AIR Theory Document (Reference 1) for more details on these methods.

8.3.1 Overall Structure

The main overlays of MDG are briefly summarized in this paragraph. The top level MDG program, overlay (0,0), initializes the program variables. It calls each of the seven overlays into execution in a sequence. If a fatal error occurs during execution of any overlay, a skip is made to overlay (7,0) named EASY which summarizes the errors. The overall structure of the MDG module is illustrated in Fig. 8.1. The MDG functional decomposition and the tree structure diagram are presented in Appendices 8-B and 8-A respectively.

8.3.2 Overlay Descriptions

8.3.2.1 (1,0) Overlay (OPDBI)

The program opens and checks all data bases to be used by MDG. If the data bases are usable, MDG processing begins. The MDGX GLOBAL, SOLUTION-DATA, and NETWORK-SPEC datasets of the temporary data base MDGX are created from data on the DIP data base, the DQG data base and RHS SOLUTION dataset. For each network, panel points data is determined. This consists of the control points and grid point data set for each panel. For each panel, geometry data is obtained from the DQG PANEL-SPEC dataset and written to the MDG CP-GEOM and GP-GEOM datasets. The panel points data is obtained from the DQG CONTROL-POINT-SPEC and SPECIAL-POINTS datasets. More details are given in Appendix 8-F under the panel points library routines and Appendix 8-G panel points library usage. The PANEL-POINTS dataset is written to the MDGF temporary dataset for use in the down stream overlays (3,0), (4,0), and (6,0). Figure 8.2 illustrates the execution and data flow for overlay (1,0).

8.3.2.2 (2,0) Overlay (PMPY)

This overlay stores singularities by blocks of images and solutions to be used as input to the (3,0) overlay. The known singularities are available in rows already unsymmetrized from the RHS SING-KNOWN dataset and stored on the MDGC data base, BLOCKED-LAMBDA dataset. The unknown singularities are created in blocks of maximum size, 100 singularity rows by 10 solution columns. A separate direct dataset BLOCKED-LAMBDA-MATRIX from RHS contains information about the size and partitioning of these blocks. These blocks are converted to unsymmetrized rows by writing partial rows onto a temporary dataset ROW-BLOCK-LAMBDA dataset. They are read back into core to form full rows which are reblocked into blocks for a single singularity row for images and solutions, and placed the same BLOCKED-LAMBDA dataset created for known singularities. The MAG singularity indices are converted to DQG global singularity indices used as keys. Figure 8.3 illustrates the execution and data flow for overlay (2,0).

8.3.2.3 (3,0) Overlay (SNGCP)

The third overlay converts BLOCKED-LAMBDA singularities at singularity locations to values at panel grid points and at the control point locations. The DQG B-SPLINE-SOURCE and B-SPLINE-DOUBLET datasets contain the continuous splines vectors for a single grid point . The splines are stored as two arrays, an index array which points to the singularity locations needed to compute the singularity at the grid point, and a second array of weights to be applied to the corresponding singularity values at singularity locations. The absolute values of the spline weights lie in the range from 0 to 1, where a

value 1 represents an infinite weight. For doublet grid points a spline vector exists for all grid point locations. For sources, spline vectors exist only at the four panel corner points and the panel center point. To compute source values at the edge midpoints the 5 panel source grid point values are used in conjunction with the subpanel splines. With this splining a set of library routines, called the table manager, is used to store the values at singularity locations, so that multiple accesses to the disk are reduced for the same singularity values required by different splines, or for computing a value at a grid point which has already been computed for an adjacent panel. When the table is full the least recently used values are deleted from the table. The table manager is described in Appendix 8-F under the table manager library routines and Appendix 8-G under the table manager library usage. Values at control points are calculated using the 5 or 9 grid point values obtained from the spline vectors for each panel. The computation converts the reference coordinates to local coordinates by application of the the reference to local coordinate transformation from the DQG PANEL-SPEC dataset. Multipling the local coordinates by the subpanel spline produces coefficients which are multiplied by the 5 or 9 grid point values to give a value of the singularity strength at the control point. Figure 8.4 illustrates the execution and data flow for overlya (3,0).

8.3.2.4 <u>(4,0) Overlay (AQCP)</u>

Values of average potential, average normal mass flux, and the three components of the average velocity will be referred to as flow quantities. Potential and mass flux can be calculated from either the IC (Influence Coefficient) matrices or from the boundary conditions imposed by the user. Average velocity can only be calculated from the IC matrix. The IC matrix for the average normal mass flux is the dot product of the VIC (Velocity Influence Coefficient) matrix and the conormal at the control point. The storage of IC matrices is known to MDG through the PDP parameter written by DQG on the BOUNDARY-COND dataset. For each network the availability of the IC matrices in the MAK data base is also known from the method-of-velocity-computation variable. The PDP parameter determines if the flow quantity for each control point is obtained from the IC-matrix value or from the boundary condition data. In processing, the network data is read and a loop over panel is executed reading the MDGC PANEL-POINTS dataset to determine the control points within each panel. For each control point the BNDRY-CONDN-SPEC data is read to determine if the boundary condition data or the IC-matrix data will be used as the value of the flow quantities at the control point. If the IC data is used, the IC matrix is read together with singularity data, is post multiplied, and unsymmetrized. If it is needed for any flow quantity, the RHS-UNKNOWN values are read in and unsymmetrized. The control point singularities are read from the CP-LAMBDA dataset for storage on the CP-DATA dataset and possibly for calculating the values of the boundary condition used in computing a flow quantity. The potential and mass flux computations are essentially parallel. Both have special cases to check for stagnation boundary conditions, in addition to a general boundary condition calulation using the RHS values. For cases where the IC matrix exists, these values are read directly from the IC-MATRICES dataset. All flow quantities and singularities are blocked by images and solutions on the CP-BLOCK dataset for use in splining with the potential splines in the (6,0) overlay. Figure 8.5 illustrates the execution and data flow for overlay (4,0).

8.3.2.5 (5,0) Overlay (BPSV)

Potential splines are created on network edges first. If an edge of a network collapses, all points along the edge take on the value of the spline at the network corner. A general algorithm is used to compute network edge splines. On the interior all center point splines are unit splines weighted infinitely to the center control point value. For interior corners and edge midpoints another algorithm is applied which uses values from surrounding center control point values. The weights applied to these values are determined by doing a constrained least squares fit on the spline vector. Figure 8.6 illustrates the execution and data flow for overlay (5,0).

8.3.2.6 (6,0) Overlay (GPQTY)

This overlay obtains average flow quantities on the fine grid. All data for a given network is assembled in memory. A spline data structure which can extend data from the grid of panel center control points to the course grid is built. It is an outer spline and is constructed in a fashion similar to the continuous source analysis splines. That technique is described in Appendix I of reference 1. The normal mass flux, velocity and potential for control points is retrieved from the MDG database. The spline data structure is used to spread normal mass flux and velocity over the entire fine grid. The BP-SPLINE-VECTOR dataset is used to spread the potential over the fine grid. The normal mass flux and potential may be computed by stagnation if it is requested. The fine grid data is written to the MDG database. Also the normal mass flux and velocity at edge control points are revised to conform to grid point values. Figure 8.7 illustrates the execution and data flow for overlay (6,0).

8.3.2.7 (7,0) Overlay (EASY)

If any fatal errors have occurred during the run of MDG, an error summary is printed reading the MDGF ERROR dataset created by the library subroutine SDMSRR for fatal SDMS errors or from the (4,0) overlay if a boundary condition value cannot be calculated. The SDMS error processing is described in Appendix 8-G under SDMS error library usage. The condition of the MDG data base is written on the DATA-BASE-HEADER dataset of MDG. Figure 8.8 illustrates the execution and data flow for overlay (7,0).

8.3.2 MDG Data Bases

The master definitions for the MDG, MDGC, MDGF, and MDGM data bases are given in Appendix 8-D.

8.3.4 MDG Interfaces

Figure 8.1 summarizes the internal and external data interfaces between MDG and other PANAIR modules (data bases).

8.3.4.1 External Interfaces

MDG receives its input data from the MEC, DQG, MAG (MAK data base), and RHS modules. The MEC data base furnishes data base names, accounts, and status information for all data bases. The DQG data base provides global, network, panel geometry, control point, boundary condition, special points,

and spline data. RHS furnishes the solution data (also available from DQG), singularity data, right hand side constraints, and columns maps to convert from the MAG indices to the DQG singularity indices. From MAG comes the IC matricies, symmetry data, and the row map to convert MAG indices to DQG control point indices.

The output data consists of the MDG data base used by FDP, PDP and CDP modules. It contains global, network, solution, control point geometry, grid point geometry and the solution data for control and grid points. The geometry data consists of coordinates, normal and tangent vectors, subpanel splines for doublets, and moment matrices. The solution data consists of average potential, average normal mass flux, sometimes average velocity, and the source and doublet singularities.

8.3.4.2 Internal Interfaces

The MDGF data base is used to keep track of fatal errors during the MDG run. In (1,0) overlay panel points data is generated and stored for use in the (3,0), (4,0), and (6,0) overlays. In the (3,0) overlay the grid point singularities are blocked and added to the GP-LAMBDA dataset on MDGF. In the (4,0) overlay the CP-BLOCK-DATA dataset is created on MDGF. The BP-SPLINE dataset is written to MDGF in the (5,0) overlay and read by the (6,0) overlay. If fatal errors occurred during the run the MDGF ERROR dataset is read by the (7,0) overlay to produce the error summary.

The MDGM data base is used for reblocking the unknown singularities in the (2,0) overlay. The MDGM data base is returned at the end of the (2,0) overlay. It is used again in the (6,0) overlay to create the temporary blocked grid point data from the final GP-DATA on the MDG data base is created.

The MDGC data base is created in the (2,0) overlay. It contains the BLOCKED-LAMBDA singularities which are read by the (4,0) and (3,0) overlays. MDGC is returned at the end of the (4,0) overlay.

8.3.5 Data Flow

The flow of data for each MDG overlay can be found by consulting Figures 8.2 thru 8.8, Appendix 8-C and the glossaries of the program/subroutines.

8.4 LOWER LEVEL FUNCTIONS

This section describes the general structure and purpose of the subroutines used in MDG.

8.4.1 Functional Decomposition

See Appendix 8-B for a description of the MDG Functional Decomposition.

8.4.2 Subroutine Descriptions

ANALP Computes potential splines for each grid point of a network.

AQCP Forms control point data for each point for average potential, mass flux, average velocity (if available), singularities, and local onset flow.

BCDAT Obtains boundary condition data if required.

BLKS Forms blocked lambdas from RHS known singularities which are already unsymmetrized.

BLUKSG Forms blocked unsymmetrized lambdas from RHS unknown singularities.

BPSV Forms potential splines (BP-SPLINES) similar to doublet analysis splines for values taken from surrounding control points instead of singularity locations.

CMPDSV Computes doublet singularity value from 9 panel grid point singularity values using local coordinates and subpanel splines.

CMPSSV Computes source singularity value from 5 panel grid point source singularity values using local coordinates and subpanel splines.

COMSRC Computes panel point values for a panel located either at the end of a single row or column network or at a network corner.

CPCALC Computes flow values at control points from IC, Lambda matrix multiplication, and unsymmetrizes the values.

CPFDCE Computes control point flow data on collapsed edges.

CPFDSS Computes control point flow data on smooth abutment segments.

CPPED2 Computes panel points on edge 2 of network.

CPPED4 Computes panel points on edge 4 of network.

CPPE13 Computes panel points on edges 1 and 3 of network in parallel for single column case.

CPPE14 Computes panel points on edges 1 and 4 of network in parallel for single row case.

CPPINT Computes panel points interior to a panel

CSCP Forms values of the singularity at control point locations for each control point in panel.

CSSGP Applies B splines to calculate singularities at grid points from surrounding singularity values.

DATPOT Provides data for analysis splines to perform constrained least squares fit.

DCPGPS Determines control points and grid point sets from DQG data.

DFSKWC Defines the transformation from the reference coordinate system to a skewed coordinate system associated with a fine grid point.

DGPTL Defines grid point table location of array GPLOC which stores the 9 defining grid pointers to singularity data .

DINGPS Determines interior grid point set.

DTNGPS Defines interior grid point set for the default grid point set consisting of panel points 1, 5, 8, 9.

DPANCP Defines panel control points by searching along a specified network edge for a given panel and the network edge.

DPBSGP Defines panel B-spline grid points.

DQGSNG Determines the DQG global singularity indices for a block of unknown singularities using the MIC column map.

EASY Provides error and accounting summary for MDG run, printing a list of fatal errors if any occured, and writing MDG data base status information to the MDG data base

EDGECP Revises the velocity and mornal mass flux at edge control points to conform to grid point values.

FCPDAT Forms control point dataset from the DQG control-pt-spec dataset.

FCPGPG Forms control point and grid point geometry datasets.

FGLDAT Forms MDG global dataset .

FORNET Forms MDG network spec dataset.

FSAGP Forms singularity values at grid points for values not already calculated, using the BP-splines and vectors of singularity values at surrounding locations.

FSGVEC Forms singularity matrix from singularity vectors for each solution and image.

FSPSVC Forms subpanel spline vector

FSTBAD Adds deleted table entry to free space table.

FSTBDL Removes last free space to be used as location for table add.

GETSPT Reads the DQG special points data set into special points common block.

GPQTY Applies potential splines to compute flow quantity values at grid points, stores potential, mass flux, velocities, and singularities at grid points for each panel point set.

ICALC Computes and unsymmetrizes flow quantity values at control points using the IC matrices.

INITCB Initializes the common blocks /SOLLST/,/SYMM/ reading the RHS solution dataset and forming the MDG solution dataset.

LATIND Transforms course panel grid points which exist at panel corner points for any panel points into its corresponding fine grid point lattice.

LOCORD Computes local coordinates of a point given its coordinates in the reference coordinate system together with the subpanel number in which the point lies. The appropriate reference to local coordinate transformation is applied to compute the local coordinate.

LSQPOT Defines the coordinate of a point used in the construction of the least squares spline and defines the index of the singularity parameter or control point located there.

MCPDAT Moves control point data obtained from the DQG control-pt-spec dataset to the MDG /CPDAT/ common block.

NETEDG Calculates spline vectors for network edges and corner points.

NETGEM Assembles the network geometry data.

OBPHI Obtains potential values for each control point for all RHS solutions and images from either the boundary conditions, stagnation conditions or the PHIIC matrix.

OBVA Obtains average velocity values for three components of velocity from the product of the VIC and unsymmetrized singularity values and computes the normal mass flux from the velocity values at each control point where the VIC is defined.

OBVANC Obtains v_a dot n_c values at each control point for all RHS-solution and images from either the boundary conditions, stagnation conditions, or from the IC matrix.

OPCKDB Opens all data bases and checks their status through calls to CHKDB

OPDBI Checks data bases, initializes common blocks, forms network data, forms panel point datasets for control and grid point geometry, and closes data bases.

OPDBM Defines maps used in MDG third overlay, opens and closes data bases.

PMPY Unsymmetrizes and blocks known and unknown singularity values. These are read from the RHS data base datasets SING-KNOWN and LAM-MAT. After processing, they are written to the MDGC BLOCKED-LAMBDA temporary dataset.

POINT Determines the coordinates of any point from its fine grid point lattice.

PTSKWC Computes the skewed local coordinates of a point.

PUTPPT Writes panel points dataset to data base from common blocks /CPDAT/,/GPSET/.

RHSVAL Obtains and unsymmetrizes RHS unknown lambda values from RHS data base .

RNDITM Reads network data and initializes table manager routines.

RPI Reads panel data information.

SCOLPP Computes panel points data for a single column network.

SDMSRR Processes SDMS errors occuring during MDG execution by writing error message, setting fatal error flag and writing an error data set for final error summary.

SNGCGP Computes singularity at control and grid points from values at surrounding locations.

SPANCP Searches network edge to see if extra control points exist.

SPLAP Performs least squares fit to determine spline vector for a specified point in an analysis network.

SPLCP Computes mesh point spline data.

SPLCPV Evaluates a single entry in the spline data structure.

SPLCPW Computes upstream weighting factors for spline construction.

SPLCPX Constructs a spline data structure to spread center control point data to the fine grid. The spline data structure is similar to the continuous source analysis splines built by the Defining Quantities Generator (DQG) module in subroutine ANALS. This is an outer spline which is described in section I of reference 1.

SPLTRN Transforms three-dimensional coordinates of singularity or control point into the local two-dimensional coordinate system for least squares fit.

SROWPP Computes panel points data for a single column network.

STCPV Stores control point flow and singularity data for each solution and image on MDG CP-DATA dataset.

STGP Stores grid point data generated using potential splines for each image and solution for all flow types.

STOGPS Stores blocks of grid points singularities for output in the (6,0) overlay where blocks are formed by images and solutions for each grid point set.

TBADD Adds new entry to table by searching for its location in the key table and returning the location in the table of the added data.

TBDELT From least recently used (LRU) counter, determine the LRU entry and add this as the new freespace entry.

TBINIT Initialize the table manager by clearing entries when called with appropriate arguments, or initialize individual tables for later processing.

TBSRCH Finds location of entry in table by comparing key value input against values in key table. If entry is not found, a value of -1 is returned as the location.

UNIPOT Computes a spline vector of unit length for each control point index and writes it to the BP-SPLINE dataset.

USFIMG Unsymmetrizes singularities at singularity locations for 4 images and blocks them by images and solutions.

USTIMG Unsymmertizes singularities at singluarity locations for 2 images and blocks them by images and solutions.

VECUNV Computes BP spline vector at a grid point from values at surrounding control point locations.

VSPREP Spread the panel center (or panel center control point) data to the whole fine grid.

VSPRET Spreads the panel center velocity data to the whole fine grid.

WNSPRD Spreads the panel center normal mass flux data to the whole fine grid.

WRBL For use in unblocking LAMMAT matrices of singularities -the partial rows are formed for each row in a block of rows and temporarily written to the data base.

WTLSQ Computes weighting factors for least squares fit.

XPSPEC Extends the panel specifications data read from the MDG-PANEL-SPEC dataset on the DQG database.

XIETAV Computes XI and ETA vectors which define the local two-dimensional coordinate system.

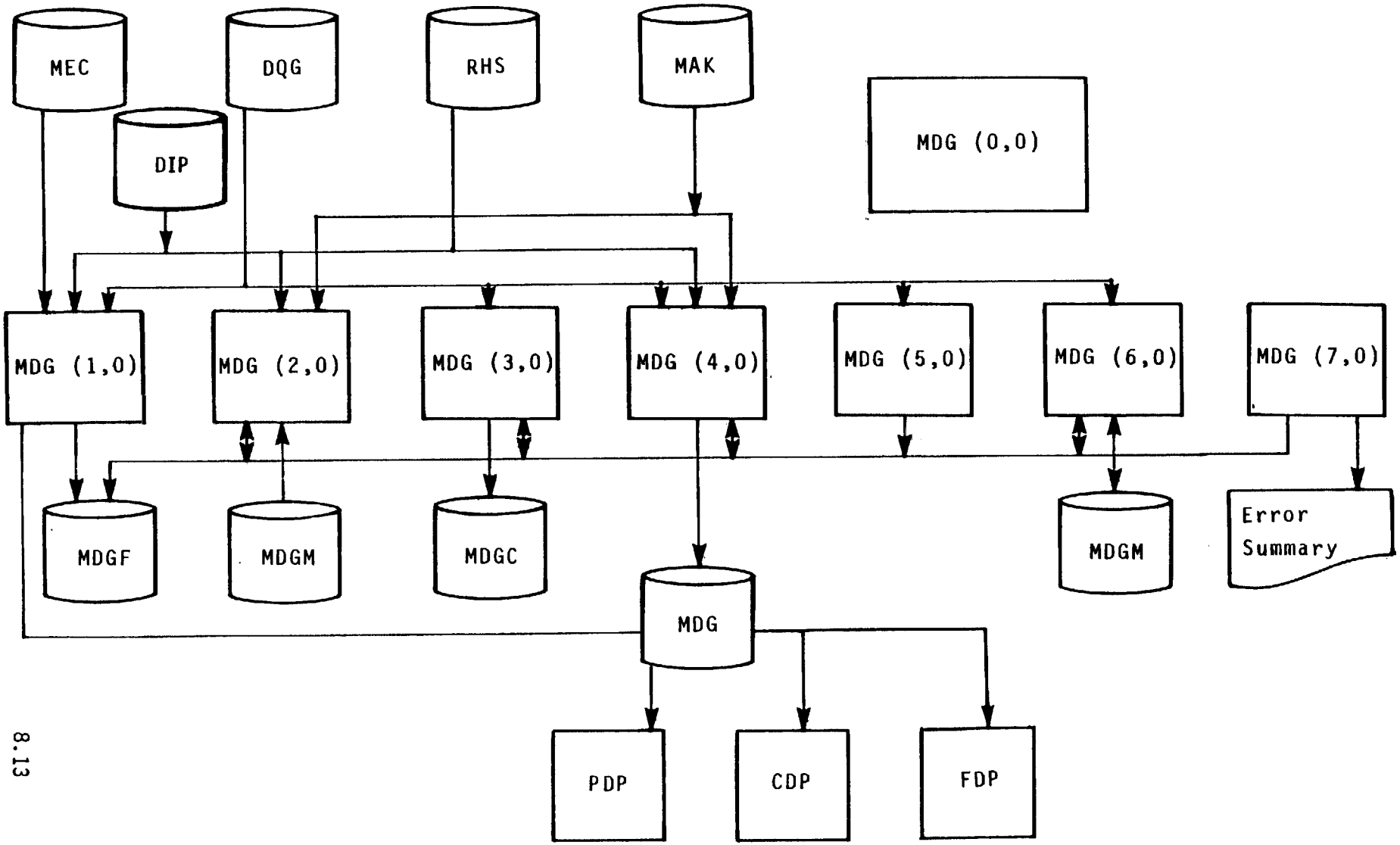


Figure 8.1 - Data Base Relationships

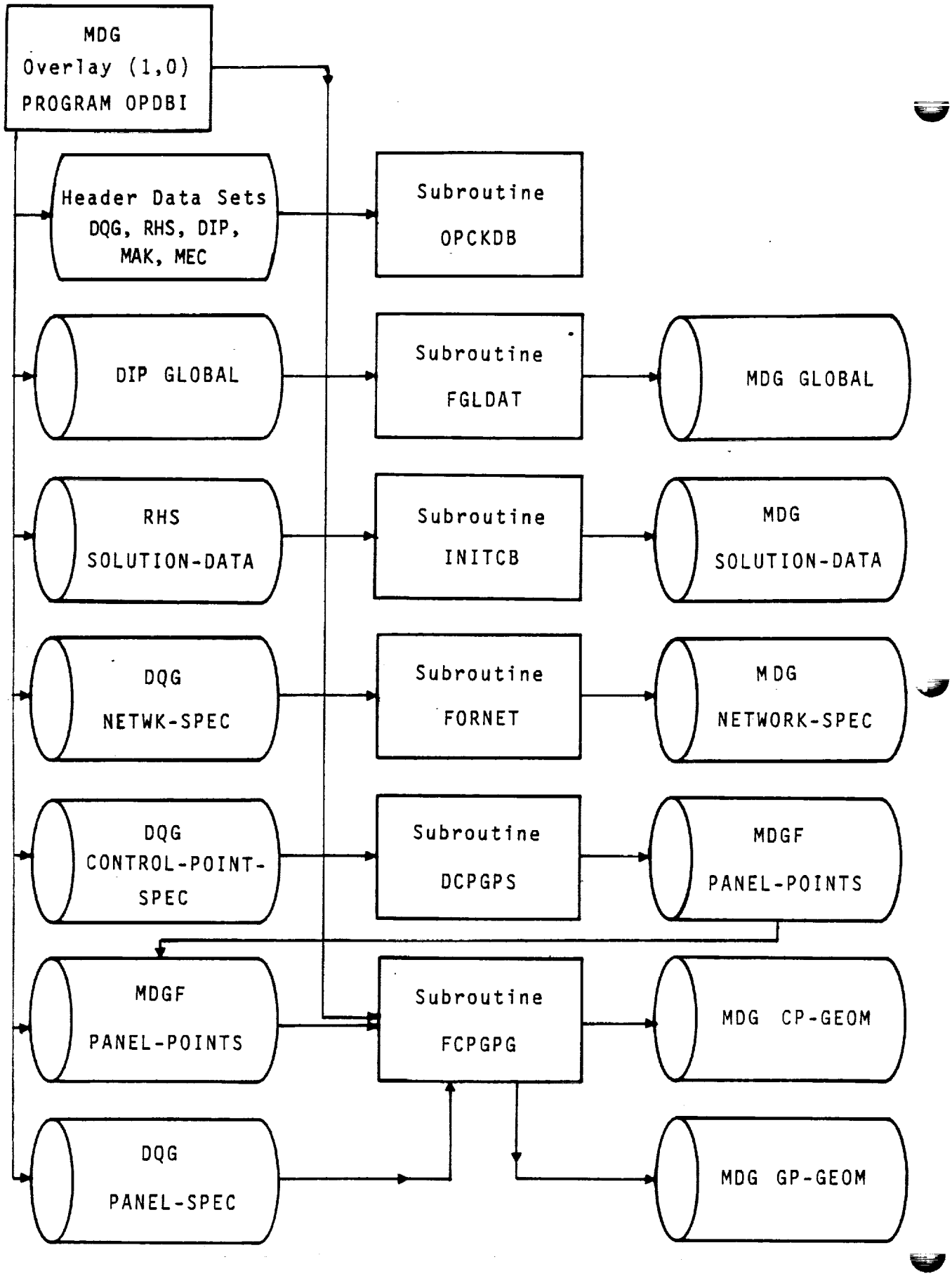


Figure 8.2 - Execution and Data Flow of Overlay (1,0)

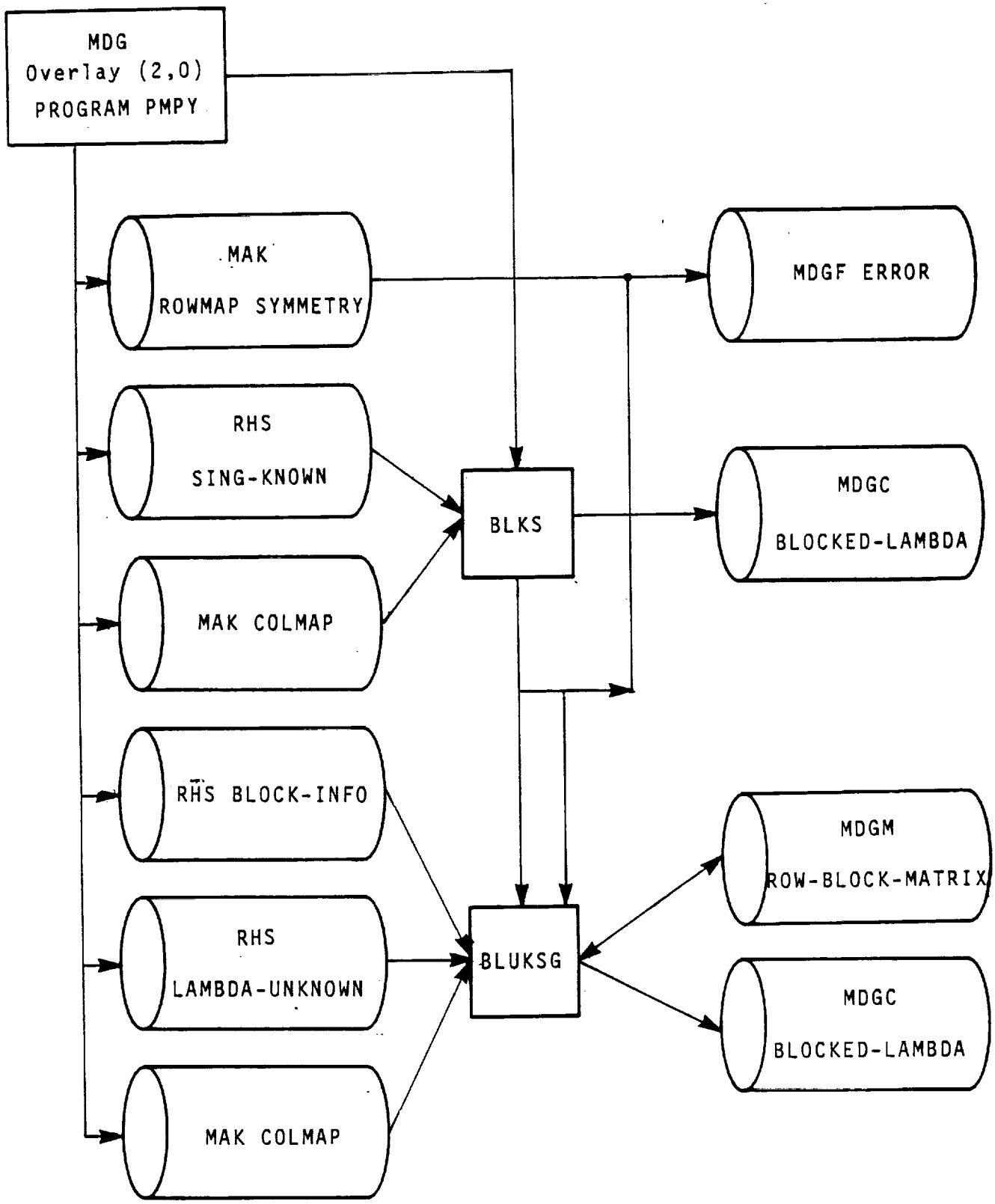


Figure 8.3 -Execution and Data Flow of Overlay (2,0)

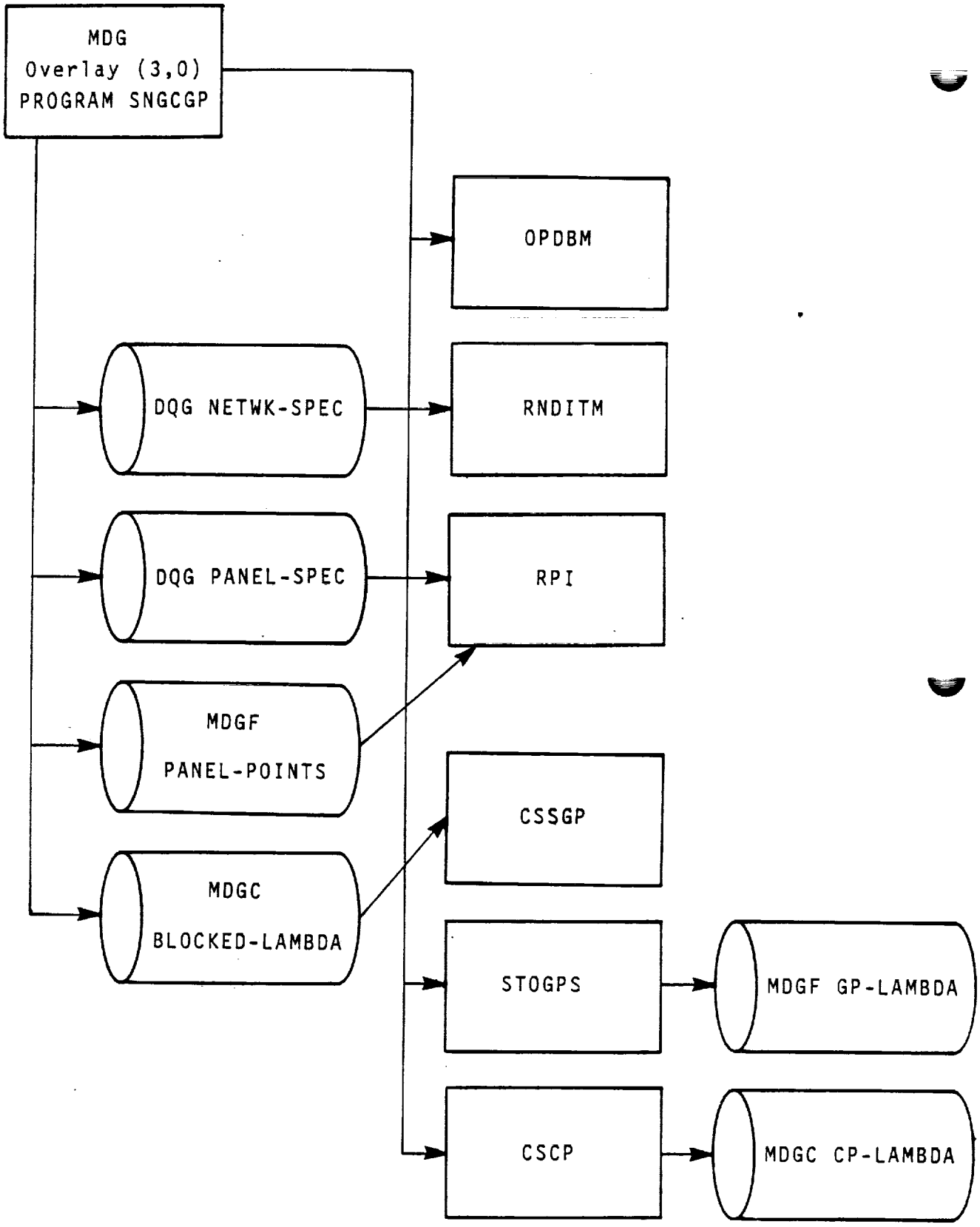


Figure 8.4 -Execution and Data Flow of Overlay (3,0)

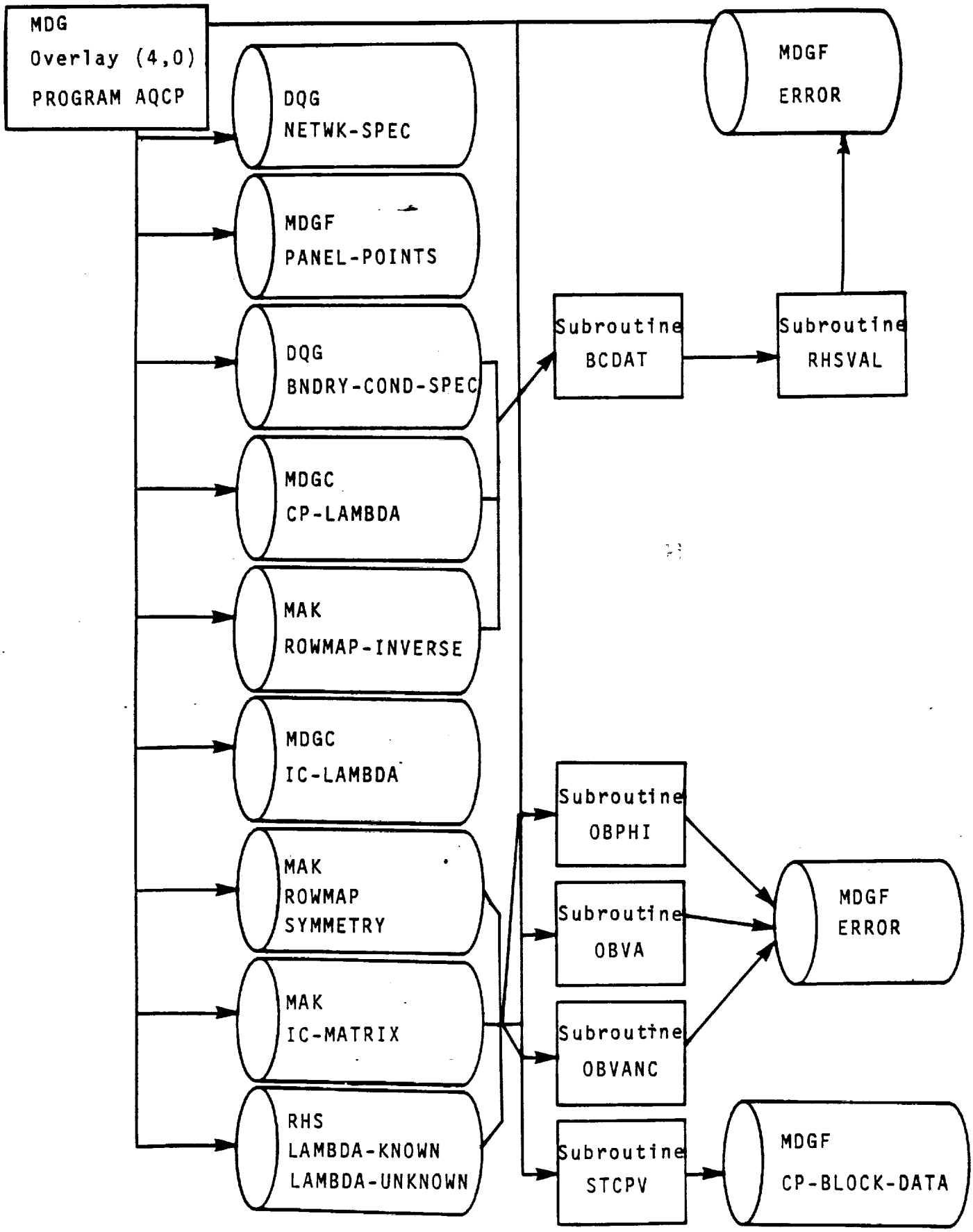
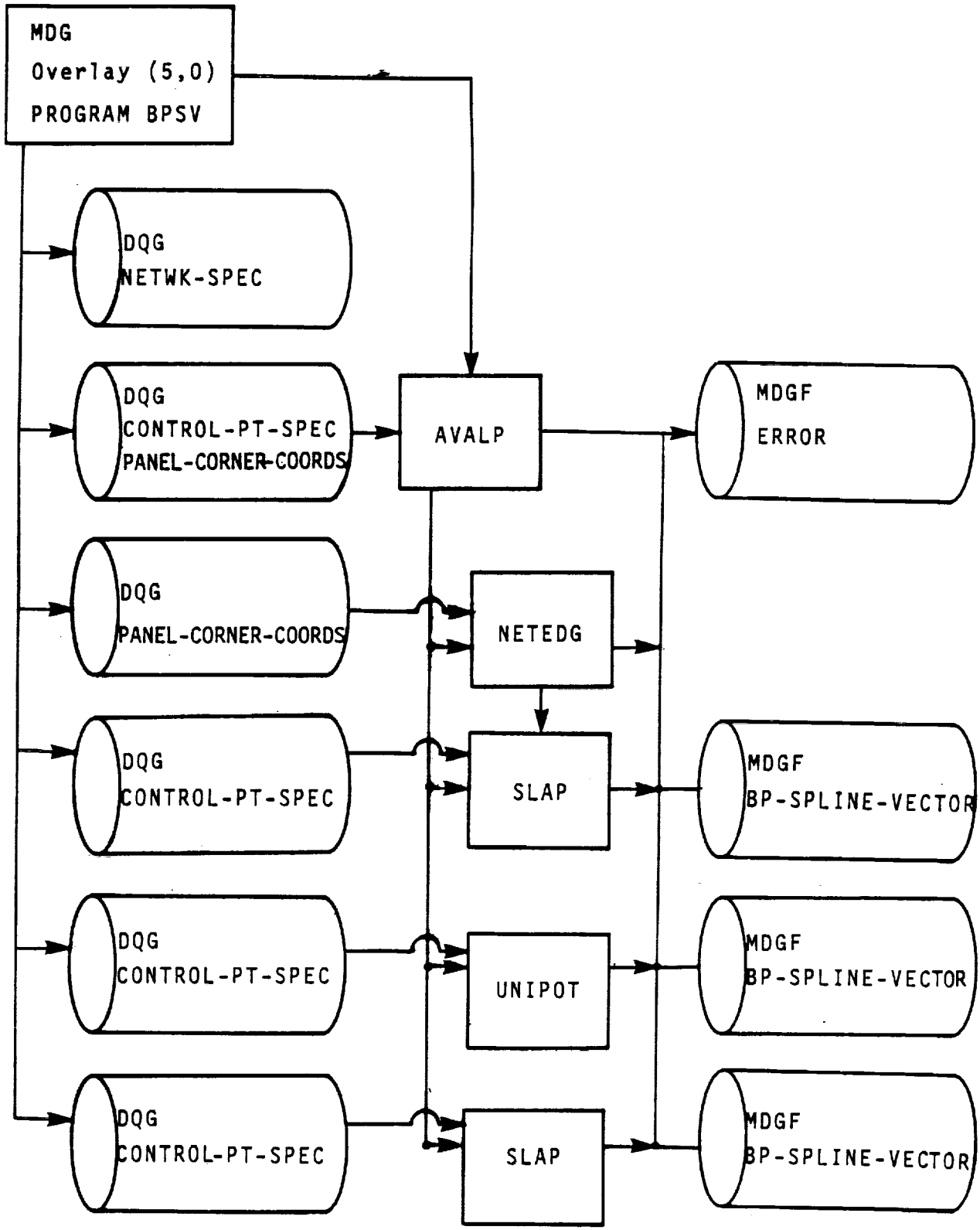


Figure 8.5 - Execution and Data Flow of Overlay (4,0)



8.18 Figure 8.6 - Execution and Data Flow of Overlay (5,0)

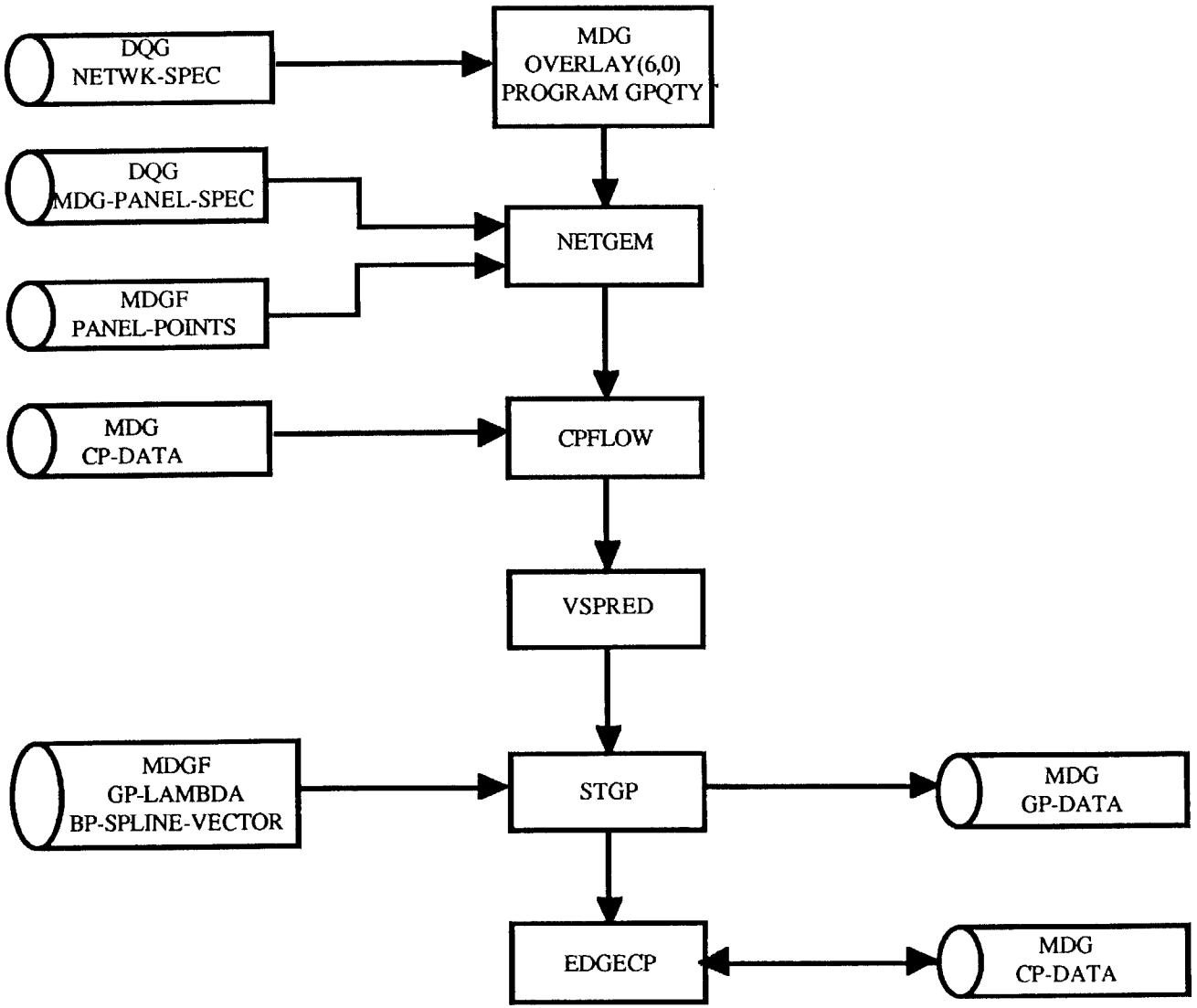


FIGURE 8.7 - EXECUTION AND DATA FLOW OF OVERLAY (6,0)

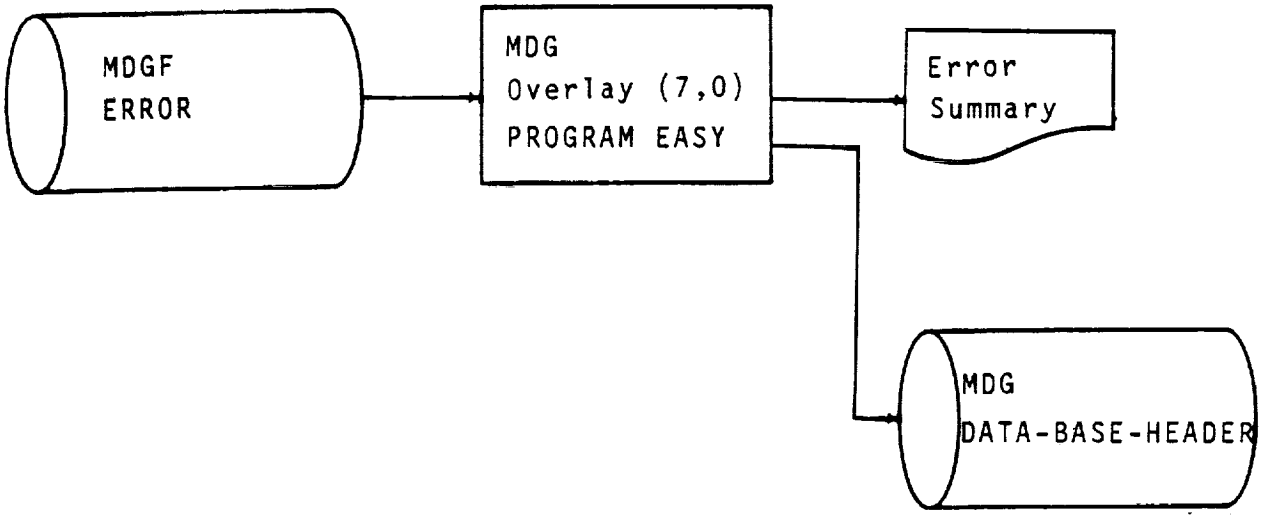


Figure 8.8 - Execution and Data Flow of Overlay (7,0)

APPENDIX 8-A TREE STRUCTURE

The tree structure diagram of the MDG module has been deleted from this document. It is, however, available on the installation tape.

APPENDIX 8-B MDG FUNCTIONAL DECOMPOSITION

The functional decomposition of the MDG module is presented here. The decomposition labels are given in the order of their execution and therefore may not be alphabetic.

A. Form global data, initialize global common block and process network control point and grid point (OPDBI) Overlay (1,0)
  A. Open and Check Data Bases (OPCKDB)
  B. Form Global Data (FGLDAT)
    A. Read Global Data
    B. Form Network List Data
    C. Put MDG Global Data
    D. Compute Reflection Matricies
  C. Initialize Global Common Blocks (INITCB)
    A. Extract Solution Data from RHS Data Base
    B. Form /SOLLST/ Common Block
    C. Write MDG Solution Data
  D. Form Network Data (FORNST)
    A. Read DQG NETWK-SPEC dataset
    B. Form MDG NETWK-SPEC dataset
    C. Put MDG NETWK-SPEC dataset
  E. Determine Control Points and Grid Points Set (DCPGPS)
    A. Compute Panel Points Edges 1 and 3 (CPPE13)
      A. Get Special Points
      C. Initialize CP, GPSET Count
      D. Define Interior Grid Point Set (DINGPS)
      B. Check for CP on Midpoint of Edge 4 (FCPDAT)
      E. Define Panel Control Points (CPANCP)
      F. Define Grid Point Set 2
      G. Put Panel Point (PUTPPP)
    B. Compute Panel Points Edge 2 (CPPED3)
      H. Compute Panel Points on First Panel
      A. Get Special Points (GETSPT)
      B. Initialize CP GPSET Count
      C. Define Interior Grid Point Set (DINGPS)
      D. Define Panel Control Points (DPANCP)
      E. Define Grid Point Set 2 PTS 2,6
      F. Put Panel Points (PUTPPP)
      G. Compute Single Row/Col Panel Points (COMSPC)
    C. Compute Panel Points Edge 4 (CPPED4)
      A. Get Special Points (GETSPT)
      B. Initialize CP GPSET Count
      C. Define Interior Grid Point Set (DINGPS)
      D. Define Panel Control Points (DPANCP)
      E. Put Panel Points (PUTPPT)
    D. Compute Panel Points Interior (CPPINT)
      A. Define Interior Grid Point Set (DINOPS)
      B. Form Control Point Data (FCPDAT)
      C. Put Panel Points (PUTPPT)
    E. Compute Single Row Panel Points (SROWPP)
      A. Compute Single Row/Col Panel Points (COMSRC)
      B. Compute Single Row/Col Panel Points (COMRRC)
      C. Get Special Points Edge 1, Edge 3 (GETSPT)

D. Initialize CP, GEPSET Count
E. Compute Panel Points Edge 1, 3
A. Define Interior Grid Point Set (DINGPP)
B. Define Panel Control Points (DPANCP)
C. Define Grid Points Set 2, PTS 4,7
D. Put Panel Points (PUTPPT)
F. For Last Column Compute the Panel Points (OMSRC)
F. Compute Single Col Panel Points (SCOLPP)
A. Compute Panel Points on First Row Panel (COMSRC)
B. Compute Panel Points on First Column Panel(COMSRC)
C. Get Special Points Edge 2, Edge 4 (GETSPT)
D. Initialize CP, GPSET Count
E. Compute Panel Points Edges 2 and 4 (CPPE24)
A. Define Interior Grid Point Set (DINGPS)
B. Define Panel Control Points (DPANCP)
C. Define Grid Point Set 2, PTS 2,6
D. Put Panel Points (PUTPPT)
F. Compute Single Row/Col Panel Points (COMSRC)

F. Form Control Point and Grid Point Geometry (FCPGPG)
A. Get Panel Data
A. Get DQG PANEL-SPEC Data
B. Get MDG PANEL-POINTS
B. Add Data to CP-GEOM
A. Determine if GRIDPT is a Control Point
B. Move Data from /CPDAT/
C. Move Data from /PANDAT/
C. Write CP-GEOM
A. Form Key
B. Put GP-GEOM
D. Define CP-GEOM Data
A. Define GP-COORDINATES
A. Determine Offset in Panel Coordinates
B. Move Panel Coord to /GPGEOM/ Coordinates
C. Store Grid Point Sequence
E. Write GP-GEOM
A. Form Key from 1st Grid Point
B. Put GP-GEOM

G. Close Data Base

B. Read, Unsymmetrize, and Block Singularity Values(LAMBDA) Overlay (2,0)
A. Define Maps and Read Matrix Information
A. Open Data Base and Define Maps
A. Define Maps for MIC Data Base
B. Define Maps for RHS Data Base
C. Define Maps for MDGC Data Base
D. Define Maps for MDGF Data Base
E. Define Maps for MDGM Data BAse
B. Read MAG Symmetry
A. Get MIC Symmetry
B. Form /SYMM/
C. Form /MAGPAR/

C. Unsymmetrize and Block LAMBDAS
  A. Block LAMBDAS from KNOWN-SINGULARITIES (BLKS)
    A. Determine Singularity Index
      A. Determine Update Type
      B. Set Update Type to Non-Updatable
      C. Set Update Type to Updatable
      D. Get COL-MAP
    B. Get RHS SING-KNOWN
    C. Form Blocked LAMBDA (KNOWN)
      A. Determine Block Offset
      B. Move Unsymmetrized Images
      C. Determine Number of Solutions Per Block
      D. Define Number of Images
      E. Define BLOCK-SIZE
      F. Put Blocked LAMBDA
  B. Re-Block LAMBDAS from UNKNOWN-SINGULARITIES (BLUKSG)
    A. Get BLOCK-RHS-INFO
    B. Determine DQG-SING-INDEXES (DQGSNG)
      A. Determine BLock Row Start
      B. Determine ROW-NUM, Update Type
        A. Form Global ROW-NO
        B. Define Row No. Non-Updatable
        C. Define Update Type
        D. Define Row No. Updatable
        E. Define Update Type
      C. Get MAG COL-MAP
      D. Store DQG SING-INDEX
    C. Unsymmetrize LAMBDA-MATRIX Blocks
      A. Get LAMBDA MATRIX IMAGE 1
      B. Unsymmetrize Two Images (USTIMG)
        A. Move ARRLAM to ARRLPP, ARRLPM, ARRLMP
        B. Read LAMBDA-MATRIX IMAGE 2
        C. Unsymmetrize 1st (PP) Image
        D. Substract ARRLAM from ARRLMP and Divide
        E. Subtract ARRLAM from ARRLPM and Divide by Two
      D. Unsymmetrize Four Images (USFIMG)
        A. Move ARRLAM to ARRLPP, ARRLAM, ARRLMP, ARRLMM
        B. Get LAMBDA-MATRIX
        C. Compute Images
      C. Move ARRLAM to ARRPP
    D. Write ROW-BLOCK-LAMBDA (WRBL)
      A. Move ARRLPP to RBLAMB
      B. Move ARRLAM to RBLAMB
      C. Move ARRLMP to RBLAMB
      D. Move ARRLMM to RBLAMB
      E. Move ARRLPM to RBLAMB
      F. Form ICE
      G. Put ROW-BLOCK-LAMBDA
    E. Form ROW-ARRAY from Row Blocks
      A. Get ROW-BLOCK-LAMBDA
      B. Determine Offset in Row Array
      C. Move ROW-BLOCK to ROW-ARRAY

F. Re-block and Output BLOCKED-LAMBDA dataset
    A. Determine ROW-ARRAY Offset
    B. Form BLOCKED-LAMBDA Data
    C. Form Key Set and State Values
    D. Put BLOCKED-LAMBDA dataset

C. Obtain Singularities at Control and Grid Points (SNGCGP) Overlay (3,0)
  A. Open Data Bases and Define Maps (OPDBM)
    A. Open Data Bases MDGC, MDG, MDGF, DQG
    B. Define Map Sequence
  B. Read Network Data and Initialize Table Manager (RNDITM)
    A. Get DQG NETWK-SPEC Data
    B. Determine ROW-COL Information
    C. Determine Network Type and Panel Grid Points
      A. Determine Network Type
      B. Define Network Corners
      C. Define Panel B-SPLINE Grid Points (DPBSGP)
        A. Define GP Set for Doublet Network
        B. Define Number of Grid Points to be 9
        C. Put Corner Points in GP Set for Source Network
        D. Put Panel Center Grid Points in GP Set for Source Network
        E. Define Number of Grid Points to be 5
    D. Initialize Table Manager
      A. Form Calling Argument for S-ARRAY
      B. Initialize Table for S-ARRAY (TBINIT)
      C. Form Calling Argument for GP-ARRAY
      D. Initialize Table for GP-ARRAY (TBINIT)
  C. Read Panel Information (RPI)
    A. Get DQG PANEL-SPEC DATA
    B. Get PANEL-POINTS DATA
  D. Compute Singularities at Spline Grid Points (CSSGP)
    A. Initialize for Singularity Type
      A. Define Source Grid Points
      B. Define Doublet Grid Points
      C. Define Last Singularity Type
    B. Check if Singularity Grid Point in Table
      A. Form Key Values for Table
      B. Search Table for Grid Point Value (TBSRCH)
    C. Get B-SPLINE Data Set
      A. Get B-SPLINE Source
      B. Get B-SPLINE Doublet
    D. Form Singularity at Grid Points (FSAGP)
      A. Search for SING-INDEX in Table
      B. Read BLOCKED-LAMBDA
      C. Add Arrays to Table
      D. Form Singularity Vector (FSGVEC)
        A. Initialize Singularity Vector to Zero (ZERO)
        B. Define SNG-VECTOR Entry
      E. Multiply Spline and Singularity Vectors
      F. Store Grid Point Values in Table
    E. Define Grid Point Table Location (DGPTL)
      A. Define Singularity Location in GP-ARRAY Table
      B. Define Panel Grid Point
      C. Define Last Source Grid Point
      D. Define Last Source GP-ARRAY Location

F. Form Subpanel Spline Vector (FSPSVC)
    A. Determine Table Location
    B. Move Table Values to Singularity Vector

E. Store Grid Point Singularities (STOGPS)
    A. Search for Grid Point
        A. Set FONND False
        B. Set FOUND True
        C. Define GP-ARRAY Location
    B. Compute GP from Subpanel Spline SPSPL
        A. Determine Subpanel of Grid Point
        B. Compute Local Coordinates
        C. Compute Source GP-VALUE (CMPSSV)
        D. Add Computed Grid Singularities to Table (TBADD)
    C. Format GP-LAMBDA
        A. Initialize SRC, DBLT Strengths to Zero
        C. Get Offset in LAMBDA and GP-ARRAYS
        D. Define Source Strength
        E. Move GP-ARRAY to SRC-LAMBDA
        F. Define Doublet Strength
        G. Move GP-ARRAY to DBLT-LAMBDA
    E. Get and Replace GP-LAMBDA
        A. Get GP-LAMBDA (ESGET)
        B. Replace GP-LAMBDA (ESGET)
    D. Put GP-LAMBDA (ESPUT)

F. Compute Singularities at Control Points (CSCP)
    A. Compute Local Control Point Coordinates (LOCORD)
        A. Define Matrix Lengths
        B. Translate Reference Coordinate by SUBPAN Origin
        C. Multiply A-Matrix With Translated Reference Coordinates (CAB)
    B. Compute Source Control Point Values (CMPSSV)
    C. Compute Doublet Control Point Values (CMPDSV)
    D. Store CP-LAMBDA
        A. Form Key Values
        B. Compute Singularity Lengths
            A. Get CP-LAMBDA (ESGET)
            B. Replace CP-LAMBDA (ESREP)
        C. Put CP-LAMBDA (ESPUT)

D. Obtain Average Quantities at Control Points (AQCP)
    A. Open Data Bases and Define Maps
        A. Open Data Bases (DBOPEN)
        B. Define Maps
    B. Read Network and Panel Data
        A. Read NETWORK-SPEC Data (ESGET)
        B. Read Panel Points Data (ESGET)
    C. Get Boundary Condition Data (BCDAT)
        A. Get DQG BNDRY-COND-SPEC-DATA (ESGET)
        B. Get MIC ROW-MAP-INVERSE (ESGET)
        C. Determine RHS-VLAUE (RHSVAL)
            A. Initialize Right Hand Side Values (ZERO)
            B. Get MAG Row Map (ESGET)
            C. GEt RHS-UNKBOW (ESGET)
            D. Move RHS Values to BCVALU

E. Symmetrize RHS-Values
A. Define Scale Factor
B. Initialize Unsymmetrized Array to Zero
C. Compute Unsymmetrized Values
D. Replace Unsymmetrized Value in BCVALU

D. Set Fatal Error
A. Print Fatal Error Message
B. Write Fatal Error Message

E. Get CP-LAMBDA Data
A. Initialize Singularities to Zero
B. Get Singularities of Each Type

D. Obtain PHI-AVE Quantities (OBPHI)
A. Compute PHI-AVE for Stagnation BC
A. PHI-AVE by UPPER SURFACE ANALYSIS
B. PHI-AVE by LOW SURFACE ANALYSIS

B. Calculate PHI from BC
A. Check if PHI is Available from Non-Stagnation Boundary Condition
A. Check Boundary Condition for Zero Entries
B. Set B.C. Flag
B. Decrement Boundary COndition
C. Get PHI from B.C.
D. Print Fatal Error Message
E. Write Fatal Error Message

C. Compute PHI from IC Matrix (ICALC)

E. Obtain VA, VANC, from VIC (OBVA)
A. Compute RHS Index
B. Compute Velocity from IC Matrix (ICALC)

F. Obtain VANC Only (OBVANC)
A. Compute VANC for Stagnation
A. Compute VANC by Upper Surface Analysis
B. Compute VANC by Lower Surface Analysis

B. Calculate VANC from BC
A. Check if VANC is Available from Non-Stagnation Boundary Conditions
A. Check Boundary Condition for Zero Entries
B. Set B.C. Flag
B. Decrement Boundary Condition
C. Get VANC from B.C.
D. Print Fatal Error Message
E. Write Fatal Error Message

C. Get IC-LAMBDA
A. Compute RHS Index
B. Compute VANC from IC Matrix

G. Store Control Point Values (STCPV)
B. Get ONSET-FLOW RHS-DB
C. Form CP-DATA
D. Put CP-DATA (ESPUT)

E. Form CP-BLOCK-DATA
   A. Put PHI CP-BLOCK-DATA (ESPUT)
   B. Put WANC CP-BLOCK-DATA (ESPUT)
   C. Put VA-X CP-BLOCK-DATA (ESPUT)
   D. Put VA-Y CP-BLOCK-DATA (ESPUT)
   E. Put VA-Z CP-BLOCK-DATA (ESPUT)

I. Compute Control Point Flow Data on Smooth Abutment Segments (CPFDSS)
   A. Set up Data on Segment Length
   B. Initialize Arrays
   C. Get Control Point Data and DQG Spline Vector (ESGET)
   D. Find Control Point Index Which Corresponds to Singularity Parameter Index
   E. Write Fatal Error Message
   C. Get Control Point Block Data (ESGET)
   F. Accumulate Data to Output Arrays
   G. Write Block Data for Null Control Point on Edge of Smooth Abutment (ESPUT)

J. Compute Control Point Flow Data on Collapsed Edges (CPFDCE)
   A. Define Lattice Indices 1st Point
   B. Define Lattice Increments
   C. Get Control Point Index
   D. Clear Flow Data Array
   E. Get Blocked Row Data for 1st Point
   F. Get Control Point Index Edge Midpoint
   G. Write Flow Data for Null Control Point

H. Close Data Bases (PACLOS)

E. Compute BP Spline Vectors (BPN) Overlay (5,0)
  A. Open Database and Define Maps (PAOPEN/DSMAP/SVMAP/ENDMAP)
  B. Get NETWK-SPEC Data (ESGET)
  C. Compute Doublet Analysis Potential Spline (ANALP)
    A. Initialize BP-SPLINE Data
    B. Calculate Network Edge Spline Vectors (NETEDG)
      A. Compute Lattice Indices for First Corner Point (LATIND)
        A. Get CONTROL-PT-SPEC Data
        B. Define Unit Values
        C. Move Control Point Data
      B. Compute Spline Vector for First Corner Point (SPLAP)
        A. Compute LSQ Data for Surrounding Points (DATPOT)
          A. Initialize
            A. Initialize Number of Points
            B. Initialize Reference Point
          B. Define Lattice Indicees
            A. Compute Lattice Indices for Point
            B. Determine New Lattice Indices
          C. Increment Counter
          D. Define LSQ Data for Point (LSQPOT)
            A. Get CONTROL-PT-SPEC
            B. Define Unit Values
            C. Move Control Point Data
          F. Define XI, ETA and ZETA Vectors (XIETAV)
            A. Define Lattice Coordinates of Point
            B. Compute Coordinate of Point
            C. Define XI and ETA Vectors
            D. Define ZETA Vector
              B. Compute Magnitude of ZETA Vector
              C. Normalize ZMAQ by Taking Fourth Root
              D. Normalize ZETA Vector
              E. Define Zero Vector
          G. Compute Coordinate Transformation (SPLTRN)
            A. Define Vector from Two Points
            B. Define XIBAR Component
            C. Define ETABAR Component
            D. Define ZETBAR Component
            E. Define Factor for Scaling Two Dimensional Coordinate
              A. Define Unit Factor
              B. Defing Factor
            F. Define Two Dimensional Coordinate
            G. Program Error in Selection of XI and ETA Vectors
              A. Increment Error Count
              B. Take Error Exit
              C. Print Messages
          H. Compute Weights (WTLSQ)
            A. Define Vector from Origin to Point
            B. Compute Weight
            D. Compute Two Dimensional Radius Squared
            E. Normalize Weight
            F. Define Constrained Quadratic Least Squares

B. Perform Constrained Quadratic Least Squares Fit (CQLSF)
C. Spline Error
A. Increment Error
B. Take Error Exit
C. Print Messages
D. Clear Spline Vector
D. Print Warning Messages
E. Accumulate Spline Vector Contributions (VECUNV)
A. Initialize Index Array
A. Take Error Exit
E. Define New Component of Union Vector
A. Increment Numer of Components
B. Take Error Exit
C. Define Union Vector (ESPUT)
C. Compute Lattice Indices
D. Write BP-SPLINE-VECTOR
F. Compute Lattice Indices of Column Edge Midpoint (Corner) (LATIND)
G. Compute Spline Vector Row Edge Midpoint (SPLAP)
C. Compute Unit Potential Spline Vector
A. Compute Lattice Indices of Center Point
B. Form Unit BP-SPLINE (UNIPOT)
D. Get Arrays of Corner and Edge Midpoints
B. Copy Columns Two Through Five into Columns One Through Four
C. Get Next Column of Corner Points
F. Compute Spline Vector for Corner Point (SPLAPO
D. Print Fatal Error Message
E. Write Error Data Set
A. Form Error Information
B. Put MDGF Error Data
C. Take Fatal Error Exit
F. Obtain Average Data at Grid Points (Overlay (6,0))
A. Open Data Base and Define Maps
A. Open Data Bases
B. Define Maps
B. Get Network Specification Data
C. Determine BC Stagnation Option
D. Assemble Network Geometry Data (NETGEM)
A. Initialize
B. Get Panel Points Geometry Data from MDGF
C. Form Grid of Panel Center Control Points
D. Form Fine Grid Geometry Data for Network
E. Compute Spline Data for Entire Network (SPLCP)
A. Define Fine Grid Dimensions
B. Define Spline Vectors for Mesh Points
C. Adjust Splines for Edge Mesh Points Along any Collapsed Edges
D. Evaluate Panel Center Spline Vectors

E. Assemble Flow Data for Center Control Points (CPFLOW)
   A. Initialize
   B. Get Flow Data for Center Control Points from MDG
   C. Store VIC Velocity Components and Normal Perturbation Mass Flux
   D. Read Potential Data from Edge and Additional Control Points
F. Compute Flow Data at Grid Points (VSPRED)
   A. Spread Normal Mass Flux Data (WNSPRD)
   B. Spread Velocity Data (VSPRET)
G. Compute and Store Average Flow Data at Grid Points (STGP)
   A. Initialize
   B. Form Grid Point Set Sequence
   C. Get Singularity Values for Grid Points in the Set
   D. Assemble Singularity Data for Grid Points
   E. Compute Averae Perturbation Potential and Normal Mass Flux for the Grid Points
      A. Determine Fine Grid Row and Column Indicies
      B. Get BP-Spline Vectors from MDGF
      C. Compute Potential Using Spline Vectors
      D. Compute Potential and Normal Mass Flux for Lower Surface Stagnation Solution
      C. Compute Potential and Normal Mass Flux for Upper Surface Stagnation Solution
   F. Assemble VIC Velocity Data for Grid Points
H. Compute Flow Data at Edge Control Points (EDGECP)
   A. Form and Initialize Loop Control Table
   B. Get Loop Control Parameters from Table
   C. Read Edge Control Point Data from MDG
   D. Compute VIC Velocity and Normal Mass Flux from Corresponding Grid Point Values
   E. Store Control Point Data in MDG
I. Close Data Bases
G. Print Error and Accounting Information (Overlay (7,0)) (EASY)
   A. Read Error Dataset (ESGET)
   B. Print Error Summary
   C. Formulate Accounting Summary
   D. Print Accounting Summary
   E. Write Accounting Summary
      B. Write DB-HEADER
      C. Close and Release MDGF-DB (PACLOS)
      D. Close MDG-DB (PACLOS)

APPENDIX 8-C

DATA BASE COMMUNICATIONS CHART

The Data Base Communications Chart is presented in three forms. Each form is alphabetized by columns, from left to right. The first form has a column order of Data Base, Dataset Name, Map Name, Common Block, and Program/Subroutine. The second form has a column order of the Data Base, Map Name, Dataset Name, Common Block, and Program/Subroutine. The third form has a column order of Common Block, Data Base, Map Name, Dataset Name, and Program/Subroutine. Thus a person can get a cross reference on a data element by knowing either the Dataset Name, Map Name or Common Block name.

FIRST FORM

Overlay (1,0)

DATA BASE	DATASET NAME	MAP NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
DIP	GLOBAL	GLOBAL	/GLOBDT/	OPCKDB
DQG	CONTROL-PT-SPEC	CPSPEC	Dynamic*	OPCKDB
DQG	GLOBAL	GLODQG	/GLOBDT/	OPCKDB
DQG	MDG-PANEL-SPEC	PANELS	/GPGEOM/ /PANDAT/	OPCKDB
DQG	NETWK-SPEC	NETWK	/NETDAT/ Dynamic*	OPCKDB
DQG	PANEL-SPEC	PANELS	/PANDAT/ Dynamic*	OPCKDB
DQG	SPECIAL-POINTS	SPECPT	Dynamic*	OPCKDB
MAK	MAG-PANEL-DATA	MAG-PAN	/PANPKW/	OPCKDB
MAK	SYMMETRY	MAGSYMM	/SYMMET/	OPCKDB
MDG	CP-GEOM	CPFGEOM	/CPGEOM/	OPCKDB
MDG	GLOBAL	GLOBMD	/GLOBDT/ Dynamic*	OPCKDB
MDG	GLOBAL	GLOBMID	/NETLST/	OPCKDB
MDG	GP-GEOM	GPGEOM	/GPGEOM/	OPCKDB
MDG	MAG-PANEL-DATA	MDGPAN	/PANPKW/	OPCKDB
MDG	NETWORK-SPEC	NETWKS	Dynamic*	OPCKDB
MDG	SOLUTION-DATA	MDGSOL	/SOLLID/	OPCKDB
MDGF	ERROR	ERR	/MDGERR/	OPCKDB
MDGF	PANEL-POINTS	PANPTS	/CPDAT/ /GPSET/ Dynamic*	OPCKDB
RHS	SOLUTION-DATA	SOLDAT	/SOLLID/ /SOLLST/	INITCB

Overlay (2,0)

DATA BASE	DATASET NAME	MAP NAME	COMMON BLOCKS	PROGRAM/ SUBROUTINE
MDGC	BLOCKED-LAMBDA	BLKLAM	/BLKLAM/ Dynamic*	PMPY
MDGC	IC-LAMBDA	ICLAMB	/ICLAM/ Dynamic*	PMPY
MDGC	ERROR	ERR	/MDGERR/	PMPY
MDGM	ROW-BLOCK-MATRIX	RBLMAT	/BLAMB/ Dynamic*	PMPY
MAK	COLMAPO	COL3	Dynamic*	PMPY
MAK	SYMMETRY	SYMMAP	/SYMMET/	PMPY
RHS	BLOCK-INFO	BLRHIN	/BLKIFO/	PMPY
RHS	LAM-MAT	LAMMAP	Dynamic*	PMPY
RHS	LAMBDA-KNOWN	LAMKWN	/MAGAP/ Dynamic*	PMPY
RHS	LAMBDA-UNKNOWN	LAMUNK	/MAGAR/	PMPY

Overlay (3,0)

DATA BASE	DATASET NAME	MAP NAME	COMMON BLOCKS	PROGRAM/ SUBROUTINE
DQG	NETWK-SPEC	NETWK	/NETDAT/ Dynamic*	OPDBM
DQG	PANEL-SPEC	PANELS	/PANDAT/ Dynamic*	OPDBM
DQG	CONTROL-PT-SPEC	CPSPEC	Dynamic*	OPDBM
DQG	B-SPLINE-DOUBLET	SPLINE	/BSPLIN/ Dynamic*	OPDBM
DQG	B-SPLINE-SOURCE	SSPLINE	/BSPLIN/ Dynamic*	OPDBM
MDGC	CP-LAMBDA	CPLMB	/CPLAMB/ Dynamic*	OPDBM
MDGC	BLOCKED-LAMBDA	BLKLAM	/LAMBLK/ Dynamic*	OPDBM
MDGF	PANEL-POINTS	PANPTS	/CPDAT/ /GPSET/ Dynamic*	OPDBM
MDGF	GP-LAMBDA	GPLAM	/GPBLAM/ Dynamic*	OPDBM

Overlay (4,0)

DATA BASE	DATASET NAME	MAP NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
DQG	NETWK-SPEC	NETWK	/NETDAT/ Dynamic*	AQCP
DQG	BNDRY-CONDN-SPEC	BNDRY	/BCVALU/ Dynamic*	AQCP
MDG	CP-DATA	CPDATA	/DATACP/	AQCP
MDG	CP-DATA	CPDATA	Dynamic*	AQCP
MDGC	CP-LAMBDA	CPLMB	Dynamic*	AQCP
MDGF	PANEL-POINTS	PANPTS	/CPDAT/ /GPSET/ Dynamic*	AQCP
MDGF	ERROR	ERR	/MDGERR/	AQCP
MDGF	CP-BLOCK-DATA	CPBLKD	Dynamic*	AQCP
MAK	IC-MATRICES	ICMAT	/MAGPAR/ Dynamic*	AQCP
				AQCP
MAK	ROWMAP-INVERSE	ROWIN3	Dynamic*	AQCP
MAK	ROWMAP	ROW3	Dynamic*	AQCP
RHS	SING-KNOWN	SNGKWN	Dynamic*	AQCP
RHS	LAMBDA-UNKNOWN	LAMUNK	Dynamic*	AQCP
RHS	RHS-KNOWN	SKWN	Dynamic*	AQCP
RHS	RHS-UNKNOWN	SUNK	Dynamic*	ACQP
RHS	ONSET-FLOW	RHSOSF	Dynamic*	AQCP

Overlay (5,0)

DATA BASE	DATASETNAME	MAP NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
DQG	CONTROL-PT-SPEC	CPSPEC	Dynamic*	BPSV
DQG	PANEL-CORNER-COORDS	COORDS-GEN	Dynamic*	BPSV
DQG	NETWK-SPEC	NETWK	/NETDAT/ Dynamic*	BPSV
MDGF	ERROR	ERR	/MDGERR/	BPSV
MDGF	B-SPLINE-VECTOR	BPSVEC	/BPSPL/	BPSV

Overlay (6,0)

DATA BASE	DATASET NAME	MAP NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
DQG	MDG-PANEL-SPEC	PANELS	/GPSET/ Dynamic*	GPQTY
DQG	NETWK-SPEC	NETWK	/NETDAT/ Dynamic*	GPQTY
MDG	CP-DATA	CPDATA	/DATACP/ Dynamic*	GPQTY
MDG	GP-DATA	GPDATA	/DATGP/ Dynamic*	GPQTY
MDGF	PANEL-POINTS	PANPTS	/CPDAT/ /GPSET/ Dynamic*	GPQTY
MDGF	BP-SPLINE-VECTOR	SPSVEC	/BPSPL/ Dynamic*	GPQTY
MDGF	CP-BLOCK-DATA	CBBLKD	Dynamic*	GPQTY
MDGF	GP-LAMBDA	GPLAM	/GPBLAM/ Dynamic*	GPQTY
MDGF	ERROR	ERR	/MDGERR/	GPQTY

Overlay (7,0)

DATA BASE	DATASET-NAME	MAP NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
MDG	DATA-BASE-HEADER	DBHEAD	/RUNID/	EASY
MDGF	ERROR	ERR	/MDGERR/	EASY

SECOND FORM

Overlay (1,0)

DATA BASE	MAP NAME	DATASET NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
DIP	GLOBAL	GLOBAL	/GLOBDT/	OPCKDB
DQG	CPSPEC	CONTROL-PT-SPEC	Dynamic*	OPCKDB
DQG	GLODQG	GLOBAL	/GLOBDT/	OPCKDB
DQG	NETWK	NETWK-SPEC	/NETDAT/ Dynamic*	OPCKDB
DQG	PANELS	MDG-PANEL-SPEC	/PANDAT/ /GPGEOM/	OPCKDB
DQG	SPECPT	SPECIAL-POINTS	Dynamic*	OPCKDB
MAK	MAG-PAN	MAG-PANEL-DATA	/PANPKW/	OPCKDB
MAK	MAGSYMM	SYMMETRY	/SYMMET/	OPCKDB
MDG	CPFGEOM	CP-GEOM	/CPGEOM/	OPCKDB
MDG	GLOBMD	GLOBAL	/GLOBDT/ Dynamic*	OPCKDB
MDG	GLOBMID	GLOBAL	/NETLST/	OPCKDB
MDG	GPGEOM	GP-GEOM	/GPGEOM/	OPCKDB
MDG	NETWKS	NETWORK-SPEC	Dynamic*	OPCKDB
MDG	MDGPAN	MAG-PANEL-DATA	/PANPKW/	OPCKDB
MDG	MDGSOL	SOLUTION-DATA	/SOLLID/	OPCKDB
MDGF	ERR	ERROR	/MDGERR/	OPCKDB
MDGF	PANPTS	PANEL-POINTS	/CPDAT/ /GPSET/ Dynamic*	OPCKDB
RHS	SOLDAT	SOLUTION-DATA	/SOLLID/ /SOLLST/	INITCB

Overlay (2,0)

DATA BASE	MAP NAME	DATASET NAME	COMMON BLOCKS	PROGRAM/ SUBROUTINE
MDGC	BLKLAM	BLOCKED-LAMBDA	/BLKLAM/ Dynamic*	PMPY
MDGC	ICLAMB	IC-LAMBDA	/ICLAM/ Dynamic*	PMPY
MDGC	ERR	ERROR	/MDGERR/	PMPY
MDGM	RBLMAT	ROW-BLOCK-MATRIX	/BLAMB/ Dynamic*	PMPY
MAK	COL3	COLMAPO	Dynamic*	PMPY
MAK	SYMMAP	SYMMETRY	/SYMMET/	PMPY
RHS	BLRHIN	BLOCK-INFO	/BLKIFO/	PMPY
RHS	LAMMAP	LAM-MAT	Dynamic*	PMPY
RHS	LAMKWN	LAMBDA-KNOWN	/MAGAP/ Dynamic*	PMPY
RHS	LAMUNK	LAMBDA-UNKNOWN	/MAGAR/	PMPY

Overlay (3,0)

DATA BASE	MAP NAME	DATASET NAME	COMMON BLOCKS	PROGRAM/ SUBROUTINE
DQG	NETWK	NETWK-SPEC	/NETDAT/ Dynamic*	OPDBM
DQG	PANELS	PANEL-SPEC	/PANDAT/ Dynamic*	OPDBM
DQG	CPSPEC	CONTROL-PT-SPEC	Dynamic*	OPDBM
DQG	SPLINE	B-SPLINE-DOUBLET	/BSPLIN/ Dynamic*	OPDBM
DQG	SSPLINE	B-SPLINE-SOURCE	/BSPLIN/ Dynamic*	OPDBM
MDGC	CPLMB	CP-LAMBDA	/CPLAMB/ Dynamic*	OPDBM
MDGC	BLKLAM	BLOCKED-LAMBDA	/LAMBLK/ Dynamic*	OPDBM
MDGF	PANPTS	PANEL-POINTS	/CPDAT/ /GPSET/ Dynamic*	OPDBM
MDGF	GPLAM	GP-LAMBDA	/GPBLAM/ Dynamic*	OPDBM

Overlay (4,0)

DATA BASE	MAP NAME	DATASET NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
DQG	NETWK	NETWK-SPEC	/NETDAT/ Dynamic*	AQCP
DQG	BNDRY	BNDRY-CONDN-SPEC	/BCVALU/ Dynamic*	AQCP
MDG	CPDATA	CP-DATA	/DATACP/	AQCP
MDG	CPDATA	CP-DATA	Dynamic*	AQCP
MDGC	CPLMB	CP-LAMBDA	Dynamic*	AQCP
MDGF	PANPTS	PANEL-POINTS	/CPDAT/ /GPSET/ Dynamic*	AQCP
MDGF	ERR	ERROR	/MDGERR/	AQCP
MDGF	CPBLKD	CP-BLOCK-DATA	Dynamic*	AQCP
MAK	ICMAT	IC-MATRICES	/MAGPAR/ Dynamic*	AQCP
			UPDCPKNUKICT	AQCP
MAK	ROWIN3	ROWMAP-INVERSE	Dynamic*	AQCP
MAK	ROW3	ROWMAP	Dynamic*	AQCP
RHS	SNGKWN	SING-KNOWN	Dynamic*	AQCP
RHS	LAMUNK	LAMBDA-UNKNOWN	Dynamic*	AQCP
RHS	SKWN	RHS-KNOWN	Dynamic*	AQCP
RHS	SUNK	RHS-UNKNOWN	Dynamic*	ACQP
RHS	RHSOSF	ONSET-FLOW	Dynamic*	AQCP

Overlay (5,0)

DATA BASE	MAP NAME	DATASETNAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
DQG	CPSPEC	CONTROL-PT-SPEC	Dynamic*	BPSV
DQG	COORDS-GEN	PANEL-CORNER-COORDS	Dynamic*	BPSV
DQG	NETWK	NETWK-SPEC	/NETDAT/ Dynamic*	BPSV
MDGF	ERR	ERROR	/MDGERR/	BPSV
MDGF	BPSVEC	B-SPLINE-VECTOR	/BPSPL/	BPSV

Overlay (6,0)

DATA BASE	MAP NAME	DATASET NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
DQG	NETWK	NETWK-SPEC	/NETDAT/ Dynamic*	GPQTY
DQG	PANELS	MDG-PANEL-SPEC	/GPSET/ Dynamic*	GPQTY
MDG	CPDATA	CP-DATA	/DATACP/ Dynamic*	GPQTY
MDG	GPDATA	GP-DATA	/DATGP/ Dynamic*	GPQTY
MDGF	PANPTS	PANEL-POINTS	/CPDAT/ /GPSET/ Dynamic*	GPQTY
MDGF	SPSVEC	BP-SPLINE-VECTOR	/BPSPL/ Dynamic*	GPQTY
MDGF	CBBLKD	CP-BLOCK-DATA	Dynamic*	GPQTY
MDGF	GPLAM	GP-LAMBDA	/GPBLAM/ Dynamic*	GPQTY
MDGF	ERR	ERROR	/MDGERR/	GPQTY

Overlay (7,0)

DATA BASE	MAP NAME	DATASET-NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
MDG	DBHEAD	DATA-BASE-HEADER	/RUNID/	EASY
MDGF	ERR	ERROR	/MDGERR/	EASY

THIRD FORM

Overlay (1,0)

COMMON BLOCK	DATA BASE	MAP NAME	DATASET NAME	PROGRAM/ SUBROUTINE
/GLOBDT/	DIP	GLOBAL	GLOBAL	OPCKDB
/GLOBDT/	DQG	GLODQG	GLOBAL	OPCKDB
/GPGEOM/	DQG	PANELS	MDG-PANEL-SPEC	OPCKDB
Dynamic*	DQG	CPSPEC	CONTROL-PT-SPEC	OPCKDB
/NETDAT/	DQG	NETWK	NETWK-SPEC	OPCKDB
Dynamic*				
/PANDAT/	DQG	PANELS	MDG-PANEL-SPEC	OPCKDB
Dynamic*	DQG	SPECPT	SPECIAL-POINTS	OPCKDB
/PANPKW/	MAK	MAG-PAN	MAG-PANEL-DATA	OPCKDB
/PANPKW/	MAK	MAGSYMM	SYMMETRY	OPCKDB
/CPGEOM/	MDG	CPFGEOM	CP-GEOM	OPCKDB
/GLOBDT/	MDG	GLOBMD	GLOBAL	OPCKDB
Dynamic*				
/NETLST/	MDG	GLOBMID	GLOBAL	OPCKDB
/PANPKW/	MDG	MDGPAN	MAG-PANEL-DATA	OPCKDB
/GPGEOM/	MDG	GPGEOM	GP-GEOM	OPCKDB
Dynamic*	MDG	NETWKS	NETWORK-SPEC	OPCKDB
/SOLLID/	MDG	MDGSOL	SOLUTION-DATA	OPCKDB
/MDGERR/	MDGF	ERR	ERROR	OPCKDB
/CPDAT/	MDGF	PANPTS	PANEL-POINTS	OPCKDB
/GPSET/				
Dynamic*				
/SOLLID/	RHS	SOLDAT	SOLUTION-DATA	INITCB
/SOLLST/				

Overlay (2,0)

COMMON BLOCKS	DATA BASE	MAP NAME	DATASET NAME	PROGRAM/ SUBROUTINE
/BLKLAM/	MDGC	BLKLAM	BLOCKED-LAMBDA	PMPY
Dynamic*				
/ICLAM/	MDGC	ICLAMB	IC-LAMBDA	PMPY
Dynamic*				
/MDGERR/	MDGC	ERR	ERROR	PMPY
/BLAMB/	MDGM	RBLMAT	ROW-BLOCK-MATRIX	PMPY
Dynamic*				
Dynamic*	MAK	COL3	COLMAPO	PMPY
/SYMMET/	MAK	SYMMAP	SYMMETRY	PMPY
/BLKIFO/	RHS	BLRHIN	BLOCK-INFO	PMPY
Dynamic*	RHS	LAMMAP	LAM-MAT	PMPY
/MAGAP/	RHS	LAMKWN	LAMBDA-KNOWN	PMPY
Dynamic*				
/MAGAR/	RHS	LAMUNK	LAMBDA-UNKNOWN	PMPY

Overlay (3,0)

COMMON BLOCKS	DATA BASE	MAP NAME	DATASET NAME	PROGRAM/ SUBROUTINE
/NETDAT/	DQG	NETWK	NETWK-SPEC	OPDBM
Dynamic*				
/PANDAT/	DQG	PANELS	PANEL-SPEC	OPDBM
Dynamic*				
Dynamic*	DQG	CPSPEC	CONTROL-PT-SPEC	OPDBM
/BSPLIN/	DQG	SPLINE	B-SPLINE-DOUBLET	OPDBM
Dynamic*				
/BSPLIN/	DQG	SSPLINE	B-SPLINE-SOURCE	OPDBM
Dynamic*				
/CPLAMB/	MDGC	CPLMB	CP-LAMBDA	OPDBM
Dynamic*				
/LAMBLK/	MDGC	BLKLAM	BLOCKED-LAMBDA	OPDBM
Dynamic*				
/CPDAT/	MDGF	PANPTS	PANEL-POINTS	OPDBM
/GPSET/				
Dynamic*				
/GPBLAM/	MDGF	GPLAM	GP-LAMBDA	OPDBM
Dynamic*				

Overlay (4,0)

COMMON BLOCK	DATA BASE	MAP NAME	DATASET NAME	PROGRAM/ SUBROUTINE
/NETDAT/	DQG	NETWK	NETWK-SPEC	AQCP
Dynamic*				
/BCVALU/	DQG	BNDRY	BNDRY-CONDN-SPEC	AQCP
Dynamic*				
/DATACP/	MDG	CPDATA	CP-DATA	AQCP
Dynamic*	MDG	CPDATA	CP-DATA	AQCP
Dynamic*	MDGC	CPLMB	CP-LAMBDA	AQCP
/CPDAT/	MDGF	PANPTS	PANEL-POINTS	AQCP
/GPSET/				
Dynamic*				
/MDGERR/	MDGF	ERR	ERROR	AQCP
Dynamic*	MDGF	CPBLKD	CP-BLOCK-DATA	AQCP
/MAGPAR/	MAK	ICMAT	IC-MATRICES	AQCP
Dynamic*				
UPDCPKNUKICT				AQCP
Dynamic*	MAK	ROWIN3	ROWMAP-INVERSE	AQCP
Dynamic*	MAK	ROW3	ROWMAP	AQCP
Dynamic*	RHS	SNGKWN	SING-KNOWN	AQCP
Dynamic*	RHS	LAMUNK	LAMBDA-UNKNOWN	AQCP
Dynamic*	RHS	SKWN	RHS-KNOWN	AQCP
Dynamic*	RHS	SUNK	RHS-UNKNOWN	ACQP
Dynamic*	RHS	RHSOSF	ONSET-FLOW	AQCP

Overlay (5,0)

COMMON BLOCK	DATA BASE	MAP NAME	DATASETNAME	PROGRAM/ SUBROUTINE
Dynamic*	DQG	CPSPEC	CONTROL-PT-SPEC	BPSV
Dynamic*	DQG	COORDS-GEN	PANEL-CORNER-COORDS	BPSV
/NETDAT/ Dynamic*	DQG	NETWK	NETWK-SPEC	BPSV
/MDGERR/	MDGF	ERR	ERROR	BPSV
/BPSPL/	MDGF	BPSVEC	B-SPLINE-VECTOR	BPSV

Overlay (6,0)

COMMON BLOCK	DATA BASE	MAP NAME	DATASET NAME	PROGRAM/ SUBROUTINE
/DATACP/ Dynamic*	MDG	CPDATA	CP-DATA	GPQTY
/GPSET/ Dynamic*	DQG	MDG-PANEL-SPEC	PANELS	GPQTY
/NETDAT/ Dynamic*	DQG	NETWK	NETWK-SPEC	GPQTY
/DATGP/ Dynamic*	MDG	GPDATA	GP-DATA	GPQTY
/CPDAT/ /GPSET/ Dynamic*	MDGF	PANPTS	PANEL-POINTS	GPQTY
/BPSPL/ Dynamic*	MDGF	SPSVEC	BP-SPLINE-VECTOR	GPQTY
Dynamic*	MDGF	CBBLKD	CP-BLOCK-DATA	GPQTY
/GPBLAM/ Dynamic*	MDGF	GPLAM	GP-LAMBDA	GPQTY
/MDGERR/	MDGF	ERR	ERROR	GPQTY

Overlay (7,0)

COMMON BLOCK	DATA BASE	MAP NAME	DATASET-NAME	PROGRAM/ SUBROUTINE
/RUNID/	MDG	DBHEAD	DATA-BASE-HEADER	EASY
/MDGERR/	MDGF	ERR	ERROR	EASY

* Dynamic mapping is used for some or all elements of a dataset thus requiring no common block storage. See section 13 of this document for details of dynamic mapping.

APPENDIX 8-D MASTER DEFINITION

The data base master definition listing of the MDG module has been deleted from this document. It is produced from the PAN AIR tape during installation.

APPENDIX 8-E SYMMETRIZATION

The PAN AIR Theory Document (Reference 1) shows how the cost of solutions to the potential flow problem can be significantly reduced in the case that the configuration (and/or flow) is symmetric about some plane called a plane of symmetry (POS). When flow symmetry exists, the singularity parameters provided by RHS and the IC matricies provided by MAG are linear combinations of values on either side of the plane of symmetry. However, for the evaluation of the pressures, forces, and moments on the configuration by module PDP and CDP, it is necessary to use the values of singularity parameters and flow quantities at the actual points on the configuration. This process of obtaining the values of parameters at the points from the linear combinations of parameters (symmetrized parameters) is referred to as unsymmetrization. A general algorithm for unsymmetrizing 1, 2 or 4 distinct images is given.

The known singularities from RHS are already unsymmetrized. Quantities unsymmetrized by MDG are the unknown singularities stored on the RHS data base and the symmetrized flow quantity values obtained by multiplying symmetrized singularities by symmetrized IC matrix values. The latter flow quantities are potential, normal mass flux, and average velocity. The unsymmetrized quantities stored by MDG have not been scaled in sign when reflected across the POS.

There are potentially four quadrants defined by up to two planes of symmetry. (For visualization consider the x-z plane as the first POS and the x-y plane as the second POS.) If no POS exists, or there is no flow asymmetry with respect to either POS then there is only one distinct image. If flow asymmetry exists with respect to only one POS then there are two distinct images. If flow asymmetry exists with respect to both POS, four distinct images exist.

MDG stores data on the CP-DATA and GP-DATA datasets by distinct images and solutions. However, throughout its internal processing, unsymmetrized quantities are stored in blocks, by images, for each solution with a maximum of 36 quantities per block; for one distinct image up to 36 solutions, for two distinct images up to 18 solutions, and for four distinct images up to 9 solutions, respectively for each block of data.

The general algorithm is as follows:

Data Definition:

U unsymmetrized quantities indexed by distinct image and solution

F scale factor equal to 1/2 raised to the number of planes of symmetry

S symmetrized quantities indexed by distinct image and solution

SIGN a four by four matrix defined as

```
SIGN(I,J) =  1   if I=1 or J=1 or I+J=6
          = -1   otherwise
```

Execution sequence:

```
Initialize U to zero

For each distinct image I do

    For each solution K do

        For each distinct image J do

            Retrieve S(J,K)

            U(I,K) = U(I,K) + F * SIGN(I,J) * S(J,K)

        Enddo on distinct image J

    Enddo on solution K

Enddo on distinct image I
```

APPENDIX 8-F MDG LIBRARY FUNCTIONAL DECOMPOSITION

This functional decomposition of the subprograms, which are used in different levels of the MDG module, is presented here. These subprograms are used only in the MDG module of the PAN AIR system and, so, physically reside in the module.

MDG Library - General

CMPDSV Compute Doublet Singularity

A. Multiply Subpanel Doublet Spline and Singularity Vector
C. Compute MV
D. Store Doublet Singularity

CMPSSV Compute Source Singularity Value

A. Multiply Subpanel Source Spline and Singularity Vector
B. Compute Source Singularity at Point
C. Store Source Singularity

ICALC Compute IC Values at Control Point

A. Clear Result Array (ZERO)
B. Define Updatability and Known Indicies for Partition
C. Get Pointer of IC Row
D. Define Pointer to Singularity Vector
E. Get Partition of Singularities
F. Compute Inner Product of IC Row and Singularity Vector
G. Accumulate Contribution to Result Array and Unsymmetrize
H. Fatal Error/IC Row and Singularity Vector

TBADD Add to Table

A. Determine Table Number
B. Increment REF-COUNT
C. Get Free Space Table
D. Add Entry to Table
E. Add Key to Table Data
F. Print Errors

TBDELT Delete Least Recently Used from Table

A. Initialize Smallest
B. Set Smallest
C. Delete Key Table Entry
D. Move Key Up One

TBINIT Initialize Table Manager for Table

A. Edit Inputs
B. Define Key Table and Initialize Ref-Count
C. Initialize Number of Table Entries
D. Initialize Free Space Table
E. Print Error Message
F. Set Error Return Code

TBSRCH Search Table for Entry

A. Determine Table Number
B. Determine Number of Table Entries
C. Set Not Found
D. Binary Search on Key Entries
E. Print Errors

FSTBAD Free Space Table Add

A. Initialize Merge Flag
B. Check for Merge
C. Increment Block Count
D. Increment Number of Free Space Sets
E. Change Last Location
F. Define New Free Space Loc, NUM-BLKS = 1
G. Merge Blocks of Free Space
H. Define Free Space Block

FSTBDL Free Space Table Delete

A. Define Free Space Location
B. Define Last Location
C. Decrement BLOCK-COUNT
D. Decrement NUM-LOCS
E. Define Last Location

DPANCP Define Panel Control Points

A. Define PANEL POINTS
B. Search of Panel Corner Points (SPANCP)
C. Form Control Point Data for Corner Points (FCPDAT)
D. Form Control Point Data for Edge Midpoint (FCPDAT)
E. Form Control Point Data for Center Point (FCPDAT)

DINGPS Define Interior Grid Point Set

A. Set IPNROW equal to PANEL-ROW
B. Set IPNCOL equal to PANEL-COL
C. Set number of GPSETS to one
D. Set number of points in set to four
E. Define grid point set 1 points 1,5,8,9
F. Define panel fine grid point

FCPDAT Form Control Point Data

A. Define CP Lattice Key
B. Get CONTROL-PT-SPEC DATA (ESGET)
C. Move Data to /CPDAT/
D. Take SDMSRR

GETSPT Get SPECIAL-POINTS

A. Form Key
B. Get Special Points (ESGET)
C. Get Special Points (ESGET)
D. Take SDMS Error Exit (SDMSRR)

MCPDAT Move Data to /MCPDAT/

A. Increment CP Count
B. Move Subpanel Number
C. Move CP+INDEX
D. Move CP-COORDS
E. Move Panel Points Number to CPANPT

PUTPPT Put Panel Points

A. Form Key
B. Put Panel Points (ESPUT)

SPANCP Search for Panel Corner on Edge

A. Set Flag and Initialize
B. Determine Row/Col Parameters
C. Compare Points

SDMSRR SDMS Error Exit

A. Set Fatal Flag
B. Increment Fatal Error Count
C. Print Fatal Error Diagnostic
D. Form MDGERR Data
E. Form Non-Fatal MDG Error Data
F. Increment Total Error Count
G. Put MDGF-ERROR Data (ESPUT)

APPENDIX 8-G MDG LIBRARY USAGE

8-G.1 Panel Points Library Usage

In order to determine the control points and grid point sets for each panel in a network, a separate set of routines were written in the MDG (1,0) overlay for use in the MDG (3,0), (4,0), and (6,0) overlays by reading the MDGF data base PANEL-POINTS dataset. These routines consist of both general and special purpose applications to generate panel points data for different network configurations. The general routines define the default grid point set (panel points 1,5,8,9) as shown below:

```
4-----7-----3
'           '
8     9     6
'           '
1-----5-----2
```

These routines also process control point data from the DQG CONTROL-SPEC dataset, getting special points from DQG for a network edge, moving the control point data to the output buffer, executing an SDMS ESPUT of the panel points data, searching for special points on an edge, getting center control point data, and computing the panel points for a single panel network, single row or column network. Other routines handle special cases such as single row or single column network, calculating the control points for parallel edges simultaneously, or the case for networks consisting of two panel rows or columns. The control point data for each panel is contained on the /CPDAT/ common block and consists of the number of control points contained in the panel, the DQG control point index, the control point coordinates, the subpanel number of the control point, the panel point number, the fine grid row lattice, and column lattice of each control point. The grid point data contains: the fine grid lattices of the first panel point in each grid point set used as key values for grid point data stored in the overlays (3,0) and (6,0). It also contains the number of grid point sets, the number of grid points in each grid point set and the panel point numbers of the grid points in the sequence.

8-G.2 Table Manager Library Usage

The table manager is a family of programs designed to provide an in-core storage of data associated with splining operations in the MDG (3,0) and (6,0) overlays (although it is more general in its implementation). Use of the table manager is started by a call to TBINIT with arguments (1,1,1,0). This clears out previous table information and initializes the table manager to begin processing. For each table used subsequent calls to TBINIT are made telling the name of the table, number of entries to be contained in the table (must be less than maximum allowable, or an error occurs), number of keys required to retrieve an item from the table (maximum of 5 allowed), and size of a cell in the table (maximum size is 36). Because of the fixed dimensions of the common block arrays used in the table manager only two tables are allowed. This is the maximum currently used by MDG. After initialization the tables are considered empty and reference counts are set to zero. Hereafter, each addition (TBADD call) or search for an entry in the table (TBSRCH) increments the global reference count of the entry if found. This count determines the least recently used entries for the table which will be deleted when the table is full. The actual tables of data are stored by the user and passed to the table manager as a formal parameters in calls to TBADD and TBSRCH. Data in the table is stored through the TBADD routine and is not moved which in the table, except by the user, or through deletion when the table is full. In order to search for an entry in the table a keytable is kept in sorted order by ascending keys. Each time an add is made the entry point of the table data is returned as the formal parameter ILOC. This is the same entry returned on a call to TBSRCH, but ILOC is equal to -1 if the entry is not found in the table. When the table is full the table manager internally calls TBDELT from TBADD which searches through all reference counts until the smallest value is found. Another internal call to FSTDBL deletes this entry from the keytable (KEYTBL common block) and a call to FSTBAD is made adding the location of the newly deleted item as the new free space location (FREESP common block). In its current implementation only one item is deleted when the table is full, but the FSTDBL routine is designed to accommodate multiple deletions from the table which would be more efficient when a large number of table entries are initialized. When entries are added to or deleted from the keytable, the length of the table or equivalently the number of entries in the table is updated. The common block TBDAT has global information formed in the call to TBINIT about the key size, cell size, and maximum number of entries.

8-G.3 SDMS Error Processing

In MDG after each call to a data base using the SDMS calls ESGET, ESPUT, ESREP, DESGET, DESPUT, a check is made on the value of NERR in common block SDMSER which reflects the status of the SDMS call. If the status reflects a fatal error for further MDG processing, SDMSRR is called passing to it the value of NERR, the number of keys used in the SDMS call and their values, and an MDG assigned map number, the location in the program from which the call is made, and which overlay the call is being made from. SDMSRR increments the count of fatal errors contained in NFERR and sets FATAL true in the FATAL common block. SDMSRR writes a record on the error dataset with the information passed to it through the formal parameter list. In the last MDG overlay (7,0) the ERROR dataset is read and a summary of errors is printed.

9.0 POINT DATA PROCESSOR (PDP) MODULE

9.1 INTRODUCTION

The PDP module is a post processor of the PAN AIR system. It presumes the existence of a solution to the potential flow problem and computes perturbation and total surface flow quantities from these data.

Data bases from three PAN AIR modules are required to run PDP. These are the MEC (Module Execution Control), DIP (Data Input Processor) and the MDG (Minimal Data Generator) data bases. The MEC data base contains locational information of all data bases, the DIP data base describes the surface flow options selected by user and the MDG data base provides PDP the configuration geometry and the solution data.

The PDP module consists of a top level program which calls three main overlays. The first overlay checks the status of all data bases required by PDP and processes network specifications and global data. The second overlay computes average and difference velocities at each control and grid point of the selected networks and writes these data in a temporary data base. The third overlay reads these data to compute velocities, mass flux, pressure coefficients and local Mach numbers on selected surfaces (upper, lower, upper minus lower, lower minus upper and average). The computed data is printed out and/or stored in the PDP data base depending upon user selections.

Details of computation of surface flow quantities is given in Section N of the PAN AIR Theory Document (Reference 1). The structure and format of the user input data for surface flow properties computation is described in Section 7.6.1 of the PAN AIR User's Manual (Reference 2).

9.2 PDP OVERVIEW

9.2.1 Purpose of PDP

PDP computes perturbation and total mass flux, velocity, potential and pressures and local Mach numbers at user selected point types (panel center, edge midpoint, and additional control points, fine grid points and points arbitrarily defined by user) and surfaces. The pressures and local Mach numbers are computed for isentropic, linear, second order, reduced second order and slender body approximations, depending upon user selection.

9.2.2 PDP Input/Output Data

9.2.2.1 Input

Input data to PDP module comes from the MEC, DIP and MDG data bases. The MEC data base provides data base names, account number under which each data base resides on disk, date of creation and the status of the data bases, whether complete or incomplete. The DIP data base provides the user selected options for the PDP module. These options include the total number of cases of options for PDP, networks, solutions velocity computation and correction methods to be used and the pressure rules selected.

The MDG module provides PDP with network geometry, source and doublet singularity strengths, perturbation potential, pertubation normal mass flux

and perturbation velocity computed from the IC (Influence Coefficient) matrices. The geometry data consists of network specifications, coordinates of panel control and grid points and doublet spline matrices for subpanels. The potential, velocity and mass flux data provided by MDG are the average values.

9.2.2.2 Output

The surface flow properties (velocities, mass flux, pressure coefficients, local Mach numbers, etc.) computed by PDP for control, grid and arbitrary points are printed and/or stored in the PDP permanent data base for later retrieval by the PPP module. Selection of each item in the output data, whether it should be printed and/or stored in the data base or not to be computed, is controlled by user specifications residing on the DIP data base.

The printed output consists of an estimate on disk storage required for the PDP run, a summary of global options, a list of computation options selected by the user for each case, surface flow properties data selected for printing for each velocity computation method, velocity correction scheme and surface selected by the user.

The PDP permanent data base provides global data, user options for each case, network specifications for the configuration and surface flow properties for control and grid points of the selected networks. Computed flow data for arbitrary points are only printed out and not stored in the PDP data base.

9.2.3 Data Base Interfaces

The MEC, DIP and MDG data bases provide input data to the PDP module. The MEC data base gives the name, account number, date of creation and other locational information for each data base. The DIP data base provides the user options for PDP. The MDG data base provides the global data, solution information, network geometry data, singulatities and perturbation values for velocity, potential and normal mass flux for control and grid points.

PDP creates a temporary data base named PDPT which saves intermediate data computed in the second overlay COMVEL for use in the third overlay FLPROP. This temporary data base contains average and difference perturbation velocities and gradients of doublet strength as computed in COMVEL along with the minimal set of data for control and/or grid points obtained from the MDG data base.

Figure 9.1 illustrates the relationships between PDP and all used data bases.

9.3 MODULE DESCRIPTION

9.3.1 Overall Structure

The main overlays and their subroutines are briefly summarized in this paragraph. The functional decomposition of PDP is given in Appendix 9-B and a subroutine tree structure diagram is presented in Appendix 9-A. Figure 9.2 illustrates the top level structure of the PDP module.

9.3.2 Overlay Descriptions

A summary description of each overlay of the PDP module is given in the following paragraphs.

9.3.2.1 PDP - Overlay (0,0)

The top level overlay (Figure 9.3) of PDP initializes program variables, data bases and controls access to the three primary overlays.

It calls Overlay OPDBI to check the status of DIP and MDG data bases, and defines the necessary data base maps. Then for each case of user requested options, the module calls into execution the two overlays COMVEL and FLPROP in that order. After processing the last case of options, it writes the DATA-BASE-HEADER dataset of the PDP data base, if generated.

9.3.2.2 OPDBI Overlay (1,0)

The second level overlay OPDBI (Figure 9.4) reads the MEC data base to retrieve run and problem identification (RID and PID). It checks the DIP and MDG data bases for completeness and if found unusable, returns a fatal error flag to the main overlay for end of execution. If the DIP and MDG data bases are found complete, OPDBI processes global data and network specifications from the MDG data base and user options from the DIP data base. It generates the PDP data base (if selected by user) and writes GLOBAL abd NETWK-SPEC datasets. While processing the user option cases, OPDBI estimates the disk storage requirements for each case and prints out this information along with the total for the run.

9.3.2.3 COMVEL Overlay (2,0)

The second overlay COMVEL (Figure 9.5) computes the gradients of perturbation potential and doublet strength and then the perturbation average and difference velocities at each control and grid point from a minimal set of data from the MDG data base. Details of velocity computation can be found in Section N.1 of the PAN AIR Theory Document (Reference 1). The computations are performed panel by panel along each column of a network selected by user. This is done for each distinct image, i.e., across a plane of configuration symmetry with asymmetric flow (see description of input record G4 in Section 7 of Reference 2). The gradient of the perturbation potential and the average velocity are computed only if the boundary condition method of velocity computation is selected by user. If the user selects the VIC (Velocity Influence Coefficient method), the average velocity at each point of a panel is obtained from the MDG data base.

COMVEL stores the computed data along with singularity values, perturbation potential, perturbation normal mass flux, etc., as obtained from the MDG data base in the PDPT temporary data base.

9.3.2.4 FLPROP Overlay (3,0)

The third level overlay FLPROP (Figure 9.6) prints global and surface flow options data, defines all necessary maps for PDP data base, calls program PANPTS (Overlay 3,1) to compute surface flow data at control and grid points

and calls program ARBPTS (Overlay 3,2) to compute these data at arbitrary points.

9.3.2.4.1 PANPTS Overlay (3,1)

This secondary overlay PANPTS (Figure 9.7) retrieves the velocity data from PDPT data base and computes the perturbation and total surface flow quantities for each surface selected by the user (UPPER, LOWER, UPLO, LOUP and AVERAGE). Table 9.1 lists these data items. All vector components in the table are in the reference axis system.

From the perturbation average and difference velocities for each point of a panel, PANPTS computes the perturbation and total velocities for each selected surface, corrects these by the user selected correction schemes (SA1, SA2 or NONE, see records SF10b and G11, Section N.3 of Reference 1) and then uses the velocities in computing perturbation, total and total normal mass flux, pressure coefficients and local Mach numbers for isentropic, linear, second order, reduced second order and slender body rules (see Section N.4.2 of Reference 1).

These data are printed and/or stored in the PDP data base depending upon user selections.

9.3.2.4.2 ARBPTS Overlay (3,2)

The secondary overlay ARBPTS processes points arbitrarily defined by the user by providing the network identification, panel row and column indices and coordinates. Program ARBPTS reads these data from DIP data base dataset 'ARBITRARY-POINTS'. For each arbitrary point, the program assembles the geometry and flow data for the grid points of the specified panel by accessing the PDP data base. The given point is then projected onto the panel surface and the index of the subpanel where the projected point lies, is computed. The surface flow quantities are then computed at the point by linear extrapolation of the values at the three grid points forming the subpanel. The computed flow data, the user specified coordinates and the program computed coordinates are then printed out.

9.3.3 PDP Data Bases

A permanent data base named PDP is created by the module if selected by the user. PDP also creates a temporary data base named PDPT for intermediate storage of data. The Master Definitions of PDP and PDPT data bases are described in Appendix 9-D.

9.3.4 PDP Interfaces

9.3.4.1 System Interfaces

The PDP module is accessed through MEC by user control cards and a system procedure file generated by MEC. This interface is described in Sections 1.0 and 2.0 of this document.

9.3.4.2 External Interfaces

The MEC, DIP and MDG data bases are the source of input data for the PDP module. The PDP permanent data base, if generated, is used by the PPP module.

9.3.4.3 Internal Interfaces

The PDPT temporary data base is used by PDP as a scratch file for storing intermediate data.

The interface between the overlays and subprograms is defined by a tree structure diagram in Appendix 9-A.

9.3.5 Data Flow

During execution, data flows between programs, subprograms and the data bases. Figure 9.2 depicts this activity. Subprograms may also communicate with each other by using labelled common or subroutine formal parameters. Information concerning data flow in this manner can be found by consulting the glossaries of the subprograms which are of interest. Section 1, Paragraph 1.4 of this document can be consulted for more detailed information of the use of the tools available for analysis of data flow. Also, Appendix 9-C has been included to aid analysis of data flow between PDP and its data bases.

9.4 LOWER LEVEL FUNCTIONS

The following paragraphs present the functional decompositions (hierarchial structure) of the overlays and their subprograms and purposes of each subprogram.

9.4.1 Functional Decomposition

See Appendix 9-B for a description of the PDP functional decomposition. Section 1, Paragraph 1.4.1 of this document can be consulted for information on the use of the functional decomposition.

9.4.2 Subroutine Descriptions

ANALOP
: Analyzes a case of user options to find disk storage requirement.

ANALYZ
: Reads GLOBAL and options data from DIP data base and analyzes all the user option cases.

ARBFLO
: Computes flow quantities at an arbitrary point by calling subroutine EXTPLT.

CMPORG
: Defines origin of a give subpanel and translates the coordinates of a point in the subpanel to the subpanel coordinate system.

CPGPFL
: Computes flow properties at control and grid points and prints out the data and/or stores the data on PDP data base.

CPVEL

Computes average and difference velocities at control points.

DATAGP

Determines what grid point sets (singularity data from MDG) need to be in core and then assembles the the data by calling subroutine RDGPDT

DATMOV

Transfers grid point geometry and velocity data from common block /PANBLK/ to /GPDATA/.

EDGFLO

Processes control points (edge midpoint and additional) on network edges for computing surface flow properties.

EDGVEL

Processes control points (edge midpoint and additional) on network edges for computing average and difference velocities.

EXTPLT

Computes a flow quantity (either vector or scalar) at an arbitrary point by linear extrapolation of the values at the vertices of the subpanel where it lies.

FNDSUB

Projects an arbitrary point onto the given panel, determines the subpanel where the point lies and then computes the subpanel unit normal vector and its area.

FLXSRF

Computes perturbation, total and total normal mass flux on selected surfaces.

GEOGP

Determines what grid point sets (geometry data from MDG data base) need to be in core and then assembles the data by calling subroutine RDGPGM.

GPGEOM

Computes additional grid point geomtry data needed by PDP. These include normal and conormal vectors and doublet subpanel spline matrices.

GPVEL

Computes the average and difference velocities at grid points of panels.

KAPVEC

Computes the KAPPA vectors (a 9 by 3 matrix) given the subpanel index and the skewness parameters for subpanel vertices.

LAYOUT

Prints out a summary of the global and case options data selected by user.

LOADVL

Loads computed surface flow quantities into common blocks for convenience in printing or storing in the PDP data base. Common blocks /PNTCTL/ is used for data to be printed and /FLQNT/ for data to be stored.

MAPDIP

Defines all SDMS maps for DIP data base datasets needed by PDP.

MAPMDG

Defines all SDMS maps for MDG data base datasets needed by PDP.

MAPDPT

Defines all SDMS maps for the PDP temporary data base (PDPT).

MAPPDP

Defines all SDMS maps for all PDP data base datasets.

NETFLO

Processes control and grid points data from the PDPT data base for networks except for edge control points in order to compute pressures and other surface flow data.

PANSRF

Assembles the geometry and flow data for the nine grid points of a panel by reading the PDP dataset 'FLOW-QUANT'.

PDPGEN

Generates the PDP data base and writes network specifications data (dataset NETWK-SPEC) and global data (dataset GLOBAL).

PNTHDR

Prints output header lines (run identification, problem identification, user identification, etc.).

PNTRPT

Prints report on user selected surface flow quantities from data in common block /PNTCTL/.

PNTSUB

Prints subheader lines for identification of point type, network, solution, image, velocity computation and correction method, and pressure computation options.

PTGEOM

Computes normal and conormal vectors at a point on a subpanel and the inner product of the normal and the transpose of the conormal vectors.

PTLSRF

Computes perturbation and total potential at a point on a surface.

RDCPVL

Reads control point data from the PDP temporary data base dataset CP-VEL into common block /CPDATA/.

RDGPDT

Reads grid point velocity and singularity data from the MDG data base dataset GP-DATA into common block /GPDATA/.

RDGPGM

Reads grid point geometry data from MDG data base dataset GP-GEOM into common block /GPDATA/.

RDGPVL

Reads grid point velocity data from PDP temporary data base (PDPT) dataset GP-VEL into common block /GPDATA/.

RDSRF

Reads and processes user options data for a case from the DIP dataset SURF-FLOW.

REFMAT

Computes matrix for transformation from reference axis to subpanel local coordinate system.

REQPNT

Prints estimate on disk storage requirements for the PDP run.

RESGP

Restores grid point data from common block /PANBLK/ into /GPDATA/ for a panel.

RESTORE

Restores surface flow quantities (velocity, mass flux, pressures, local Mach numbers, etc.) from common block /TEMPQN/ into /PNTCTL/.

SA1COR

Computes stagnation to ambient velocity correction by the first method (see Section N.3.1 of Reference 1).

SUBSPL

Computes doublet subpanel spline matrix for a subpanel.

TRNMAT

Computes transformation matrix from the reference axis to another coordinate system (e.g., where the X axis is the uniform onset flow direction).

VELCOR

Computes SA1 and SA2 velocity corrections (see Section N.3 of Reference 1).

VELSRF

Computes perturbation and total velocities on a surface.

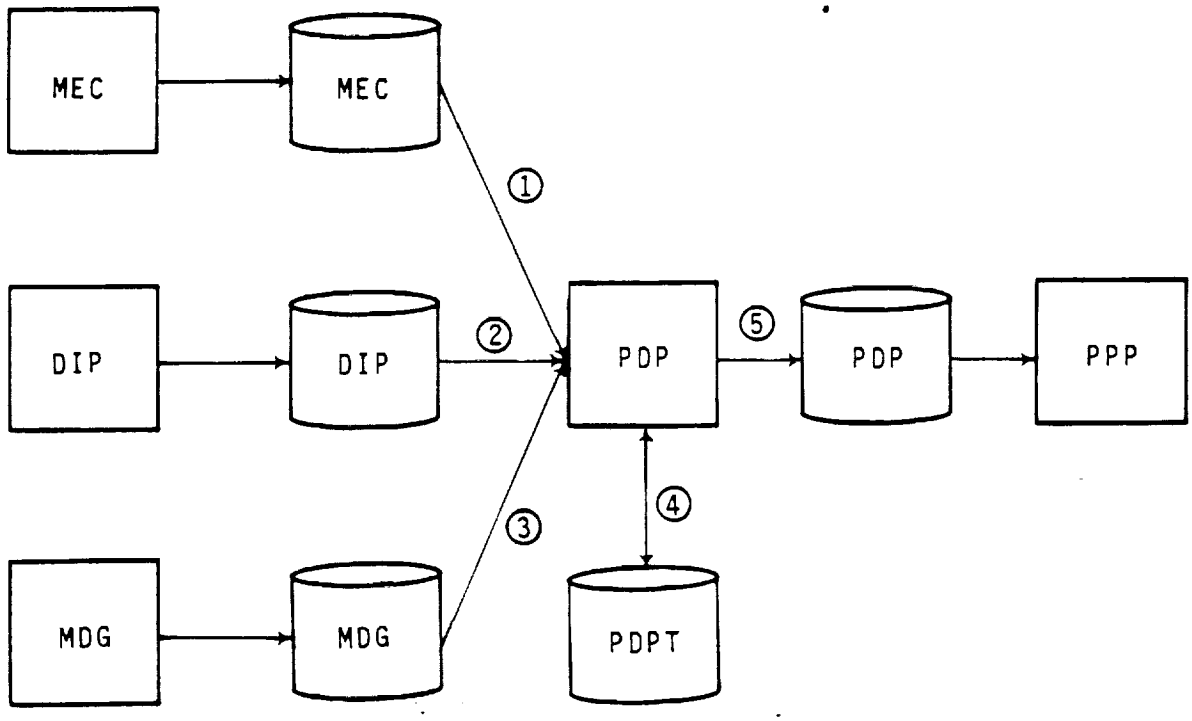
VORCTY

Computes the corticity angle for a wake surface.

TABLE 9.1 - List of Surface Flow Quantities

DIP Index*	Headings	Quantity
2	X	Point, x-coordinate
2	Y	Point, y-coordinate
2	Z	Point, z-coordinate
3	PWX	Perturbation mass flux, x-component
3	PWY	Perturbation mass flux, y-component
3	PWZ	Perturbation mass flux, z-component
4	WX	Total mass flux, x-component
4	WY	Total mass flux, y-component
4	WZ	Total mass flux, z-component
5	WMAG	Total mass flux, magnitude
6	WN	Total mass flux, normal component
7	PVX	Perturbation velocity, x-component
7	PVY	Perturbation velocity, y-component
7	PVZ	Perturbation velocity, z-component
8	VX	Total velocity, x-component
8	VY	Total velocity, y-component
8	VZ	Total velocity, z-component
9	VMAG	Total velocity, magnitude
10	PHI	Perturbation potential
11	PHIT	Total potential
12	MLISEN	Local Mach number, isentropic
12	MLLINE	Local Mach number, linear
12	MLSECO	Local Mach number, second-order
12	MLREDU	Local Mach number, reduced second-order
12	MLSLEN	Local Mach number, slender body
13	CPISEN	Pressure coefficient, isentropic
13	CPLINE	Pressure coefficient, linear
13	CPSECO	Pressure coefficient, second-order
13	CPREDU	Pressure coefficient, reduced second-order
13	CPSLEN	Pressure coefficient, slender body
14	GMUX	Doublet strength gradient, x-component
14	GMUY	Doublet strength gradient, y-component
14	GMUZ	Doublet strength gradient, z-component
15	PSI	Angle between average velocity and surface vorticity vectors (degrees)
16	SINGS	Singularity strength, source
16	SINGD	Singularity strength, doublet
17	SPDMAX	Maximum total speed
18	SPDCRT	Critical speed
19	CPVAC	Pressure coefficient, vacuum
	CPCISN	Critical pressure coefficient, isentropic
	CPCLIN	Critical pressure coefficient, linear
	CPCSO	Critical pressure coefficient, second order
	CPCRSO	Critical pressure coefficient, reduced second order
	CPCSB	Critical pressure coefficient, slender body

* See a description of user input record SF10 in Section 7 of the PAN AIR User's Manual (Ref. 2). The last five parameters in this table (the critical pressure coefficients) are not user selectable; instead, the PDP modules computes and prints these (except for arbitrary points) for subsonic flow if the corresponding pressure rules are selected by the user.



① - Data base directory information
② - User input surface flow options
③ - Global, network specification, control and grid point geometry and singularity data
④ - Control and grid point velocities, singularities, perturbation potential, and geometry data
⑤ - Global and network specification data and surface flow quantities

Figure 9.1 - Data Base Relationships

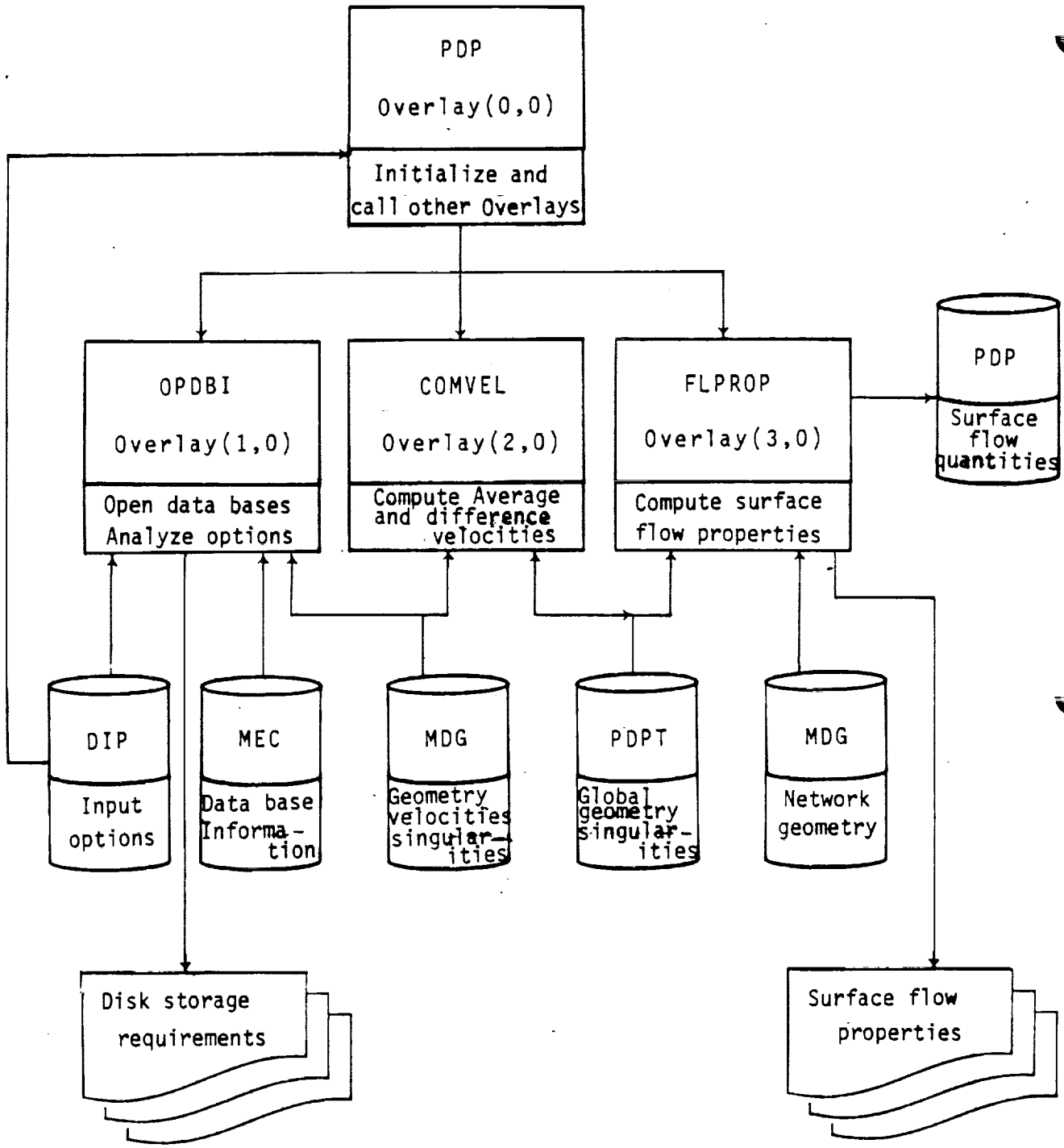


Figure 9.2 - PDP Structure and Data Interfaces

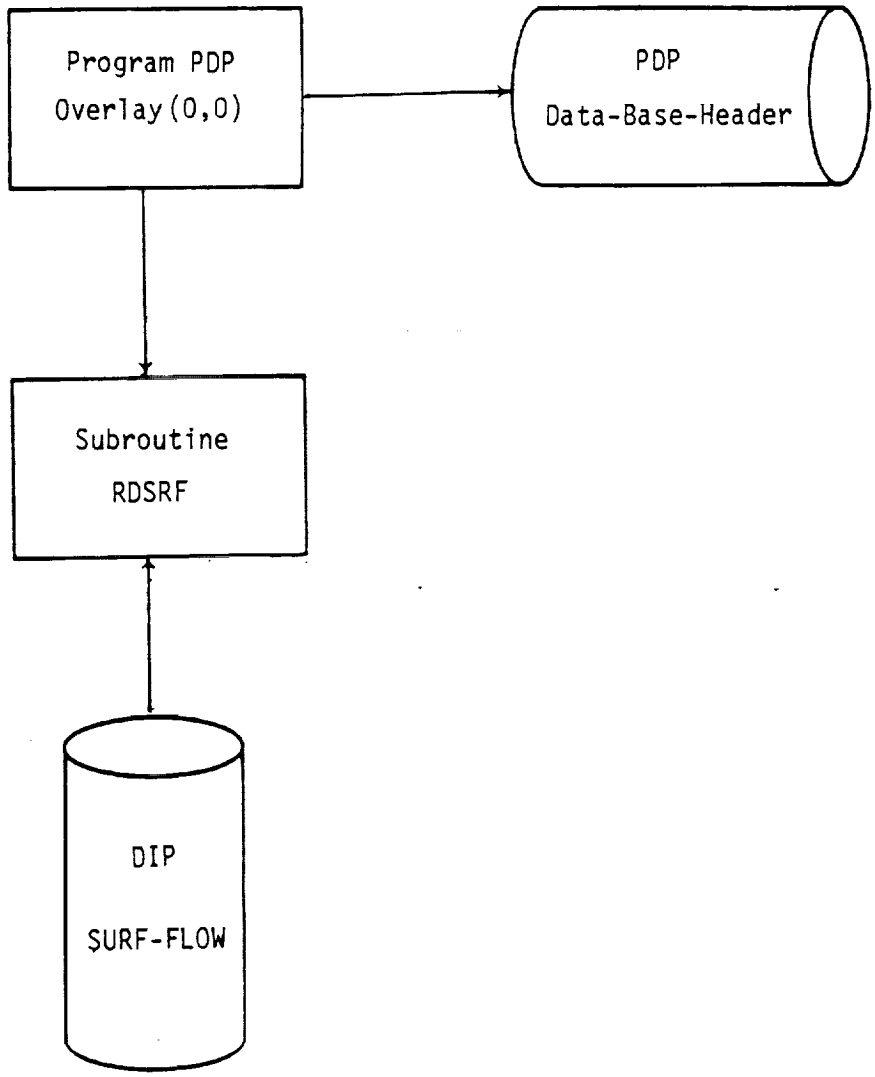


Figure 9.3 - Structure and Data Flow of Overlay (0,0)

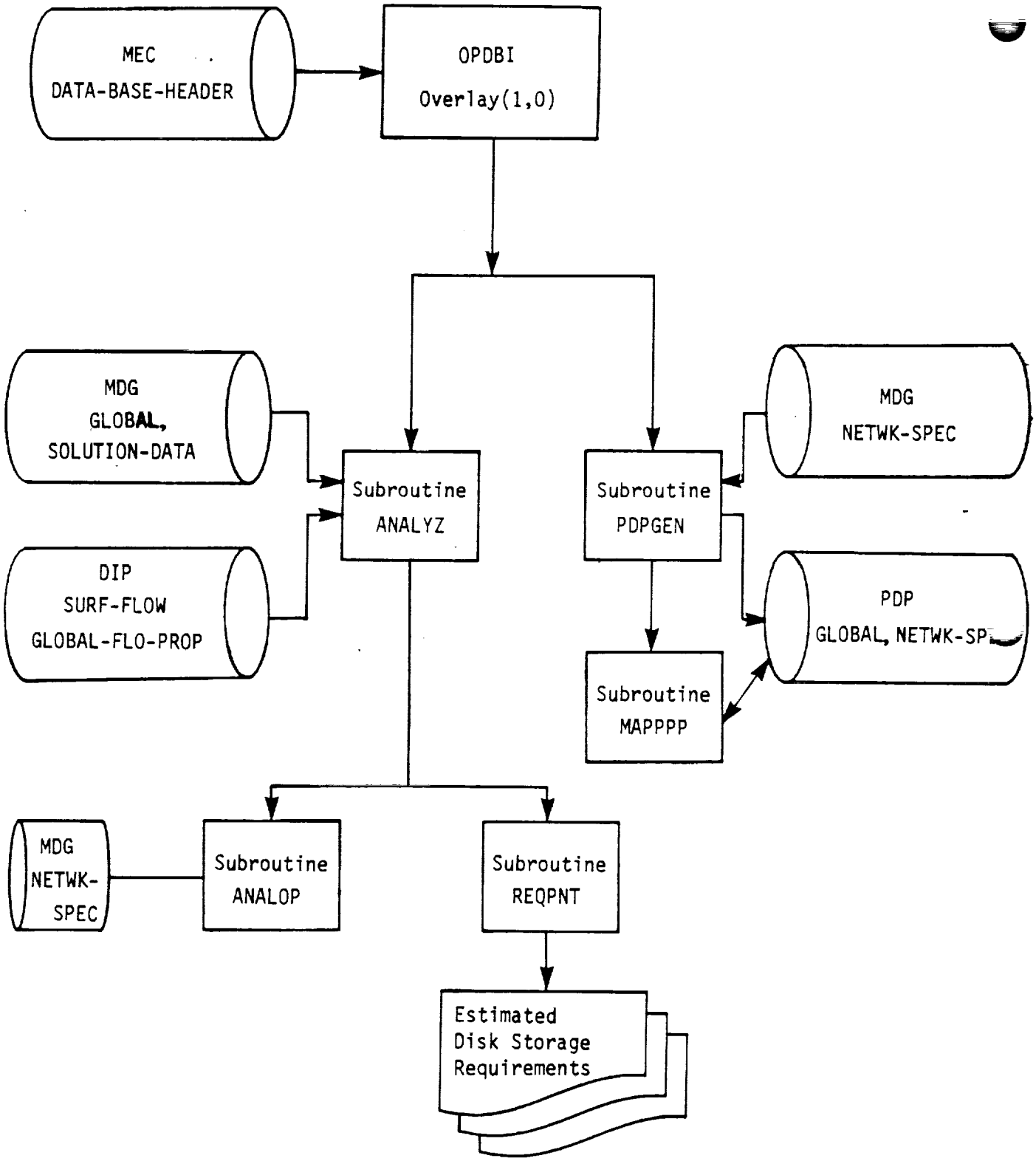


Figure 9.4 - Structure and Data Flow of Overlay(1,0)

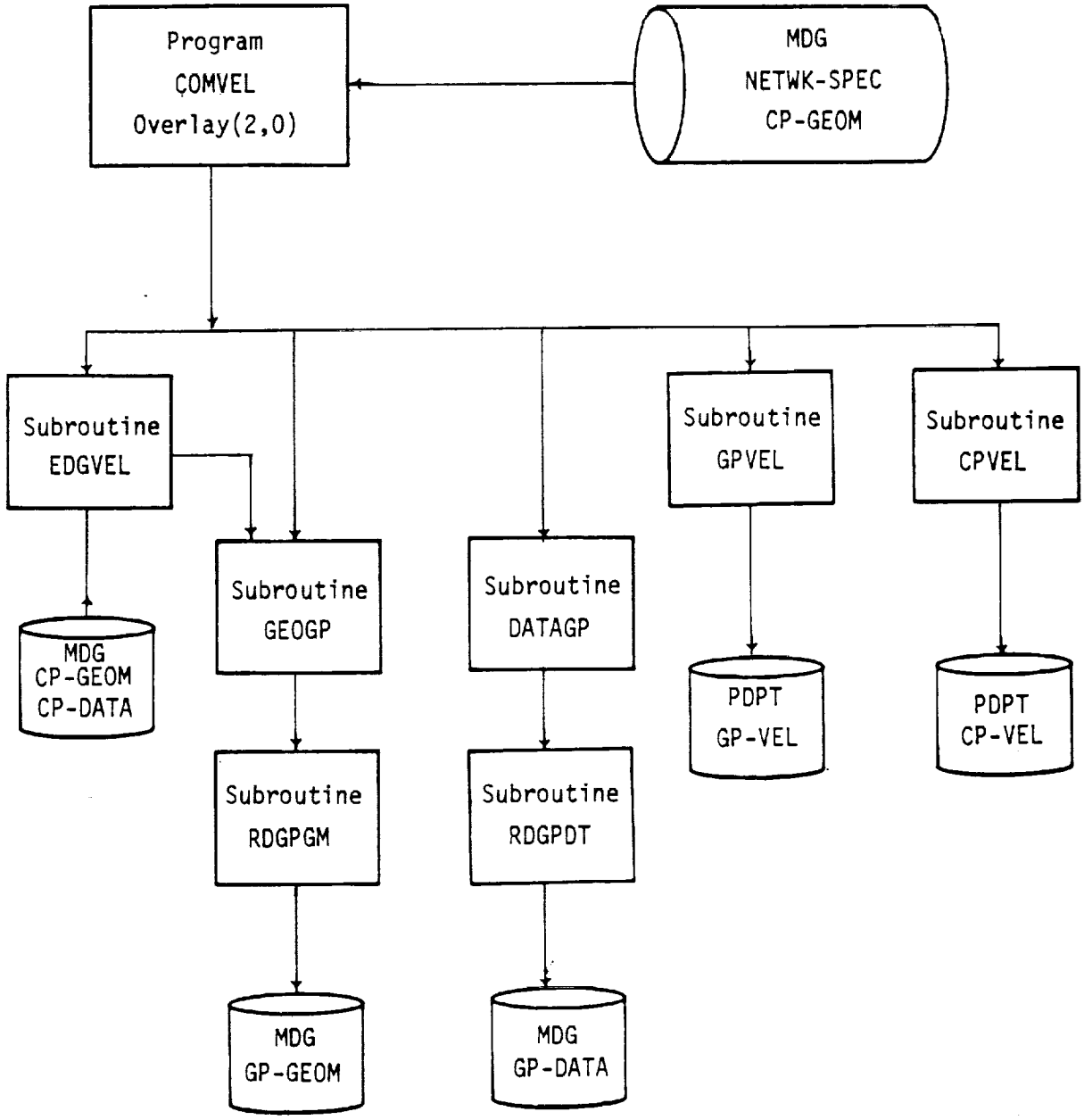


Figure 9.5 - Structure and Data Flow of Overlay(2,0)

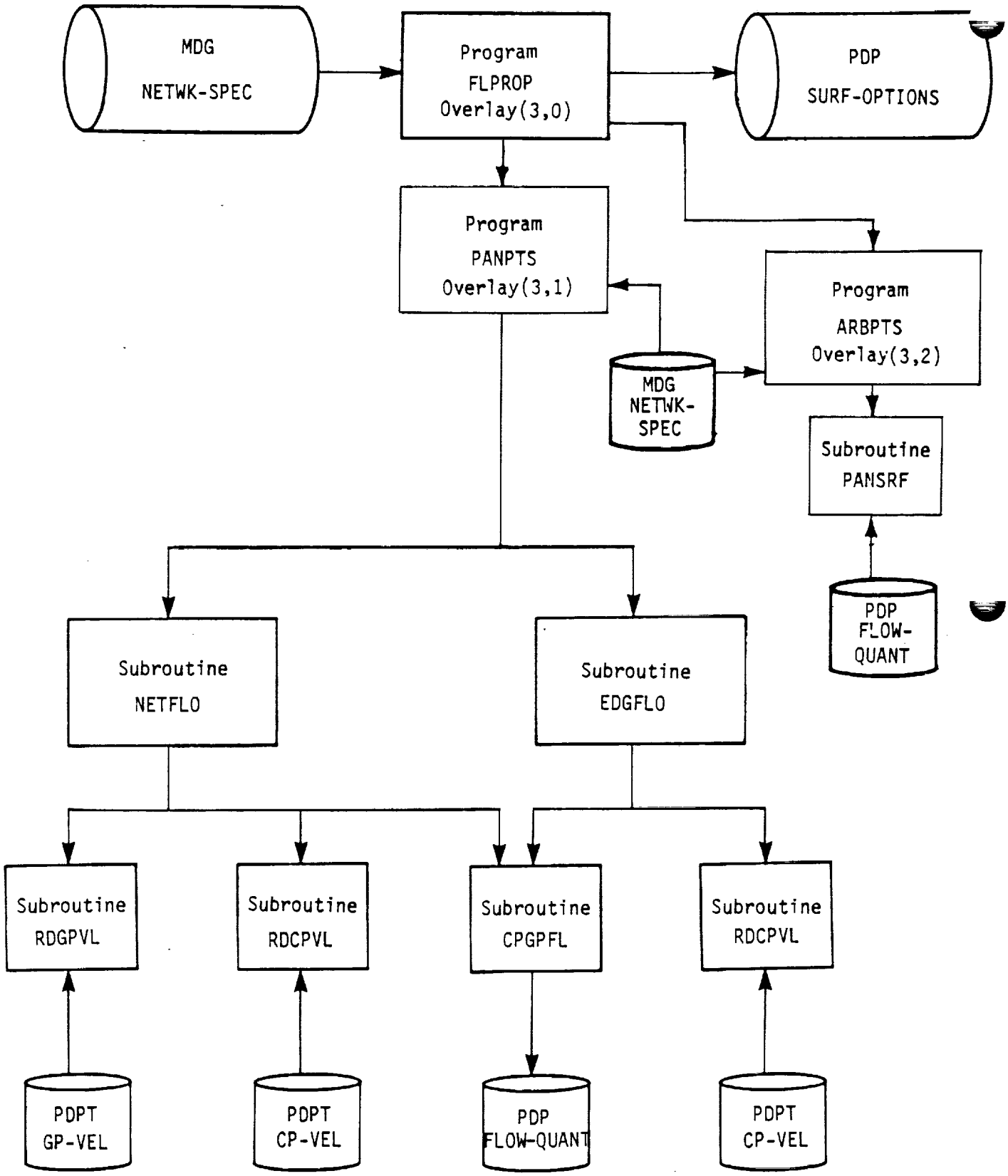


Figure 9.6 - Structure and Data Flow of Overlay (3,0)

APPENDIX 9-A TREE STRUCTURE

The tree structure diagram of the PDP module has been deleted from this document. It is, however, available on the installation tape.

APPENDIX 9-B PDP FUNCTIONAL DECOMPOSITION

The functional decomposition of the PDP module is presented here. The decomposition labels are given in the order of their execution and therefore may not be alphabetic.

H Initiate Program Execution
- A Initialize variables
- B Initiate program start (PRGBEG)
- C Initiate SDMS (ISDMS)

A Check data bases and initialize - Overlay (1,0) Program OPDBI
- A Open MEC data base (DBOPEN)
- B Get run identification information
  - A Define MEC header dataset map (DSMAP, SVMAP, ENDMAP)
  - B Get MEC header dataset (ESGET)
- C Check data bases required by PDP
  - A Check DIP data base (CHPADB)
  - B Check MDG data base (CHPADB)
  - C Generate PDPT data base (CHPADB)
- D Open data bases and define SDMS maps
  - A Open DIP and MDG data bases (PAOPEN)
  - B Define SDMS maps for DIP datasets (MAPDIP)
  - C Define SDMS maps for MDG data base (MAPMDG)
  - D Analyze and print resource requirements (ANALYZ)
- F Check and generate PDP data base, write "GLOBAL" and "NETWK-SPEC" datasets (PDPGEN)
- G Close DIP and MDG data bases (PACLOS)
- H Close MEC data base (DBCLOS)

I Open DIP, MDG and PDP data bases and defined needed SDMS maps
- A Open DIP, MDG
- B Define needed DIP maps (MAPDIP)
- C Define needed MDG maps (MAPMDG)
- D Open PDP data base (PAOPEN)

B
- A Read and analyze surface flow options data (RDSRF)
  - A Read DIP dataset "SURF-FLOW" (ESGET)
  - B Determine image list for each network
  - C Determine velocity computation options selected
  - D Determine velocity correction options selected
  - E Form PDP option-index vector
    - A Get from reference table the start and end position of an item in the index vector
    - B Flags these word appropriately (0, 1, 2, or 3)
    - C Determine flags to indicate print and data base storage
    - D Flag local Mach number option elements, if selected
  - F Find number of surfaces selected
- B Open PDPT data base (PAOPEN)
- C Define SDMS maps for PDPT data base (MAPDPT)

C Compute average and difference velocities at points - Overlay (2,0) - Program COMVEL
- A Analyze point types selected
- B Get network data
  - A Get index and ID
  - B Read MDG dataset "NETWORK-SPEC" (ESGET)
- C Process edge control points, if selected (EDGVEL)
- D Get geometry data from MDG data base
  - A Read control point geometry data (ESGET)
  - B Get grid point geometry data (GEOGP)

E Get singularity and velocity data for point
  A Get for grid points (DATAGP)
  B Get for control points from MDG data base (ESGET)
F Compute and write the average and difference velocity data to PDPT data base
  A Form grid points in panel (GPVEL)
    A Assemble data for the grid points
    B Compute additional geometry data required (GPGEOM)
    C Compute subpanel local coordinates for a point (CMPORG)
    D Assemble the perturbation potential and doublet strength matrix for the panel (PANMAT)
    E Compute and accumulate average velocity by the boundary condition method for all subpanels where the point lies
      A Compute gradient of potential at point (COMGRD)
      B Compute the first terms of the equation for average velocity (CAB)
      C Compute and accumulate average velocity at the point for the subpanel
    F Compute and accumulate difference velocity at the point for all subpanels where the point lies
      A Compute gradient of doublet strength (COMGRD)
      B Compute the first terms of equation (CAB)
      C Compute and accumulate difference velocity at point for the subpanel
    G Compute average (for all subpanels where the point lies) of the velocities
    H Compute gradient of doublet strength in reference axis (CAB)
    I Write grid point data onto PDPT dataset "GP-VEL"
  B Compute average and difference velocities at center control points and write to PDPT data base dataset (CPVEL)
    A Compute additional geometry data for the control point
      A Compute conormal and inner product of normal and transpose of cornormal vectors (PTGEOM)
        A Initialize error code to zero
        B Compute conormal vector (CAB)
        C Compute sub/super inclination of subpanel (VIP)
        D Increment error count, if any
        E Compute inner product of normal and transpose of conormal vectors (CAB)
      B Compute transformation matrix for subpanel local to reference axis (REFMAT)
        A Compute unit V vector
          A Compute vector cross product of unit normal and compressibility vector (CROSS)
          B Normalize V vector (UNIVEC)
          C Define new V vector if normal vector is parallel to compressibility vector
          D Normalize V vector
        B Compute unit vector U perpendicular to V and normal vectors (CROSS)
        C Compute first term of matrix
          A Find scale factor
          B Compute unscaled first column (CAB)
          C Scale and form first column (CAB)

D Compute second column of matrix
A Find scale factor
B Compute unscaled column
C Scale and form second column
E Compute third column of matrix
A Find column factor
B Compute, scale and form third column
F Error exit
C Define subpanel local coordinate system and tranform point coordinates in reference axis to subpanel local system (CMPORG)
B Compute first part of the first term for equations defining average and difference velocities
C Form doublet strength matrix (9x1) for panel (PANMAT)
A Determine column, row and point number
B Get average perturbation potential and doublet strength values for point
C Form elements of matrix
D Compute average perturbation velocity by the boundary condition method
A Compute gradient of potential (COMGRD)
B Compute first term of equation (CAB)
C Compute the three components of velocity
E Compute perturbation difference velocity at the point
A Compute gradient of doublet strength at point (CAB)
B Compute first term of the difference velocity equation (CAB)
C Compute the three components of velocity
F Compute gradient of doublet strength in reference axis
G Reflect coordinates, velocities, gradients of doublet strength and normal vector for image (IMAGE)
H Write computed data to PDPT data base dataset CP-VEL (ESPUT)

D Compute surface flow properties data - Overlay (3,0), Program FLPROP
A Print global and surface flow case option data (LAYOUT)
B Define SDMS maps for PDP dataset FLOW-QUANT and write SURF-OPTIONS datasets
A Define SDMS maps (DSMAP, SVMAP, ENDMAP)
B Write SURF-OPTIONS dataset for this case (ESPUT)
C Compute constant quantities dependent on solution only
D Compute flow properties at panel points - Overlay (3,1), Program PANPTS
A Get network index
B Get network specifications data from MDG data base (ESGET)
C Compute flow properties at network edges (EDGVEL)
A Form and initialize loop control table
B Get loop control parameters from table for the edge
C Get and process velocity data for control point from PDPT data base (RDCPVL)
D Compute and output flow quantities at the point on edge (CPGPFL)
A Select average perturbation velocity at point (B.C. or VIC)
B Compute perturbation and total potential on surface (PTLSRF)

C Compute perturbation and total velocities on surface (VELSRF)
D Compute mass flux (perturbation, total and total normal) on surface

E Compute velocity correction (SA1 and SA2) (VELCOR)
A Compute perturbation velocity component in the compressibility axis direction (VIP)
B Compute SA1 (stagnation to ambient correction number one)
A Compute mass flux along compressibility direction
B Compute correction to velocity (SA1COR)
C Compute SA2 correction (stagnation to ambient correction number 2)
A Multiply perturbation mass flux by the ratio of perturbation velocity and perturbation mass flux magnitudes to get total velocity
B Compute correction to velocity

F Compute local Mach numbers and pressure coefficients (COMPRS)
A Compute perturbation and local incremental onset flow velocities along user preferred direction
B Compute maximum and critical speed and total velocity
C Compute pressure coefficients
A for isentropic rule
B for second order rule
C for reduced second order rule
D for slender body rule
E for linear rule
D Compute local Mach numbers
A for isentropic and second order rules
B for reduced second order rule
C for slender body rule
D for linear rule

G Save the computed flow quantities for the point on the upper and lower surfaces in common block /TEMPQN/

H Restore the computed flow quantities for the point to common block /COMPQN/ and compute vorticity angle
A Restore data
B Compute total mass flux and magnitudes of total velocity and total mass flux (VIP)
C Compute vorticity (VORCTY)

I Load flow quantities data into print and data base buffers (common blocks /PNTCTL/ and /FLQNT/) (LOADVL)

J Produce printed output (PNTRPT)
A Initialize program variables
B Prepare and print header lines for new page of output, if necessary (PNTSUB)
C Print flow quantities data from common block /PNTCTL/
D Save the indices for the current solution, network, image and point type

K Write flow quantities data to PDP data base dataset FLOW-QUANT, if data base storage is selected (ESPUT)

D Compute flow properties for the network points (except on edges) (NETFLO)
A Determine row and column indices for grid points in panel

B Read and process grid point data from PDPT data base dataset GP-VEL (RDPVL)
C Determine row and column indices for control point
D Read and process control point data from PDPT dataset CP-VEL (RDCPVL)
E Compute and output flow quantities at the points (CPGPFL)

A Select average perturbation velocity at point (B.C. or VIC)
B Compute perturbation and total potential on surface (PTLSRF)
C Compute perturbation and total velocities on surface (VELSRF)
D Compute mass flux (perturbation, total and total normal) on surface
E Compute velocity correction (SA1 and SA2) (VELCOR)
  A Compute perturbation velocity component in the compressibility axis direction (VIP)
  B Compute SA1 (Stagnation to Ambient correction number one)
    A Compute mass flux along compressibility direction
    B Compute correction to velocity (SA1COR)
  C Compute SA2 correction
    A Multiply perturbation mass flux by the ratio of perturbation velocity and perturbation mass flux magnitudes to get total velocity
    B Compute correction to velocity
F Compute local Mach numbers and pressure coefficients (COMPRS)
  A Compute perturbation and local incremental onset flow velocities along user preferred direction
  B Compute maximum and critical speed and total velocity
  C Compute pressure coefficients
    A for isentropic rule
    B for second order rule
    C for reduced second order rule
    D for slender body rule
    E for linear rule
  D Compute local Mach numbers
    A for isentropic and second order rules
    B for reduced second order rule
    C for slender body rule
    D for liear rule
G Save the computed flow quantities for the point on the upper and lower surfaces in common block (TEMPQN)
H Restore the computed flow quantities for the point to common block /COMPQN/ and compute vorticity angle
  A Restore data
  B Compute total mass flux and magnitudes of total velocity and total mass flux (VIP)
  C Compute vorticity (VORCTY)
I Load flow quantities data into print and data base buffers (common blocks /PNTCTL/ and /FLQNT/) (LOADVL)
J Produce printed output (PNTRPT)
  A Initialize program variables
  B Prepare and print header lines for new page of output, if necessary (PNTSUB)

C Print flow quantities data from common block /PNTCTL/
D Save the indices for current solution, network, image and point type

K Write flow quantities data to PDP data base dataset FLOW-QUANT, if data base storage is selected (ESPUT)

E Compute flow properties at arbitrary points - Overlay (3,2), Program ARBPTS

A Define map for DIP dataset 'ARBITRARY-POINTS', if first-time execution

B Get arbitrary points specification data (ESGET) from DIP data base

C Get network dimensions data from MDG dataset 'NETWK-SPEC' (ESGET)

D Assemble flow data from PDP data base for given panel where the arbitrary point may lie (PANSRF)

A Initialize data arrays to zeroes (ZERO)

B Compute fine grid row and column indices for needed grid data blocks

C Read dataset 'FLOW-QUANT' of PDP data base for a grid block (ESGET)

D Compute panel unit normal vector

E Save network and panel row and column indices

F Determine subpanel where the arbitrary point lies (FNDSUB)

A Initialize error code to zero

B Compute projection of point on panel surface

C Compute parameters needed to determine the subpanel number

D Compute subpanel index

E Assemble coordinates of the subpanel vertices and compute subpanel unit normal vector

F If point lies outside panel, set error code to 1.

G Compute and print flow properties at the arbitrary point (ARBFLO)

A Initialize print and data base arrays

B Compute flow quantities at point by linear extrapolation of the values at the subpanel vertices (EXTPLT)

C Load computed data into print and data base arrays (LOADVL)

D Print computed flow data for the arbitrary point (PNTRPT)

E Close and return temporary data base, PDPT (PACLOS)

F Initiate end of program execution

A Close DIP and MDG data bases, write PDP header dataset DATA-BASE-HEADER and close PDP data base (PACLOS)

B Announce end of execution (PRGEND)

APPENDIX 9-C DATA BASE COMMUNICATIONS CHART

The Data Base Communications Chart is presented in three forms. Each form is alphabetized by columns, from left to right. The first form has a column order of Data Base, Dataset Name, Map Name, Common Block, and Program/Subroutine. The second form has a column order of Data Base, Map Name, Dataset Name, Common Block, and Program/Subroutine. The third form has a column order of Common Block, Data Base, Map Name, Dataset Name, and Program/Subroutine. Thus a person can get a cross reference on a data element by knowing either the Dataset Name, Map Name or Common Block name.

FIRST FORM

DATA BASE	DATASET NAME	MAP NAME	COMMON BLOCK*	PROGAM/ SUBROUTINE
DIP	ARBITRARY-POINTS	DIPARBPT	/ARBGEO/	MAPDIP ARBPTS
DIP	GLOBAL-FLO-PROP	DIPGLBFL	/PDGLOB/	MAPDIP RDSRF
DIP	SURF-FLOW	DIPSRFOP	/PDOPT/	MAPDIP RDSRF
MEC	DATA-BASE-HEADER	MECMAP	/RUNIDS/	OPDBI
MDG	CP-DATA	MDGCPDAT	/CPDATA/	MAPMDG COMVEL EDGVEL
MDG	CP-GEOM	MDGCPGEM	/CPDATA/	MAPMDG COMVEL EDGVEL
MDG	GLOBAL	MDGGLOBL	/PDGLOB/	MAPMDG ANALYZ
MDG	GP-DATA	MDGGPDAT	/GPDATA/	MAPMDG RDGPDT
MDG	GP-GEOM	MDGGPGEM	/GPDATA/	MAPMDG RDGPGM
MDG	NETWORK-SPEC	MDGNETMP	/NETSPC/	MAPMDG ANALYZ COMVEL PANPTS ARBPTS
PDP	DATA-BASE-HEADER	PDPHDR	/RUNIDS/	MAPPDP FLPROP
PDP	FLOW-QUANT	FLQNTMAP	/FLQNT/	MAPPDP FLPROP CPGPPL PANSRF
PDP	GLOBAL	GLOBMAP	/PDGLOB/	MAPPDP PDPGEN
PDP	NETWK-SPEC	NETMAP	/NETSPC/	MAPPDP PDPGEN
PDP	SURF-OPTIONS	OPTNMAP	/PDOPT/	MAPPDP FLPROP
PDPT	CP-VEL	CPVEL	/CPDATA/	MAPDPT RDCPVL CPVEL
PDPT	CP-VEL	GPVEL	/GPDATA/	MAPDPT GPVEL

SECOND FORM

DATA BASE	MAP NAME	DATASET NAME	COMMON BLOCK*	PROGAM/ SUBROUTINE
DIP	DIPARBPT	ARBITRARY-POINTS	/ARBGEO/	MAPDIP ARBPTS
DIP	DIPGLBFL	GLOBAL-FLO-PROP	/PDGLOB/	MAPDIP RDSRF
DIP	DIPSRFOP	SURF-FLOW	/PDOPT/	MAPDIP RDSRF
MEC	MECMAP	DATA-BASE-HEADER	/RUNIDS/	OPDBI
MDG	MDGCPDAT	CP-DATA	/CPDATA/	MAPMDG COMVEL EDGVEL
MDG	MDGCPGEM	CP-GEOM	/CPDATA/	MAPMDG COMVEL EDGVEL
MDG	MDGGLOBL	GLOBAL	/PDGLOB/	MAPMDG ANALYZ
MDG	MDGGPDAT	GP-DATA	/GPDATA/	MAPMDG RDGPDT
MDG	MDGGPGEM	GP-GEOM	/GPDATA/	MAPMDG RDGPGM
MDG	MDGNETMP	NETWORK-SPEC	/NETSPC/	MAPMDG ANALYZ COMVEL PANPTS ARBPTS
PDP	PDPHDR	DATA-BASE-HEADER	/RUNIDS/	MAPPDP FLPROP
PDP	FLQNTMAP	FLOW-QUANT	/FLQNT/	MAPPDP FLPROP CPGPPL PANSRF
PDP	GLOBMAP	GLOBAL	/PDGLOB/	MAPPDP PDPGEN
PDP	NETMAP	NETWK-SPEC	/NETSPC/	MAPPDP PDPGEN
PDP	OPTNMAP	SURF-OPTIONS	/PDOPT/	MAPPDP FLPROP
PDPT	CPVEL	CP-VEL	/CPDATA/	MAPDPT RDCPVL CPVEL
PDPT	GPVEL	CP-VEL	/GPDATA/	MAPDPT GPVEL

THIRD FORM

COMMON BLOCK*	DATA BASE	MAP NAME	DATASET NAME	PROGRAM/ SUBROUTINE
/ARBGEO/	DIP	DIPARBPT	ARBITRARY-POINTS	MAPDIP ARBPTS
/CPDATA/	MDG	MDGCPDAT	CP-DATA	MAPMDG COMVEL EDGVEL
/CPDATA/	MDG	MDGCPGEM	CP-GEOM	MAPMDG COMVEL EDGVEL
/CPDATA/	PDPT	CPVEL	CP-VEL	MAPDPT RDCPVL CPVEL
/FLQNT/	PDP	FLQNTMAP	FLOW-QUANT	MAPPDP FLPROP CPGPPL PANSRF
/GPDATA/	MDG	MDGGPDAT	GP-DATA	MAPMDG RDGPDT
/GPDATA/	MDG	MDGGPGEM	GP-GEOM	MAPMDG RDGPGM
/GPDATA/	PDPT	GPVEL	CP-VEL	MAPDPT GPVEL
/NETSPC/	MDG	MDGNETMP	NETWORK-SPEC	MAPMDG ANALYZ COMVEL PANPTS ARBPTS
/NETSPC/	PDP	NETMAP	NETWK-SPEC	MAPPDP PDPGEN
/PDGLOB/	DIP	DIPGLBFL	GLOBAL-FLO-PROP	MAPDIP RDSRF
/PDGLOB/	MDG	MDGGLOBL	GLOBAL	MAPMDG ANALYZ
/PDGLOB/	PDP	GLOBMAP	GLOBAL	MAPPDP PDPGEN
/PDOPT/	DIP	DIPSRFOP	SURF-FLOW	MAPDIP RDSRF
/PDOPT/	PDP	OPTNMAP	SURF-OPTIONS	MAPPDP FLPROP
/RUNIDS/	MEC	MECMAP	DATA-BASE-HEADER	OPDBI
/RUNIDS/	PDP	PDPHDR	DATA-BASE-HEADER	MAPPDP FLPROP

*Dynamic mapping (see Section 13 of this document for details) is used for all or some of the keys for each data set, thus requiring no common block storage for the keys.

APPENDIX 9-D MASTER DEFINITION

The data base master definition listing of the PDP module has been deleted from this document. It is produced from the PAN AIR tape during installation.

10.0 CONFIGURATION DATA PROCESSOR (CDP) MODULE

10.1 INTRODUCTION

The CDP module is a post procesor for the PAN AIR system. It presumes the existence of a solution to the potential flow problem and converts the data describing that solution into forces and moments. The module may compute added mass coefficients instead of forces and moments.

Data bases from three PAN AIR modules are required to run CDP. These are the Module Execution Control (MEC) data base, which controls the sequence of module execution, the DATA Input Procesor (DIP) data base, which desribes the force and moment options desired by the user, and the Minimal Data Generator (MDG) data base. The MDG data base provides input data for the forces and moment computations. Output from CDP, in the form of the CDP data base, is used by the Print Plot Processor (PPP) module.

The CDP module consists of a top level program which calls four main overlays to compute forces and moments. The first overlay checks the status of all the data bases required by CDP and analyzes the global data. The second overlay computes forces and moments for a particular case. The third overlay computes the component of the force on a thin surface due to the infinite velocity at the leading edge. The fourth overlay transforms the forces and moment data from the reference axis system to the selected axis systems, and writes the results on the CDP data base and/or on the output listings. The fifth overlay computes and displays added mass coefficients. It may be executed instead of the second, third and fourth overlays. The results can be accumulated over sets of user options, called cases.

10.2 CDP OVERVIEW

10.2.1 Purpose of CDP

Given an existing solution, CDP will compute forces and moments or added mass coefficients on portions of a configuration, transform these according to user requests, and print or store in the data base the transformed results.

10.2.2 CDP Input/Output Data

The CDP module requires access to the data bases brom three PAN AIR modules in order to execute. These are the MEC, the DIP and the MDG data bases.

The MEC data base provides run, problem and user identification in addition to current data base identification.

The DIP data base provides user problem descriptions of global flow properties and user requests for surface force and moments.

The MDG data base provides global data, network specificiations, control point data, control point geometry, grid point data, grid point geometry and solution data.

CDP output data consists of printed output and the CDP permanent data base which is used by the Print Plot Processor (PPP) module. Output data is controlled by user specifications which reside on the DIP data base. The printed output may provide a summary of user options, resource estimates, user requested results and errors and diagnostics. The CDP permanent data base provides case options, network specifications, edge force coefficients, and force and moment coefficients for panels, networks, configuration and network edges. Added mass coefficients for selected portions of the configuration may also reside on the data base.

10.2.3 Data Base Interface

The CDP module creates a temporary data base which saves data computed in the second overlay COPMPFM and the third overlay LEDGF. The third overlay reads doublet strengths from the temporary data base. The fourth overlay GENOUT accesses the temporary data base in order to produce the final output. The temporary data base provides force and moment coefficients, edge forces and moments, panel areas, and doublet strength.

10.3 MODULE DESCRIPTION

10.3.1 Overall Structure

The high level module design is described in this paragraph. The lower level subroutines are described in Paragraph 10.4. The functional decomposition of CDP is illustrated in Appendix 10-B. The overall structure of CDP is depicted in Figure 10.1.

10.3.2 Overlay Descriptions

10.3.2.1 CDP Overlay (0,0)

The top level overlay initializes the data base and other program parameters. It calls overlay OPDBI, to check the satus of the data bases and to initialize global data, and defines the necessary data base maps. Then, to compute forces and moments for each case of user requested options, the module calls the three overlays COMPFM, LEDGF AND GENOUT. The forces and moments for the reference coordinate system are computed in COMPFM. Edge forces are computed in LEDGF. The overlay GENOUT transforms the forces and moments data to user requested axis systems and writes the information on the CDP data base and the output file. To compute and display added mass coefficients, the module calls the fifth overlay AMCOEF.

10.3.2.2 OPDBI Overlay (1,0)

The second level overlay OPDBI (Figure 10.2) reads the MEC data base to retrieve run and data base identification. It checks the data bases for completeness. Then the module reads global data from the DIP and MDG data bases, and it calculates parameters required to reflect input configurations across planes of symmetry.

10.3.2.3 COMPFM Overlay (2,0)

The second level overlay COMPFM (Figure 10.3) computes forces and moments from minimal data from the MDG data base and stores the results on the CDP temporary data base. The computations are performed panel by panel along a given column. The minimal data for a particular panel is assembled from four data sets which are keyed by the panel's four corner points.

10.3.2.4 LEDGF Overlay (3,0)

The second level overlay LEDGF (Figure 10.4) computes edge forces and moments from minimal data from the MDG data base and stores the results on the CDP temporary data base. The computations are performed panel by panel along a given edge. The panel corner point geometry is extrapolated from the fine grid geometry.

10.3.2.5 GENOUT Overlay (4,0)

The second level overlay GENOUT (Figure 10.5) prints out the computed forces and moments data and stores the data in the CDP data base for later retrieval. For each panel, the forces and moments are retrieved from the CDP temporary data base and transformed into the requested axis systems. The results are written to the CDP permanent data base and the line printer. This is also done for forces calculated along the edge.

10.3.2.6 AMCOEF Overlay (5,0)

The second level overlay AMCOEF (Figure 10.6) computes and displays added mass coefficients. The computations are performed for an individual panel and displayed for sums of panels as selected by the user. Displayed coefficients are printed or written to the CDP permanent data base.

10.3.3 CDP Data Base

The permanent data base CDP is created by CDP for used by the Print Plot Processor (PPP) module. The Master Definition is described in Appendix 10-D.

10.3.4 CDP Interfaces

10.3.4.1 System Interfaces

The CDP module is assessed by user control cards or a procedure file generated by the MEC module.

10.3.4.2 External Interfaces

The MEC data base, the DIP data base and the MDG data base are input vehicles for the CDP module. The CDP permanent data base and the output listing are the output vehicles for CDP.

10.3.4.3 Internal Interfaces

The interfaces between the overlays and the subprograms is defined by a tree structure diagram in Appendix 10-A.

10.3.5 Data Flow

The flow of execution is depicted in Figures 10.7 and 10.8. During execution, data flows between subprogram and data bases. Figures 10.7 and 10.8 depict this activity. Subprograms may also communicate with each other by using labeled common or formal parameters. Information concerning data flow in this manner can be found by consulting the glossaries of the subprograms which are of interest. Section 1, Paragraph 1.4 of this document can be consulted for more detailed information of the use of the tools available for analysis of data flow. Also, Appendix 10-C has been included to aid analysis of data flow between CDP and its data bases.

10.4 LOWER LEVEL FUNCTIONS

The following paragraphs present the functional decompositions (hierarchial structure) of the overlays and their subprograms and give the purpose of each subroutine.

10.4.1 Functional Decomposition

See Appendix 10-B for a description of the CDP functional decomposition. Section 1, paragraph 1.4.1 of this document can be consulted for more detailed information of the use of functional decomposition.

10.4.2 Subroutine Descriptions

The subroutines used in the CDP module which do not reside on the PALIB library or the SDMS library are described below. Also refer to the tree structure in Appendix 10-A.

AINVERS

Computes the inverse of the coefficient matrix (A matrix) used in velocity calculations.

ANALYZ

Reads and sets up global data.

ASPADA

Assembles geometry and surface flow data for a particular panel.

BCXFER

Transfers data from blank common to a local storage area.

BLOCK

Assembles four 3 x 3 matrices into a single 6 x 6 matrix.

CASEPG

Writes the case summary page.

CEFDIR

Computes a unit vector in the direction of the edge force.

CELPAR

Computes the edge limit parameter which is a measure of the degree of singularity of the velocities at the panel column edge.

CEPRDB

Accumulates and stores the edge forces and moments.

CFIGAM

Computes the added mass coefficients for a configuration.

COLAM

Computes the added mass coefficients for a column of panels.

COMPRS

Computes selected pressure coefficients and local Mach numbers for a point on a specified surface.

CORREC

Determines a correction factor to be used for edge force calculations.

CPAGEQ

Computes various panel quantities which are functions of geometry.

CPPRDB

Accumulates panel forces and moments and generates the CDP data bases and printed output.

FLOCHK

Checks the doublet distribution near the edges of networks to see if they follow patterns established for the 2-D flat plate cases from which edge force corrections have been developed.

GENFM

Computes force and moment coefficients in the reference axis system.

GLOBPG

Writes the global summary page.

INTAM

Performs the surface integration required to compute added mass coefficients for a panel.

LOADBC

Assembles data from the MDG module in blank common arrays. The data is sufficient to compute added mass coefficients for a column of panels.

MAPCDP

Defines selected SDMS maps for the CDP data base.

MAPCDT

Defines selected SDMS maps for the CDT data base.

MAPDIP

Defines selected SDMS maps for the DIP data base.

MAPMDG

Defines selected SDMS maps for the MDG data base.

NETWAM

Computes added mass coefficients for a network.

OUTPAM

Outputs added mass coefficients to the OUTPUT file or the CDP permanent data base.

PANLAM

Computes added mass coefficients for a panel.

PNTGLOB

Prints CDP global data.

PNTHDR

Prints CDP report header.

PNTOPT

Prints the CDP case options data as the first page for each case.

PNTRPT

Prints forces and moments data for panels, columns, networks, configurations and edges.

PNTSUB

Prints a new page of the report.

PRINAM

Prints added mass coefficients on the OUTPUT file.

PVEL

Computes perturbation velocities for the specified surface.

RCASFM

Reads and analyzes options for the current case.

REFLAM

Adjusts added mass coefficients for a panel to correspond to the correct image.

REFLFM

Reflects a force and its corresponding moment about given planes of symmetry.

SAICOR

Computes the SAI correction on velocity.

SCALAM

Scales added mass coefficients.

SPACER

Characterizes panel spacing used near a network edge.

TRANAM

Translates the added mass coefficients of a panel to alternative axis systems.

TRNMAT

Computes the transformation matrices for moving from the reference axis system to a user requested axis system.

TRNSFM

Transforms the force and moment data for a particular panel to a user selected axis system.

UNLDBC

Retrieves the MDG data from blank common arrays which is required to compute added mass coefficients for a panel.

VELCOR

Computes user selected corrections on velocity.

VELOC

Computes difference and average panel velocities.

XFERBC

Transfers data from a local storage array to a blank common array.

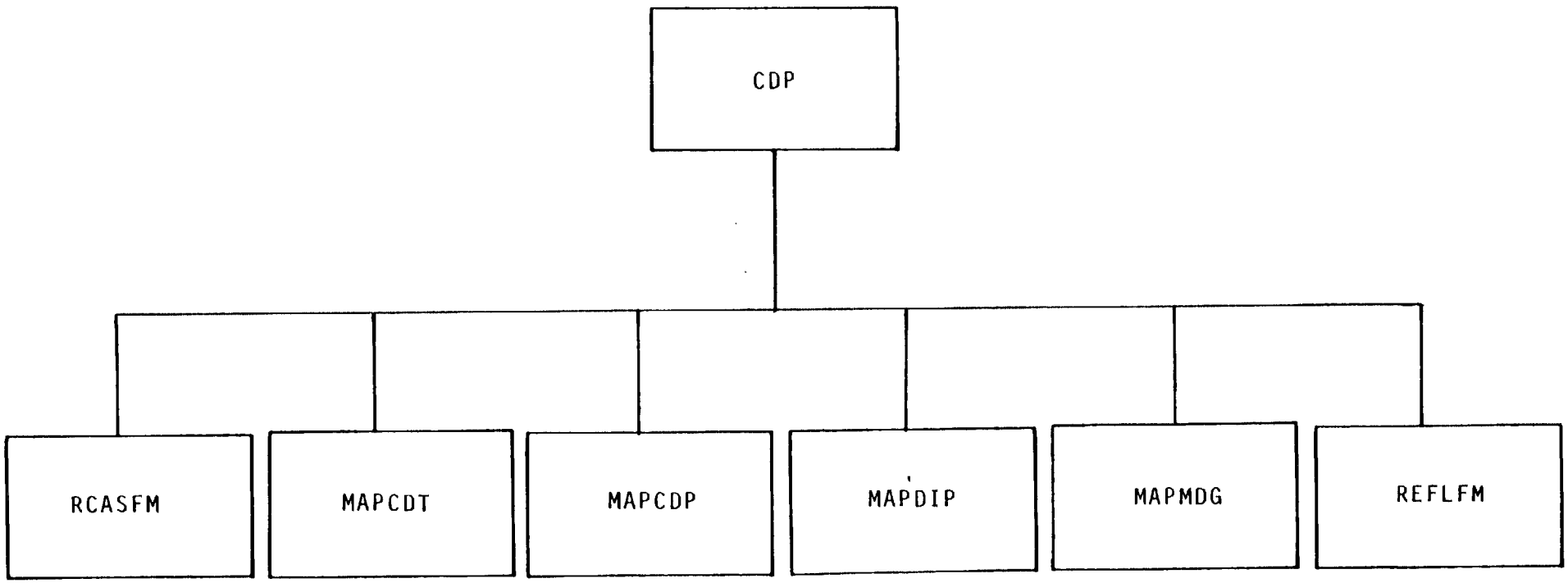


Figure 10.1 - CDP Structure Overlay (0,0)

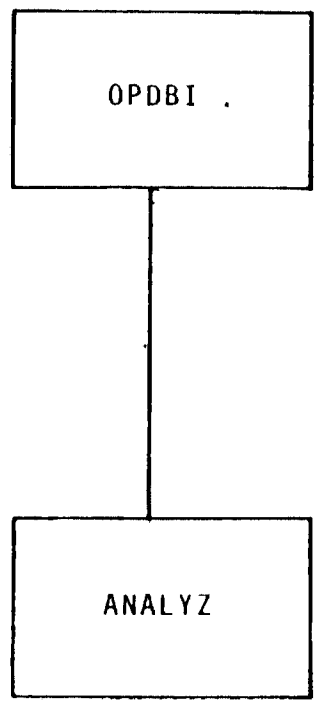


Figure 10.2 - CDP Structure Overlay (1,0)

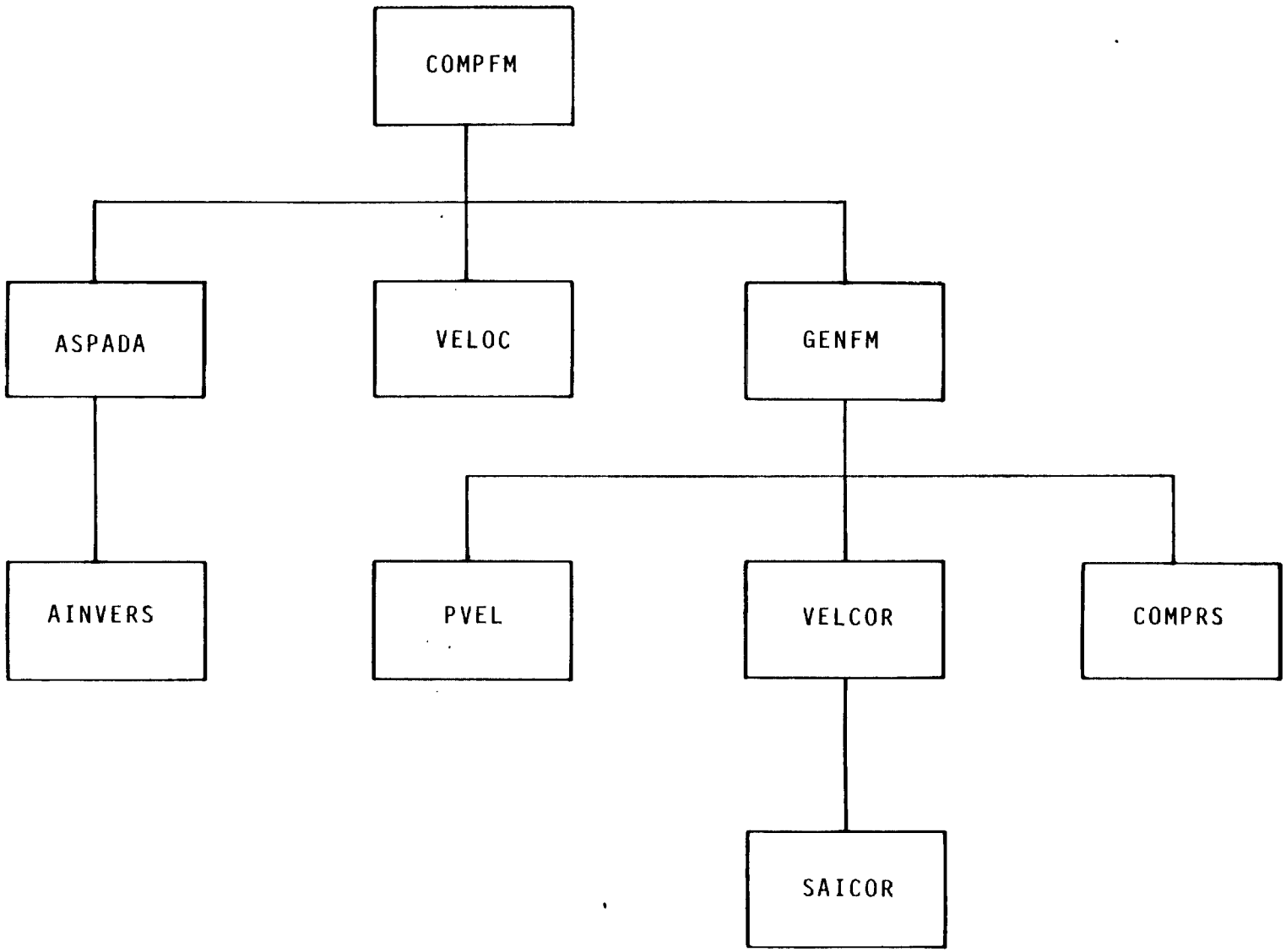


Figure 10.3 - CDP Structure Overlay (2,0)

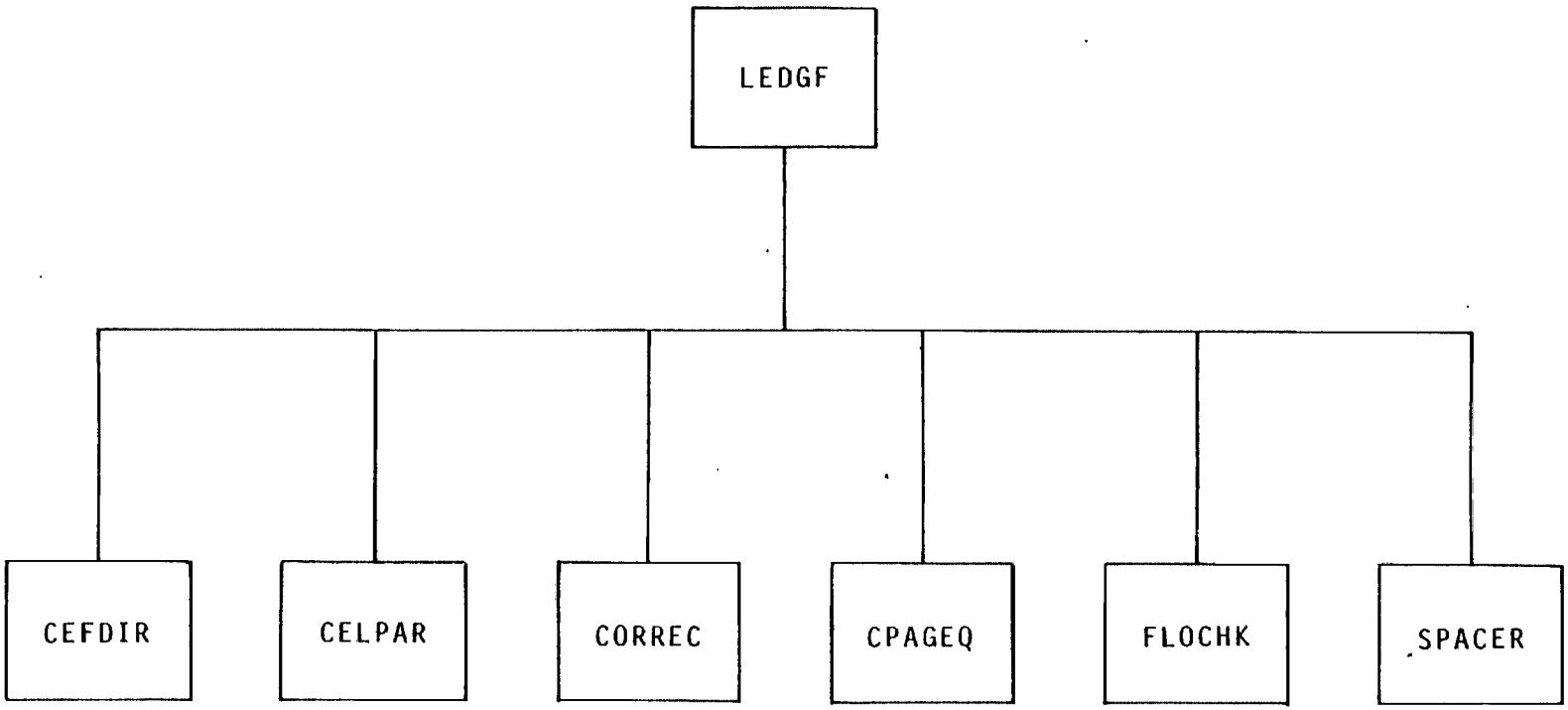


Figure 10.4 - CDP Structure Overlay (3,0)

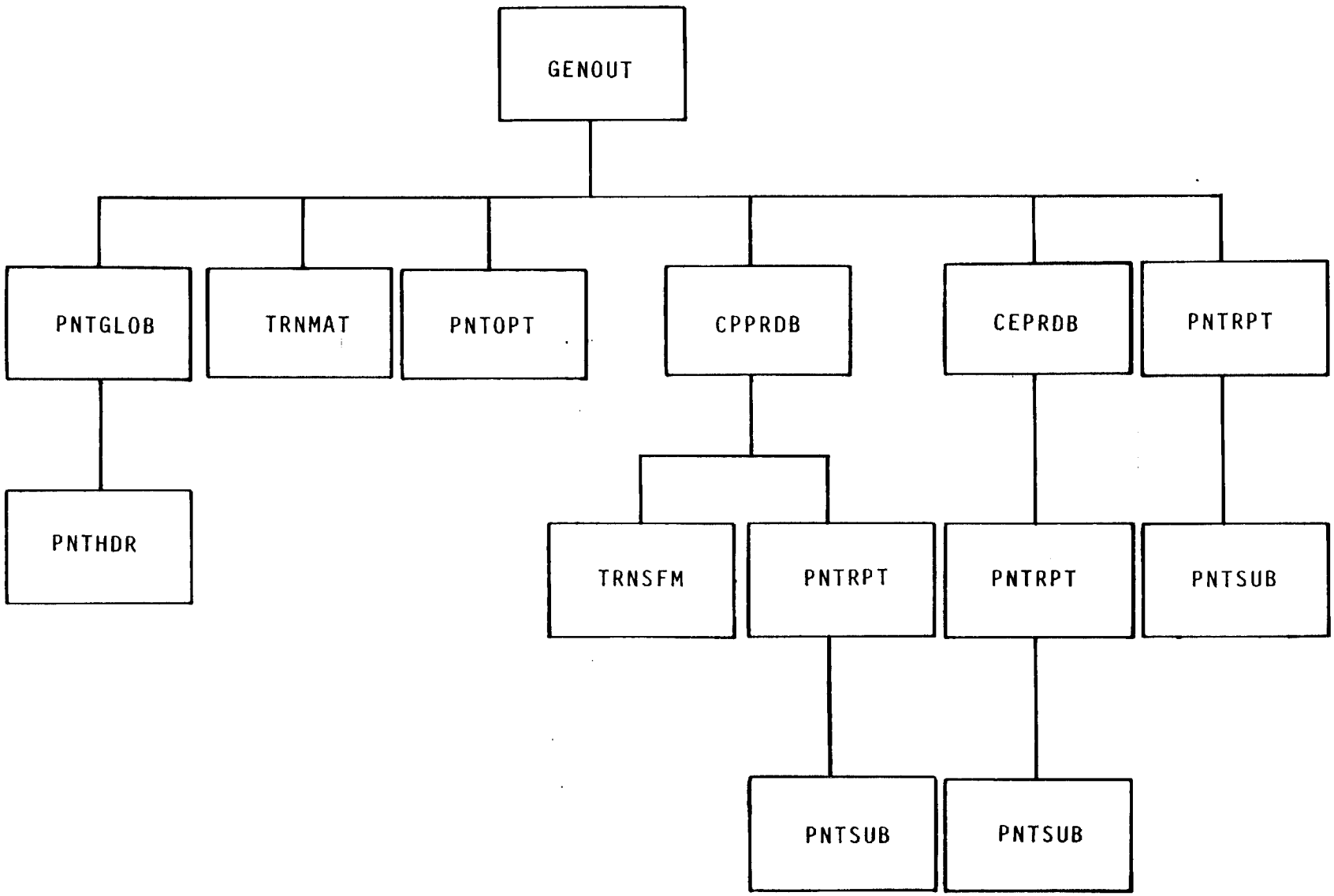


Figure 10.5 - CDP Structure Overlay (4,0)

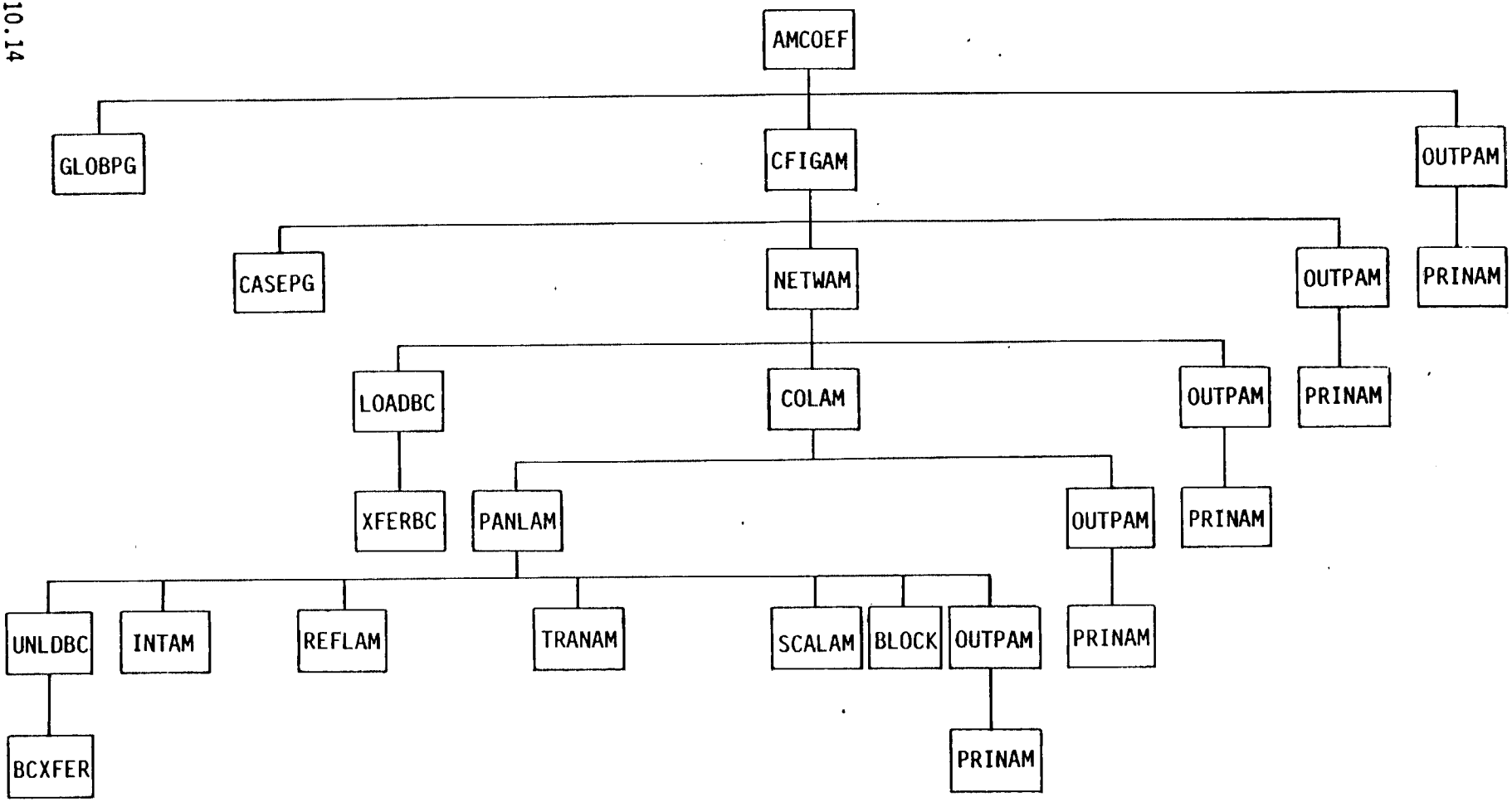


Figure 10.6 - CDP Structure Overlay (5,0)

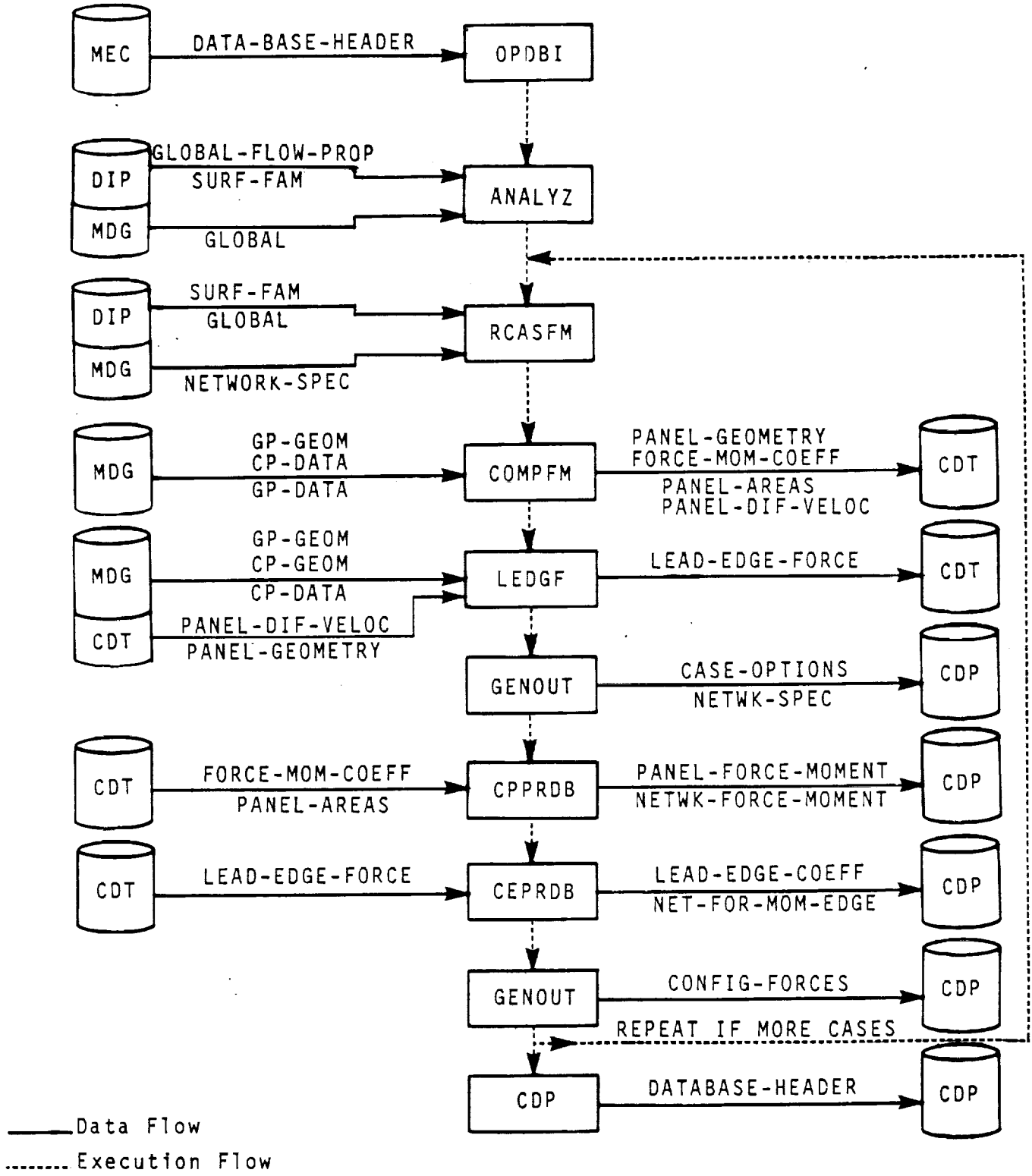


Figure 10.7 - Data Execution Flow for Forces and Moments

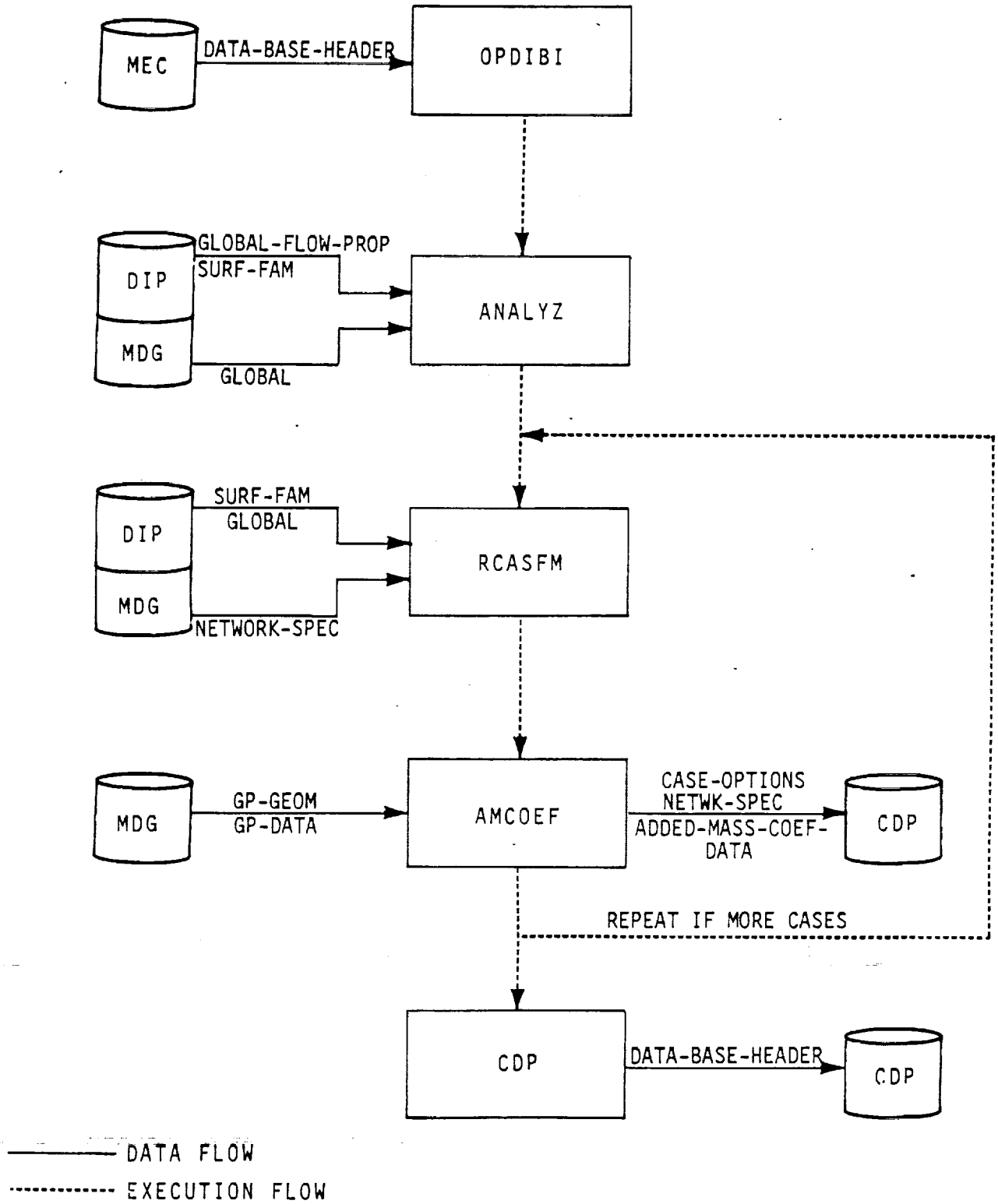


Figure 10.8 - Data Execution Flow for Added Mass Coefficients

APPENDIX 10-A TREE STRUCTURE

The tree structure diagram of the CDP moule has been deleted from this document. It is, however, available on the installation tape.

APPENDIX 10-B FUNCTIONAL DECOMPOSITION

The functional decomposition of the CDP module is presented here. The decomposition labels are given in the order of their execution and therefore may not be alphabetic.

A Initiate Program Execution
- A Initialize
- B Initiate Program Start
- C Initiate SDMS

B Overlay (1,0), Program OPDBI - Check Databases and Initialize Global Data
- A Open MEC Database
- B Get Run Identification Information
  - A Define MEC Header Map
  - B Get MEC Header Dataset
- C Check Databases Required by CDP
  - A Check DIP Database
  - B Check MDG Database
  - C Check CDP Temporary Database
- D Open Databases and Define SDMS Maps
  - A Open DIP and MDG Databases
  - B MAPDIP - Define SDMS Maps for DIP Datasets (see label CB)
  - C MAPMDG - Define SDMS MAps for MDG Datasets (see label CC)
- E ANALYZ - Analyze Datasets and Setup Global Data
  - A Initialize
  - B Read Global Data
  - C Determine Database/Print Option
    - A Obtain DIP Option Data
    - B Set CDP Print Flag
    - C Set CDP Database Flag
  - D Check Permanent CDP Databases
  - E Form the Image Index Array
    - A Initialize
    - B Set Images 2, 3, 4 Equivalent to Image 1
    - C Set Image 2 Equivalent to Image 1 and Set Image 3 Equivalent to Image 4
    - D Set Image 3 Equivalent to Image 2 and Set Image 4 Equivalent to Image 1
    - E Set Image 2 Equivalent to Image 1
    - F Set Image 4 Equivalent to Image 1
  - F Compute transformation Matrix for Images
    - A Compute for First and Second Quadrants
    - B Get Third Reflection Matrix
- F Close DIP and MDG Databases

C Open Databases and Define SDMS Maps
- A Open DIP and MDG Databases
- B MAPDIP - Define DIP Maps
  - A Initialize
  - B Select Maps for DIP Database Datasets
    - A Define Global Flow Properties Map
    - B Define Surface Forces and Moments Map
    - C Define Case Map
    - D Define Global Solution Number Map
- C MAPMDG - Define MDG Maps
  - A Initialize
  - B Select Maps for MDG Database Datasets
    - A Define GLobal Map
    - B Define Network Specifications Map

C Define Control Point Geometry Map
D Define Grid Point Geometry Map
E Define Control Point Data Map
F Define Grid Point Data Map
G Define Solution Map
H Define Grid Point COordinates Map
I Define Control Point Geometry Sub Map
J Define Control Point Data Sub Map

D Open CDP Permanent Database

E MAPCDP - Define SDMS Maps for CDP Datasets

A Define Map for Dataset CASE-OPTIONS
A Define Map Name
B Define Static Maps for Keys and Elements
C End Map Definition

B Define Map for Dataset NETWK-SPEC
A Define Map Name
B Define Combination Static and Dynamic Map for Keys and Elements
C End Mapping

C Define Map for Dataset PANEL-FORCE-MOMENT
A Define Map Name
B Define Static Map for Keys and Elements
C End Mapping

D Define Map for Dataset LEAD-EDGE-COEFF
A Define Map Name
B Define Combination Static and Dynamic Maps for Keys and Elements
C End Mapping

E Define Map for Dataset NETWORK-FORCE-ELEMENT
A Define Map Name
B Define Combination Static and Dynamic Maps for Keys and Elements
C End Mapping

F Define Map for Dataset CONFIG-FORCES
A Define Map Name
B Define Combination Static and Dynamic Maps for Keys and Elements
C End Mapping

G Define Map for Dataset NET-FOR-MOM-EDGE
A Define Map Name
B Define Combination Static and Dynamic Maps for Keys and Elements
C End Mapping

H Define Map for Dataset DATA-BASE-HEADER
A Define Map Name
B Define Combination Static and Dynamic Maps for Keys and Elements
C End Mapping

D Read Forces and Moments Options and Prepare Temporary Database

A RCASFM - Read Forces and Moments Options for Case and Analyze Requirements
A Initialize
B Read Surface Forces and Moments Data on DIP
C Get Global Solution List
D Set Edge Force Flag

E Rearrange Data in Vectors
  A Arrange Pressure Rule Requests
  B Generate Print Request Vector
  C Generate Database Request Vector
  D Generate Axis System Vectors
  E Generate Computation Options Vectors
F Print Output Requirements for this Case
  A Print Options Selected
  B Estimate Resources
    A Read Network Data and Options
    B Print Resource Estimates
B Open CDP Temporary Database
C MAPCDT - Define SDMS Maps for CDPT Datasets
  A Define Map for Dataset FORCE-MOM-COEFF
    A Define Map Name
    B Define Static Map for Keys and Elements
    C End Mapping
  B Define Map for Dataset LEAD-EDGE-FORCE
    A Define Map Name
    B Define Static Map for Keys and Elements
    C End Mapping
  C Define Maps for Dataset PANEL-AREAS
    A Define Map Name
    B Define Static Map for Keys and Elements
    C End Mapping
  E Define Map for Dataset DATABASE-HEADER
    A Define Map Name
    B Define Static Map for Keys and Elements
    C End Mapping

E Overlay (2,0), Program COMPFM - Compute Forces and Moments for the RCS
  A Initialize for the Current Case
  B Obtain Network Images, Options and Data
    A Decipher Network Images
    B Read Network DAta and Options
  C Get Geometry and Minimal Data
    A Construct Key Sets
    B Read Grid Point Geometry Dataset
    C ASPADA - Assemble Panel Geometry and Data
      A Initialize
      B Assemble Vector for Grid Point Correspondence
      C Assemble Grid Point Geometry for 1st Solution
        A Assmeble Doublet Strength Integral Vector
        B Assemble Grid Point Doublet Integrals
          A Assemble Doublet Far Field Integral
          B Assemble Doublet Dipole Moment Integral
          C Assmeble Normal Cross Product Moment Integral
        C Compute Panel Conormal Vectors
        D AINVERS - Compute A^{1} for Velocity Calculation
          A Initialize
            A Zero Local Storage
            B Set Flags for Triangular Panel
          B Form A Matrix
          C Modify Conormal
          D Form A^{-1}

```
                E    Reset A^-1 for Triangular Panel
                F    Compute Panel Area and Volume Flow
        D    Assemble Grid Point Minimal Data
             A    Assmble Mass Flux
             B    Assemble Doublet Singularity
             C    Assemble Source Singularity
             D    Assemble Potential
             E    Assemble Average Velocities
D    Obtain Minimal Data for this Solution
     A    Read Grid Point MDG Database Dataset
     B    Get Local Onset Flow from MDG Dataset
     C    ASPADA - Assemble Panel Minimal Data (see label ECC)
E    Compute Panel Velocities
     A    VELOC - Compute Difference Velocities
          A    Initialize
          B    Compute Velocities at Panel Corner Points
          C    Compute Velocities at Edge Midpoints
          D    Compute Velocities at Panel Center
          E    Reset Velcoity for a Collapsed Edge
     B    VELOC - Compute Average Velocities (see label EEA)
     C    Write Doublet to CDT-DB
     D    Write Edge Midpoints to CDT-DB
F    GENFM - Generate Forces and Moments
     A    Initialize
          A    Zero Local Storage
          B    Set Constants
          C    Set Requested Surface Indicator
          D    PVEL - Computer Perturbation Velocity
               A    Compute Velocity on Upper Surface
               B    Compute Velocity on Lower Surface
          E    Compute Corrected Velocity
               A    Compute the Magnitude of Local Onset Flow
               B    Compute Total Velocity
               C    Compute the Magnitude of Total Velocity
               D    Compute the Total Mass Flux
               E    Compute the Magnitude of Total Mass Flux
               F    VELCOR - Perform the Velocity Correction
                    A    Compute the Component of Perturbation
                         Velocity in the Freestream Velocity Direction
                    B    Compute SA1 Correction
                         A    SA1COR - Compute Correction to Total
                              Velocity by Newton's Method
                              A    Compute Constant Quantities
                              B    Set X-Component of Total Velocity
                              C    Compute Increment to X-Component
                                   of Velocity by Newton's Method
                              D    Compute New Value of Total Velocity
                              E    Repeat Iteration
                              F    Branch Out to Loop
                              G    Generate Informative Message
                              H    Correct X-Component of Total
                                   Velocity
                         B    Compute Correction to X-Component of
                              Perturbation Velocity
                         C    Compute Magnitude of Total Velocity
```

- A Compute Corrected Velocity by Method 1 (Mulitply Perturbation Mass Flux by the Ratio of Perturbation Velocity and Perturbation Mass Flux)
- B Compute Corrected Velocity by Method 2
  - A Compute Ratio of Densities (Local to Freestream)
  - B Compute Velocity Correction (Multiply Perturbation Mass Flux by Ratio of Densities)

B Compute Upper and Lower Force and Moment
- A Compute Surface Mass Flux
- B Compute Momentum Transfer Terms
  - A Compute Momentum Flux
  - B Compute Momentum Transfer Term for Force
  - C Compute Momentum Transfer Term for Moment
    - A Compute Term Under Summation
    - B Cross Summation Term with Momentum Flux
    - C Combine Terms
- C Compute Pressure Coefficient
  - A Get Grid Point Geometry and Velocity
  - B Compute Total Velocity
  - C Compute Magnitude of Total Velocity
  - D COMPRS - Perform Pressure Coefficient Computation
    - A Compute Perturbation and Local Incremental Onset Flow Velocities Along User Preferred Direction
      - A Save Uniform Onset Flow Vector and Magnitude
      - B Save Compressibility Vector and Magnitude
      - C Transform Perturbation Velocity
      - D Transform Local Incremental Onset Flow Velocity
      - E Transform Uniform Onset Flow Velocity
      - F Find DLTAE and DLTAE2
    - B Compute Maximum and Critical Speed and Total Velocity, Squares of Perturbation and Local Incremental Onset Flow Velocities
      - A Compute Maximum Speed
      - B Compute Critical Speed
    - C Compute Pressure Coefficients
      - A Compute for Isentropic Approximation
        - A Compute Pressure Coefficient in Flow
        - B Compute Pressure Coefficient at Vacuum Condition
      - B Compute for Second Order Approximation
        - A Compute CPSO
        - B Compute Corresponding Pressure Coefficient at Vacuum Condition
      - C Compute for Reduced Second Order
        - A Compute Pressure Coefficient in Flow

B Compute Corresponding Pressure Coefficient at Vacuum Condition
D Compute for Slender Body Approximations
A Compute Pressure Coefficient in Flow
B Compute Pressure Coefficient at Vacuum Condition
E Compute for Linear Approximation
A Compute Pressure Coefficient in Flow
B Compute Pressure Coefficient at Vacuum Condition
D Compute Local Mach Numbers
A Compute for Isentropic Approximation
B Compute for Reduced Second Order
C Compute for Slender Body Approximation
D Compute for Linear Approximation
E Retrieve Pressure Coefficients
D Compute Pressure Terms
A Compute Pressure Terms for Force
B Compute Pressure Terms for Moment
A Compute First Term
B Compute Second Term
C Combine Pressure Terms in Moment Coefficient
E Compute Surface Force and Moment
C Compute Force and Moment for Requested Surface
A Combine Force and Moment from Current Surface
B REFLEM - Reflect Force and Moment
A Initialize
A Compute Scaling Constant
B Retrieve Input Image Force and Moment
B Get Force and Moment from Previous Image
C Transform Force to Next Image
D Compute Moment Under Current Reflection
A Compute Second Term
B Compute Third Term
C Combine All Terms
E Return Transformed Force and Moment
G Store Forces and Moments on CDPT
H Rearrange Data for Next Panel in Column

F Overlay (3,0), Program LEDGF - Compute Leading Edge Forces
A Edge Preparation
A Check for Valid Edge Request
B Analyze Panel Spacing and Required Corrections
A Assemble Geometry Required to Check Spacing
B SPACER - Check Panel Spacing
C CORREC - Compute Correction Factor
B Get Geometry for Panel
C CPAGEQ - Compute Associated Geometric Quantities
A Assemble Panel Geometry
B Compute Edge Length and Edge Tangent
C Compute Panel Normal
D Compute Edge Normal
E Get Edge Midpoint

D Get Doublet Strength
E Perform Computations
A CEFDIR - Compute Edge Force Direction
B CELPAR - Compute Edge Parameter
C Compute Edge Force Magnitude
D Compute Edge Force and Moment
F Write Edge Force and Moment to Temporary Database

G Overlay (4,0), Program GENOUT - Transform to Requested Axis Systems and Write Forces and Moments on CDP Database and Output File
A Initialize
A Set Constants
B Zero Network Accumulated Data
C PNTGLOB - Print CDP Global Data
A PNTHDR - Print Header Lines
A Increment Output Page Number by One
B Write First Output Line - Program Announcement
C Write Second Line - Problem Identification
D Write Third Line - Run Identification
E Write Fourth Line - User Identification
F Initialize Line Count Appropriately
B Print Networks Information
C Print Solution Information
D Print Symmetry Information
E Print Mach Number, CALPHA and CBETA
A Print Mach Number
B Compute CALPHA and CBETA
C Print CALPHA and CBETA
D PNTOPT - Print CDP Case - Options Data
A Print Page Header
B Print Number of Networks, Velcoity Options and Surface Selection Data
C Print Information on Selected Axis Systems
D Print Global Reference Values
E Print Local Reference Values
F Print Solution Information
E Write CDP Case Options
B TRNMAT - Compute Transformation Matrices for Selected Axis Systems
A Initialize Transformation Matrix to Zeroes
B Get Index for the Axis from the List
C Compute Matric Constant Terms
A Use Angle of Attack and Sideslip
B Use Input Euler Angles
C Perform Computation
D Compute First Column of Matrix
E Compute Second Columne of Matrix
F Compute Third Columne of Matrix
C Write Network Specification Dataset
D CPPRDB - Compute Panel Forces and Moments and Produce Print and Database Output
A Initialize
A Zero Local Storage
B Get Panel Areas
C Include Any Edge Forces

B Read Forces and Moment Data for the Panel in the Reference Axis System from the CDPT Database
C TRNSFM - Transform Forces and Moments
  A Transform Panel Forces and Moments and Store in Temporary Arrays TEMPF and TEMPM
  B Restore to PANFOR and PANMOM Arrays
D Accumulate Forces and Moments Data
E PNTRPT - Print Panel Forces and Moments
  A PNTSUB - Print New Page of Report
    A Increment Output Page Number by 1
    B Write Standard CDP Header
    C Write Header for Network, Solution, Image Identification, etc.
    D Print Subheaders for Report
    E Save Solution, Network, Image Indices
  B Prepare Option Names and Data
    A Process Option Names For Printing
    B Scale Forces and Moments
  C Print Forces and Moments Data
    A Print Individual Panel Data
    B Print Column Sum Data
    C Print Network Sum Data
    D Print Configuration Sum DAta
    E Print Panel Edge Data
    F Print Network Edge Data
    G Print Accumulated Data
  D Save the Solution, Network, Image and Option Indices
F Store Panel Forces and Moments in the CDP Database
  A SCALFM - Scale Forces and Moments To Be Written
  B Write Data
G Output Column Sum of Forces and Moments Data
  A Get Accumulated Data
  B PNTRPT - Print Column Sum (see label GDE)
  C Store Columne Sum in CDP Database
    A SCALFM - Scale Forces and Moments To Be Written
    B Write Data
H Accumulate Forces and Moments Data for Network
I Output Network Sum of Forces and Moments
  A Get Accumulated Data for Network
  B PNTRPT - Print Network Sum (see label GDE)
  C Output Network Sum to CDP Database
    A SCALFM - Scale Forces and Moments To Be Written
    B Write Data

E CEPRDB - Compute and Store Edge Forces and Moments
  A Obtain Edge Flag
  B Zero Edge Arrays
  C Get Number of Panels
  D Read Edge Forces and Moments
  E Transform Edge Results
  F Accumulate Edge Results
  G PNTRPT - Print Edge Forces and Moments for Panel (see label GDE)
  H Store Panel Edge Forces and Moments on Database
    A SCALFM - Scale Forces and Moments To Be Written
    B Write Data

```
        I    PNTRPT - Print Total Edge Forces and Moments (see label GDE)
        J    Store Total Edge Forces and Moments on Database
             A    SCALFM - Scale Forces and Moments To Be Written
             B    Write Data
   F    Output Configuration Sum
        A    PNTRPT - Print Configuration Sum (see label GDE)
        B    Store Configuration Sum on Database
             A    SCALFM - Scale Forces and Moments To Be Written
             B    Write Data
   G    Output Accumulation Sum
        A    PNTRPT - Print Accumulation Sum (see lable GDE)
        B    Store Accumulation Sum on Database
             A    SCALFM - Scale Forces and Moments To Be Written
             B    Write Data

H  Return and Close CDPT Database

I  Close DIP, MDG and CDP Database

J  End Program
K  Overlay (5,0) Program AMCPEF - Compute Added Mass Coefficients
   A    Initialize
        A    GLOBPG - Write Global Summary Page
        B    Zero Accumulation Total
        C    Initiate Blank Common
   B    CFIGAM - Compute Configuration Total
        A    CASEPG - Write Case Summary Page
        B    Write Case Options Dataset
        C    Zero Configuration Total
        D    Write Network Specifications Dataset
        E    NETWAM - Compute Network Total
             A    Initialize
                  A    Zero Network Total
                  B    Set Blank Common Array Dimensions
                  C    Add Array to Blank Common
             B    LOADBC - Load Blank Common with MDG Data
                  A    Store Geometry Data in Blank Common
                       A    Set Fine Grid Keys
                       B    Get Geometry Data
                       C    XFERBC - Store Moment Matrices in Blank Common
                       D    XFERBC - Store Panel Center in Blank Commen
                  B    Store Solution Data in Blank Common
                       A    Retrieve Panel Lower Grid Point Sets
                       B    Retrieve Panel Upper Grid Point Sets
                       C    Realign Grid Point Sets
                       D    XFERBC - Move Data to Blank Common
                       E    Shift Grid Point Data
             C    COLAM - Compute Column Total
                  A    Zero Column Total
                  B    PANLAM - Compute Panel Coefficients
                       A    UNLDBC - Unload Data from Blank Common
                            A    BCXFER - Retrieve Geometry Data
                            B    BCXFER - Retrieve Solution Data
                            C    Align Solution Data
```

B INTAM - Perform Integration
A Initialize
B Compute Surface Potential
C Compute Surface Matrices
D Add Surface Matrix Contribution
C REFLAM - Reflect Coefficients
A Compute Reflection Constants
B Perform Added Mass Reflection
D TRANAM - Translate Coefficients
A Compute Transformation Matrices
B Translate Coefficients to New Axis System
E SCALAM - Scale Coefficients
F OUTPAM - Output Panel Total
A PRINAM - Print Added Mass Coefficients
B Write Added Mass Coefficients
C Increment Column Total
D OUTPAM - Output Column Total
A PRINAM - Point Added Mass Coefficients
B Write Added Mass Coefficients
D Increment Network Total
E Delete Array from Blank Common
F OUTPAM - Output Network Total
A PRINAM - Print Added Mass Coefficients
B Write Added Mass Coefficients
F Increment Configuration Total
G OUTPAM - Output Configuration Total
A PRINAM - Print Added Mass Coefficient
B Write Added Mass Coefficients
C Increment Accumulation Total
D OUTPAM - Output Accumulation Total
A PRINAM - Print Added Mass Coefficients
B Write Added Mass Coefficients

APPENDIX 10-C DATA BASE COMMUNICATIONS CHART

The data base communications chart is presented in three forms. The first form has a column order of Data Base, Dataset Name, Map Name, Common Block, and Program/Subroutine. The second form has a column order of Data Base, Map Name, Dataset Name, Common Block, and Program/Subroutine. The third form has a column order of Common Block, Data Base, Map Name, Dataset Name, and Program/Subroutine. Thus a person can get a cross reference on a data element by knowing either the Dataset Name, Map Name or Common Block.

FIRST FORM

DATA BASE	DATASET NAME	MAP NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
CDP	ADDED-MASS-COEF-DATA	ADDMASSCO	/KEYLIST/	OUTPAM
CDP	CASE-OPTIONS	OPTNMAP	/CASES/	RCASFM
			/SOLS/	
			/NETWK/	
			/ACCUM/	
CDP	DATABASE-HEADER	DRMAP	/RUNIDS/	CDP
CDP	CONFIG-FORCES	CONFIMAP	/CASES/	GENOUT
			/ACCUM/	
			dynamic	
CDP	PANEL-FORCE-MOMENT	PANELMAP	/CASES/	CPPRDB
			/NETWK/	
			/LEGEOM/	
			/AACUM/	
			/PANDAT/	
CDP	LEAD-EDGE-COEFF	EDGEMAP	/CASES/	CEPRDB
			dynamic	
CDP	NETWK-SPEC	NETSPCMAP	/CASES/	GENOUT
			/NETWK/	
CDP	NETWORK-FORCE-MOMENT	NETWKMAP	/CASES/	CPPRDB
			/ACCUM/	
			/FMACCU/	
CDPT	FORCE-MOM-COEFF	COEFFMAP	/CASES/	COMPFM
			/LEGEOM/	
			/ACCUM/	
			/PANDAT/	

DATA BASE	DATASET-NAME	MAP-NAME	COMMON-BLOCK	PROGRAM/ SUBROUTINE
CDPT	LEAD-EDGE-FORCE	EDGEMAPT	/CASES/	LEDGF
			/EDGSIN/	
			dynamic*	
CDPT	PANEL-AREAS	AREAMAP	/CASES/	COMPFM
			/PANDAT/	
CDPT	PANEL-DIF-VELOC	VELDMAP	dynamic*	COMPFM
				LEDGF
CDPT	PANEL-GEOMETRY	PAGEOMAP	dynamic*	COMPFM
				LEDGF
DIP	GLOBAL	GLOBNVMRHS	/CASES/	RCASFM
DIP	GLOBAL-FLOW-PROP	GLOBALPRO	/CASES/	ANALYZ
DIP	SURF-FAM	CASEMAP	/CASES/	ANALYZ
			/ACCUM/	
DIP	SURF-FAM	SURFAM	/CASES/	RCASFM
			/SOLS/	
			/ACCUM/	
			/NETWK/	
MDG	GLOBAL	MDGGLOBAL	/NETWK/	ANALYZ
MDG	NETWORK-SPEC	NETSPEC	/NETWK/	RCASFM
MDG	CP-GEOM	CPGEOM	dynamic*	

DATA BASE	DATASET-NAME	MAP-NAME	COMMON-BLOCK	PROGRAM/ SUBROUTINE
MDG	CP-GEOM	GPGEOM	/CASES/ /PGEOM/	COMPFM
MDG	CP-DATA	CPDATA	/GLOBAL/ /CASES/ dynamic*	COMPFM
MDG	GP-DATA	GPDATA	/CASES/ /MINDAT/ /GLOBAL/ /PGEOM/	COMPFM
MDG	SOLUTION-DATA	SOLDATA	/SOLS/	
MDG	GP-GEOM	GPCOORDS	/PGEOM/ /LEGEOM/	LEDGF
MDG	CP-GEOM	CPCOORDS	/CASES/ /LEGEOM/	LEDGF
MDG	CP-DATA	CONDATA	/CASES/ /GLOBAL/ /CPVEL/	LEDGF

SECOND FORM

DATA BASE	MAP NAME	DATASET-NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
CDP	ADDMASSCO	ADDED-MASS-COEF-DATA	/KEYLIST/	OUTPAM
			dynamic	
CDP	OPTNMAP	CASE-OPTIONS	/CASES/	RCASFM
			/SOLS/	
			/NETWK/	
			/ACCUM/	
CDP	DRMAP	DATABASE-HEADER	/RUNIDS/	CDP
CDP	CONFIMAP	CONFIG-FORCES	/CASES/	GENOUT
			/ACCUM/	
			dynamic*	
CDP	PANELMAP	PANEL-FORCE-MOMENT	/CASES/	CPPRDB
			/NETWK/	
			/LEGEOM/	
			/AACUM/	
			/PANDAT/	
CDP	EDGEMAP	LEAD-EDGE-COEFF	/CASES/	CEPRDB
			dynamic*	
CDP	NETSPCMAP	NETWK-SPEC	/CASES/	GENOUT
			/NETWK/	
CDP	NETWKMAP	NETWORK-FORCE-MOMENT	/CASES/	CPPRDB
			/ACCUM/	
			/FMACCU/	

DATA BASE	MAP NAME	DATASET-NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
CDPT	COEFFMAP	FORCE-MOM-COEFF	/CASES/ /LEGEOM/ /ACCUM/ /PANDAT/	COMPFM
CDPT	EDGEMAPT	LEAD-EDGE-FORCE	/CASES/ /EDGSIN/ dynamic*	LEDGF
CDPT	AREAMAP	PANEL-AREAS	/CASES/ /PANDAT/	COMPFM
CDPT	VELDMAP	PANEL-DIF-VELOC	dynamic*	COMPFM LEDGF
CDPT	PAGEOMAP	PANEL-GEOMETRY	dynamic*	COMPFM LEDGF
DIP	GLOBNVMRHS	GLOBAL	/CASES/	RCASFM
DIP	GLOBALPRO	GLOBAL-FLOW-PROP	/CASES/	ANALYZ
DIP	CASEMAP	SURF-FAM	/CASES/ /ACCUM/	ANALYZ
DIP	SURFAM	SURF-FAM	/CASES/ /SOLS/ /ACCUM/ /NETWK/	RCASFM
MDG	MDGGLOBAL	GLOBAL	/NETWK/	ANALYZ

DATA BASE	MAP NAME	DATASET-NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
MDG	NETSPEC	NETWORK-SPEC	/NETWK/	RCASFM
MDG	CPGEOM	CP-GEOM	dynamic*	
MDG	GPGEOM	CP-GEOM	/CASES/	COMPFM
			/PGEOM/	
MDG	CPDATA	CP-DATA	/GLOBAL/	COMPFM
			/CASES/	
			dynamic*	
MDG	GPDATA	GP-DATA	/CASES/	COMPFM
			/MINDAT/	
			/GLOBAL/	
			/PGEOM/	
MDG	SOLDATA	SOLUTION-DATA	/SOLS/	
MDG	GPCOORDS	GP-GEOM	/PGEOM/	LEDGF
			/LEGEOM/	
MDG	CPCOORDS	CP-GEOM	/CASES/	LEDGF
			/LEGEOM/	
MDG	CONDATA	CP-DATA	/CASES/	LEDGF
			/GLOBAL/	
			/CPVEL/	

THIRD FORM

COMMON BLOCK	DATA BASE	MAP NAME	DATASET NAME	PROGRAM/ SUBROUTINE
/KEYLIST/	CDP	ADDED-MASS-COEF-DATA	ADDMASSCO	OUTPAM
/CASES/	CDP	CASE-OPTIONS	OPTNMAP	RCASFM
/SOLS/				
/NETWK/				
/ACCUM/				
/RUNIDS/	CDP	DATABASE-HEADER	DRMAP	CDP
/CASES/	CDP	CONFIG-FORCES	CONFIMAP	GENOUT
/ACCUM/				
dynamic*				
/CASES/	CDP	PANEL-FORCE-MOMENT	PANELMAP	CPPRDB
/NETWK/				
/LEGEOM/				
/AACUM/				
/PANDAT/				
/CASES/	CDP	LEAD-EDGE-COEFF	EDGEMAP	CEPRDB
dynamic*				
/CASES/	CDP	NETWK-SPEC	NETSPCMAP	GENOUT
/NETWK/				
/CASES/	CDP	NETWORK-FORCE-MOMEN	NETWKMAPT	CPPRDB
/ACCUM/				
/FMACCU/				
/CASES/	CDPT	FORCE-MOM-COEFF	COEFFMAP	COMPFM
/LEGEOM/				
/ACCUM/				
/PANDAT/				

COMMON BLOCK	DATA BASE	MAP NAME	DATASET NAME	PROGRAM/ SUBROUTINE
/CASES/	CDPT	LEAD-EDGE-FORCE	EDGEMAPT	LEDGF
/EDGSIN/				
dynamic*				
/CASES/	CDPT	PANEL-AREAS	AREAMAP	COMPFM
/PANDAT/				
dynamic*	CDPT	PANEL-DIF-VELOC	VELDMAP	COMPFM
				LEDGF
dynamic*	CDPT	PANEL-GEOMETRY	PAGEOMAP	COMPFM
				LEDGF
/CASES/	DIP	GLOBAL	GLOBNVMRHS	RCASFM
/CASES/	DIP	GLOBAL-FLOW-PROP	GLOBALPRO	ANALYZ
/CASES/	DIP	SURF-FAM	CASEMAP	ANALYZ
/ACCUM/				
/CASES/	DIP	SURF-FAM	SURFAM	RCASFM
/SOLS/				
/ACCUM/				
/NETWK/				
/NETWK/	MDG	GLOBAL	MDGGLOBAL	ANALYZ
/NETWK/	MDG	NETWORK-SPEC	NETSPEC	RCASFM
dynamic*	MDG	CP-GEOM	CPGEOM	
/CASES/	MDG	CP-GEOM	GPGEOM	COMPFM
/PGEOM/				
/GLOBAL/	MDG	CP-DATA	CPDATA	COMPFM

COMMON BLOCK	DATA BASE	MAP NAME	DATASET NAME	PROGRAM/ SUBROUTINE
/CASES/				
dynamic*				
/CASES/	MDG	GP-DATA	GPDATA	COMPFM
/MINDAT/				
/GLOBAL/				
/PGEOM/				
/SOLS/	MDG	SOLUTION-DATA	SOLDATA	
/PGEOM/	MDG	GP-GEOM	GPCOORDS	LEDGF
/LEGEOM/				
/CASES/	MDG	CP-GEOM	CPCOORDS	LEDGF
/LEGEOM/				
/CASES/	MDG	CP-DATA	CONDATA	LEDGF
/GLOBAL/				
/CPVEL/				

* Dynamic mapping of the dataset is used for all or some of the dataset elements, thus requiring no common block storage for these elements. See Section 13 of this document for details of dynamic mapping.

APPENDIX 10-D MASTER DEFINTION

The data base master definition listings of the CDP module has been deleted from this document. These are produced from the PAN AIR tape during installation.

11.0 PRINT PLOT PROCESSOR (PPP) MODULE

11.1 INTRODUCTION

The scope of the PPP module has changed since the conception of PAN AIR. Originally, PPP was to generate all of the PAN AIR printed and plotted output. Now, however, each of the PAN AIR modules produces its own printed output while cycling through its calculations. This reduces the amount of information which must be saved for later processing.

PPP functions as a simple editor, extracting user selected data from the PAN AIR data bases and reformatting the information for convenience in preparing plot files of PAN AIR results.

PPP provides a running account of the module progress in the form of Run Name descriptors to identify the type of options executed and label on the plot files for DQG, PDP and CDP data. The plot file formats are described in Appendices E, F, and G for geometry plot file, point data plot file and configuration forces and moments data plot file respectively. Tables 11.2 and 11.3 give the lists of the parameter names for point data and configuration forces and moments data. Also, a major function of identifying point data and configuration data are accomplished by creating parameter name list headings for the selected data items on the plot files as given in Tables 11.2 and 11.3 respectively.

11.2 PPP OVERVIEW

In order to prepare the plot files of geometry data from the DQG data base, point data from the PDP data base and the configuration forces and moments data from the CDP data base, the following steps are performed:

Step 1. User Data Selection Process

In describing the problem the user selects from a limited menu of PAN AIR data falling in three categories (see Section 7.0 of the PAN AIR USER'S Document, Ref. 2 for details):

Geometry data - Network panel corner points as input by the user (and sometimes slightly modified by DQG) in module GEOMPR.

Point data - Pressures and velocities to be plotted against columns/rows of network control or grid points in POINTP.

(NOTE: The current program allows only the center control point type to be selected by columns)

Configuration data- Forces and moments over panels, panel rows/columns, networks, and configurations - groups of networks to be plotted against solution parameter in module CONFIG.

(NOTE: The current program allows only configuration data to be selected by columns)

Step 2. Prepare Geometry Plot File (Corner Points)

For geometry data the user specifies the problem to be considered (this is equivalent to specifying the DQG data base name from which the geometry will be read) and the list of networks to be processed. Geometry plot file format preparation is accomplished in module GEOMPR. A description of geometry plot file is also shown in Appendix 11-E.

Step 3. Prepare Point Data Selection Plot File (Pressures and Velocities)

With point data, the user may want to consider more classifications of data to indicate the problem (PDP data base name) plus the cases, solutions, and networks to be processed. In addition, for each network a choice of plotting by panel rows or columns may be made (default = columns).

Given the above choices, PPP will extract data for all images, velocity computation options, and velocity corrections. Also, PPP selection options of data items placed on the data base (per point, by PDP) will be processed.

The data for plotting along a column/row of panels will include results at all of the column's/row's center control points, plus the two control points falling on the network edges at the ends of the column/row. Optionally, the results will be plotted at the network (fine) grid points. Point data plot file format preparation is accomplished in module POINTP. A description of point data plot file is shown in Appendix 11-F.

Step 4. Prepare Configuration Data Selection Plot File (Forces and Moments)

Configuration data selection is similar to point data. Again, the user may want to indicate the problem (CDP data base name) plus the cases, solutions, and networks to be processed. In addition, configurations (groups of networks) may be specified. Configuration data plot file format preparation is accomplished in module CONFIG. A description of configuration data plot file is shown in Appendix 11-G.

The current PPP version restricts the geometry data, point data and configuration data to be accessed from only one data base each. If further plot data from other data bases are desired, then PPP module will have to be rerun. This will limit the capability of intermixing data from different data bases in the same run.

11.2.1 Purpose of PPP

The Print Plot Processor (PPP) is a module of the PAN AIR system. Its purpose is to prepare plot data files for geometry, point data and configuration data that were generated in DQG, PDP and CDP modules as shown in Appendices 11-E, 11-F and 11-G respectively.

11.2.2 PPP Input/Output Data

11.2.2.1 Input

Inputs into PPP are fetched from the following data bases,

DIP User selections of PPP options.

MEC Data Base names, account numbers, data base status, date of execution and other similar information.
PDP User selected point data.
CDP User selected configuration data.
DQG User selected network geometry data.

11.2.2.2 Output

Printed outputs and/or plot files as shown in Appendices 11-E, 11-F, and 11-G are produced from PPP with the following information,

- User selected geometry data from DQG data base on a disk file named FT09.
- User selected point data from PDP data base on a disk file named FT10.
- User selected configuration data from CDP data base on a disk file named FT11.

The only necessary restrictions on the preparation of plot files is that the user have enough disk/file space to store the data generated for plotting. See Table 11.1 for the maximum problem limits allowable by PPP. Also, the problem sizes for a typical run are shown.

Appendix 11-D lists the possible PPP error diagnostics.

If a printout of the outputs stored on the plot files are desired, then it is necessary to copy the information stored on the temporary disk file to the print output file by Job Control Cards as follows,

```
REWIND(DN=FT09)
REWIND(DN=FT10)
REWIND(DN=FT11)
LS(FT09)
LS(FT10)
LS(FT11)
```

where,

FT09 contains Geometry Data, FT10 contains Point Data,
FT11 contains Configuration Data, and
LS is a PAN AIR JCL procedure on dataset PAPROCS

11.2.3 Data Base Interfaces

The data bases from MEC and DIP are always required as input. The data base(s) from DQG, PDP, and CDP are selected by user options and are used as input data.

11.3 MODULE DESCRIPTION

The main overlays and its subroutines are briefly summarized in this paragraph. The PPP functional decomposition is shown in Appendix 11-B. A chart of the subroutine tree structure is presented in Appendix 11-A.

11.3.1 Overall Structure

Figure 11.1 illustrates the top level structure of the PPP module.

11.3.2 Overlay Descriptions

11.3.2.1 PPP Overlay (0,0)

The top level overlay (0,0) PPP initializes and controls access to the 4 primary overlays.

11.3.2.2 PPPINT (Overlay 1,0)

Module PPPINT checks the status of the DIP, DQG, PDP and CDP data bases. The PPP selection options are initialized in a common block with data from the DIP module. Figure 11.2 shows the structure and data flow of overlay (1,0).

11.3.2.3 GEOMPR (Overlay 2,0)

Module GEOMPR gets geometry data from the DQG data base and calls subroutine GEOPLT to prepare geometry plot file (FT09). The network panel corner points data are processed in subroutine CORPTS. Figure 11.3 shows the structure and data flow of overlay (2,0).

11.3.2.4 POINTP (Overlay 3,0)

Module POINTP processes and prepares point data plot file (FT10). MAPPDP is called to perform SDMS mappings of PDP data sets. Subroutine LAYOUT is called to store global data, run identification data and surface flow quantity options in common blocks. The preparation and formatting of the point data plot file is processed in PNTPLT. Subroutine PTYP23 is called to process point type 2 for network edges and point type 3 for additional control points data. Subroutine PTYP14 is called to process point type 1 for panel center control point and point type 4 for grid points data. Figure 11.4 shows the structure and data flow of overlay (3,0).

11.3.2.5 CONFIG (Overlay 4,0)

Module CONFIG gets forces and moments on portions of a configuration from the CDP data base according to user requests and prepares a configuration data plot file (FT11). For each user defined case, the module loops over all selected solutions. The SDMS mappings for DIP and CDP data sets are processed in subroutines MAPDIP and MAPCDP. The forces and moments options are read in subroutine RCASFM and the case options are stored in a common block. Subroutine GENOUT is called to prepare the plot file for the configuration options data for panel sums in PANLSM, column sums in PCOLSM, network sums in PNETSM, configuration sums and accumulation sums. Figure 11.5 shows the structure and data flow of overlay (4,0).

11.3.3 PPP Data Base

There is no data base created in PPP. The data base communications chart is shown in Appendix 11-C.

11.3.4 PPP Interfaces

Figure 11.6 summarizes the external data interfaces between the PPP module and the MEC, DIP, DQG, PDP and CDP databases.

11.3.5 PPP Data Flow

Figures 11.2 thru 11.5 illustrate the data flow for the major sections of PPP.

11.4 LOWER LEVEL FUNCTIONS

11.4.1 Functional Decompositions

Functional decompositions are shown in Appendix 11-B and the tree structure is given in Appendix 11-A.

11.4.2 Subroutine Descriptions

BLKDAT

The Block Data subprogram initializes the items in labeled common used for print/plot purposes for DQG, PDP and CDP data.

CDPARM

Prepares CDP parameter name headings for selected options of forces and moments, pressure rules and axes systems. In addition, the first heading line consists of solution number, magnitude of uniform onset flow velocity, alpha and beta values.

CORPTS

Reads geometry corner points data from DQG-DB and writes it onto the plot file FT09.

CRUNAM

Prepares CDP run names for configuration options data.

DPRINT

Writes DQG geometry network identifications onto plot file FT09.

EDGTAB

Computes table for additional control points and edge data for PDP data.

GENOUT

Prepares the forces and moments data to be stored on the CDP configuration plot file FT11.

GEOPLT

Prepares DQG geometry plot file of corner point data for all selected networks.

HEADPR

Print out page header information including page count for DQG, PDP and CDP data.

HEADR

Writes DQG geometry plot header information onto the plot file FT09.

LAYOUT

Stores PDP global data with appropriate headers for the first time. It also stores the surface options data for all subsequent calls.

LINDX

Locates the starting and ending indices for row or column loops for PDP point type center, edge, additional and grid (i.e. point types 1, 2, 3, and 4, respectively).

LOADVL

Loads computed PDP flow quantities data into buffer arrays to be stored on the plot file FT10.

MAPCDP

Defines SDMS maps for CDP data base data sets CASE-OPTIONS, NETWK-SPEC, PANEL-FORCE-MOMENT, LEAD-EDGE-COEFF, NETWORK-FORCE-MOMENT, CONFIG-FORCES, NET-FOR-MOM-EDGE, and DATA-BASE-HEADER.

MAPDIP

Defines SDMS maps on DIP data base for data sets GLOBAL-FLOW-PROP, SURF-FAM, GLOBAL and CONFIG-PRINT-PLOT for CDP data.

MAPPDP

Defines SDMS maps on PDP database for data sets GLOBAL, NETWK-SPEC, SURF-OPTIONS, FLOW-QUANT, and DATA-BASE-HEADER for PDP data.

ONSETF

Computes the magnitude of uniform onset flow velocity, alpha and beta values for a solution for CDP data.

PANLSM

Prepares panel sums for forces and moments plot data to be stored on CDP plot file FT11.

PCOLSM

Prepares column sum for forces and moments plot data to be stored on CDP plot file FT11.

PLTDAT

Prepares panel forces and moments data to be stored on the CDP configuration plot file FT11.

PLTFIL

Stores surface flow quantities data onto PDP plot file FT10. The surface flow data is loaded into the arrays by calling LOADVL.

PLTHDC

Writes CDP header information of problem, run and user identifications onto plot file FT11.

PLTHDR

Stores PDP plot headers on plot file FT10.

PLTHED

Writes headers for CDP data on the configuration plot file FT11.

PLTOPT

Prepares CDP case options data as the first page of each case.

PLTRPT

Prepares CDP plot file data for forces and moments for panels and the total for columns, networks, and configuration.

PLTSUB

Writes header for pressure and velocity selections onto PDP plot file FT10.

PNETSM

Prepares CDP panel network sums for forces and moments plot data for the axes and pressure rules selected.

PNTGLOB

Prints global data for CDP data.

PNTPLT

Reads data from PDP data base and stores it onto plot file FT10.

PPPROC

The network, solution and case data are written out on the print and/or plot file.

PRUNAM

Produces point data run names to be put on the PDP plot file FT10 to identify point data type.

PTYP14

Processes PDP point type 1 for panel center control point and point type 4 for grid points data.

PTYP23

Processes PDP point type 2 for network edges and point type 3 for additional control points data.

RCASFM

Reads and analyzes CDP user options for current case data. Also, the solution list is prepared.

STUFF

Processes all selected PDP flow quantities data and saves these in array named FLQDAT.

TABLE 11.1 - Maximum and Typical Counts on Problem Options

PLOT DATA	ITEM/OPTION	MAXIMUM NUMBER (PROBLEM SIZES)	TYPICAL NUMBER (PROBLEM SIZES)
DQG	Data Base(s)	1	1
	Networks	100	30
	Panels	2000	400
PDP	Data Base(s)	1	1
	Cases	100	20
	Solutions	200	20
	Network Selections	400	50
	Array Types	4	1
	Parameter List	39	20
	Surface Selections	5	2
	Sel. Vel. Comp.	2	1
	Comp. Option Press	1	1
	Velocity Correction	3	2
	CP Options	5	2
CDP	Data Base(s)	1	1
	Cases	100	20
	Solutions	200	20
	Surface Configuration	400	40
	Axis Systems	4	2
	Data Base Options	9000	2200
	o Panels	2000	1000
	o Columns, etc.	250	100
	Surface Selection	1	1
	Sel. Vel. Comp	2	1
	Comp. Option Press	1	1
	Velocity Correction	3	2
	CP Options	5	2
	Parameter Lists	124	30

TABLE 11.2 - PDP Parameter Name List

DIP Index*	Headings	Quantity
2	X	Point, x-coordinate
2	Y	Point, y-coordinate
2	Z	Point, z-coordinate
3	PWX	Perturbation mass flux, x-component
3	PWY	Perturbation mass flux, y-component
3	PWZ	Perturbation mass flux, z-component
4	WX	Total mass flux, x-component
4	WY	Total mass flux, y-component
4	WZ	Total mass flux, z-component
5	WMAG	Total mass flux, magnitude
6	WN	Total mass flux, normal component
7	PVX	Perturbation velocity, x-component
7	PVY	Perturbation velocity, y-component
7	PVZ	Perturbation velocity, z-component
8	VX	Total velocity, x-component
8	VY	Total velocity, y-component
8	VZ	Total velocity, z-component
9	VMAG	Total velocity, magnitude
10	PHI	Perturbation potential
11	PHIT	Total potential
12	MLISEN	Local Mach number, isentropic
12	MLLINE	Local Mach number, linear
12	MLSECO	Local Mach number, second-order
12	MLREDU	Local Mach number, reduced second-order
12	MLSLEN	Local Mach number, slender body
13	CPISEN	Pressure coefficient, isentropic
13	CPLINE	Pressure coefficient, linear
13	CPSECO	Pressure coefficient, second-order
13	CPREDU	Pressure coefficient, reduced second-order
13	CPSLEN	Pressure coefficient, slender body
14	GMUX	Doublet strength gradient, x-component
14	GMUY	Doublet strength gradient, y-component
14	GMUZ	Doublet strength gradient, z-component
15	PSI	Angle, between average velocity and surface vorticity vectors (degrees)
16	SINGS	Singularity strength, source
16	SINGD	Singularity strength, doublet
17	SPDMAX	Maximum total speed
18	SPDCRT	Critical speed
19	CPVAC	Pressure coefficient, vacuum

* See a description of user input record SF10 in Section 7 of the PAN AIR USER'S MANUAL (Ref. 2).

TABLE 11.3 - CDP Parameter Name List

Character Positions	CDP Headings	Quantity
1 and 2 (Forces and Moments)	FX	Force component in X direction
	FY	" " " Y "
	FZ	" " " Z "
	MX	Moment component in X direction
	MY	" " " Y "
	MZ	" " " Z "
3-5 (CP Rules)	ISE	Isentropic
	LIN	Linear
	SEC	Second-order
	RED	Reduced second-order
	SLE	Slender body
6-7 (Axis Systems)	RC	Reference coordinate system
	WA	Wing axis system
	BA	Body axis system
	SA	Stability axis system

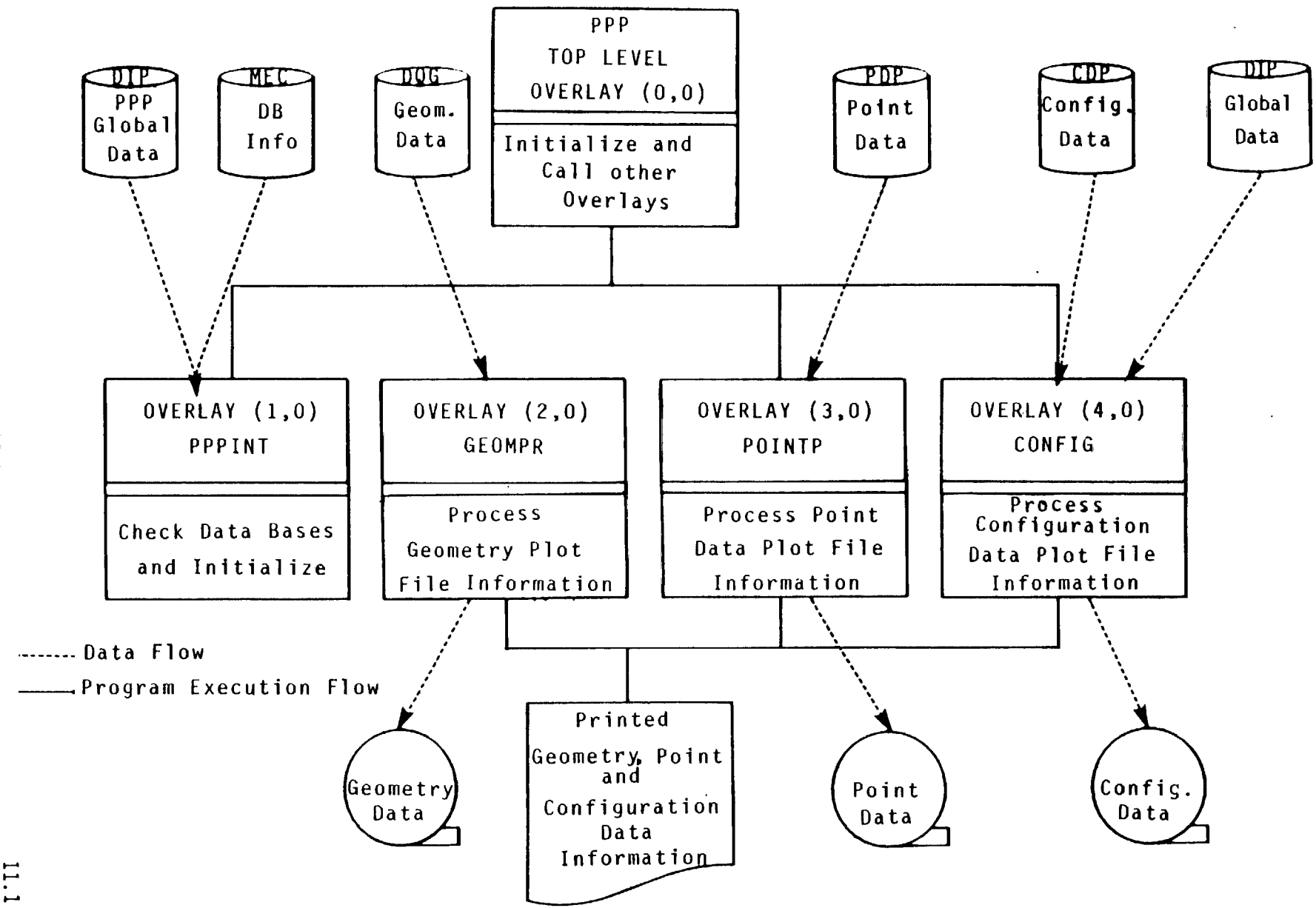


Figure 11.1 - Top Level Structure of PPP

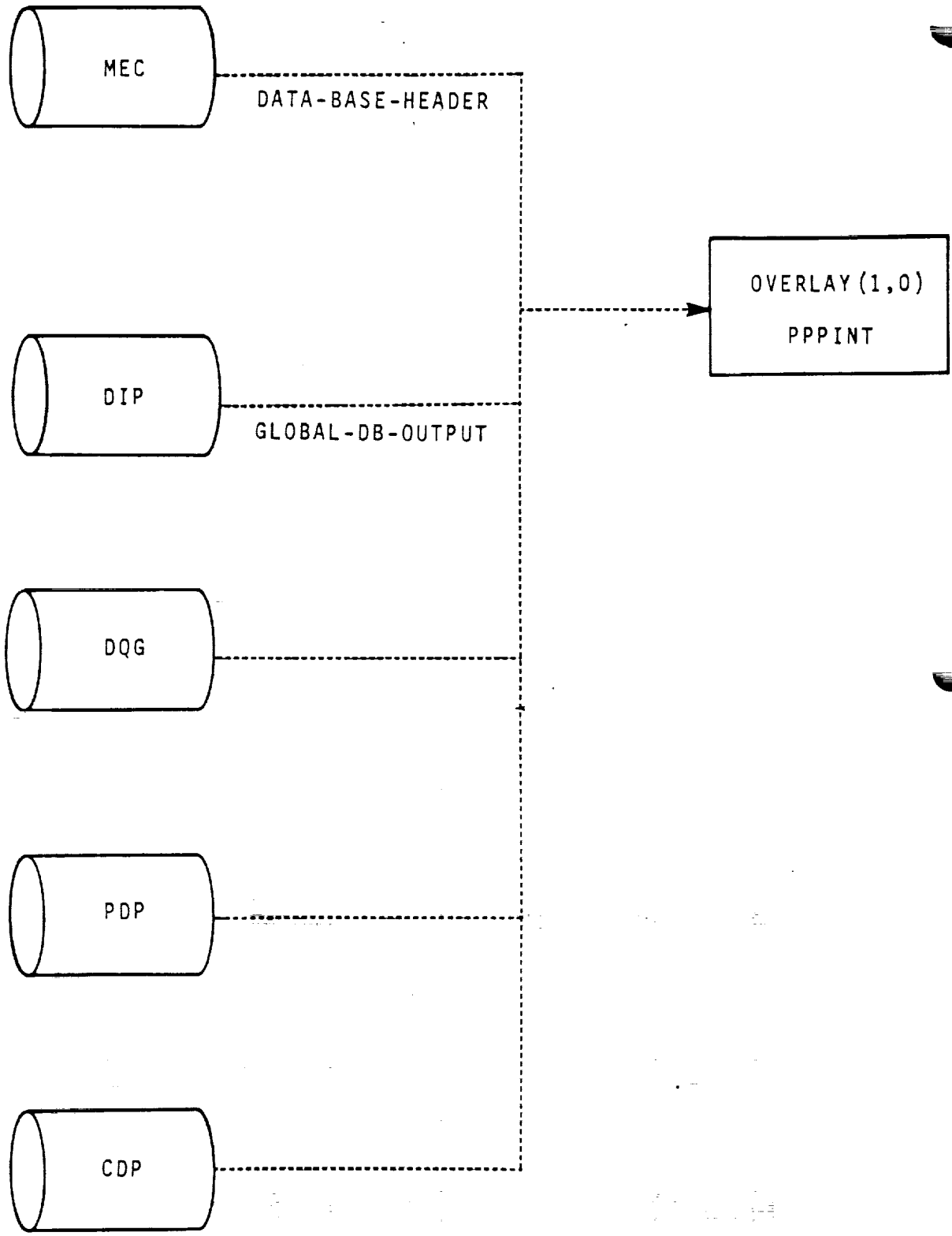


Figure 11.2 - Structure and Data Flow of Overlay (1,0)

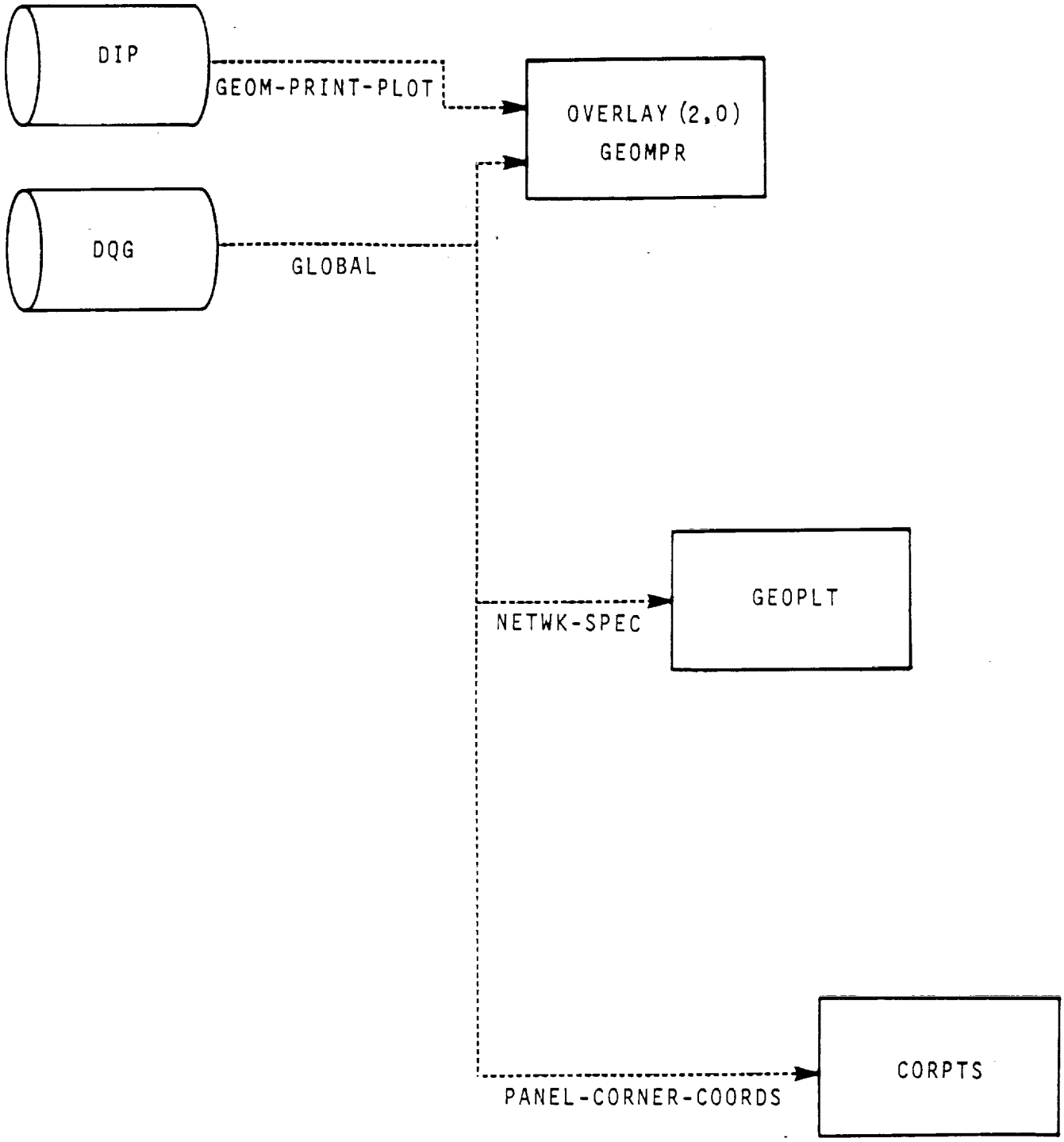


Figure 11.3 - Structure and Data Flow of Overlay (2,0)

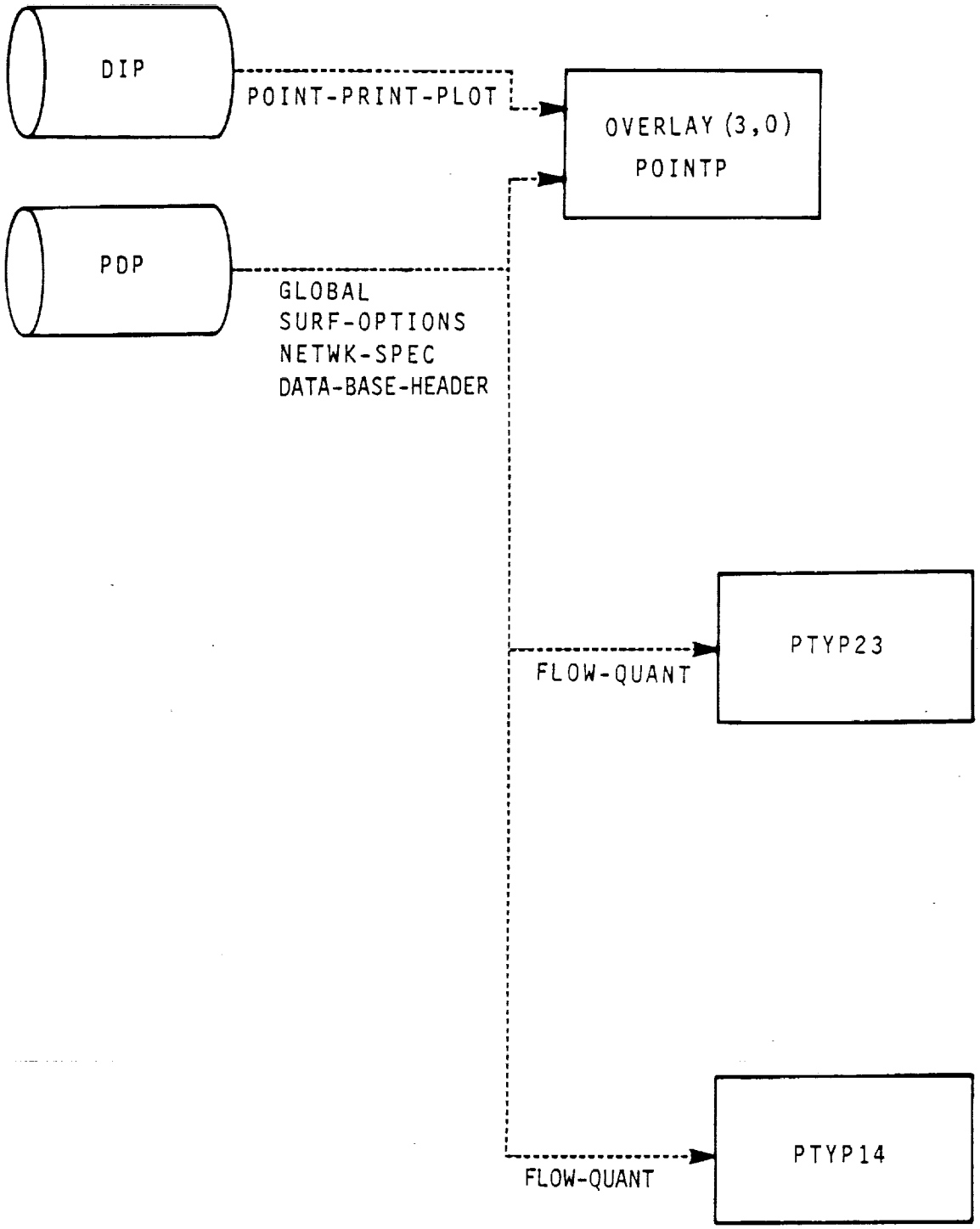


Figure 11.4 - Structure and Data Flow of Overlay (3,0)

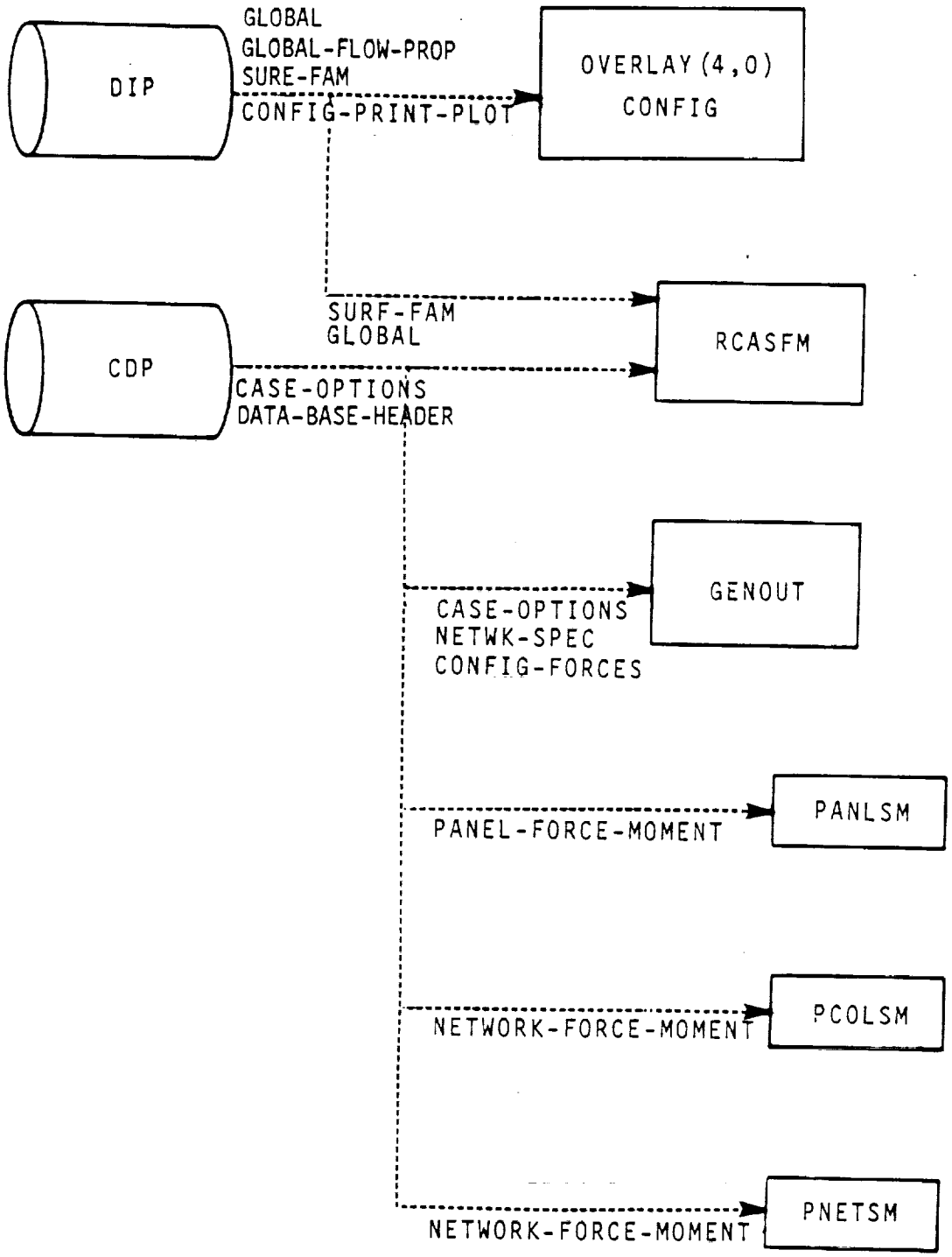
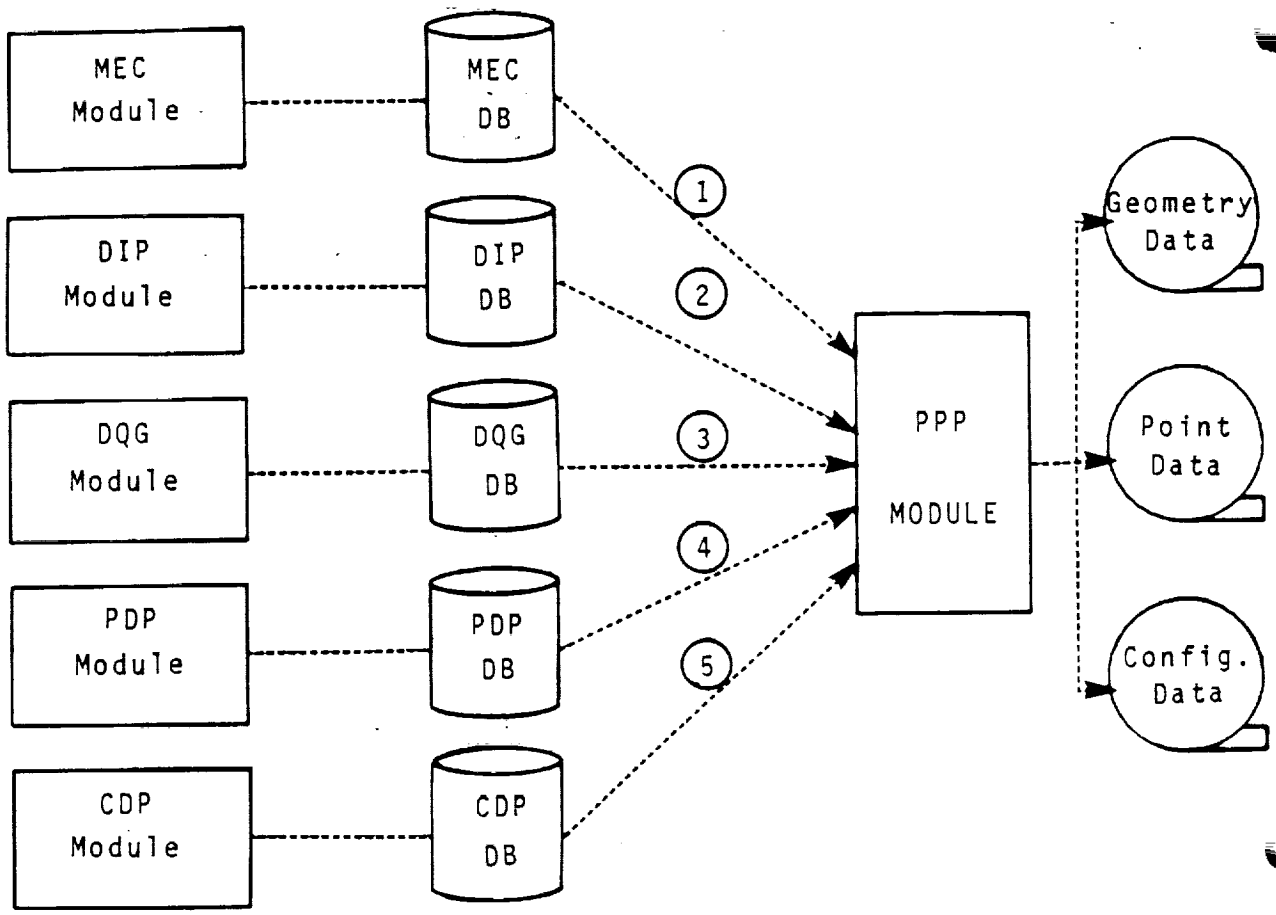


Figure 11.5 - Structure and Data Flow of Overlay (4,0)



(1) Data Base Directory Information

(2) PPP Selection Information

(3) Geometry Data

(4) Point Data

(5) Configuration Data

Figure 11.6 - External Data Inferfaces

APPENDIX 11-A TREE STRUCTURE

The tree structure of the PPP moudle has been deleted from this document. It is, however, available on the installation tape.

APPENDIX 11-B PPP FUNCTIONAL DECOMPOSITION

The functional decomposition of the PPP module is presented here. The decomposition labels are given in the order of their execution and therefore may not be alphabetic.

A Initialization and control of PPP Execution [Overlay 0,0] (PPP).
  A Initialize printout (PRGBEG).
  B Initialization of SDMS for Execution (ISDMS).
  C Check Data Base(s) and initialize [Overlay 1,0] (PPPINT).
    A Initialization.
    B Open MEC Data base (SDSLIB/DBOPEN).
    C Get data base header information (SDSLIB/ESGET).
    D Check data base location (PALIB/CHPADB).
      A Check DIP.
      B Check DQG.
      C Check PDP.
      D Check CDP.
    E Open DIP data bases (SDSLIB/PAOPEN).
    F Get global PPP information from DIP (SDSLIB/ESGET).
    G Close MEC data base (SDSLIB/DBCLOS).

B Process geometry data [Overlay 2,0] (GEOMPR).
  A Open DIP data base (SDSLIB/PAOPEN).
    A Open DQG data base (SDSLIB/PAOPEN).
    B Define mappings for DQG data base.
  B Process and prepare geometry plot file.
    A Get geometry specifications from DIP data base (SDSLIB/ESGET).
    B Setup MAP for Global data set for DQG data base (SDSLIB/SVMAP).
    C Setup MAP for Network specs for DQG data base (SDSLIB/SVMAP/DSMAP).
    D Get Global geometry data from DQG-DB (SDSLIB/ESGET).
    E Calculate CALPHA, CBETA and set number of networks.
  C Process geometry plot file (GEOPLT).
    A Write header titles, run ID, problem ID, user ID and DQG global data on plot file (HEADR).
    B Determine network selections.
      A Set networks to all active DQG selections (Default).
      B Set networks to PPP selections.
    C Get network specs from DIP (SDSLIB/ESGET).
    D Print run name (DPRINT).
    E Process geometry corner points from DQG data base (CORPTS).
      A Get panel corner point data from DQG data base (SDSLIB/ESGET).
      B Prepare and write a column of corner points on plot file.
    F Write ending identification on plot file.
  D Close DQG and DIP data bases (SDSLIB/PACLOS).

C Process point data [Overlay 3,0] (POINTP).
  A Open DIP data base (SDSLIB/PAOPEN).
    A Setup map for POINT-PRINT-PLOT (SDSLIB/DSMAP).
    B Define static map (SDSLIB/SVMAP).
  B Get point data specs from DIP for PPP (SDSLIB/ESGET).
  C Open PDP data base (SDSLIB/PAOPEN).
  D Form maps for PDP data (MAPPDP).
    A Get global PDP data (SDSLIB/ESGET).
  E Process and prepare point data plot file.
    A Get surface flow data (SDSLIB/ESGET).
    B Prepare layout of outputs (LAYOUT).
      A Print global data.

B Print surface flow options data.
A Print header information (HEADPR).
B Print option selection information.
C Print option index values for flow quantities.
C Get network data (SDSLIB/ESGET).
D Prepare point data plot file (PNTPLT).
A Initiate program execution.
B Process point types for network edge and additional control point.
A Process edge data table (EDGTAB).
A Initialize row/column indices.
B Prepare panel center control point table.
C Prepare additional control point table.
B Process network edge and additional control points data (PTYP23).
A Initialize loop control parameters.
B Get flow quantities data (SDSLIB/ESGET).
C Process point type.
A Unpack flow quantities (STUFF).
A Initialization of unpacked flow quantities data.
B Unpack flow data.
C Store flow data.
B Load flow quantities data (LOADVL).
A Initialization.
B Fill flow data selections into buffers.
C Fill subheader flow data.
D Load subheader flow data into buffers.
E Load flow data into buffers.
D Process plot file data (PLTFIL).
A Prepare new page of report (PLTSUB).
B Write new page headings (HEADPR).
C Prepare run name (PRUNAM).
D Print/store flow quantities data.
E Save current point indices.
C Process center control points and grid points (PTYP14).
A Initialize row/column indices (LINDX).
B Load flow data from PDP data base (SDSLIB/ESGET).
C Set surface option.
D Set velocity correction option.
E Unpack flow data (STUFF).
F Load flow data (LOADVL).
G Store flow data (PLTFIL).
F Close PDP plot file (SDSLIB/PACLOS).

D Process configurations data [Overlay 4,0] (CONFIG).
A Open data bases and define SDMS maps.
A Open DIP data base (SDSLIB/PAOPEN).
B Define DIP maps (MAPDIP).
A Initialize.
B Select maps for DIP data base.

C Map SURF-FAM.
D Map SURF-FAM for CASEMAP.
E Map GLOBAL.
F Map CONFIG-PRINT-PLOT PPP selection.

B Process and prepare plot file.
A Get DIP-CFG-PL specs (SDSLIB/ESGET).
B Determine selection options for CDP.
C Open CDP data base (SDSLIB/PAOPEN).
D Define SDMS maps for CDP (MAPCDP).
E Define SDMS maps for DIP (MAPDIP).

C Get forces and moments data.
A Read forces and moments case-options data (RCASFM).
A Initialize.
B Get case-options and surface data from CDP and DIP data base.
C Generate computation options vectors.
D Print output case requirements.
E Prepare solutions list array.

D Get forces and moments data from CDP data base and store output onto plot file (GENOUT).
A Print CDP Global and case-options data.
B Print Global data (PNTGLOB).
A Print header lines.
A Print header information for top of new page (PLTHED).
B Store header information for plot file (PLTHDC).
B Print networks information.
C Print global data.
C Print CDP case-options data (PLTOPT).
D Get CDP case-options and network specs data (SDSLIB/ESGET).
E Setup CDP parameter name list (CDPARM).
F Get CDP network specs data (SDSLIB/ESGET).
G Get panel forces and moments data and prepare configuration plot data (PLTDAT).
A Initialization.
B Prepare panel forces and moments data (PANLSM).
A Setup panel run name (CRUNAM).
B Prepare alpha, beta and magnitude of uniform onset flow velocity data (ONSETF).
C Get panel forces and moments data (SDSLIB/ESGET).
D Process panel forces and moments plot data (PLTRPT).
C Prepare column sum forces and moments data (PCOLSM)
A Setup column sum run name (CRUNAM).
B Prepare alpha, beta and magnitude of uniform onset flow velocity data (ONSETF).
C Output column sum of forces and moments data.
A Get column sum data (SDSLIB/ESGET).
B Process column sum forces and moments plot data (PLTRPT).
D Prepare network sum forces and moments data (PNETSM).
A Setup network sum run name (CRUNAM).
B Prepare alpha, beta and magnitude of uniform onset flow velocity data (ONSETF).
C Get network sum data (SDSLIB/ESGET).
D Process network sum forces and moments plot data (PLTRPT).
H Prepare configuration sum plot data.

A Setup CDP option run name.
B Prepare alpha, beta and magnitude of uniform onset flow velocity solution data (ONSETF).
C Get configuration data (SDSLIB/ESGET).
D Process configuration plot data (PLTRPT).

I Process accumulations option data.
A Setup CDP parameter name list (CDPARM).
B Setup CDP run name (CRUNAM).
C Prepare alpha, beta and magnitude of uniform onset flow velocity solution data (ONSETF).
D Get forces and moments data (SDSLIB/ESGET).
E Process configuration plot data (PLTRPT).

E Close DIP and CDP data bases (SDSLIB/PACLOS).

E Terminate Program Execution (PRGEND).

APPENDIX 11-C DATABASE COMMUNICATIONS CHARTS

The Data Base Communications Chart is presented in three forms. Each form is alphabetized by columns, from left to right. The first form has a column order of Data Base, Dataset Name, Map Name, Common Block, and Program/Subroutine. The second form has a column order of Data Base, Map Name, Dataset Name, Common Block, and Program/Subroutine. The third form has a column order of Common Block, Data Base, Map Name, Dataset Name, and Program/Subroutine. Thus a person can get a cross reference on a data element by knowing either the Dataset Name, Map Name or Common Block name.

FIRST FORM

DATA BASE	DATA SET NAME	MAP NAME	COMMON BLOCK	SUBROUTINE
		Overlay (1,0)		
MEC	DATA-BASE-HEADER	MECHDR	/RUNIDS/	PPPINT
DIP	GLOBAL-DB-OUTPUT	DIP-GLOBPP	/GLOPPP/	PPPINT
		Overlay (2,0)		
DIP	GEOM-PRINT-PLOT	DIPGEOM	/PPPDAT/	GEOMPR
DQG	DATA-BASE-HEADER	DQGHDR	/RUNIDS/	GEOMPR
DQG	GLOBAL	GLOBAL	/NETWX/	GEOMPR
DQG	NETWK-SPEC	NETMAP	/NETWX/ Dynamic	GEOPLT
DQG	PANEL-CORNER-COORDS	PANCORD	Dynamic	CORPTS
		Overlay (3,0)		
DIP	POINT-PRINT-PLOT	DIPPTPLT	/PPPDAT/	POINTP
PDP	DATA-BASE-HEADER	PDPHDR	/RUNIDS/	POINTP
PDP	FLOW-QUANT	FLQNTMAP	/PDOPT/ /FLQNT/ Dynamic	PTYP23
PDP	FLOW-QUANT	FLQNTMAP	/PDOPT/ /FLQNT/ Dynamic	PTYP14
PDP	GLOBAL	GLOBMAP	/PDGLOB/	POINTP
PDP	SURF-OPTIONS	OPTNMAP	/PDOPT/	POINTP
PDP	NETWK-SPEC	NETMAP	/NETSPC/	POINTP
		Overlay (4,0)		
DIP	CONFIG-PRINT-PLOT	DIP-CFG-	/PPPDAT/	CONFIG
DIP	GLOBAL-FLOW-PROP	GLOBALPR	/CASES/	CONFIG
DIP	SURF-FAM	SURFAM	/CASES/ /SOLS/ /ACCUM/ /NETWK/	CONFIG
DIP	GLOBAL	GLOBDIP	/CASES/	CONFIG
DIP	GLOBAL	GLOBDIP	/CASES/	RCASFM
DIP	SURF-FAM	CASEMAP	/CASES/ /ACCUM/	RCASFM
CDP	DATA-BASE-HEADER	HDRMAP	/RUNIDS/	CONFIG
CDP	CASE-OPTIONS	OPTNMAP	/CASES/	RCASFM
CDP	CASE-OPTIONS	OPTNMAP	/SOLS/ /NETWK/ /ACCUM/	GENOUT
CDP	NETWK-SPEC	NETSPCMP	/CASES/ /NETWK/	GENOUT
CDP	CONFIG-FORCES	CONFIMAP	/CASES/ /ACCUM/ Dynamic	GENOUT

DATA BASE	DATA SET NAME	MAP NAME	COMMON BLOCK	SUBROUTINE
CDP	PANEL-FORCE-MOMENT	PANELMAP	/CASES/ /ACCUM/ /NETWK/ /LEGEOM/ /PANDAT/	PANLSM
CDP	NETWORK-FORCE-MOMENT	NETWKMAP	/CASES/	PCOLSM
CDP	NETWORK-FORCE-MOMENT	NETWKMAP	/ACCUM/ /FMACCU/	PNETSM

SECOND FORM

DATA BASE	MAP NAME	DATA SET NAME	COMMON BLOCK	SUBROUTINE
		Overlay (1,0)		
MEC	MECHDR	DATA-BASE-HEADER	/RUNIDS/	PPPINT
DIP	DIP-GLOBPP	GLOBAL-DB-OUTPUT	/GLOPPP/	PPPINT
		Overlay (2,0)		
DIP	DIPGEOM	GEOM-PRINT-PLOT	/PPPDAT/	GEOMPR
DQG	DQGHDR	DATA-BASE-HEADER	/RUNIDS/	GEOMPR
DQG	GLOBAL	GLOBAL	/NETWX/	GEOMPR
DQG	NETMAP	NETWK-SPEC	/NETWX/ Dynamic	GEOPLT
DQG	PANCORD	PANEL-CORNER-COORDS	Dynamic	CORPTS
		Overlay (3,0)		
DIP	DIPPTPLT	POINT-PRINT-PLOT	/PPPDAT/	POINTP
PDP	PDPHDR	DATA-BASE-HEADER	/RUNIDS/	POINTP
PDP	FLQNTMAP	FLOW-QUANT	/PDOPT/ /FLQNT/ Dynamic	PTYP23
PDP	FLQNTMAP	FLOW-QUANT	/PDOPT/ /FLQNT/ Dynamic	PTYP14
PDP	GLOBMAP	GLOBAL	/PDGLOB/	POINTP
PDP	OPTNMAP	SURF-OPTIONS	/PDOPT/	POINTP
PDP	NETMAP	NETWK-SPEC	/NETSPC/	POINTP
		Overlay (4,0)		
DIP	DIP-CFG-	CONFIG-PRINT-PLOT	/PPPDAT/	CONFIG
DIP	GLOBALPR	GLOBAL-FLOW-PROP	/CASES/	CONFIG
DIP	SURFAM	SURF-FAM	/CASES/ /SOLS/ /ACCUM/ /NETWK/	CONFIG
DIP	GLOBDIP	GLOBAL	/CASES/	CONFIG
DIP	GLOBDIP	GLOBAL	/CASES/	RCASFM
DIP	CASEMAP	SURF-FAM	/CASES/ /ACCUM/	RCASFM
CDP	HDRMAP	DATA-BASE-HEADER	/RUNIDS/	CONFIG
CDP	OPTNMAP	CASE-OPTIONS	/CASES/	RCASFM
CDP	OPTNMAP	CASE-OPTIONS	/SOLS/ /NETWK/ /ACCUM/	GENOUT
CDP	NETSPCMP	NETWK-SPEC	/CASES/ /NETWK/	GENOUT
CDP	CONFIMAP	CONFIG-FORCES	/CASES/ /ACCUM/ Dynamic	GENOUT

DATA BASE	MAP NAME	DATA SET NAME	COMMON BLOCK	SUBROUTINE
CDP	PANELMAP	PANEL-FORCE-MOMENT	/CASES/ /ACCUM/ /NETWK/ /LEGEOM/ /PANDAT/	PANLSM
CDP	NETWKMAP	NETWORK-FORCE-MOMENT	/CASES/	PCOLSM
CDP	NETWKMAP	NETWORK-FORCE-MOMENT	/ACCUM/ /FMACCU/	PNETSM

THIRD FORM

COMMON BLOCK	DATA BASE	MAP NAME	DATA SET NAME	SUBROUTINE
/RUNIDS/	MEC	MECHDR	DATA-BASE-HEADER	PPPINT
/GLOPPP/	DIP	DIP-GLOBPP	GLOBAL-DB-OUTPUT	PPPINT
		Overlay (2,0)		
/PPPDAT/	DIP	DIPGEOM	GEOM-PRINT-PLOT	GEOMPR
/RUNIDS/	DQG	DQGHDR	DATA-BASE-HEADER	GEOMPR
/NETWX/	DQG	GLOBAL	GLOBAL	GEOMPR
/NETWX/	DQG	NETMAP	NETWK-SPEC	GEOPLT
Dynamic				
Dynamic	DQG	PANCORD	PANEL-CORNER-COORDS	CORPTS
		Overlay (3,0)		
/PPPDAT/	DIP	DIPPTPLT	POINT-PRINT-PLOT	POINTP
/RUNIDS/	PDP	PDPHDR	DATA-BASE-HEADER	POINTP
/PDOPT/	PDP	FLQNTMAP	FLOW-QUANT	PTYP23
/FLQNT/				
Dynamic				
/PDOPT/	PDP	FLQNTMAP	FLOW-QUANT	PTYP14
/FLQNT/				
Dynamic				
/PDGLOB/	PDP	GLOBMAP	GLOBAL	POINTP
/PDOPT/	PDP	OPTNMAP	SURF-OPTIONS	POINTP
/NETSPC/	PDP	NETMAP	NETWK-SPEC	POINTP
		Overlay (4,0)		
/PPPDAT/	DIP	DIP-CFG-	CONFIG-PRINT-PLOT	CONFIG
/CASES/	DIP	GLOBALPR	GLOBAL-FLOW-PROP	CONFIG
/CASES/	DIP	SURFAM	SURF-FAM	CONFIG
/SOLS/				
/ACCUM/				
/NETWK/				
/CASES/	DIP	GLOBDIP	GLOBAL	CONFIG
/CASES/	DIP	GLOBDIP	GLOBAL	RCASFM
/CASES/	DIP	CASEMAP	SURF-FAM	RCASFM
/ACCUM/				
/RUNIDS/	CDP	HDRMAP	DATA-BASE-HEADER	CONFIG
/CASES/	CDP	OPTNMAP	CASE-OPTIONS	RCASFM
/SOLS/	CDP	OPTNMAP	CASE-OPTIONS	GENOUT
/NETWK/				
/ACCUM/				
/CASES/	CDP	NETSPCMP	NETWK-SPEC	GENOUT
/NETWK/				
/CASES/	CDP	CONFIMAP	CONFIG-FORCES	GENOUT
/ACCUM/				
Dynamic				

COMMON BLOCK	DATA BASE	MAP NAME	DATA SET NAME	SUBROUTINE
/CASES/	CDP	PANELMAP	PANEL-FORCE-MOMENT	PANLSM
/ACCUM/				
/NETWK/				
/LEGEOM/				
/PANDAT/				
/CASES/	CDP	NETWKMAP	NETWORK-FORCE-MOMENT	PCOLSM
/ACCUM/	CDP	NETWKMAP	NETWORK-FORCE-MOMENT	PNETSM
/FMACCU/				

APPENDIX 11-D PPP ERROR MESSAGES

Module/ Submodule Name	Error Message
PPP (0,0)	********ERROR IN MODULE __ERR____ where, ERR = PPPINT or GEOMPR or POINTP or CONFIG
PPPINT	********NERR = _____ IS COUNT OF FATAL ERRORS FROM PPPINT
	********ERROR IN FETCHING DATASET GLOBAL-DB-OUTPUT FROM DIP-DB***FATAL ERROR*****EXIT SDMS ERROR CODE = _____
	DATA BASE NAME IS _____ IRO = _____ *****ERROR IN CALLING CHPADB-WARNING ONLY*****
GEOMPR (2,0)	*****ERROR IN FETCHING DATASET __ERR__ FROM DQG-DB*****FATAL ERROR-SKIP GEOMETRY DATA SDMS ERROR CODE = _____
	where, ERR = DIP-GEOMPP or = GLOBAL-IN
GEOPLT	*****ERROR IN FETCHING NETWORK SPECS DATASET FROM DQG-DB*****WARNING MESSAGE - CONTINUE SDMS ERROR CODE = ____ MAP NAME = NET MAP KEYSET = _____

Module	Error Message
POINTP	*****PPP SELECTION FOR GRID DIRECTION OF ROWS WERE MADE --- WARNING ONLY***** COLUMN SELECTION IS ONLY AVAILABLE IN PAN AIR VERSION I(GRID DIRECTION FLAG WILL BE SET TO COLUMNS)
	*****PPP SELECTION FOR POINT TYPE OF GRIDS WERE MADE --- WARNING ONLY ***** CENTER CONTROL POINT TYPE IS ONLY AVAILABLE IN PAN AIR VERSION I(POINT TYPE FLAG WILL BE SET TO CENTER)
	*****ERROR IN FETCHING DATASET POINT-PRINT-PLOT FROM DIP-DB*****FATAL ERROR - EXIT SDMS ERROR CODE = _____ LNOSOL = _____ LNOCAS = _____ LNONET = _____ IARDIR = _____ IARTYP = _____
	*****ERROR IN FETCHING DATASET GLOBAL FROM PDP-DB*****FATAL ERROR - EXIT SDMS ERROR CODE = _____ NCASE = _____ NNET = _____ NSOL = _____ NPOS = _____
	*****ERROR IN FETCHING DATASET SURF-OPTIONS FROM PDP-DB*****WARNING MESSAGE - CONTINUE NEXT CASE SDMS ERROR CODE = _____ ICASE = _____ NNETR = _____ NSOLR = _____ NLOCR = _____ NSURFR = _____ IPROPT = _____
	*****ERROR IN FETCHING DATASET NETWK-SPEC FROM PDP-DB*****WARNING MESSAGE - CONTINUE NEXT NETWORK SDMS ERROR CODE = _____ ICASE = _____ ISOL = _____ INET = _____ NROW = _____ NCOL = _____
	IMAGE SELECTED IS NOT ON PDP-DB FOR THE PPP IMAGE LIST** IMAGE = _____

Module	Error Message
PNTPLT	POINT TYPE OF EDGE (ILOC = 2) OR ADDITIONAL (ILOC = 3) WAS SPECIFIED. THESE OPTIONS ARE NOT AVAILABLE IN VERSION I-----SKIP THIS OPTION AND CONTINUE EXECUTION ILOC = _____ ICASE = _____ ISOL = _____ INET = _____ IMAGE = _____
	ERROR IN FETCHING DATA FROM PDP-DB SDMS ERROR - ERROR CODE = _____
PTYP23	*****ERROR IN FETCHING FLOW QUANTITIES DATA FROM PDP-DB - WARNING MESSAGE ONLY***** (ERROR IN EDGE AND ADDIT CALC) SDMS ERROR CODE = _____
PTYP14	*****ERROR IN FETCHING FLOW QUANTITIES DATA FROM PDP-DB - WARNING MESSAGE ONLY***** (ERROR IN CENTER AND GRID) SDMS ERROR CODE = _____ ICASE = _____ INET = _____ ISOL = _____ IMAGE = _____ ILOC = _____ IX = _____ JX = _____
CONFIG	***SDMS ERROR IN GETTING DATASET CONFIG-PRINT-PLOT*** SDMS ERROR CODE = _____
	*****ERROR IN FETCHING DATASET GLOBAL FROM DIP-DB*****FATAL ERROR - EXIT SDMS ERROR CODE = _____
RCASFM	***SDMS ERROR IN READING CDP DATA*** ERROR CODE = _____
	SDMS ERROR IN READING DIP DATA ERROR CODE = _____

Module	Error Message
GENOUT	*****ERROR IN FETCHING DATASET CONFIG-FORCES FROM CDP-DB*****FATAL ERROR - SKIP CONFIGURATION SUMS SDMS ERROR CODE = _____
	*****ERROR IN FETCHING DATASET CONFIG-FORCES FROM CDP-DB*****FATAL ERROR - SKIP ACCUMULATION SUMS SDMS ERROR CODE = _____
	*****ERROR IN FETCHING DATASET CASE-OPTIONS FROM CDP-DB*****FATAL ERROR - EXIT SDMS ERROR CODE = _____
	*****ERROR IN FETCHING DATASET NETWK-SPEC FROM CDP-DB*****FATAL ERROR - EXIT SDMS ERROR CODE = _____
PANELSM	*****ERROR IN FETCHING DATASET PANEL-FORCE-MOMENT FROM CDP-DB*****FATAL ERROR - SKIP PANEL SUMS SDMS ERROR CODE = _____ ERROR OCCURED WHEN SDMS KEYS WERE SET TO: CASE NUMBER (ICASE) = _____ SOLUTION NUMBER (ISOLA) = _____ NETWORK NUMBER (INETA) = _____ IMAGE INDEX (IMAGEA) = _____ VEL COMPUTATION (IVECPA) = _____ VEL CORRECTION (IVECRA) = _____ PANEL COL NUMBER (IPACOL) = _____ PANEL ROW NUMBER (IPAROW) = _____ AXIS NAME (NAMAXS) = _____
PCOLSM	*****ERROR IN FETCHING DATASET NETWORK-FORCE-MOMENT FROM CDP-DB*****FATAL ERROR - SKIP COLUMN SUMS SDMS ERROR CODE = _____
PNETSM	*****ERROR IN FETCHING DATASET NETWORK-FORCE-MOMENT FOR NETWORK SUMS FROM CDP-DB*****WARNING MESSAGE - CONTINUE EXECUTION SDMS ERROR CODE = _____

APPENDIX 11-E GEOMETRY PLOT FILE

11-E.1 Plot File Format for Geometry Data

The network panel corner points data along with its identification information is written onto a plot file (logical unit 9), as given below:

Record Set(s)	Item	Columns	Description
1	DQG Plot Titles		DQG Plot Title Information consisting of 4 lines of title information as follows:
	a) NETWORK GEOMETRY FROM DQG DATA BASE	1-35	DQG Title (Format 3A10,A5)
	b) Run ID	1-72	DQG Run Identification (RID). (Format 7A10,A2)
	c) Problem ID	1-72	DQG Problem Identification (PID). (Format 7A10,A2)
	d) User ID	1-72	DQG User Identification (UID). (Format 7A10,A2)
2	*START	1-5	Signifies start of data (Format A5)
3	$GLOBAL DATA	1-12	Global Data (see paragraph 11-E.2 for details)
4	(DQG Run Id)	1-28	DQG Run Name Identification (Format A1, I3, 2I2, 2A10) (see Paragraph 11-E.3).
5	(Geometry Data from DQG)		Network Geometry corner points data X, Y and Z along with its identification data [Format I4, 6X, 3(I4, 1X), 3(F12.6, 1X)]
		1-4	Sequence Number
		5-10	Blanks
		11-14	Row Number
		15	Blank
		16-19	Column Number
		20	Blank
		21-24	Network Number
		25	Blank
		26-37	X-coordinate
		38	Blank
		39-50	Y-coordinate
		51	Blank
		52-63	Z-coordinate
		64	Blank
			(Repeat record sets 4 and 5 above for each Network selected.)
6	*END	1-4	The last line of data contains *END to signify the end of DQG data (Format A4)

11-E.2 Global Data for Geometry File

A description of the Global Data for DQG is written on the geometry plot file (logical unit 9) following record set 2 (i.e., *START descriptor record signifying the start of data).

Record Set(s)	Record Subset(s)	Item(s)	Columns	Description	Format
3		$GLOBAL DATA	1-12	Global Data	A12
	1	DATE	1-5	The heading DATA	A5
		DATECR	6-15	Date of creation in the form Yr/Mo/Date**	A10
	2	AMACH	1-10	Mach Number	F10.5
		CALPHA	11-20	Angle of attack (degrees)	F10.5
		CBETA	21-30	Angle of sideslip (degrees)	F10.5
		NUMPOS	31-35	Number of planes of symmetry, =0 unsymmetric =1 one plane of sym. =2 two planes of sym.	I5
		NNET	36-40	Number of Networks	I5
	3	POSNRM	1-60	Normal to first and second planes normal to the planes of symmetry (3 by NUMPOS)	6F10.5
	4	POSLOC	1-30	Coordinates of point common to first and second planes	3F10.5
	5	NETPPP,NETID	1-70	Network index and ID, two networks per record subset [network number (I4) and network id (2A10)]	2(I4,1X,2A10,10X)

** For the Ames System, the form is Date/Mo/Yr

11-E.3 DQG Run Name Format

ITEM NUMBERS	COLUMNS	LITERAL NAME/VALUE	FORMAT	DESCRIPTION
1	1	D	A1	DQG Identification
2	2-4		I3	Network Number
3	5-6		I2	Number of Rows
4	7-8		I2	Number of Columns
5	9-28		A20	Network ID

APPENDIX 11-F POINT DATA PLOT FILE

11-F.1 Plot File Format for Point Data .

The format of the point data plot file (on logical unit 10) is described below:

Record Set(s)	Item	Columns	Description
1	(6F10.5)	1-8	Data Format Specification (Format A8)
2	PDP PLOT TITLE(S)		PDP Plot Title Information. Starts in column 1 with a $ and consists of 4 lines of title information as follows:
	a) $POINT DATA FROM PDP DATA BASE	1-30	PDP Plot Title (Format 3A10)
	b) $(RID)	1-72	PDP Run Identification (RID).(Format 7A10, A2)
	c) $(PID)	1-72	PDP Problem Identification (PID).(Format 7A10, A2)
	d) $(UID)	1-72	PDP User Identification (UID).(Format 7A10,A2)
3	*RUN 40	1-7	Identifies maximum run name length of 40 alpha/numeric characters in PDP run name (record set 6).
4	$GLOBAL DATA	1-12	Global Data (see paragraph 11-F.2 for details)
5	(PDP Parameter Name List)	1-76	Identifies parameters available for plotting. If more than one line is needed to specify parameters, the word MORE must be entered in columns 73-76 on that line except for the last line of a parameter list. The parameter name list is written on the plot file at the beginning of each solution. The parameter list is written 6 per line. A detailed description is given in the Table 11.2 (Format 6A10, 12X, A4).
6	(PDP Run Name)	1-40	A detailed description of the PDP run name is described in paragraph 11-F.3 (Format A1, I2, I3, I2, 4A4, A1, I3, A3, A4, A1, I2, 2X).
7	(Point data from PDP in order of Parameter name list)	1-60	PDP Data list in order of parameter name list in the format specified in Record Set 1 above. (Repeat Record Sets 6 and 7 above for all selected data options.)
8	*EOF	1-4	The last line of dataset contains *EOF to signify the end of data for that run (Format A4).

11-F.2 Global Data for PDP file

A description of the Global Data for PDP is written on the point data plot file (logical unit 10) following record set 3 (i.e., *RUN 40 descriptor record identifying maximum run name length of 40).

Record Set(s)	Record Subset(s)	Item(s)	Columns	Description	Format
4		$GLOBAL DATA	1-12	Global Data	A12
	1	DATE	1-5	The heading DATA	A5
		DATECR	6-15	Date of creation in the form Yr/Mo/Date**	A10
	2	AMACH	1-10	Mach Number	F10.5
		CALPHA	11-20	Angle of attack (degrees)	F10.5
		CBETA	21-30	Angle of sideslip (degrees)	F10.5
		NUMPOS	31-35	Number of planes of symmetry, =0 unsymmetric =1 one plane of sym. =2 two planes of sym.	I5
		NNET	36-40	Number of Networks	I5
		NSOL	41-45	Number of solutions	I5
		NCASE	46-50	Number of cases	I5
	3	POSNRM	1-60	Normal to first and second planes normal to the planes of symmetry (3 by NUMPOS)	6F10.5
	4	POSLOC	1-30	Coordinates of point common to first and second planes	3F10.5
	5	NETPPP,NETID	1-70	Network index and ID, two networks per record subset [network number (I4) and network id (2A10)]	2(I4,1X,2A10,10X)
	6	ALPHA	1-70	Angle of attack for each solution (max 200)	7F10.5

** For the Ames System, the form is Date/Mo/Yr

Record Set(s)	Record Subset(s)	Item(s)	Columns	Description	Format
	7	BETA	1-70	Angle of sideslip for each solution (max 200)	7F10.5
	8	SOLLST,SOLID	1-70	Solution index and ID, two solutions per record subset [solution number (I4) and solution id (2A10)]	2(I4,1X,2A10,10X)
	9	CASLST,CASEID	1-70	Case index and ID, two cases per record subset [case number (I4) and case id (2A10)]	2(I4,1X,2A10,10X)

11-F.3 PDP Run Name Format

Item Number	Columns	Literal Name(s) or Value(s)	For Literal name or Associated Integer Format*	Description
1	1	P	A1	PDP Identification
2 (a)	2-3		I2	Case Number
(b)	4-6		I3	Solution Number
3	7-8	99	I2	Job number, preset to 99 (not used)
4	9-12	UPPE - 1 LOWE - 2 UPLO - 3 LOUP - 4 AVER - 5	A4/I4	Surface Selection
5	13-16	BOUN - 1 VIC - 2	A4/I4	Velocity computation option
6	17-20	UNIF - 1 LOCA - 2	A4/I4	Pressure computation option
7	21-24	NONE - 0 SA1 - 1 SA2 - 2	A4/I4	Velocity correction option
8 (a)	25	N	A1	Network ID
(b)	26-28		I3	Network number
9	29-31	INP - 1 IST - 2 2ND - 3 3RD - 4	A3/I3	Images
10	32-35	CENT - 1 EDGE - 2 ADDI - 3 GRID - 4	A4/I4	Point type
11(a)	36	R or C	A1	Row or Column ID
(b)	37-38		I2	Row or Column Number
12*(a)	39	C	A1	Column ID
(b)	40-41		I2	Column Number
13*	42-44		I3	Run Sequence Number

* Note that the PDP plot file has 2 similar names for each dataset option. Item numbers 12 and 13 in the Run Name are used for only the second run name descriptive with associated integer values for item numbers 4, 5, 6, 7, 9 and 10 above. Also, the second run name length is 44 characters instead of the maximum length of 40 specified in record set 3 described in paragraph 11-F.1.

APPENDIX 11-G CONFIGURATION FORCES AND MOMENTS PLOT FILE

11-G.1 Plot File Format for Configuration Data

The format of the configuration data plot file on logical unit 11 is described below:

Record Sets(s)	Item	Columns	Description
1	(6F10.5)	1-8	Data Format Specification (Format A8)
2	CDP Plot Title(s)		CDP Plot Title Information. Starts in column with a $ and consists of 4 lines of title information as follows:
	a)$CONFIGURATION DATA FROM CDP DATA BASE	1-38	CDP Plot Title (Format 3A10, A8)
	b) $(RID)	1-72	CDP Run Identification (RID).(Format 7A10,A2)
	c) $(PID)	1-72	CDP Problem Identification (PID).(Format 7A10,A2)
	d) $(UID)	1-72	CDP User Identification (UID).(Format 7A10,A2)
3	*RUN 40	1-7	Identifies maximum run name length of 40 alpha/numeric characters in CDP run name (record set 6).
4	$GLOBAL DATA	1-12	Global Data (see paragraph 11-G.2 for details)
5	(CDP Parameter Name List)	1-76	Identifies parameters available for plotting. If more than one line is needed to specify parameters, the word MORE must be entered in columns 73-76 on that line except for the last line of a parameter list. The CDP parameter name list is written on the plot file at the beginning of the plot file data for each solution and at the beginning of the accumulation sum data. The parameter list is written 6 per line. A detailed description of the CDP parameter name list is described in Table 11.3, (Format 6A10, 12X, A4)
6	(CDP Run Name)	1-40	A detailed description of the CDP Run Name is described in paragraph 11-G.3 (Format A1, **, I2, 4A4, A1, I3, A3, A1, A1, I2, A1, I2, 2X).

Record Sets(s)	Item	Columns	Description
7	(Configuration data from CDP in order of parameter name list)	1-60	CDP data is written in order of parameter name list specified in record set 5 above. The first record lists the solution number, magnitude of uniform onset flow velocity, alpha (angle of attack) and beta (angle of sideslip) values using format (I10, 3F10.5). The forces and moments data for the selected pressure rules and axis systems as shown in TABLE 11.3 are written on the plot file in the format specified in record set 1 above. (Repeat record sets 6 and 7 above for all selected data options.)
8	*EOF	1-4	The last line of dataset contains *EOF to signify the end of data for that run (Format A4).

**Formats for configuration options in columns 2-6 of the CDP Run Name are described in paragraph 11-G.3, item number 2.

11-G.2 Global Data for CDP file

A description of the Global Data for CDP is written on the configuration data plot file (logical unit 11) following record set 3 (i.e., *RUN 40 descriptor record identifying maximum run name length of 40).

Record Set(s)	Record Subset(s)	Item(s)	Columns	Description	Format
4		$GLOBAL DATA	1-12	Global Data	A12
	1	DATE	1-5	The heading DATA	A5
		DATECR	6-15	Date of creation in the form Yr/Mo/Date**	A10
	2	AMACH	1-10	Mach number	F10.5
		CALPHA	11-20	Angle of attack (degrees)	F10.5
		CBETA	21-30	Angle of sideslip (degrees)	F10.5
		NUMPOS	31-35	Number of planes of symmetry, 0= unsymmetric 1= one plane of sym. 2= two planes of sym.	I5
		NNET	36-40	Number of networks	I5
		NSOL	41-45	Number of solutions	I5
		NCASE	46-50	Number of cases	I5
	3	POSNRM	1-60	Normal to first and second planes normal to the planes of symmetry (3 by NUMPOS)	6F10.5
	4	POSLOC	1-30	Coordinates of point common to first and second planes	3F10.5
	5	NETPPP,NETID	1-70	Network index and ID, two networks per record subset [network number (I4) and network id (2A10)]	2(I4,1X,2A10,10X)
	6	ALPHA	1-70	Angle of attack for each solution (max 200)	7F10.5

** For the Ames System, the form is Date/Mo/Yr

Record Set(s)	Record Subset(s)	Item(s)	Columns	Description	Format
	7	BETA	1-70	Angle of sideslip for each solution (max 200)	7F10.5
	8	SOLLST,SOLID	1-70	Solution index and ID, two solutions per record subset [solution number (I4) and solution ID (2A10)]	2(I4,1X,2A10,10X)
	9	CASLST,CASEID	1-70	Case index and ID, two cases per record subset [case number (I4) and case ID (2A10)]	2(I4,1X,2A10,10X)
	10	REFPAR		List of reference data coefficient values	
		SR	1-10	Area reference parameter	F10.5
		CR	11-20	Chord reference parameter	F10.5
		BR	21-30	Span reference parameter	F10.5
	11	NUMAXS	1-4	Number of axis systems selected	I4
		AXISAR		List of selected axis systems allowable	
			5-8	1= reference coordinate system (RCS)	I4
			9-12	2= wind axis system (WAS)	I4
			13-16	3= body axis system (BAS)	I4
			17-20	4= stability axis system (SAS)	I4
	12	MOMLST	1-72	Coordinates of moment reference values for above axis system (3 by NUMAXS)	12F6.2
	13	ELRLST	1-72	Euler angles in degrees to go from RCS to selected axis system only for BAS	12F6.2

11-G.3 CDP Run Name Format

Item Numbers	Column(s)	Literal Name(s)	Format	Description
1	1	C	A1	CDP Identification
2(a)Panel Data	2-3		I2	Network Number
	4-6		I3	Panel Number
(b)Column Sum	2-3		I2	Network Number
	4	C	A1	Column Sum ID
	5-6		I2	Column Number
(c)Network Sum	2-3		I2	Network Number
	4-6		3X	Blanks
(d)Config. Sum	2-4	CON	A3	Configuration ID
	5-6		I2	Case Number
(e)Accum. Sum	2-4	ACC	A3	Accumulation ID
	5-6		I2	Case Number
3	7-8	99	I2	Job number, Preset to 99 (Not used)
4	9-12	UPPE LOWE UPLO LOUP AVER	A4	Surface Selection Option
5	13-16	BOUN VIC	A4	Velocity Computation Option
6	17-20	UNIF LOCA	A4	Pressure Computation Option
7	21-24	NONE SA1 SA2	A4	Velocity Correction Option
8(a)	25	C	A1	Case ID
(b)	26-28		I3	Case Number
9	29-31	INP 1ST 2ND 3RD	A3	Images
10(a)	32	P	A1	Panel ID
(b)	33	R	A1	Row ID
(c)	34-35		I2	Row Number
11(a)	36	C	A1	Column ID
(b)	37-38		I2	Column Number

12.0 FIELD DATA PROCESSOR (FDP) MODULE

12.1 Introduction

The Field Data Processor (FDP) is a stand alone program which is a module of the PAN AIR system. It presumes that singularity strengths have been calculated for points on the configuration and computes flow quantities for points in the field. It is a post processing module like the Point Data Processor (PDP) and Configuration Data Processor (CDP) modules. The two basic functions of FDP are to compute flow quantities at user selected points in the field (offbody points) or along streamlines. The computational core of routines in FDP was taken from the PAN AIR pilot code and does not conform to PAN AIR coding standards. This section will describe all of the higher level routines in FDP but will not describe some of the lower level routines. Also much of the code used in FDP was taken from the Matrix Generator (MAG) module. In particular, the PIVC subassembly, described in section 5, is used in FDP. The reader should review appendix P of reference 1 for an explanation of the computations performed in FDP and section 7.6.2 of reference 2 for a description of the user specifications for FDP.

12.2 FDP Overview

12.2.1 Purpose of FDP

FDP will examine flow behavior in the field away from the configuration surface. It will trace streamlines from a user specified starting point and compute flow quantities along the streamline. The user may also select individual points or grids of points off the body at which flow quantities are computed. The flow quantities output include all those available from the PDP module except for those associated with singularities. In addition, the streamline arclength and transit time can be output. The output quantities from FDP are enumerated in table 7.10 of reference 2.

12.2.2 FDP Input/Output Data

12.2.2.1 Input

Input data to the FDP module, like other post processing modules, comes from the MEC, DIP and MDG databases. (The reader should be familiar with the use of SDMS described in section 14.) The datasets required are given in appendix 12-C along with the routines where database maps are defined. To precisely define which quantities are input to FDP, compare the database maps with the MEC, DIP and MDG master definitions in appendices 2-A, 3-A and 8-A respectively.

In general, the MEC data base provides the names of the databases and DIP provides the user specifications for offbody point and streamline cases. (See section 7.6.2 of reference 2.) DIP also provides global problem data such as the number of networks, Mach number and planes of symmetry. The MDG database provides the panel geometry data, such as splines and normals, and the calculated singularities.

12.2.2.2 Output

The flow properties computed by FDP for offbody points and streamlines are printed and also written to a plot data file (logical unit 12). See figure

12.1. FDP does not produce an SDMS database. The FDP plot data file is similar to the plot data file (logical unit 10) which is written by the Print Plot Processor (PPP) module. The specific format of logical unit 12 is described in tables 8.41 through 8.43 of reference 2.

The printed output is similar to that of the PDP module. In all the offbody points cases, the selected flow quantities are preceded by a summary of the solutions selected and the points selected. The streamline cases begin with a summary of the selected starting points and integration parameters, followed by a status summary of each streamline. The status summary will indicate if the streamline integration has terminated abnormally. The flow quantities along the streamlines will be displayed after the summary. An example of FDP printed output is given in figure 8.9 of reference 2.

12.2.3 Internal Data Files

FDP uses four internal files for temporary data storage. They contain column singularities (logical unit 28), panel data (logical unit 18), panel singularities (logical unit 19) and streamline data (logical unit 8). These are not SDMS databases and so their contents are described in appendix 12-D. The internal data flow is shown in figures 12.2 through 12.5.

12.3 Module Description

12.3.1 Overall Structure

The FDP module does not use overlays but its operation can be divided into three basic tasks and one basic function. The module will perform preparation processing by initializing the contents of the internal data files. Flow quantities at offbody points will be computed and displayed during offbody processing. During streamline processing the streamline integration is performed and flow quantities along the streamlines are output. Both offbody and streamline processing will have to perform potential and velocity calculations.

12.3.2 Detailed Descriptions

12.3.2.1 Preparation Processing

This task is performed by the main routine FDPPRG prior to the offbody case loop. The MEC and DIP data bases are accessed in routine OPENDB and their data is loaded into labeled common blocks. The columns of singularity data on the MDG data base are unsymmetrized and transferred to the column singularity dataset (logical unit 28). The routine BLOCK loads various program constants. The routine PPPDQ takes the essential panel data from the MDG data base and packages it in the panel data dataset (logical unit 18). In FDPPRG, the number of offbody and streamline cases is retrieved from the DIP database and the global headings are written to the plot data file (logical unit 12) by routine PLTHDR.

12.3.2.2 Offbody Processing (OFFBD)

All the processing for offbody cases is performed inside the offbody case loop in FDPPRG. The user case selections are retrieved from the DIP database and the data is directed to labeled common blocks with the SDMS static map

option. The routine PANSNG takes the unsymmetrized column singularities for only the solutions selected for the case and the panel data and writes the panel singularity data (logical unit 19). The routine OUTPREP interprets the case selections so that FDPOUT will print the data in the proper format. The offbody points are represented internally as a list of points and a matching list of solution numbers. Thus for a case with multiple solutions, one point would have multiple occurrences in the point list each with a different corresponding solution index. The data is ultimately passed to the potential and velocity calculations in this form. The call to OFFBD begins the calculation and display of offbody points for a selected subset (usually all) of the case solutions. The total velocity and perturbation potential for each point and solution index pair is calculated by a call to PVCAL. (See section 12.3.2.4.) Then for each point and solution, the full set of requested quantities is computed from these basic quantities using procedures explained in appendix N of reference 1. The routine FDPOUT writes the printed output and FDPPLT writes the plot data file.

12.3.2.3 Streamline Processing (STMLNE)

All of the processing for streamline cases is performed inside the streamline case loop in FDPPRG. The user case selections are retrieved from the DIP database. The routines PANSNG and OUTPREP perform the same function here as in section 12.3.2.2. The starting point list and corresponding solution index list is prepared in the same fashion as in section 12.3.2.2. Additional lists are prepared which indicate the streamline limits and direction. The data is ultimately passed to the integrator in this form. The call to STMLNE begins the calculation and display of streamline points for a selected subset of the case solutions. The routine STMLNE2 and subordinates perform the integration. They were extracted from the PAN AIR pilot code and perform the asynchronous integration of multiple streamlines. That is, several streamlines are integrated together to minimize multiple accesses to the same set of panel data. The integration technique is described in appendix P of reference 1. The function evaluation by the integrator is directed by the routine FSTLMN which calls PVCAL. (See section 12.3.2.4.) The basic streamline data is written by the integrator to logical unit 8 as it is computed. (See appendix 12-D.) The file is sequential so data for a particular streamline may be scattered throughout the dataset. When the integration is completed and the streamline data has been written, the routine STMOUT reorders the data, expands the basic data to the full set of selected output quantities, and writes the printed output and plot data file. The routine will read through the streamline data (unit 8) looking for data for the first streamline, rewind the dataset and repeat the process for subsequent streamlines. The selected output quantities are computed in the same fashion as in section 12.3.2.2. The streamline arclength is calculated by the integrator. The streamline travel time is computed by averaging the velocity (or mass flux) between adjacent streamline points. The routines FDPOUT and FDPPLT perform the same function as in section 12.3.2.2.

12.3.2.4 Potential and Velocity Calculation (PVCAL)

Both the streamline and offbody routines require the calculation of velocity and potential at selected points. (See appendix P.1 of reference 1.) The routine PVCAL retrieves singularity data for panels from unit 19 and panel geometry data from unit 18 and uses the PIVC subassembly to compute perturbation velocity and potential. The PIVC subassembly is also used in MAG

and it is described in section 5. The perturbation velocity is converted to total mass flux or velocity per request of the calling routine.

12.3.3 Module Database

FDP does not generate an SDMS database.

12.3.4 Data Interfaces

12.3.4.1 System Interfaces

Figure 12.1 illustrates the external interfaces between the FDP module and the MEC, DIP and MDG data bases. Figures 12.2 through 12.5 illustrate the internal interfaces between routines in FDP and the datasets on units 8, 18, 19 and 28. Figure 12.1 illustrates the printed output and plot file generated by FDP. They are not required by any other PAN AIR module.

12.3.4.2 Subprogram Interfaces

A tree diagram of all the routines in FDP is given in Appendix 12-A. This show the interrelationships among the subroutines which make up FDP. It also shows the calls to routines in the SDMS library (see section 14) and the PAN AIR library (see section 13). Figures 12.1 through 12.5 also show the called-by relationships of routines in FDP without the references to library and low level routines. Each subroutine is described briefly in section 12.4.2.

12.3.5 Data Flow in FDP

Figures 12.2 to 12.5 illustrate the data flow between routines in FDP and external databases and datasets. All accesses to SDMS databases are performed in routines FDPPRG, OPENDB and PPPDQ. Access to the FDP internal datasets is described in Appendix 12-D. The flow of data is labeled common and formal parameters may be traced with the use of comments in the code.

12.4 LOWER LEVEL FUNCTIONS

The following paragraphs present the functional decompositions of the routines and gives the purpose of each routine.

12.4.1 Functional Decomposition

The FDP functional decomposition is given in Appendix 12-B.

12.4.2 Subroutine Descriptions

The subroutines used in FDP are described below.

BLOCK

Initializes selected labeled common blocks.

BLTRNS

Initializes the panel singularity dataset (unit 19).

BRTRNS

Initializes the panel data dataset (unit 18).

COEFP

Computes pressure coefficients, local Mach numbers and critical pressure coefficients.

CSCAL2

Scales the component of a vector parallel to the compressibility axis. It is used to convert between perturbation mass flux and velocity.

DUALXF

Converts between total mass flux and velocity.

ELTRNS

Clears the data buffer for the panel singularity dataset (unit 19).

ERTRNS

Clears the data buffer for the panel data dataset (unit 18).

FDPOUT

Writes the selected flow quantities for a point to the printed output dataset. It also writes pages headings and labels as appropriate.

FDPPLT

Writes the selected flow quantities for a point to the plot data dataset (unit 12).

FDPPRG

Controls the preparation for FDP cases and the processing of offbody and streamline cases.

FSTLMN

Performs the evaluation of velocity or mass flux for the streamline integration. It arranges the data in the proper form and then calls PVCAL.

ILTRNS

Writes data to the panel singularity dataset (unit 19).

IRTRNS

Writes data to the panel data dataset (unit 18).

LTRNS

Reads data from the panel singularity dataset (unit 19).

OFFBD

Computes and prints flow quantities at offbody points for selected solutions.

ONSTFL

Adds any rotational component to the freestream velocity.

OPENDB

Opens and defines the maps for all the SDMS databases. Reads and initializes global run parameters, such as Mach number and compressibility axis. It also unsymmetrizes singularity values and writes them to the column singularity dataset (unit 28).

ORIENT

Computes the orientation of a vector with respect to the positive x axis in the reference coordinate system. It is used to compute quantities such as VALPHA and VBETA (see table 7.10 in reference 2) or to regenerate α_C and β_C from the compressibility direction.

OUTPREP

Prepares to output FDP quantities. It distinguishes between offbody and streamline cases and initializes the appropriate headings. Since any subset of the possible flow quantities can be selected for printing, OUTPREP computes how they will be formatted. This information is used by FDPOUT to actually perform the write.

PAKLAM

Packs the panel singularities into a single buffer.

PAKPQF

Packs the panel data into a single buffer.

PANSNG

Converts the column singularity data to panel singularities on unit 19. It writes singularities only for the current set of selected solutions.

PIVC

Computes a panel's influence on a point. It is described in section 5.

PLTHDR

Writes the headings for the plot data dataset (unit 12).

PPPDQ

Transfers the panel data on the MDG data base to the panel data dataset (unit 18).

PVCAL

Computes perturbation potential and total velocity or mass flux.

RTRNS

Reads data from the panel data dataset (unit 18).

SETUP

Prepares to compute streamlines.

SETUP1

Controls the streamline integration and writes the streamline data as computed to unit 8.

STEP

Performs the streamline integration.

STMLNE

Computes the streamlines and outputs flow quantities along the streamlines.

STMLNE2

Computes streamlines.

STMOUT

Outputs flow quantities along the streamlines.

UPKLAM

Unpacks a buffer of panel singularities.

UPKPQF

Unpacks a buffer of panel data.

VELCOR

Performs velocity corrections.

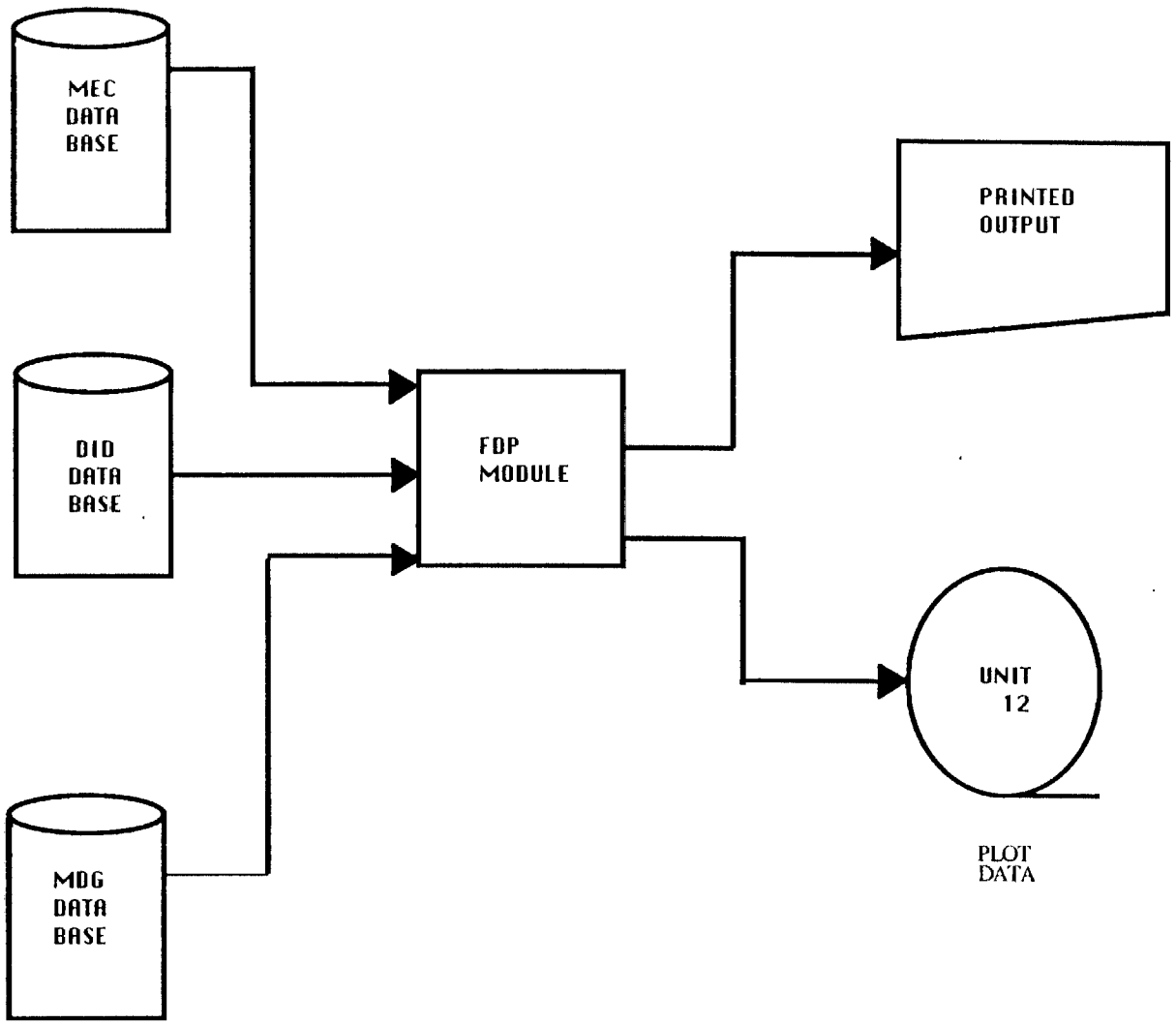


FIGURE 12.1 - FDP EXTERNAL INTERFACES

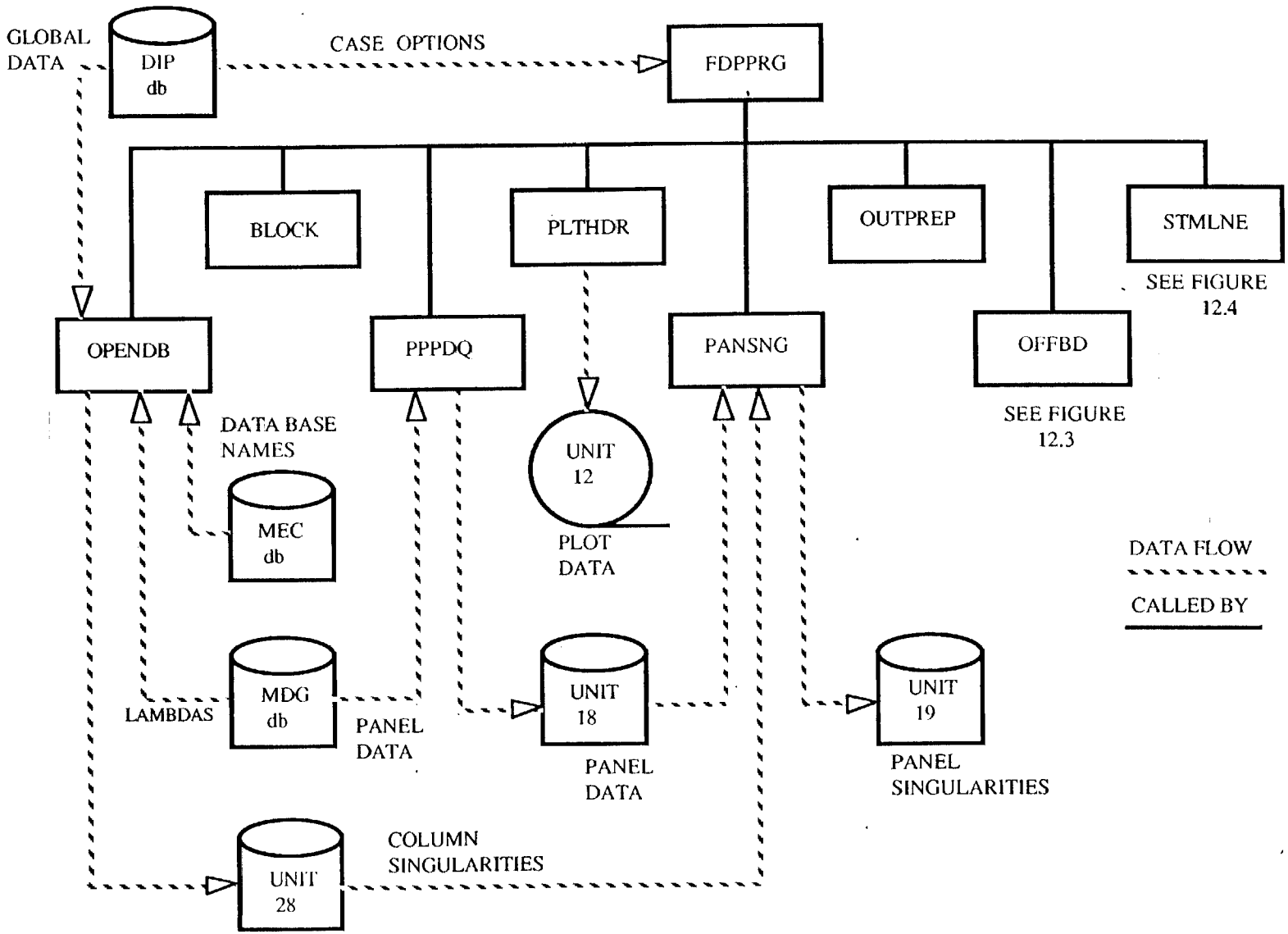


FIGURE 12.2 - FDP INTERNAL INTERFACES

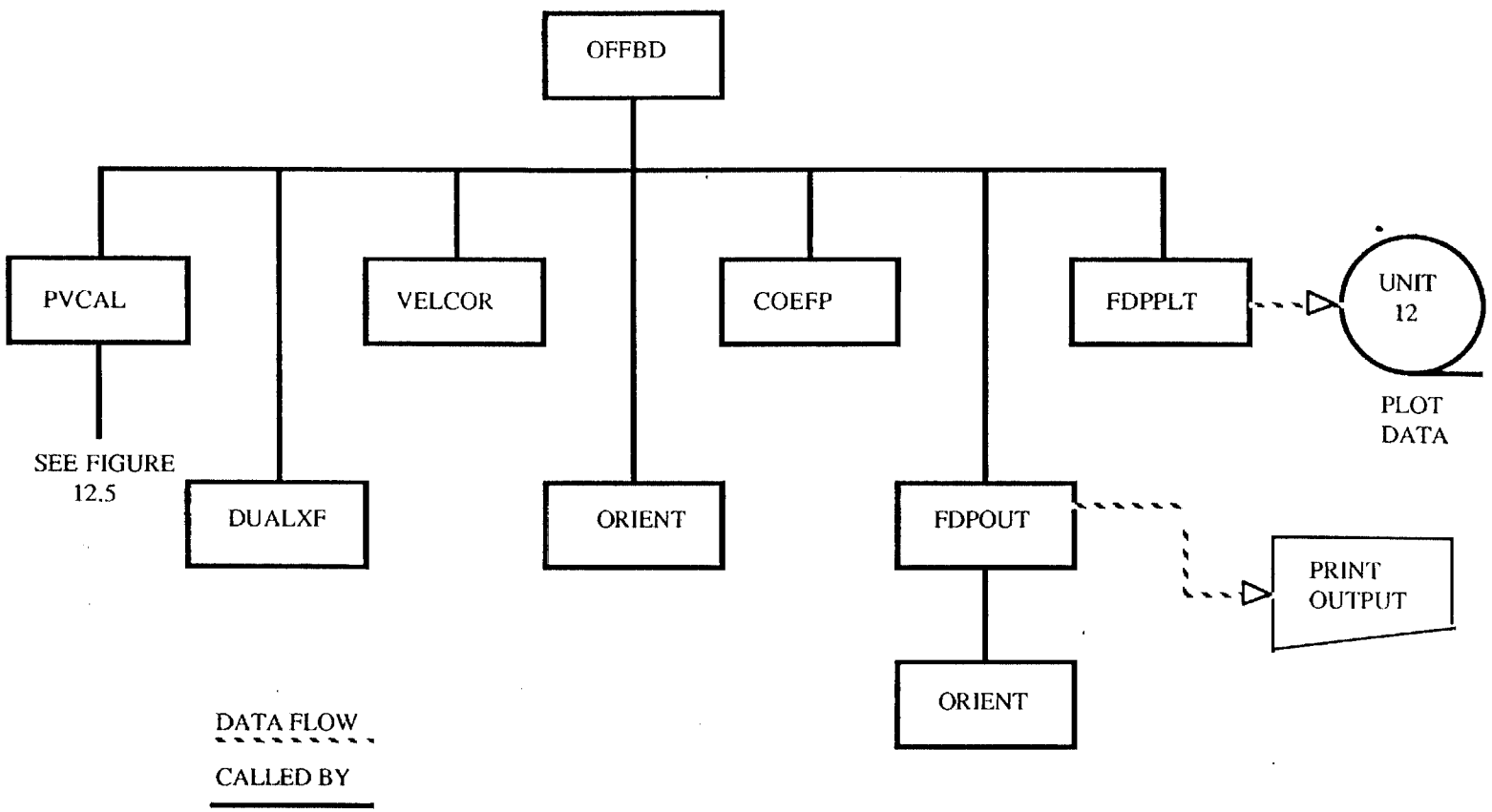


FIGURE 12.3 - OFFBODY INTERNAL INTERFACES

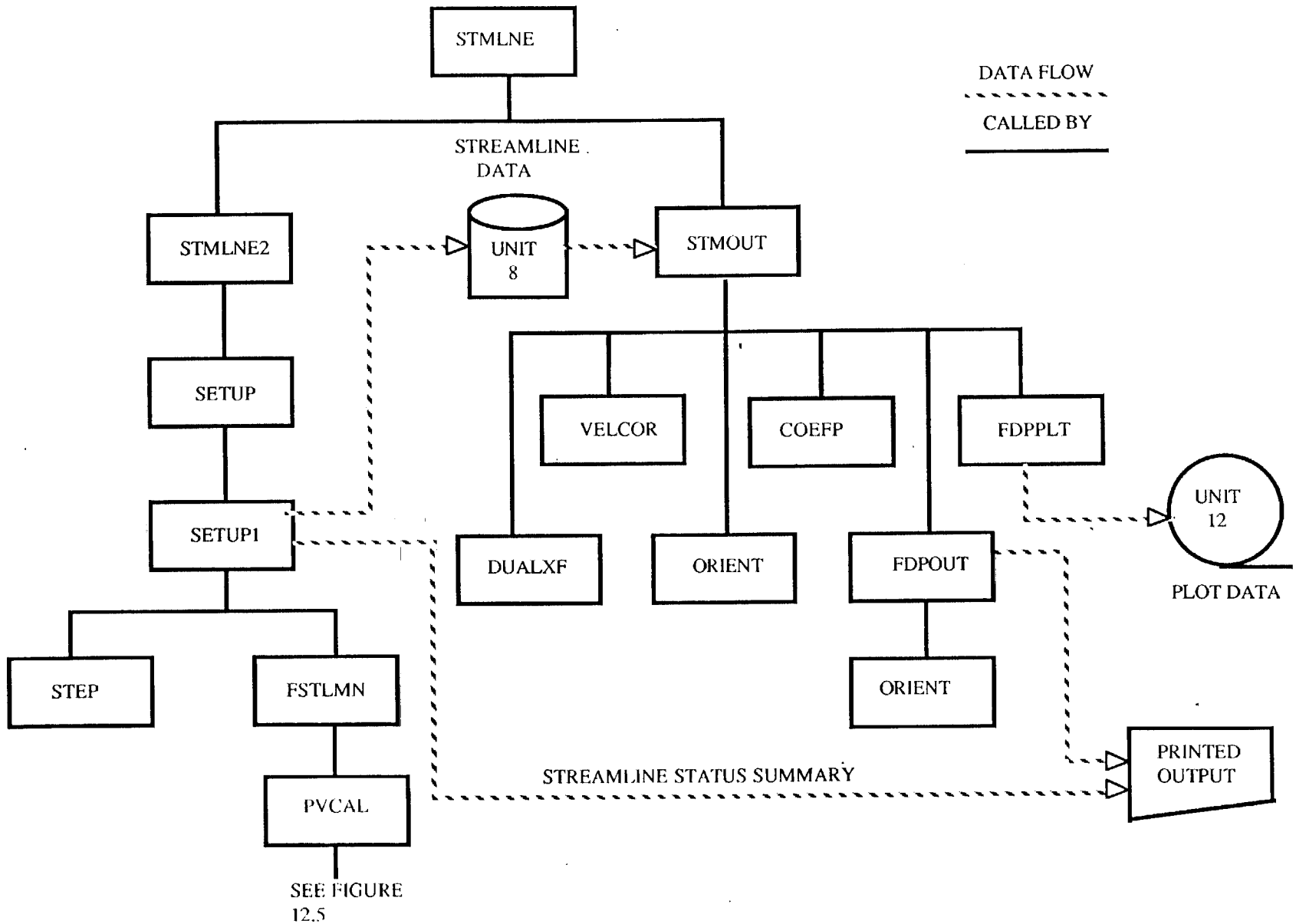


FIGURE 12.4 - STREAMLINE INTERNAL INTERFAACES

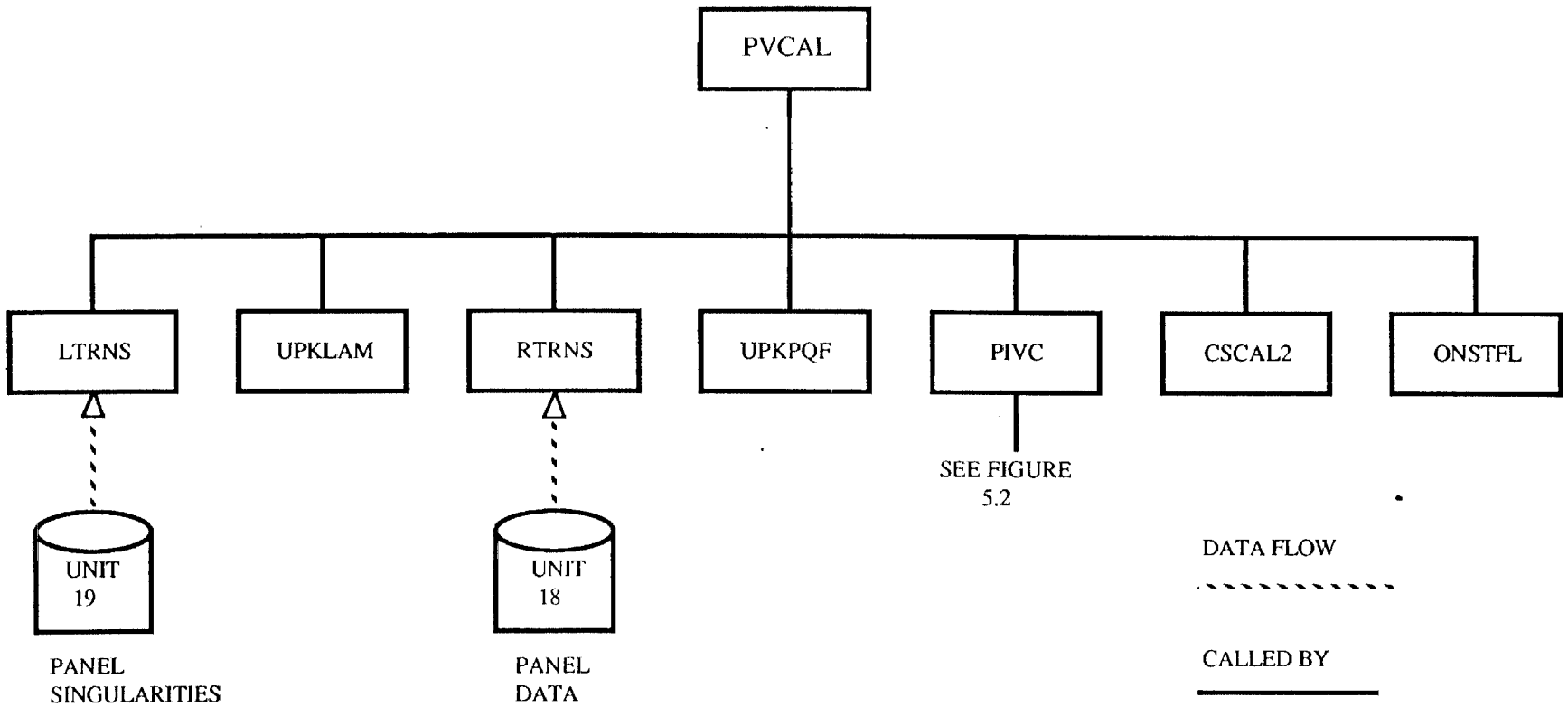


FIGURE 12.5 - PVCAL INTERNAL INTERFACES

APPENDIX 12-A

TREE STRUCTURE

The tree structure diagram of the FDP module has been deleted from this document. It is, however, available on the installation tape.

APPENDIX 12-B

FUNCTIONAL DECOMPOSITION OF FDP

This appendix describes the functional decomposition of FDP, that is, an outline form of the structure of FDP organized by task. Where a particular task is realized as a subroutine, the subroutine name is listed in parenthesis along the right margin.

A. Prepare for offbody and streamline data processing

A. Define the SDMS maps for the DIP and MDG databases. Retrieve global case data. Transfer the singularity data (MDG datasets LAMBDA-UNKNOWN and LAMBDA-KNOWN) to a random access (READMS/WRITEMS) dataset (FT28) (OPENDB)

B. Initialize labeled common blocks (BLOCK)

C. Transfer the panel data (MDG dataset MAG-PANEL-DATA) to a sequential binary dataset (FT18) (PPPDQ)

D. Write the plot dataset (FT12) heading (PLTHDR)

B. Perform offbody data processing

A. Get offbody case options

B. Write source and doublet parameters for each panel and selected solution to a sequential binary dataset (FT19) by combining the global singularity data from FT28 and the panel data from FT18. (PANSNG)

C. Prepare the output heading format (OUTPREP)

D. Compute and output flow quantities at offbody points (OFFBD)

A. Compute total velocity/mass flux and perturbation potential (PVCAL)

A. Get singularity data from FT19 (LTRNS)

B. Interpret the singularity data in a usable form (UPKLAM)

C. Get panel data from FT18 (RTRNS)

D. Interpret the panel data in a usable form (UPKPQF)

E. Compute perturbation velocity and potential (see section 5.4.2 and figure 5.2 for a decomposition of the PIVC subassembly) (PIVC)

F. Compute perturbation mass flux (CSCAL2)

G. Compute total velocity/mass flux (ONSTFL)

B. Perform velocity corrections (VELCOR)

C. Compute pressure coefficients (COEFP)

D. Compute total potential

E. Output field flow properties (FDPOUT)

F. Write flow quantities to the plot dataset (FT12) (FDPPLT)

C. Perform streamline data processing

A. Get streamline case options

B. Write source and doublet parameters for each panel and selected solution to a sequential binary dataset (FT19) by combining the global singularity data from FT28 and the panel data from FT18 (PANSNG)

C. Prepare the output heading format (OUTPREP)

D. Compute and output flow quantities along streamlines (STMLNE)

A. Compute flow quantities along streamlines (STMLNE2)

A. Prepare to compute streamlines (SETUP)

B. Control the asynchronous integration of multiple streamlines and write the results as computed to a binary sequential dataset (FT08) (SETUP1)

A. Move along the appropriate integration path (STEP)

B. Prepare to evaluate the total velocity (FSTLMN)

C. Compute the total velocity/mass flux andperturbation potential (see paragraph BDA) (PVCAL)

B. Output flow quantities along streamlines (STMOUT)

A. Get all the data for just the current stream line from FT08

B. Transform velocity to mass flux (DUALXF)

C. Transform mass flux to velocity (DUALXF)

D. Compute velocity corrections (VELCOR)

E. Compute pressures (COEFP)

F. Compute arc length and time

G. Output field flow properties (FDPOUT)

H. Write flow quantities to the plot dataset (FT12)

APPENDIX 12-C

DATA BASE COMMUNICATIONS CHART

The Data Base Communications Chart is presented in three forms. Each form is alphabetized by columns from left to right. The first form has a column order of Data Base, Dataset Name, Map Name, Common Block and Subroutine. The second form has a column order of Data Base, Map Name, Dataset Name, Common Block and Subroutine. The third form has a column order of Common Block, Data Base, Map Name, Dataset Name and Subroutine. Thus a person can get a cross reference on a data element by knowing either the Dataset Name, Map Name or Common Block name.

FIRST FORM

DATA BASE	DATASET NAME	MAP NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
MEC	DATA-BASE-HEADER	MECHED	/RUNIDS/	OPENDB
MEC	MACRO-OPTIONS	MACRO	/MAGNUM/	OPENDB
DIP	GLOBAL	FSVMAP	/ACASE/	OPENDB
DIP	GLOBAL	FSVMAP	/SYMM/	OPENDB
DIP	GLOBAL	GLOBAL	/MAGGLO/	OPENDB
DIP	GLOBAL	GLOBAL	/MAGNUM/	OPENDB
DIP	GLOBAL	GLOBAL	/NSOLN/	OPENDB
DIP	GLOBAL	GLOBAL	/SYMTRY/	OPENDB
DIP	GLOBAL-FLOW-PROP	DIP-GLOFLO	dynamic	OPENDB
DIP	OFFBODY-OPTIONS	DIP-OBOPT	/FDPCAS/	OPENDB
DIP	OFFBODY-OPTIONS	DIP-OBOPT	/OFBOPT/	OPENDB
DIP	OFFBODY-OPTIONS	DIP-OBOPT	/ZFDP/	OPENDB
DIP	STREAMLINE-OPTIONS	DIP-SLOPT	/FDPCAS/	OPENDB
DIP	STREAMLINE-OPTIONS	DIP-SLOPT	/KSTMLN/	OPENDB
DIP	STREAMLINE-OPTIONS	DIP-SLOPT	/STLOPT/	OPENDB
DIP	STREAMLINE-OPTIONS	DIP-SLOPT	/STMCAS/	OPENDB
DIP	STREAMLINE-OPTIONS	DIP-SLOPT	/ZFDP/	OPENDB
MDG	GLOBAL	GLOMDG	/MAGNUM/	OPENDB
MDG	LAMBDA-KNOWN	LAM-KNOW	dynamic	OPENDB
MDG	LAMDBA-UNKNOWN	LAM-UNKN	dynamic	OPENDB
MDG	MAG-PANEL-DATA	MAG-PAN	/PANDF/	OPENDB
MDG	MAG-PANEL-DATA	MAG-PAN	/PANDQ/	OPENDB

SECOND FORM

DATA BASE	MAP NAME	DATASET NAME	COMMON BLOCK	PROGRAM/ SUBROUTINE
MEC	MACRO	MACRO-OPTIONS	/MAGNUM/	OPENDB
MEC	MECHED	DATA-BASE-HEADER	/RUNIDS/	OPENDB
DIP	DIP-GLOFLO	GLOBAL-FLOW-PROP	dynamic	OPENDB
DIP	DIP-OBOPT	OFFBODY-OPTIONS	/FDPCAS/	OPENDB
DIP	DIP-OBOPT	OFFBODY-OPTIONS	/OFBOPT/	OPENDB
DIP	DIP-OBODT	OFFBODY-OPTIONS	/ZFDP/	OPENDB
DIP	DIP-SLOPT	STREAMLINE-OPTIONS	/FDPCAS/	OPENDB
DIP	DIP-SLOPT	STREAMLINE-OPTIONS	/KSTMLN/	OPENDB
DIP	DIP-SLOPT	STREAMLINE-OPTIONS	/STLOPT/	OPENDB
DIP	DIP-SLOPT	STREAMLINE-OPTIONS	/STMCAS/	OPENDB
DIP	DIP-SLOPT	STREAMLINE-OPTIONS	/ZFDP/	OPENDB
DIP	FSVMAP	GLOBAL	/ACASE/	OPENDB
DIP	FSVMAP	GLOBAL	/SYMM/	OPENDB
DIP	GLOBAL	GLOBAL	/MAGGLO/	OPENDB
DIP	GLOBAL	GLOBAL	/MAGNUM/	OPENDB
DIP	GLOBAL	GLOBAL	/NSOLN/	OPENDB
DIP	GLOBAL	GLOBAL	/SYMTRY/	OPENDB
MDG	GLOMDG	GLOBAL	/MAGNUM/	OPENDB
MDG	LAM-KNOW	LAMBDA-KNOWN	dynamic	OPENDB
MDG	LAM-UNKN	LAMBDA-UNKNOWN	dynamic	OPENDB
MDG	MAG-PAN	MAG-PANEL-DATA	/PANDF/	OPENDB
MDG	MAG-PAN	MAG-PANEL-DATA	/PANDQ/	OPENDB

THIRD FORM

COMMON BLOCK	DATA BASE	MAP NAME	DATASET NAME	PROGRAM/ SUBROUTINE
/ACASE/	DIP	FSVMAP	GLOBAL	OPENDB
dynamic	DIP	DIP-GLOFLO	GLOBAL-FLOW-PROP	OPENDB
dynamic	MDG	LAM-KNOW	LAMDBA-KNOWN	OPENDB
dynamic	MDG	LAM-UNKN	LAMBDA-UNKNOWN	OPENDB
/FDPCAS/	DIP	DIP-OBOPT	OFFBODY-OPTIONS	OPENDB
/FDPCAS/	DIP	DIP-SLOPT	STREAMLINE-OPTIONS	OPENDB
/KSTMLN/	DIP	DIP-SLOPT	STREAMLINE-OPTIONS	OPENDB
/MAGGLO/	DIP	GLOBAL	GLOBAL	OPENDB
/MAGNUM/	DIP	GLOBAL	GLOBAL	OPENDB
/MAGNUM/	MDG	GLOMDG	GLOBAL	OPENDB
/MAGNUM/	MEC	MACRO	MACRO-OPTIONS	OPENDB
/NSOLN/	DIP	GLOBAL	GLOBAL	OPENDB
/OFBOPT/	DIP	DIP-OBOPT	OFFBODY-OPTIONS	OPENDB
/PANDF/	MDG	MAG-PAN	MAG-PANEL-DATA	OPENDB
/PANDQ/	MDG	MAG-PAN	MAG-PANEL-DATA	OPENDB
/RUNIDS/	MEC	MECHED	DATA-BASE-HEADER	OPENDB
/STLOPT/	DIP	DIP-SLOPT	STREAMLINE-OPTIONS	OPENDB
/STMCAS/	DIP	DIP-SLOPT	STREAMLINE-OPTIONS	OPENDB
/SYMN/	DIP	FSVMAP	GLOBAL	OPENDB
/SYMTRY/	DIP	GLOBAL	GLOBAL	OPENDB
/ZFDP/	DIP	DIP-OBOPT	OFFBODY-OPTIONS	OPENDB
/ZFDP/	DIP	DIP-SLOPT	STREAMLINE-OPTIONS	OPENDB

APPENDIX 12-D
FDP INTERNAL DATASETS

12-D.1 Introduction

FDP uses four internal datasets for temporary storage which are not SDMS datasets. They are described in the following sections.

12-D.2 Column Singularities (Unit 28)

This is a random access binary dataset created by READMS/WRITEMS routines on logical unit 28. Its records are keyed by solution number. Each record contains all of the calculated singularity strengths for that particular solution. The singularities have been unsymmetrized. They were derived from the symmetrized singularities on the MDG datasets LAMBDA-KNOWN and LAMBDA-UNKNOWN by the routine OPENDB. They are used by the routine PANSNG to derive the singularity strengths on a particular panel. The singularities are stored in the following order: known-nonupdatable, known-updatable, unknown-nonupdatable and unknown-updatable. That group is repeated within a record for each distinct image.

12-D.3 Panel Data (Unit 18)

This is a sequential access binary dataset created by unformatted binary writes to logical unit 18. Each physical record is 2048 words long. The physical records are logically divided into 256 word packets. These packets contain the essential panel defining quantities which are prepared by the routine PAKPQF. The panel data dataset is created by the routine PPPDQ which reads the dataset MAG-PANEL-DATA in MDG and makes calls to PAKPQF. The panel data is used in velocity calculations by PVCAL. The routines BRTRNS, IRTRNS, ERTRNS and RTRNS are used to read and write to the dataset.

12-D.4 Panel Singularities (Unit 19)

This is a sequential access binary dataset created by unformatted binary writes to logical unit 19. If NS is the number of solutions selected for the current case and NI is the number of distinct images, then the physical record length is the largest multiple of 14*NS*NI which is less than or equal to 1120. The physical records are logically divided into packets of 14*NS*NI words. The packets contain the singularity strengths (the five source parameters followed by the nine doublet parameters) for a particular panel. The group of 14 singularity values are repeated for each distinct image and that group in turn is repeated for each of the selected solutions. The dataset is rewritten for each case by the routine PANSNG and will contain only the singularities for the solutions selected for the case. PANSNG combines the column singularities and panel data to create the panel singularities. They are used by PVCAL in velocity calculations. The routines BLTRNS, ILTRNS, ELTRNS and LTRNS are used to read and write to the dataset.

12-D.5 Streamline Data (Unit 8)

This is a sequential access binary dataset created by unformatted binary writes to logical unit 8. Each record is 12 words long. It contains the basic data for a particular point of a particular streamline. Table 12-D.1 shows the contents of a record of basic streamline data. The dataset is written by the routine SETUP1 as the streamline integration is being performed. The records for a given streamline may be scattered through the dataset. The routine STMOUT reads through unit 8 searching for records for the first streamline, rewinds the dataset and repeats the process for the remaining streamlines.

Table 12.D.1 - Basic Streamline Data

Word Number	Contents
1	streamline number
2	streamline point number
3	arclength
4	x component of position
5	y component of position
6	z component of position
7	x component of total velocity*
8	y component of total velocity*
9	z component of total velocity*
10	perturbation potential
11	order of integration (see section P.2.1 of reference 1)
12	streamline direction +1 implies downstream and -1 implies upstream

* If variable TPSL is common block /KSTMLN/ is 0.0, the velocity is replaced by mass flux.

13.0 PAN AIR LIBRARY (PALIB)

13.1 INTRODUCTION

This section is a guide to the diverse collection of routines that make up the library used in PAN AIR. This guide is a short description of what each routine does. It is not a detailed description of calling sequences nor is it a detailed description of program contents (with exceptions as noted below). Note that structured programming techniques were used for only about ten percent of the routines which make up PALIB. The extent and completeness of in-line documentation in the other ninety percent of the library varies from excellent to non-existent. The basic supposition of this manual is that the programs in PALIB have been so thoroughly used in previous (non-PAN AIR) programs that no errors can remain in them, and that they are so simple that they may be easily modified even without extensive documentation.

This section also describes eleven basic classes of subroutines which make up PALIB and provides a short description of each subroutine. All of the PALIB routines required by PAN AIR are described in this section. For one of the classes (e.g. the Constrained Quadratic Least Squares Fit subroutines), a more detailed discussion of the operation is provided in Appendix 13-B.

13.2 PALIB OVERVIEW

13.2.1 Purpose of PALIB

PALIB is a collection of general purpose library routines which are used by modules within the PAN AIR system. They are divided into eleven classes of related routines:

1) matrix and vector manipulation;
2) general routines related to arbitrary geometries;
3) special routines related to PAN AIR geometries;
4) general mathematical routines;
5) constrained quadratic least squares fit routines;
6) blank common management;
7) special purpose SDMS-related routines;
8) real matrix solver routines;
9) free field format input routines;
10) miscellaneous; and
11) input data processing support routines;

13.2.2 PALIB Output

Most routines in PALIB print out only error messages. A few, however, produce printed output as part of their normal execution. They are CHPADB, LODREC and several of the miscellaneous routines.

13.2.3 Database Interfaces

The two subroutines in the "special purpose SDMS-related" class of subroutines communicate with one or more SDMS databases. One subroutine,

CHPADB, is used by all PAN AIR Modules to create all permanent and temporary databases. Note that no data is written to any of the databases by CHPADB. It merely opens and closes the database. The other subroutine in this class (MULTI) is used to perform an out of core matrix multiplication operation. The matricies are assumed to be stored as datasets of an SDMS database. The SDMS maps (see Section 1, Paragraph 1.2.3) are defined externally by the calling program.

A second class of routines in PALIB utilize SDMS databases in I/O operations. This class is the real matrix solver set of routines. We note here only that the SDMS map names are defined externally just as for the subroutine MULTI.

13.3 DESCRIPTION OF CLASSES OF SUBROUTINES IN PALIB

A brief description of the eleven classes of subroutines is given in this section. Included is a list of all subroutines which are members of the class.

Note that most PALIB routines receive their data by way of a formal parameter list. However, there are some subroutines in the library which receive data by labeled common block (notably blank common management routines). Tree diagrams for subroutines in PALIB are presented in Appendix 13-A.

13.3.1 Matrix and Vector Manipulations

The subroutines in this class perform various combinations of vector and matrix operations. Table 13.1 lists the subroutine names and a short indication of the operation which is performed.

13.3.2 General Routines Related to Arbitrary Geometry

The subroutines in this class manipulate basic geometric concepts such as a point, line or plane. There is no direct reference to their application in PAN AIR. Table 13.2 lists the subroutine names and a short indication of the operation that they perform.

13.3.3 Special Routines Related to PAN AIR Geometry

This class of operations are related to those in the first class except that they involve some specialized operations that occur frequently in PAN AIR but might not occur in some other system, (e.g. the evaluation of the compressible inner product). They deal mostly with panel defining quantities which are computed by DQG and MAG. Some routines also implement the graph theoretic approach to the assignment of matching conditions at abutment intersections. (See reference 1.) Table 13.3 lists the subroutine names and a brief indication of their operation.

13.3.4 General Mathematical Routines

These routines perform a variety of general mathematical operations such as sorting arrays, inverting matrices, and solving sets of linear equations.

The routines are listed by name in Table 13.4 along with a brief description of the function.

13.3.5 Constrained Quadratic Least Squares Fit

These routines solve the general least squares fit problem with an arbitrary number of independent exact constraints. A hierarchy exists in that subroutine CQLSF is the only routine called by a PAN AIR Module. In turn it calls subroutine DEFVEC and LSQSFX. Subroutines LSQSFX calls subroutines DCBHTX and PSINTP. The function of these routines is briefly discussed in Table 13.5. A more detailed description will be found in Appendix 13-B.

13.3.6 Blank Common Management

This set of programs keeps track of multiple arrays stored in blank common. They are relatively unsophisticated but their use simplifies the code within the modules of PAN AIR. A short description of the purpose of each subroutine in this group is given in Table 13.6. Note that subroutine STARTR must be called before any other subroutine in this class is called. All subroutines in this group are written in FORTRAN. These routines reference a blank common array with a dimension of 1. It is up to the calling module to dimension blank common and pass that length in another special common block. A PAN AIR module which uses these routines will cause the loader to indicate that the length of blank common has been redefined. This is acceptable as long the definition in the PAN AIR module is used.

13.3.7 Special Purpose SDMS-Related Routines

These are subroutines which perform specialized tasks within the PAN AIR System and which make use of the SDMS database system. Table 13.7 describes the FORTRAN subroutines.

13.3.8 Real Matrix Solver

This collection of subroutines performs an out-of-core matrix inversion and back-substitution. It makes use of an SDMS database. Table 13.8 lists the routines.

13.3.9 Free Field Format Input Routines

This class of subroutines reads an input record in free field format and decodes it into separate data items based on the occurance of certain delimiters (blanks, commas and end of card). LODREC is the controlling subroutine in this process. Table 13.9 briefly describes these subroutines.

13.3.10 Miscellaneous

These routines are those which do not seem to list well with the other classes. Table 13.10 describes these subroutines.

13.3.11 Data Input Processing Support Routine

These routines support only the DIP module. They each perform an operation which occurs in more than one place in the module. The routines are listed by name in Table 13.11 along with a brief description of the function. All routines in this class are written in FORTRAN.

Table 13.1 - Matrix and Vector Manipulation Routines

SUBROUTINE	OPERATION	COMMENTS
CAB	$C = A * B$	arrays stored by columns
CAD	$C = d * A$	
CAMB	$C = A - B$	
CAPB	$C = A + B$	
CAPDB	$C = A + d * B$	
CATB	$C = A^t * B$	
CMAB	$C = A * B$	arrays stored by rows
CROSS	$Z = X \times Y$	
CXMAB	$C_{im} = e_{ijk} A_{jl} B_{klm}$	
DET	$d = \det(A)$	3 by 3 determinant
MUL3X3	$C = A * B$	3 by 3 matricies
MXMACA	$C = C + A * B$	highly vectorized CAL
RRAATX	$Y = Y + A^t * X$	
RRAAX	$Y = Y + A * X$	
RRZAB	$C = A * B$	
RRZATB	$C = A^t * B$	
RRZATX	$Y = A^t * X$	
RRZAX	$Y = A * X$	
RRZXYT	$A = X * Y^t$	
TRANS	$C = A^t$	
UNIVEC	$Y = X$ / magnitude(X)	arbitrary dimension
UVECT	$Y = X$ / magnitude(X)	dimension 3
VADD	$C = A + d * B$	
VIP	$d = X \cdot Y$	
VIPDA	$d = d + X \cdot Y$	
VMAG	$d =$ magnitude(X)	
VMUL	$X = d * Y$	
ZERO	$X = 0$	

Note: A, B, C are matricies
X, Y are vectors
d is a scalar

Table 13.2 - General Geometry Routines

SUBROUTINE	OPERATION
DISTNC	Computes the distance between two points
IMAGE	Reflects a point through a plane of symmetry
INSIDE	Determines if a point lies inside a quadralateral
ISCAL	Determines if an edge is collapsed
LPROJ	Performs a length preserving projection
NORCAL	Computes the unit normal to a triangle
NRPTED	Finds a point on an edge that is closest to a given point
PIDENT	Determines if two points are essentially coincidental
PROJ	Projects a vector onto a plane
XXADJ	Withdraws the vertex of a triangle a fraction of the distance along its angular bisector

Table 13.3 - PAN AIR Geometry Routines

SUBROUTINE	OPERATION
ABTINT	Generates the matching condition assignments for the control points in an abutment intersection
BIQUAD	Evaluates the nine canonical biquadratic basis functions at a point on the standard isoparametric element. It is used in spline construction.
CCALN	Prepares computations for panel moment data. It invokes PANMOM.
COMPIP	Evaluates the compressible inner product. (See reference 1.)
GPHPLK	Reorganizes the node to node description of a tree so that branches occur in the proper order for pruning. It is called by ABTINT. (See appendix F in reference 1.)
GPHSCN	Identifies a spanning tree for a connected graph. It is called by ABTINT. (See appendix F in reference 1.)
GPLUCK	Assigns nodes to branches and checks for bad assignments. It is called by ABTINT. (See appendix F in reference 1.)
GTALAM	Computes the "ALAM" array for subpanel doublet splines. The "ALAM" array contains lambda vectors defining corner chord midpoint values used in subpanel spline calculations.
NRPTHP	Finds an estimate of the point on an H-P surface closest to a given control point. It is used to compute hyperbolic-paraboidal coordinates of the four corner chord midpoints which is one of the panel defining quantities. (See reference 1.)
PANMOM	Computes panel moment matricies
PDQSUB	Computes the doublet and source inner spline matricies for a subpanel. It is used to compute panel defining quantities.
RACOF	Computes source inner spline matricies. It is used for computing near field panel defining quantities.
RCSLOC	Defines a subpanel reference to local coordinate system transformation
SD2LIN	Computes a transformation of source design splines from a representation based on center and edge midpoints to one based on center and corner points
TCOF	Computes the coefficients of the linear, quadratic and cubic basis functions on a triangle. It is used for generating panel defining quantities.
UNIPAN	Transforms the representation of a position vector from universal to panel coordinates
UVCALC	Computes additional panel data
XBPOSH	Performs a projection of points onto a plane in scaled coordinates and applies an origin shift
XBPROJ	Performs a projection of a point onto a plane in scaled coordinates
XCOF	Computes the quasi-far-field doublet spline matricies. It is used for panel defining quantities.
ZCADJ	Withdraws edge control points

Table 13.4 - General Mathematical Routines

SUBROUTINE	OPERATION
AMCON	Defines constants to maximum machine accuracy
CODIM	Interpolates using a controled deviation method
DECOM	Decomposes a square matrix into lower and upper triangular matricies with partial pivoting and row equilibration
FBSUBM	Solves a matrix equation by forward and backward substitutions using the lower and upper triangular decomposition of the matrix
FSHELL	Sorts a real array using the shell sort algorithm. It keeps track of the original order of the array.
GLESOM	Solves a linear system of equations by calling DECOM and FBSUBM
ISHELL	Sorts an integer array using the shell sort algorithm. It keeps track of the original order of the array.
JORDAN	Inverts a matrix
KEYSRT	Arranges elements in an array to bring it into correspondence with an array that has been sorted
LCHVAR	Performs a linear change of variables. It is used in panel moment calculations.
SHLSRT	Sorts an integer array using the shell sort algorithm.
SORTAK	Sorts an integer array. It performs the same function as ISHELL.
SRCHOL	Searches an ordered list for an entry
UKYSRT	Arranges elements in an array to bring it into its original order after it has been sorted
ZWINDG	Computes the product of an array of complex numbers keeping track of the quadrant

Table 13.5 - Constrained Quadratic Least Squares Fit Routines

SUBROUTINE	FUNCTION
CQLSF	Sets up arrays for constrained and least squares fit, calls LSQSFX and unpacks solution.
DCBHTX	Performs a Householder Q-R factorization of the least squares part of the matrix.
DEFVEC	Defines functional form of the fit (polynomial in two dimensions of order one, bilinear or quadratic).
LSQSFX	Constructs L-U factorization of constrained part of fit and calls DCBHTX and PSINIP to solve the least squares part of problem.
PSINTP	Constructs the transpose of the pseudo-inverse of the least squares part of the fit using the Q-R factorization performed by DCBHTX.

Table 13.6 - Blank Common Management Routines

SUBROUTINE	FUNCTION
DELETR	Eliminates an array from blank common and compresses storage in blank common.
INITIR	Initializes storage parameters for a new array in blank common.
LOCATR	Returns current size, type and location of an array in blank common.
STARTR	Initializes storage scheme, creates an array catalog, sets limit for maximum number of arrays and determines maximum storage available.
REQFL	Checks that the current scratch memory request can reside within blank common

Table 13.7 - Special Purpose SDMS-Related Routines

SUBROUTINE	FUNCTION
CHPADB	This FORTRAN subroutine obtains the names of the required database files from the MEC database, opens the database, and (if the database is not a newly generated one) checks that the database is complete. This subroutine is used by all PAN AIR Modules to define new databases and to check the status of those databases generated by upstream modules which are required as input.
MULTI	This subroutine performs an out-of-core matrix multiplication between the matrices stored in SDMS datasets. The resulting matrix is stored as another SDMS dataset.
PAOPEN	Open a temporary or permanent SDMS database.
PACLOS	Close a temporary or permanent SDMS database.

Table 13.8 - Real Matrix Solver Routines

PAC	RDPIV
RDSMR	REDUCR
RMSBS	RMSCBS
RMSCFS	RMSD
RMSDC	RMSERA
RMSERG	RMSFB
RMSFS	RMSLBS
RMSLTS	RMSLUS
RMSRDB	RMSRED
RMSUBS	RMSXCH
RMSXCS	UNPAC
WTPIV	WTSMR

Table 13.9 - Free Field Format Input Routines

SUBROUTINE	OPERATION
BIT$LGN	Aligns one bit string with another. It is written in CAL.
BIT$LOC	Performs bit string location calculations. It is written in CAL.
BIT$MSK	Generates a variable length bit string mask. It is written in CAL.
CHKEOR	Checks an input record for an end-of-record delimiter. It is called by LODREC.
DCODIR	Sets error flags. It is called by LODREC.
GETT	Extracts a character from a string and places it, left-adjusted and blank-filled in another
INCBCD	Increments the numerical portion of a left justified BCD (Binary Coded Decimal) number
LODREC	Reads and decodes an input record
PUTT	Takes one left most character of a word and inserts it into a string
STRMOV	Moves a specified number of characters from one word to another

Table 13.10 - Miscellaneous Routines

SUBROUTINE	OPERATION
ABTJOB	Aborts the job and initiates a traceback
BKMOVE	Transfers one matrix to another
CSTPRT	Prints the cumulative CPU time since the last call
ERRMSG	Writes an error message
LOCF	Interfaces the call to the intrinsic LOC function
OUTLIN	Prints an intermediate output line of at most ten values
OUTMAT	Prints an intermediate output matrix
OUTMXV	Prints an intermediate output vector with up to ten values per line
OUTVEC	Prints an intermediate output vector with one value per line
PIW4	Packs four words into one word
PRGBEG	Indicates the start of a PAN AIR module
PRGEND	Indicates the completion of a PAN AIR module
REMARKF	Suppresses the logfile messages from the random I/O package
SHFTIC	Translates influence coefficients by a shift of origin
SYSTEMC	Acts as an interface to future error handling routines
UABEND	Aborts the job
UNPIW4	Unpacks one word into four words. It is the inverse of PIW4
XFERA	Transfers one array to another

Table 13.11 - Data Input Processing Support Routines

SUBROUTINE	FUNCTION
ADJCHK	Check the user specified edge control point locations to ensure that there are two edges with control points, the edges must be adjacent in order to have a control point at one corner of the network.
BALIND	Process the balance of an indexed input record once the index or indices have been evaluated into row and column ranges. Thus for a record like (1, 4 to 5) = .5 this routine checks for the right paren, the equal sign and loads the .5 into a temporary buffer for smearing.
BDTERM	Process the TERM records for the following network data sets: Closure edge boundary condition boundary set; Coefficients of general boundary condition equation data set; Tangent vectors for design data set; Specified flow data set; and Local incremental onset flow data set. Examples: TERM = AU / A closure term. TERM = AD1,CA2 / A pair of coefficient terms */ of equal value. TERM = TA1 / A tangent term. TERM = 1 / Specified flow equation number TERM = VWYZ / Local incremental onset flow option. Note: a record which begins with */ is for comment only
COLIND	Process the column index or range of indices for the indexed input option for control point data values. Examples: (row , 1) = value / Column 1 (row , 2 TO 4) = value / Columns 2 thru 4 (row , ALL) = value / Columns 1 thru max (row , 4 TO MAX) = value / Columns 4 thru max This routine is called after the row index or range of indices have been decoded.
COMP	Process the computation option for pressures record. Examples: COMP = UNIF / Compute presures from uniform set COMP = LOCA / Compute pressures from local onset flow COMP = COMP / Compute pressures from compressibility Global record default: COMP=UNIF
CUSCOE	Write defaulted general B.C. coefficient term datasets to data base. Check user inputs to verify that user has not tried to define input for these terms.

Table 13.11 - (Continued)

SUBROUTINE	FUNCTION
EDGE	Process the parameter list for the following network data record types: Edge control point locations = type(s), edge-number(s) Closure edge condition = type, edge-number Examples: EDGE = SNE, 1, 2, DNE, 3, 4 CLOS = DNE, 1
EXPIND	Expand a dataset which is homogenous (same value for all control points) and contains only data for a single representative control point to a dataset which may be heterogenous (every control point with one user specified value) and contains data for each control point. This step preceeds the updating of a homogenous dataset with indexed input.
FILIND	Fill the portion of the network defined by the indexed input option of a value(s) record for network data sets as follows: Closure; Coefficients of general B.C. equation; Tangent vectors; Specified flow; and Local incremental onset flow. The value(s) is uniform for all points and is either a single value (closure, coefficient or specified flow) or a triplet (tangent vector, local incremental onset flow). Each column number represents a separate dataset for the data base.
FMPDCK	Check the combined inputs for printout and data base output requests by user.
FMPRDA	Process the forces and moments printout and data base records.
FMREPA	Process global and local reference parameter records from the forces and moments data subgroup of the flow properties data group.
FPCASE	Process flow properties case names. New case names must be alphanumeric (1 to 20 characters). Old case names may be alphanumeric or integer (input order no). Old case names can only be referenced if the post solution update option is set to update. In surface flow properties the case name appears as the parameter list on the surface flow properties record. In forces and moments the case name appears as the parameter list on the case record.
FPDAWR	Write a flow properties calculation problem to the DIP database.

Table 13.11 - (Continued)

SUBROUTINE	FUNCTION
FPVALU	Process any record which has a single floating point value for a parameter list. Examples: GEOM = E-3 / Geometric edge matching tolerance - .001 RATI = 1.5 / Ratio for computation of pressures = 1.5 TRIA = E-6 / Triangular panel tolerance = .000001
IMAGED	Process the images in a parameter list for the network and image selection records in the flow properties calculations data group.
INDXED	Process the indexed input option of a value(s) record for network data sets as follows: Closure; Coefficients of general B.C. equation; Tangent vectors; Specified flow; and Local incremental onset flow. The value(s) is uniform for all points and is either a single value (closure, coefficient or specified flow) or a triplet (tangent vector, local incremental onset flow) Example: (row , column) = value(s)
INPUIM	Process the input-images records for network data in the specified flow data set and the local incremental onset flow data set. Examples: INPUT-IMAGES = INPUT INPUT-IMAGES = INPUT, 1ST, 2ND, 3RD
LHSTST	Test left hand side network constraint data to verify that each data set covers all solutions
NBDORT	Check the order of data records for the following groups Closure; Coefficients; Local incremental onset flows; Specified flows; and Tangent vectors.
NEDATA	Process the values for the following network data set terms: Coefficients of general boundary condition equation Tangent vectors for design; Specified flow; Local incremental onset flow; The values may appear as floating point data or as indexed input. Indexed input starts with a left paren as follows: (row , column) = value

Table 13.11 - (Continued)

SUBROUTINE	FUNCTION
NEPOIN	Process the points (control point locations) record for the following network data set terms: Coefficients for general boundary condition equation; Tangent vectors for design; Specified flow; and Local incremental onset flow; Examples: POINTS = CENTER POIN = EDGE POIN = ADDITIONAL POIN = ALL
NETWIM	Process network and image selection records in the flow properties data group Examples: NETWORK-IMAGES = WING-A, INPUT 1ST, + = WING-B, REVERSE + = WING-C, 1ST NETW = BODY-1 = BODY-2 = BODY-3 / IMAGES DEFAULTED TO INPUT, */ ORIENTATION DEFAULTED TO RETAIN Note: a record which begins with a */ is for comment only.
NORSUB	Normalize a triplet of X,Y,Z direction numbers and also return the magnitude for error testing (RSQ = Zero).
PLAN	Process either of the following two record types: Plane of symmetry deletion flag Abutments in planes of symmetry Examples: DELETE REFLECTION IN PLANE OF SYMMETRY = FIRST-PLANE DELE = SECOND-PLANE PLANE OF SYMMETRY = FIRST-PLANE-OF-SYMMETRY PLAN = FIRS PLAN = SECO PLAN = BOTH
PPCASE	Process the case list found on case records for PPP Point and Configuration Data.
PPPNET	Process the network IDs found in the parameter lists of network and surface records for PPP data.
PPPORT	Read and check the order of data records for the following DIP data subgroups for PPP: GEOMETRY - DQG POINT - PDP CONFIGURATION - CDP Determine when datasets are complete and write them on data base.

Table 13.11 - (Continued)

SUBROUTINE	FUNCTION
PRES	Process the pressure coefficent rule record. Examples: PRES = ISENTROPIC,LINEAR,SECOND-ORDER,REDUCED-SECOND-ORDER, + SLENDER-BODY / THIS CONTAINS ALL OPTIONAL TYPES. PRES=ISEN PRES=ISEN,LINE May be called from GLOBDP or any of the point data post solution processors.
REFE	Process the reference velocity for pressure record. Examples: REFE = 1.3,1.29,1.31,1.32,1.33 REFERENCE VELOCITY FOR PRESSURE = 1.305
RHSPRG	Purge right hand side (RHS) terms from data for current network.
ROWIND	Process the row index or range of indices for the indexed input option for control point data values. Examples are (1, col) = value / ROW 1 (2 TO 4 , col) = value / ROWS 2 THRU 4 (ALL , col) = value / ROWS 2 THRU MAX (4 TO MAX , col) = value / ROWS 4 THRU MAX This routine is called only after it has been determined that a data value record starts with a left paren.
SELE	Process the selection of velocity computation record. Examples: SELE = BOUN / BOUNDARY CONDITION METHOD SELE = VIC- / VIC-LAMBDA (VIC DOTTED WITH LAMBDA)
SETFLG	Process records which require an on/off flag to be set. (1.EQ.ON) Example record type: STORE LOCAL ONSET FLOWS STORE VIC MATRIX
SFOUCL	Load the output array with pressure coefficient rule data and velocity corrections data. The output array becomes the list of print/data base output requests for current surface flow properties calculation.
SFOUIC	Load the individual parameter list options encountered on the printout and data base records for surface flow properties calculations.
SFOULD	Process the parameter list for both the printout record and the data base record, within the surface flow properties data subgroup.

Table 13.11 - (Concluded)

SUBROUTINE	FUNCTION
SFPRDB	Process printout options data set and data base options data set from the surface flow properties data subgroup. Examples: PRINTOUT = 13 VELOCITY CORRECTIONS = NONE, SA1, SA2 PRESSURE COEFFICIENT RULES = ISEN, LINE, SECO, RUDU, SLEN, DATA BASE = ALL VELO = NONE PRES = ISENTROPIC, SLENDER-BODY
SMRIND	Transfer data from the temporary buffer for smear values to the output buffer for control point values. Then load the data onto the data base. Network data sets processed are defined by the MAPID contents as follows: 10HDIP-CLOSUR = Closure edge B.C. data 10HDIP-COEFBC = Coefficients of general B.C. equation 10HDIP-LOCFLO = Local incremental onset flow 10HDIP-SPCFLO = Specified flow 10HDIP-TANVEC = Tangent vectors for design
SOLSFP	Process solutions lists records. Examples: SOLUTIONS = 1, 2, SOLUTION-ID-3, 6 SOLU=1,2,3,6
SURF	Process the surface selection record. Examples: SURF=UPPER,LOWER,UPLO(UPPER-MINUS-LOWER) SURF=LOUP(LOWER-MINUS-UPPER),AVERAGE SURF=UPPE,LOWE,UPLO,LOUP,AVER / ALL OPTIONS Global default: SURF=UPPE
VALUE	Process the array of data values record(s) for network data sets as follows: Closure; Coefficients of general B.C. equation; Tangent vectors; Specified flow; and Local incremental onset flow vectors. The array shall contain data (one value for non-vectors, three values for vectors) for each control point as defined by the current control points location option.
VELO	Process the velocity corrections record. Examples: VELOCITY CORRECTIONS= NONE(NO-CORRECTION) VELO=NONE,SA1(1ST-STAGNATION-TO-AMBIENT) VELO=NONE,SA1,SA2 / ALL OPTIONS

APPENDIX 13-A TREE STRUCTURE

The tree structure of selected routines in the library is presented in this appendix. Many tasks requested by the PAN AIR modules are performed by groups of routines in the library. The purpose of this appendix is to show the calling relationships between those routines. A routine may be shown as the trunk of its own tree or a one of the branches of another tree. Routines which perform a single function and do not use other library routines are not shown in this appendix.

```
        +-GPHPLK----UKYSRT
        I
        I         +-FSHELL
        I         I-ISHELL
        I-GPHSCN--I-KEYSRT
        I         I-UKYSRT
        I         +-XFERA
        I
        I-GPLUCK----UKYSRT
        I-OUTMAT
ABTINT--I
        I-OUTMXV----OUTLIN
        I
        I-OUTVEC
        I-XFERA
        +-ZERO


        +-ABTJOB
CCALN---I-LOCF
        I
        I         +-LCHVAR
        +-PANMOM--I
                  +-ZERO


        +-DEFVEC
        I
        I         +-DCBHTX----VIP
        I         I-PSINTP----VIP
CQLSF---I-LSQSFX--I
        I         I-VIP
        I         +-VIPS
        I
        +-ZERO


        +-DECOM-----AMCON
GLESOM--I-FBSUBM
        +-VIPDA


        +-CAB
        I-CAMB
IMAGE---I
        I-CAPB
        +-XFERA
```

```
        +-CROSS
        I-RRZAB
INSIDE--I
        I-VADD
        +-XFERA


ISCAL-----PIDENT


                  +-GETT
        +-CHKEOR--I
        I         +-PUTT
        I-DCODIR
        I-GETT
        I
        I         +-GETT
LODREC--I-INCBCD--I         +-BIT$LGN
        I         +-STRMOV--I-BIT$LOC
        I                   +-BIT$MSK
        I-LOCF
        I         +-BIT$LGN
        I-STRMOV--I-BIT$LOC
        I         +-BIT$MSK
        I
        +-SYSTEMC


        +-CMAB
        I
        I         +-CMAB
        I-PROJ----I
        I         +-VADD
        I
LPROJ---I-UVECT
        I-VADD
        +-VMAG


                  +-LOCATR
        +-DELETR--I
        I         +-XFERA
        I
        I                   +-LOCATR
        I         +-DELETR--I
        I         I         +-XFERA
MULTI---I-INITIR--I-LOCATR
        I         +-LOCF
        I
        I-LOCF
        I
        I-REQFL----LOCF
        +-VIP
```

```
        +-CROSS
NORCAL--I-UVECT
        +-VADD


        +-CROSS
        I
        I          +-CROSS
        I          I-RRZAB
        I-INSIDE--I
        I          I-VADD
        I          +-XFERA
        I
        I-MUL3X3
        I
        I          +-VADD
        I-NRPTED--I-VIP
        I          +-XFERA
        I
NRPTHP--I-RRZAB
        I-RRZATB
        I-VADD
        I-VIP
        +-XFERA


        +-LCHVAR
PANMOM--I
        +-ZERO


        +-RRZXYT
        I-UNIPAN----CMAB
PDQSUB--I
        I-VMUL
        +-ZERO


        +-VIP
RACOF---I
        +-ZERO
```

```
                    +-RDPIV
                    I-RDSMR
                    I-RMSERG
                    I-RMSLBS
                    I-RMSLUS
                    I
                    I         +-REDUCR
          +-RMSDC---I-RMSRED--I-RMSERG
          I         I         +-RMSLTS----VIPS
          I         I
          I         I-RMSUBS
          I         I-RMSXCH----UNPAC
          I         I-RMSXCS----PAC
          I         I-WTPIV
          I         +-WTSMR
RMSD----I
          I         +-RDSMR
          +-RMSERA--I
                    +-RMSERG


                    +-RDPIV
                    I-RDSMR
                    I-RMSCBS
          +-RMSBS---I-RMSRDB----REDUCR
          I         I-RMSXCH----UNPAC
          I         I-RMSXCS----PAC
          I         +-WTSMR
RMSFB---I
          I         +-RDSMR
          I         I-REDUCR
          +-RMSFS---I
                    I-RMSCFS
                    +-WTSMR


          +-MUL3X3
          I-RRZAB
SD2LIN--I-XBPOSH
          I-XFERA
          +-ZERO


                              +-LOCATR
                    +-DELETR--I
                    I         +-XFERA
          +-INITIR--I-LOCATR
          I         +-LOCF
STARTR--I-LOCF
          I
          +-REQFL-----LOCF

XCOF------MUL3X3
```

Appendix 13-B CONSTRAINED QUADRATIC LEAST SQUARES FIT SUBROUTINES

13-B.1 CONSTRAINED LEAST SQUARES FIT SUBROUTINES

The PAN AIR Theory Document, Sec. I.5 (Ref. 1), discusses the theoretical basis of the constrained least squares procedure. In this Appendix we discuss the realization of the theory in a set of FORTRAN subroutines. The set of subroutines and their interrelationships are indicated in Figure 13-B.1.

13-B.2 Subroutines CQLSF and DEFVEC

Information is passed to CQLSF through its formal arguments concerning the number of points to be fit, a two dimensional coordinate for each point, a weight for each point and the order of the fit desired (linear, bilinear or quadratic).

Any point with an input weight greater than 1.0 is treated as an exact constraint. Any point with a negative weight is treated as a least squares part of the fit with a weight equal to the absolute value of its weight.

Subroutine DEFVEC defines a vector of polynomials in the two dimensional coordinates of the points according to the order of the fit. The vectors are:

$(1,\ x,\ y)$	Linear
$(1,\ x,\ y,\ xy)$	Bilinear
$(1,\ x,\ y,\ 1/2x^2,\ xy,\ 1/2y^2)$	Quadratic

These are defined for each point in the fit. In CQSLF the vectors are scaled by the weights for each point and are stored in the matrix A. This matrix is defined as a one dimensional array. The rows of the matrix which correspond to the exact constraints occur at the beginning of the matrix and those that correspond to the least squares constraints occur at the end of the array. Subroutine LSQSFX then solves for the set of equations and returns the transpose of the solution (see below). CQLSF unpacks the solution array and generates the solution matrix for the problem. A check on round-off errors is made by computing the identity matrix minus the product of the solution matrix and the fit matrix. The quadratic norm of this matrix is computed. If it is greater than 10^{-6} the fit is declared to be singular. If it is greater than 10^{-12} the fit is declared to be poor. CQSLF then copies the solution matrix into appropriate locations in the output array SOLMAT according to whether the full matrix is desired (e.g. for the panel subspline in the sixth overlay of DQG) or whether only the constant coefficient term of the fit is required (e.g. as in the computation of spline vectors in the fifth overlays of DQG and MDG).

13-B.2 Some Theoretical Remarks

The general form of the equations solved by LSQSFX is

$$[A]\,[x] = [b]$$

where A is a matrix which contains some equations which are to be solved exactly and some which are to be solved in a least squares sense. This can be written as

$$\left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right] \left[\begin{array}{c} x_1 \\ \hline x_2 \end{array}\right] = \left[\begin{array}{c} b_E \\ \hline b_L \end{array}\right]$$

These can be separated into two sets of equations

$$\left[\begin{array}{cc} A_{11} & A_{12} \\ \hline 0 & 0 \end{array}\right] \left[\begin{array}{c} x_1 \\ \hline x_2 \end{array}\right] = \left[\begin{array}{c} b_E \\ \hline 0 \end{array}\right]$$

$$\left[\begin{array}{cc} 0 & 0 \\ \hline A_{21} & A_{22} \end{array}\right] \left[\begin{array}{c} x_1 \\ \hline x_2 \end{array}\right] \cong \left[\begin{array}{c} 0 \\ \hline b_L \end{array}\right]$$

If the submatrix A_{11} is invertable then

$$\left[\begin{array}{c} x_1 \\ \hline 0 \end{array}\right] = \left[\begin{array}{c|c} A_{11}^{-1} & 0 \\ \hline 0 & 0 \end{array}\right] \left[\begin{array}{c} b_E \\ \hline 0 \end{array}\right] - \left[\begin{array}{c|c} A_{11}^{-1} & 0 \\ \hline 0 & 0 \end{array}\right] \left[\begin{array}{c|c} 0 & A_{12} \\ \hline 0 & 0 \end{array}\right] \left[\begin{array}{c} 0 \\ \hline x_2 \end{array}\right]$$

$$= \left[\begin{array}{c|c} A_{11}^{-1} & 0 \\ \hline 0 & 0 \end{array}\right] \left[\begin{array}{c} b_E \\ \hline 0 \end{array}\right] - \left[\begin{array}{c|c} 0 & A_{11}^{-1} A_{12} \\ \hline 0 & 0 \end{array}\right] \left[\begin{array}{c} 0 \\ \hline x_2 \end{array}\right]$$

We can substitute this into the expression for x_2 and rearrange terms to find

$$\left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & A_{22} - A_{21} A_{11}^{-1} A_{12} \end{array}\right] \left[\begin{array}{c} 0 \\ \hline x_2 \end{array}\right] = \left[\begin{array}{c|c} 0 & 0 \\ \hline A_{21} A_{11}^{-1} & I \end{array}\right] \left[\begin{array}{c} b_E \\ \hline b_L \end{array}\right]$$

$$= \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & Z_{22} \end{array}\right] \left[\begin{array}{c} 0 \\ \hline x_2 \end{array}\right]$$

The inverse of Z_{22} (in the least squares sense) is computed to allow a solution for x_2 in terms of b_E and b_L.

This result is substituted for x_2 in the equation for x_1.

$$\begin{bmatrix} x_1 \\ \hline 0 \end{bmatrix} = \left[\begin{array}{c|c} A_{11}^{-1} & 0 \\ \hline 0 & 0 \end{array}\right] \begin{bmatrix} b_E \\ \hline 0 \end{bmatrix} - \left[\begin{array}{c|c} 0 & A_{11}^{-1} A_{12} \\ \hline 0 & 0 \end{array}\right] \begin{bmatrix} 0 \\ \hline x_2 \end{bmatrix}$$

$$= \left[\begin{array}{c|c} A_{11}^{-1} + A_{11}^{-1} A_{12} Z_{22}^{-1} A_{21} A_{11}^{-1} & -A_{11}^{-1} A_{12} Z_{22}^{-1} \\ \hline 0 & 0 \end{array}\right] \begin{bmatrix} b_E \\ \hline b_L \end{bmatrix}$$

The complete solution is then

$$\begin{bmatrix} x_1 \\ \hline x_2 \end{bmatrix} = \left[\begin{array}{c|c} A_{11}^{-1} + A_{11}^{-1} A_{12} Z_{22}^{-1} A_{21} A_{11}^{-1} & -A_{11}^{-1} A_{12} Z_{22}^{-1} \\ \hline -Z_{22}^{-1} A_{21} A_{11}^{-1} & Z_{22}^{-1} \end{array}\right] \begin{bmatrix} b_E \\ \hline b_L \end{bmatrix}$$

where $Z_{22} = A_{22} - A_{21} A_{11}^{-1} A_{12}$ and Z_{22}^{-1} is the least squares pseudo-inverse of Z_{22}. The transpose of this matrix is returned to LSQSFX.

13-B.3 Subroutine LSQSFX

Subroutine LSQSFX and the two routines it calls (DCBHTX and PSINTP) all make use of a particularly dense style of coding (See Reference 6) which is intended to minimize both storage space and execution time. For this reason they may be difficult to unravel. In this section we give an outline of the operations of the subroutine LSQSFX without discussing the details of data storage. If this information is required it will have to be found through careful study of the code.

The matrix of equations which are to be solved may be divided into four quadrants distinguished by the exact constraints and the least squares constraints. The first operation LSQSFX performs is to do an L/U decomposition of the exact constraints quadrant. This operation is carried

out in the section of the subroutine marked phase (1,A). Simultaneously in phase (1,A) the matrix Z_{22} (see figure 13-B.2) is constructed in the lower right quarter of the matrix.

Then, in DCBHTX, a Householder QR factorization of Z_{22}, the least squares part of the matrix is performed. This is phase (1,B) of LSQSFX.

In phase (2,A) the inverse of the L_{11} and U_{11} matrices are computed and are applied to the U_{12} and L_{21} submatrices. In phase (2,B) the L_{11}^{-1} and U_{11}^{-1} matrix product is computed and the submatrices A_{11}^{-1}, $A_{11}^{-1} A_{12}$ and $A_{21} A_{11}^{-1}$ are computed, and the transpose of the pseudo-inverse of the matrix Z_{22} is computed in PSINTP.

Finally in phase 3 the pieces of the matrix are assembled to form the transpose of the solution matrix defined at the end of section 13-B.2. Figure 13-B.2 summarizes the operations.

13-B.4 Subroutine DCBHTX

Subroutine DCBHTX decomposes the least squares portion of the matrix $Z_{22} = A_{22} - A_{21} A_{11}^{-1} A_{12}$ into a product of unitary matrices Q and an upper triangular matrix R

$$Z_{22} = QR$$

The matrix Q is not computed explicitly. Rather since it is a product of elementary reflections

$$Q = H_1 \, H_2 \cdots H_n$$

with

$$H_k = I - B_k \, W_k \, W_k^T$$

and

$$B_k = -1/[d_k (W_k)_k]; \quad d_k = -\operatorname{sgn}(a_{kk}^{(k)}) \left[\sum_{i=k}^{m} (a_{ik}^{(k)})^2 \right]^{1/2}$$

$$W_k = [a_{ik}^{(k)}; \quad i=k,\ldots,m] - d_k [\delta_{ik}; \quad i=k,\ldots,m]^{i=k}$$

just the numbers d_k and the vectors W_k are stored in a packed form. In the above expression $a_{ik}^{(k)}$ denotes the entries of the partially factored Z_{22} matrix at the beginning of the kth stage of factorization.

13-B.5 Subroutine PSINTP

Subroutine PSINTP computes the least squares pseudo-inverse of the matrix Z_{22} as follows. Let

$$Z_{22} = QR = \left[Q_1 \;\vdots\; Q_2 \right] \begin{bmatrix} R \\ \hline 0 \end{bmatrix}$$

$$= \left[Q_1 \;\vdots\; 0 \right] \begin{bmatrix} R \\ \hline 0 \end{bmatrix}$$

Then the pseudo-inverse of Z_{22} is given by

$$Z_{22}^{-1} = R^{-1} \, Q_1^{\,T}$$

Since the transpose of Z_{22}^{-1} is comformable with Z_{22}, PSINTP actually computes the matrix $(Z_{22}^{-1})^T = Q_1 (R^{-1})^T$.

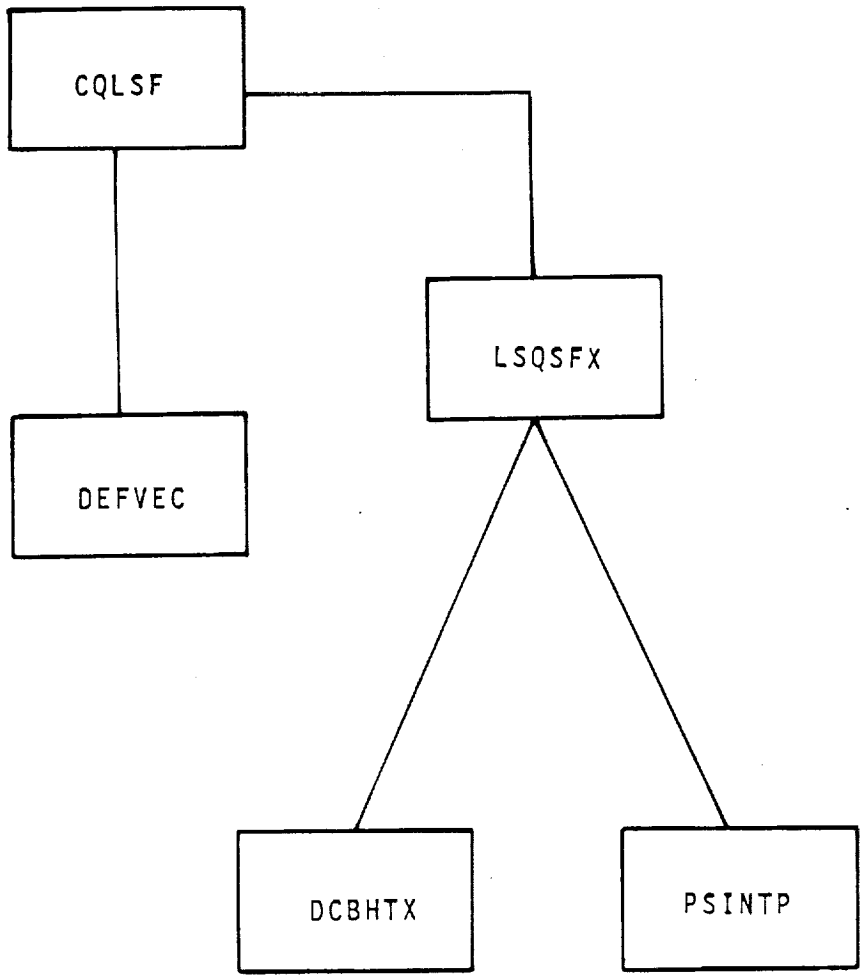


Figure 13-B.1 Tree Structure of Constrained Least Squares Subroutines

$$A = \begin{array}{c} \text{Exact} \\ \\ \text{LSQ} \end{array} \left[\begin{array}{c|c} \text{Exact} & \text{LSQ} \\ A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

$$\xrightarrow[\text{Phase (1,A)}]{\text{L/U Factorization}} \left[\begin{array}{c|c} L_{11} & 0 \\ \hline L_{21} & I \end{array}\right] \left[\begin{array}{c|c} U_{11} & U_{12} \\ \hline 0 & Z_{22} \end{array}\right]$$

$$Z_{22} = A_{22} - A_{21}\, A_{11}^{-1}\, A_{12}$$

$$\xrightarrow[\text{Phase (1,B)}]{\text{QR Factorization}} \left[\begin{array}{c|c} L_{11} & 0 \\ \hline L_{21} & I \end{array}\right] \left[\begin{array}{c|c} U_{11} & U_{12} \\ \hline 0 & Q \end{array}\right] \left[\begin{array}{c|c} I & 0 \\ \hline 0 & R \end{array}\right]$$

DCBHTX

$$\xrightarrow[\text{Phase (2,A)}]{\text{Back Solve}} \left[\begin{array}{c|c} L_{11}^{-1} & 0 \\ \hline L_{21}\, L_{11}^{-1} & I \end{array}\right] \left[\begin{array}{c|c} U_{11}^{-1} & U_{11}^{-1}\, U_{12} \\ \hline 0 & Q \end{array}\right] \left[\begin{array}{c|c} I & 0 \\ \hline 0 & R \end{array}\right]$$

$$\xrightarrow[\text{and } (Z_{22}^{-1})^T]{\text{Compute } A_{11}^{-1}} \left[\begin{array}{c|c} A_{11}^{-1} & A_{11}^{-1}\, A_{12} \\ \hline A_{22}\, A_{11}^{-1} & (Z_{22}^{-1})^T \end{array}\right]$$

(PSINTP)

Phase (2,B)

$$\xrightarrow[\text{Phase 3}]{\text{Collect Terms}} \left[\begin{array}{c|c} A_{11}^{-1T} + A_{11}^{-1T}\, A_{21}^{T}\, Z_{22}^{-1T}\, A_{12}^{T}\, A_{11}^{-1T} & A_{11}^{-1T}\, A_{21}^{T}\, Z_{22}^{-1T} \\ \hline -Z_{22}^{-1T}\, A_{12}^{T}\, A_{11}^{-1T} & Z_{22}^{-1T} \end{array}\right]$$

Figure 13-B.2 Outline of Algorithm Implemented in LSQSFX

Section 14 - Scientific Data Management System (SDMS)

The Scientific Data Management System Reference Manual is included verbatim as a section here and, as such, it does not conform to the standards of the PAN AIR Maintenance Document (i.e., page numbering). The manual is written for the CDC (Control Data Corporation) version of SDMS but it may be applied to the CRAY version used by PAN AIR with the following considerations:

1. The user number of a file may be considered the CRAY dataset identification (ID). The job user number may be considered the ownership (OWN) value. Any datasets saved by SDMS will have an identification equal to the ownership value. This includes databases and master definitions. To specify the dataset identification will require additional job control language. The options for SCOPE 2.1 do not apply to the CRAY version.

2. The job control language to access the SDMS library and the Data Definitions Processor is provided on the PAN AIR installation tape.

3. Sequential datasets (described in section 3.5 of the SDMS manual), miscellaneous database functions (section 3.6), indexed sequential datasets (appendix B), and qualified dataset search (appendix C) are not used by PAN AIR.

4. The dynamic storage capability (described in section 3.01 of the SDMS manual) and the trace option (section 5) are not available.

5. The permanent file errors in table 4-3 and table 4-4 do not apply to the CRAY version. Table D-2 (PDD Status) of reference 7 (CRAY-OS Manual) should be used instead. The permanent file error number is labeled in table D-2 as the PMST.

TABLE OF CONTENTS

TABLE OF CONTENTS

1.0 Introduction

This document describes the Scientific Data Management System (SDMS). SDMS provides a high-level, file-independent framework for external data transfers performed by scientific application programs. With SDMS, data is transferred between a program and named scalars and vectors in an external data base. These scalar and vector data elements are grouped into a data hierarchy by means of an external data definition.

The use of named, structured data means that I/O takes place at a higher level of abstraction than files provide, thus simplifying I/O design. The use of an external data definition makes data sharing between programs easier.

1.1 Data Dependence

A model of data flow in a modern system is shown in Figure 1-1. Each program P in the system communicates both with its local data L and with data S which is shared among members of the system.

The use of separate programs to perform sharply-defined functions provides a high degree of modularity. However, the use of file-oriented data transfer methods tends to impede inter-program communication in such a system. File-oriented I/O ignores two of the most important properties of data: identity and form. Record elements have values but no inherent identity, no external names. Similarly files have no inherent form. Although some files are regular in form, many are not, and none are required to be so.

The only identity given to record elements is that of position within the record. A variable stored in word 12 of a record must, of course, be retrieved from word 12. This is at best a weak kind of data identification, and it imposes on the data an ordering which is not intrinsic to it.

1.2 Data Independence

SDMS uses a data definition to define the form and identity of external data. This definition is part of the external data it represents and provides the exclusive means of access to it (Figure 1-2). This combination of definition and data is commonly referred to as a data base. This approach makes it possible for programs to issue data requests of the form "get external data element MACH-NUMBER and store it in program variable XMN".

1.3 Data Base Construction Process

Figure 1-3 shows the basic processes involved in the construction and use of a data base. The form of the data base is specified in the definition text. The definition text is converted by the Data Definition Processor (an SDMS utility) to a master definition file. Program P accesses data base elements by calling on SDMS routines which are loaded with it. These routines use a copy of the master definition to retrieve and store data base values.

1.4 SDMS Features

The important features of SDMS are listed below.

Permanent and Temporary Data Bases

SDMS permits the construction of both temporary and permanent data bases. Temporary data bases can be used to handle both the local data and transient shared data shown in Figure 1-1.

Multiple Data Bases

Several data bases may be accessed by the same program.

Master Definition Concept

A single external definition can provide the form for an arbitrary number of physical data bases.

Random Datasets

SDMS supports the list-directed transfer of element set variables to and from random datasets. A random dataset corresponds to a logical file in a file-oriented system. An element set corresponds to a logical record.

Sequential Datasets

SDMS supports list-directed transfers of element set variables to and from element set sequences. Element set sequences correspond to sequential files. They are grouped into sequential datasets.

Keyed Access

SDMS provides keyed access to random element sets and to element set sequences as well. Multiple keys are permitted.

Data Element Types

A data block consists of a set of data elements. Data element types include scalar, fixed-length vectors, and variable-length vectors. (A vector is a single-dimensional array.)

Data Element Access by Name

Data block elements are selectively accessed by name.

High-Efficiency Transfers

Random element sets can be transferred verbatim between disk and central memory without intermediate buffering.

Availability

SDMS is available to programs coded in Control Data Corporation (CDC) Fortran Extended (FTN) running under the KRONOS/NOS operating systems for CDC lower CYBER computers (6600 like), and the SCOPE 2.1 operating system for CDC 7600 computers.

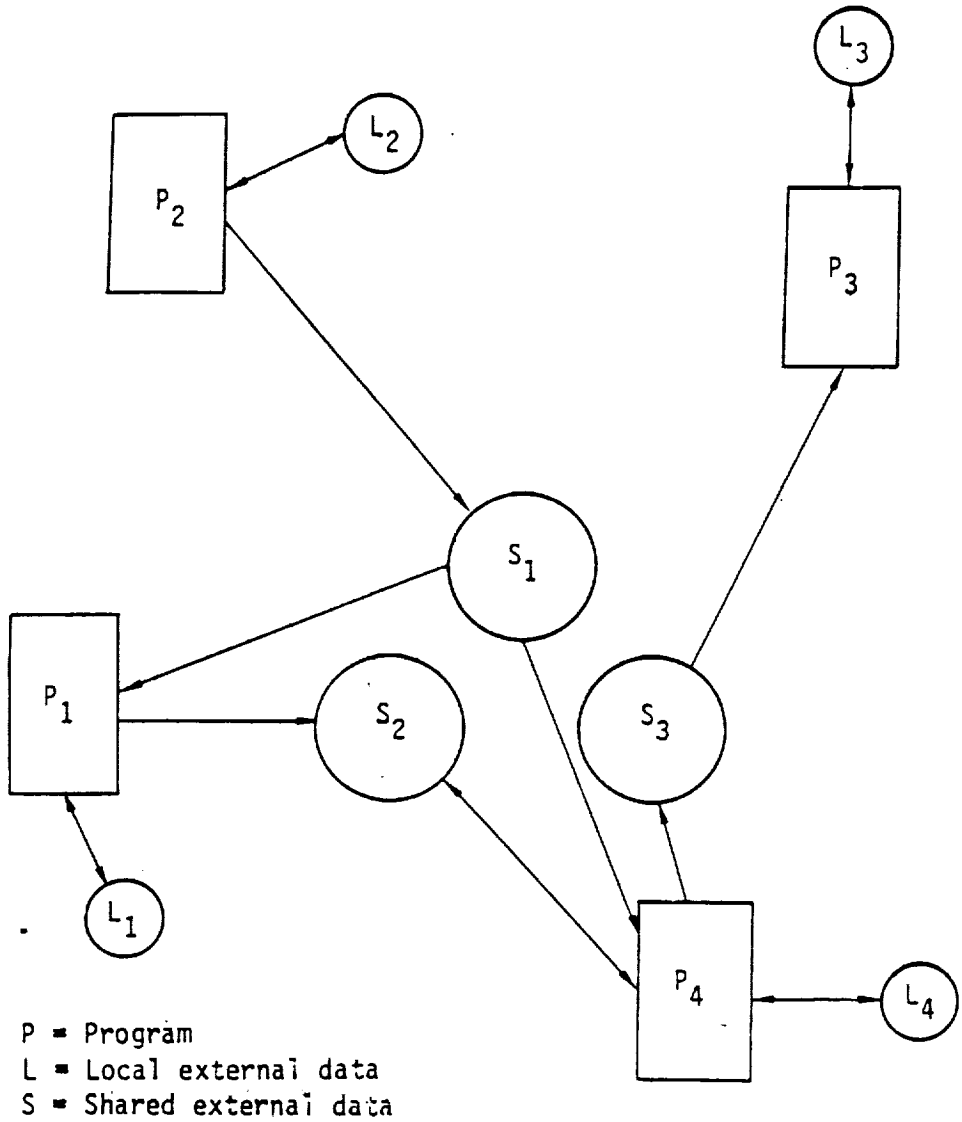


Figure 1-1: Data Communication in Typical Programs

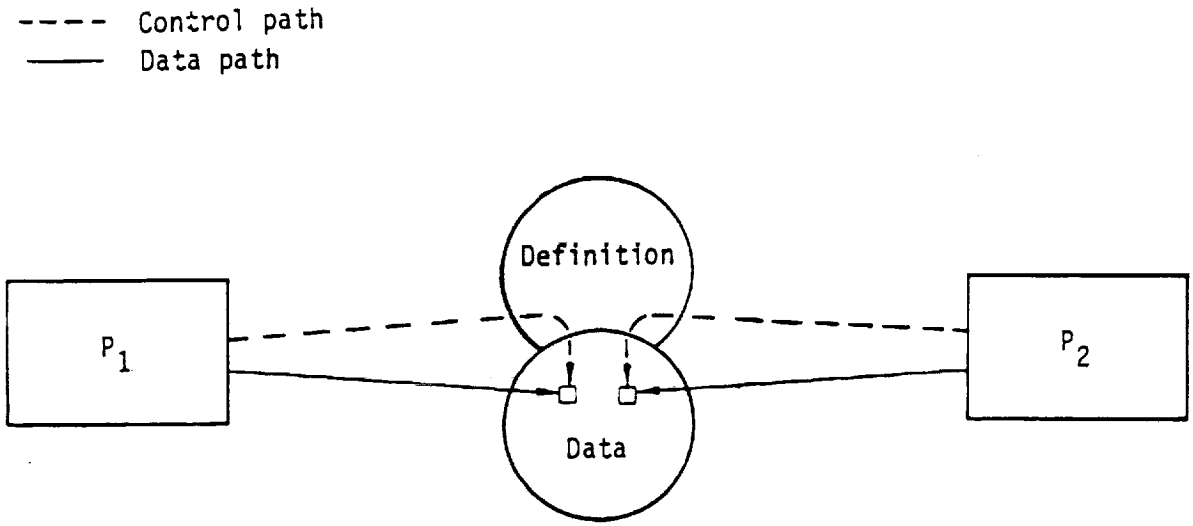


Figure 1-2: Definition Controlled Data Access

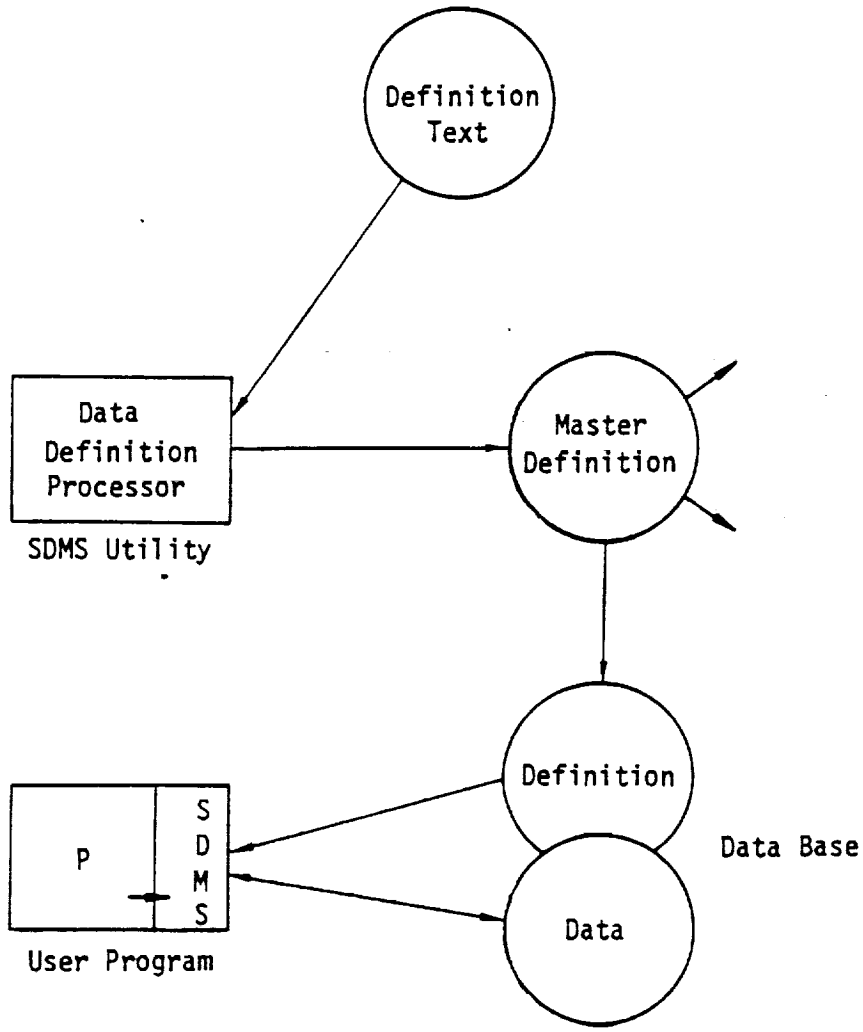


Figure 1-3: Data Base Processing

2.0 Data Base Definition

2.1 SDMS Data Base Fundamentals

Figure 2-1 shows the general form of an SDMS data base. Each SDMS data base consists of two major parts: a data base definition and a collection of random and sequential datasets. The data base definition is a copy of a master definition.

The basic SDMS information unit is the data element. All retrieval and storage of information is done by referencing data elements through names given in their dataset definition. SDMS data elements are well-suited to the expression of scientific data. Data element-forms include scalars, fixed-length vectors and variable-length vectors. Data element values may be integers, floating point numbers, and text strings.

In case of random datasets, data elements are grouped into element sets, each with a key set. The corresponding definition gives the names and attributes of element set keys and data elements. In Figure 2-1, dataset X has a single key K. Each element set has two data elements A and B. Programs store dataset X information by making statements of the form "make a new element set in dataset X such that K has the value K_1 and data element A has the value A_1." Retrieval statements have the form "from an element set in dataset X such that K has the value K_2 , transfer data element B into program area B_2."

In a random dataset, each key set is associated with one element set. In a sequential dataset, each key set is linked to an element set sequence. An element set sequence is the SDMS correspondent to a sequential file. The dataset description gives the names and attributes of sequence key sets and element set data elements.

In dataset Y of Figure 2-1, sequences are keyed by the single key Q. Each element set contains the single data element R. To create a new element set sequence requires that the program issue an SDMS request of the form "open a new sequence L in dataset Y such that Q has the value Q_1." The sequence is built by issuing SDMS requests of the form "add an element set to the end of sequence L in which data element R has the value R(i)."

DEFINITION

```
(Master Definition  Z

  Dataset X

   Key Set
     K
    End

   Element Set
     A
     B
    End

  End Dataset
        •
        •
        •

  Dataset Y

   Key Set
     Q
    End

   Element Set Sequence
     R
    End

  End Dataset

End Definition
```

CONTENT

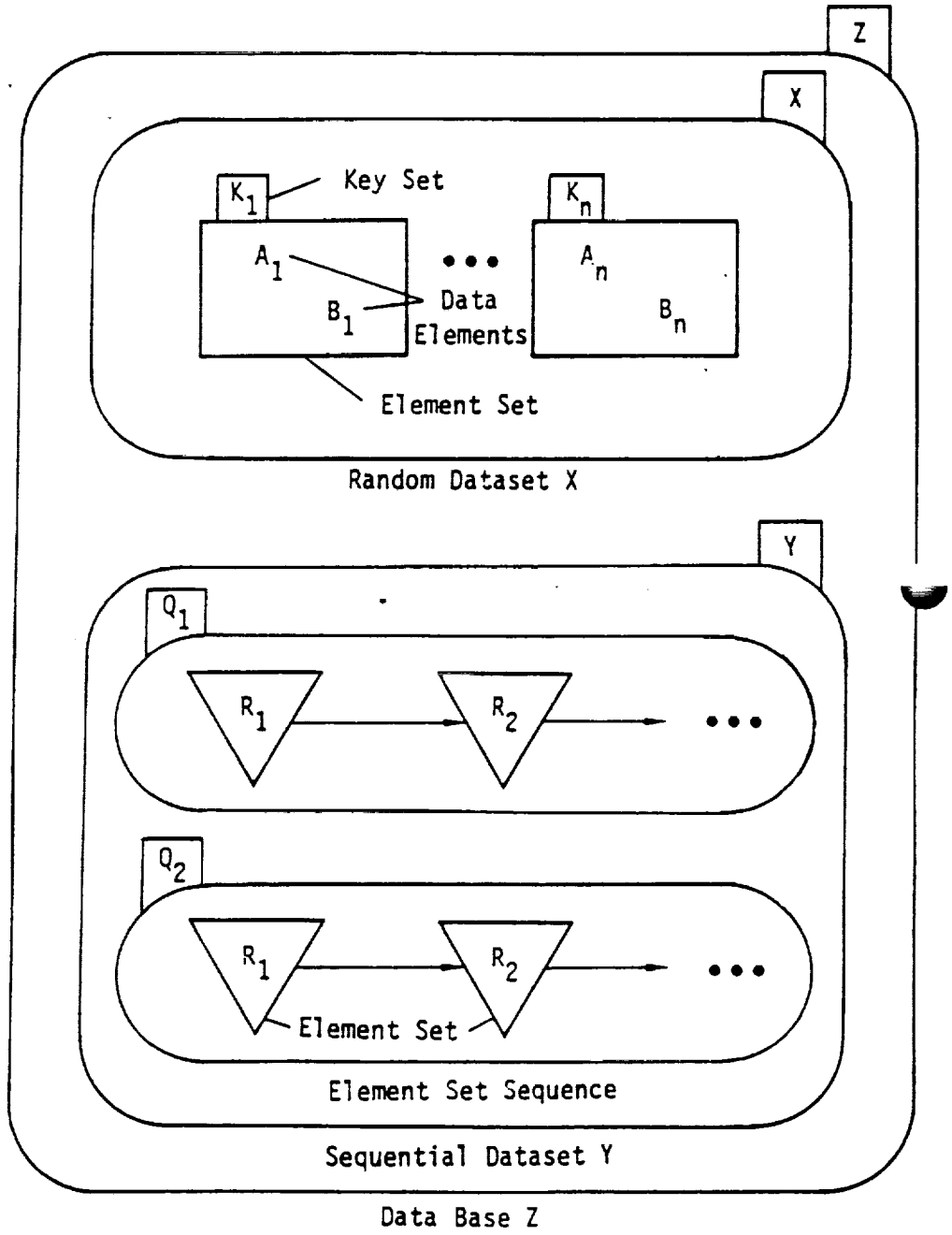


Figure 2-1: A Typical SDMS Data Base

2.2 Master Definition Structure

A master definition consists of a sequence of 80 character text lines which establishes:

1. form of a data base.
2. the permanent file on which the master definition will reside.
3. controls on dataset access within permanent data bases.
4. the names and attributes of dataset elements.

Each definition line consists of a left-to-right sequence of items separated by blanks. The one-character item '$' is reserved as an optional end-of-line terminator which may be followed by comments. Blank lines may be inserted in the definition to aid legibility.

The master definition is constructed using SDDL (Scientific Data Definition Language). In the paragraphs that follow, the syntax of SDDL will be expressed in top-down fashion, using the syntactic constructs defined in Table 2-1. Uppercase text strings denote literal items. Constructs peculiar to the CDC 7600 version (SCOPE 2.1) will be enclosed by double slashes (//).

2.2.1 Master Definition Syntax

The complete master definition has the form

```
MASTER DEFINITION mdname //user-id set-name//

  ----------------------*
     <dataset definition>
  ----------------------

END DEFINITION
```

where mdname is the name of the direct access permanent file to which the master definition is to be written. //User-id and set-name are the SCOPE 2.1 permanent file user-id and set name under which mdname is to be cataloged. For system residence (not private pack), set-name=SYSTEM.//

Note: the asterisk and horizontal lines indicate vertical repetition of the syntactic construct between them.

Syntactic Unit	Definition
<key element>	elname <data type>
<data element>	elname[<subscript>] <data type>
<subscript>	integer/name
<data type>	TEXT/INTEGER/REAL/T/I/R
<option>	READ/WRITE/R/W

Elementary Unit	Definition
mdname	master definition name (1 to 6 characters[1])
sname	dataset name (1 to 20 characters[1]).
user-id	SCOPE 2.1 user-id (1 to 9 characters[1]).
set-name	SCOPE 2.1 set name (1 to 9 characters[1]).
pw	password (1 to 10 characters[1]).
elname	element name (1 to 20 characters[1]).
lname	name of scalar integer <data element>.
TEXT or T	textual type.
INTEGER or I	integer type.
REAL or R	real type.
READ or R	read permission.
WRITE or W	write permission.

Syntactic Constructs	Definition
----------* x ----------	x . . . x
[x]	x is optional.
x/y	x or y
<x>	x is a compound syntactic unit.

Table 2-1: Scientific Data Definition Language (SDDL) Syntax

1. No imbedded blanks permitted.

2.2.2 Dataset Syntax

The syntactic unit <dataset definition> defines a random dataset or a sequential dataset. The unit <dataset definition> expands to

```
DATASET sname [DIRECT]

[<password set>]

[<key set>]

[<dataset body>]

END DATASET
```

where sname is the name of the dataset. The identifier sname can be up to 20 characters in length and may contain any legal FORTRAN character except 'blank'.

The DIRECT option specifies a random dataset which has no <element set> and therefore no structure. Its simplicity permits high-efficiency transfers to take place (3.4.2, 3.4.4, 3.4.6).

2.2.3 Password Set Syntax

The syntactic unit <password set> defines access permissions for this dataset or group in conjunction with permanent data base initialization (3.1.2). The unit <password set> expands to

```
PASSWORDS
--------------- *
  pw  <option>
---------------
END
```

where pw = password (10 characters or less)
<option> = READ or WRITE.

Options may be abbreviated by specifying R instead of READ, and W instead of WRITE.

2.2.4 Key Set Syntax

The syntactic unit <key set> defines the names and attributes of keys which are used to access dataset components. It expands into

```
KEY SET
  -------------------- *
  elname <data type>
  --------------------
END
```

where elname is the name of a scalar data element and <data type> is the element type (in the FORTRAN sense). <data type> = INTEGER, REAL or TEXT where INTEGER (or I) denotes integer type, REAL (or R) denotes real type and TEXT (or T) denotes textual type. Elname can be up to 20 characters in length and may contain any legal character except 'blank'. A key set may contain a maximum of 10 elements.

2.2.5 Dataset Body Syntax

The syntactic unit <dataset body> defines how data elements are arranged within the dataset. It expands to

```
ELEMENT SET [SEQUENCE]

      <element set>

END
```

If the word SEQUENCE is present in the header, a sequential dataset is defined. Its absence indicates a random dataset. In a sequential dataset, keyed access is provided to sequences of element sets. In a random dataset, each element set is key-accessible.

2.2.6 Element Set Syntax

The syntactic unit <element set> defines the data elements present in each element set in a dataset. It expands to

```
-----------------------------*
elname [<subscript>] <data type>
-----------------------------
```

where

elname	=	data element name (≤ 20 characters)
<subscript>	=	integer/lname
<data type>	=	TEXT/INTEGER/REAL/T/I/R
integer	=	natural number ≥ 1
lname	=	name of scalar integer data element in the same element set.

The use of the <subscript> parameter determines data element structure.

<subscript>	element structures
null (not given)	scalar
integer	fixed-length vector
lname	variable-length vector

Data element type is defined by <data type>.

<data type>	element type
TEXT (or T)	text
INTEGER (or I)	integer
REAL (or R)	real

Some data element definitions are illustrated below.

data element	definition		
integer scalar LIST-LENGTH.	LIST-LENGTH		I
text scalar NAME.	NAME		T
5 words of text named TITLE.	TITLE	5	T
3-word floating-point array POINT.	POINT	3	R
floating-point array LIST whose length is equal to the value of LIST-LENGTH.	LIST	LIST-LENGTH	R

2.3 Master Definition Example

The basis for our example is the kind of data which represents a paneled aerodynamic body. Figure 2-2 shows a wing modeled as two networks of panels; one for the wing and one for the wake. Associated with each panel are one or more control points at which boundary conditions are evaluated. An aerodynamic analysis requires that several collections of geometric properties be extracted from the modeled configuration.

Figure 2-3 shows the breakdown which we will use to represent our panelled configuration. A master definition called PANCD which corresponds to this data arrangement is shown in Table 2-2.

The dataset global-data consists of a single element set containing all global information about the configuration. In our example, this consists of the number of networks and a configuration name.

The dataset network-def-data contains information about individual networks, keyed by network number. Note that parameters with a prescribed set of values are fully described by comments.

Network-grid-points is another dataset containing information about individual networks. The data consists of a sequence of grid points in 3-space, each sequence keyed by network number before. The data element name network-no was chosen to be the same in both cases to emphasize that the key is the same in both cases. Uniqueness of reference is preserved by the fact that the data elements are in different datasets. Network grid points are accessed by specifying dataset name and network number and then reading or writing them sequentially.

The dataset panel-def-quant contains a large collection of panel-related data keyed jointly on network number and panel number. The element set includes fixed-length arrays like panel-moments and variable-length arrays like fit-wt-factors with its length- specifying element pnl-srf-fit-ordr.

The last two datasets in our master definition represent the control-point definition data of Figure 2-3. The first dataset, ctl-pt-def-quant, contains an element set sequence for each network. Each element set contains data about a particular control-point in one network. This data includes the coordinates of the control-point and the number of the panel associated with it. The data element panel-no is used as a key into the dataset panel-def-quant.

The second control-point dataset is distinct-ctl-pts, again keyed by network number. Since all control points may not be spatially distinct; the index array ctl-pt-no-lst is used to indicate those control-points which are duplicated.

Figure 2-4 shows how PANCD can serve as the basis for an arbitrary number of PAN AIR data bases.

2.4 Limitations

Item	Limit
Datasets per master definition	100
Passwords per dataset	10
Keys per key set	10
Scalar elements per dataset	100
Fixed-length arrays per dataset	100
Variable-length arrays per dataset	100

2.5 Definition Processing

At the BCS Renton Data Center, the control cards required to create a master definition are:

```
GET,DDP/UN=PAWAM1.

DDP(MDIN,OFIL)
```

where

MDIN=file of master definition text. (default=INPUT)

OFIL=print file. (default=OUTPUT)

Assume MDIN contains a valid master definition named x. At DDP completion,direct access permanent file x will be the corresponding master definition file.

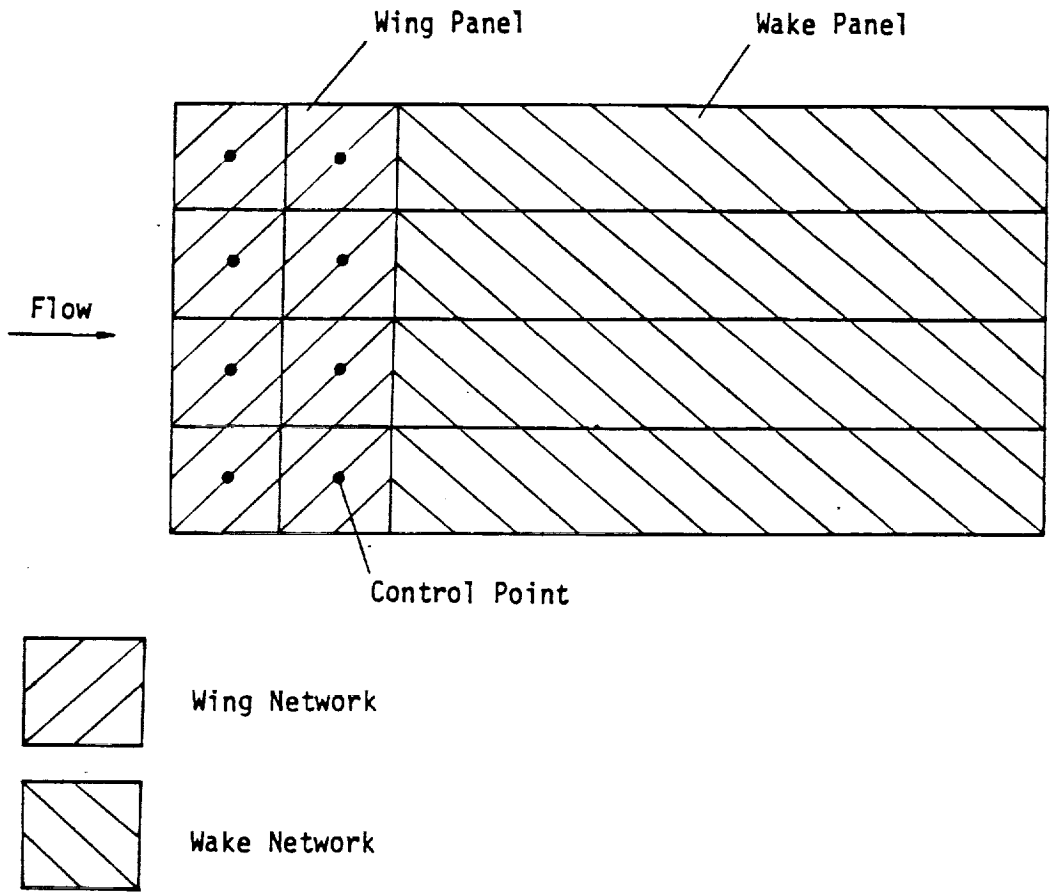


Figure 2-2: Plan View Of Wing Geometry

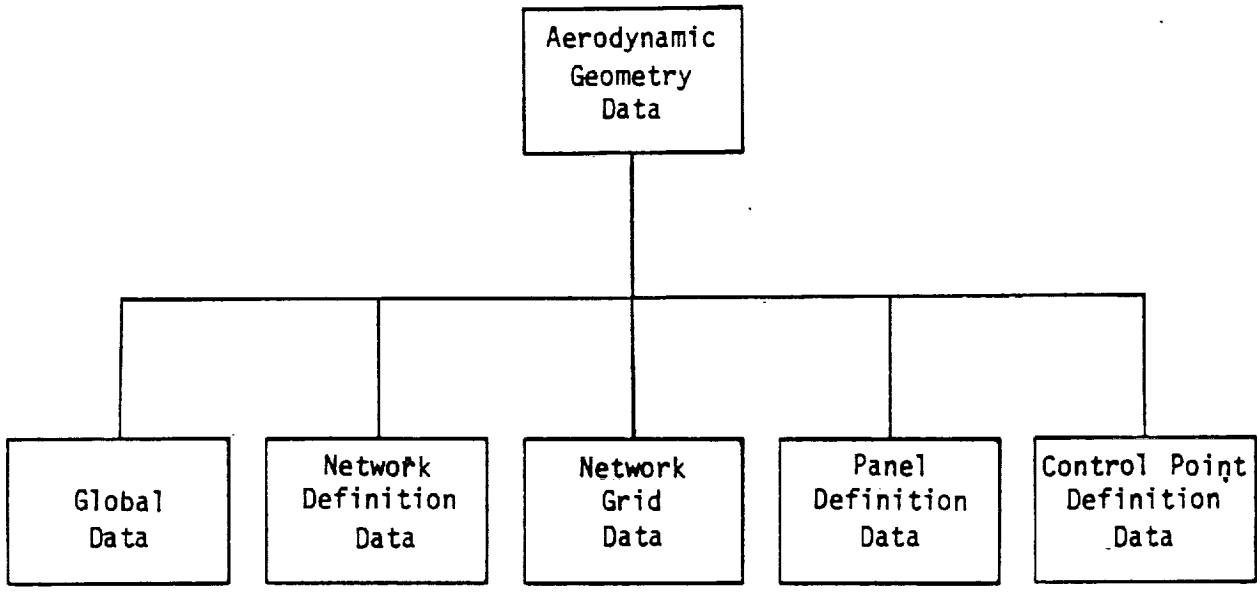


Figure 2-3: Panelled Geometry Data Structure

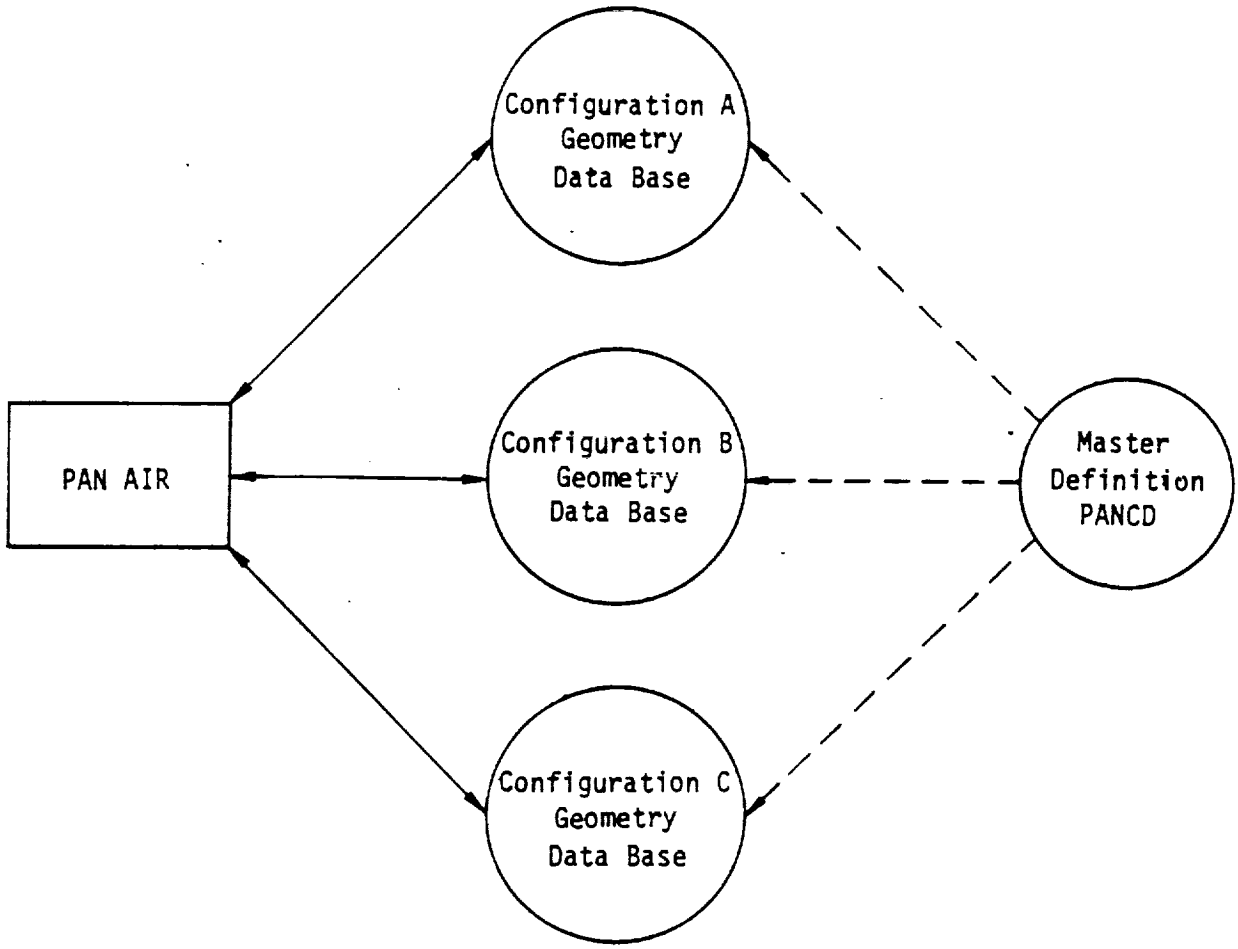


Figure 2-4: Data Base/Master Definition Relationship

```
MASTER DEFINITION pancd

DATASET global-data                    $   global configuration data.
  ELEMENT SET
    no-of-networks                 i   $   number of panel networks in
                                       $   configuration.
    config-name                    t   $   configuration name.
  END
END DATASET

DATASET network-def-data               $   network defining data.
  KEY SET
    network-no                     i   $   network number.
  END
  ELEMENT SET
    num-grid-pt-rows               i   $   number of grid point rows.
    num-grid-pt-cols               i   $   number of grid point columns.
    pnl-srf-fit-ordr               i   $   order of panel surface fit:
                                       $      1=flat panels,
                                       $      2=curved panels.
    pnl-sing-type                  i   $   panel singularity type.
                                       $      0=constant strength source,
                                       $      1=linearly-varying source,
                                       $      2=quadraticly-varying doublet,
    network-type                   i   $   network type:
                                       $      1=source/analysis,
                                       $      2=doublet/analysis,
                                       $      3=source/design no 1,
                                       $      4=doublet/design no. 1,
                                       $      5=source/design no. 2,
                                       $      6=doublet/design no. 2,
                                       $      8=doublet/wake no. 1,
                                       $      0=doublet/wake no. 2.
    num-sing-params                i   $   number of singularity parameters.
    num-ctl-pt-rows                i   $   number of control point rows.
    num-ctl-pt-cols                i   $   number of control point columns.
    num-ctl-pts                    i   $   number of control points.
  END
END DATASET

DATASET network-grid-pts               $   network grid points.
  KEY SET
    network-no                     i   $   network number.
  END
  ELEMENT SET SEQUENCE
    grid point                     r   $   grid point in x,y,z order.
  END
END DATASET
```

Table 2-2: Aerodynamic Data Definition

```
DATASET panel-def-quant                    $   panel defining quantities.
  KEY SET
    network-no                  i      $   network number.
    panel-no                    i      $   panel number.
  END
  ELEMENT SET
    corner-pt-1            3    r      $   corner point in x,y,z order.
    corner-pt-2            3    r      $   same,
    corner-pt-3            3    r      $   same,
    corner-pt-4            3    r      $   same.
    u-vector               3    r      $   (u cross v defines vector normal
                                           to flat panel.)
    v-vector               3    r      $
    pnl-ctr-pt             3    r      $   panel center point.
    lcs-rrans-mat          9    r      $   local coord. system transfor-
                                           mation matrix. (3 x 3)
    inv-lcs-tr-mat         9    r      $   inverse of lcs-trans-mat.
    panel-centroid         3    r      $   panel centroid coordinates.
    lcs-pnl-crnr-pts       8    r      $   panel corner points in local
                                       $   coordinate system.
    quad-surf-coeff        2    r      $   quadratic surface coefficients.
    lcs-origin             3    r      $   local coordinate system origin.
    max-panel-diam              r      $   maximum panel diameter. (in.)
    panel-moments         36    r      $   array of panel moments. (6 x 6)
    panel-sing-type             i      $   panel singularity type:
                                       $      0=constant-strength source,
                                              1=linearly-varying source,
                                              2=quadraticly-varying doublet.
    pnl-surf-fit-ordr           i      $   panel surface fit order:
                                       $      1 for flat panels,
                                       $      2 for curved panels.
    sing-par-index-lst          i          pnl-surf-fit-ordr
                                       $   singularity parameter index list.
    fit-wt-factors                         pnl-surf-fit-ordr
                                       $   fit weighting factors.
    leasr-sq-coeff-mat    96    r      $   least squares fit coefficient
                                       $   matrix.  (6 x 16)
  END
END DATASET
```

Table 2-2: Aerodynamic Data Definition (Cont'd)

```
DATASET ctl-pt-def-quant                          $ control point defining
                                                    quantities.
  KEY SET
    network-no                          i         $ network number.
  END
  ELEMENT SET SEQUENCE
    panel-no                            i         $ panel number.
    control-pt-coords             3     r         $ control point coordinates.
    panel-surf-normal             3     r         $ panel surface normal at control
                                                    point.
  END
END DATASET

DATASET distinct-ctl-pts                          $ definition of distinct control
                                                  $ points.
  KEY SET
    network-no                          i         $ network number.
  END
  ELEMENT SET
    ncp                                 i         $ number of control points.
    ctl-pt-num-list ncp                 i         $ list of distinct control point
                                                    indices.
  END
  END DATASET
END DEFINITION
```

Table 2-2: Aerodynamic Data Definition (Cont'd.)

3.0 Data Base Access Facilities

Figure 1-3 illustrates the relationship between the data definition process (creating data base forms) and the data manipulation process (creating and accessing data base values). Data manipulation activities are carried out by calls to SDMS subroutines using CDC Fortran Extended calling sequence conventions.

There are five classes of manipulation activity:

1. Data Base Initialization and Termination--creating new data bases, connecting to existing ones, and terminating data base processing.

2. Program Variable/Data Base Element Mapping--specifies associations between program variables and data base elements for data transfer purposes.

3. Random Dataset Data Transfer Operations--

4. Sequential Dataset Data Transfer Operations--

5. Dynamic Storage Assignment--obtain and release blocks of central memory.

The sub-sections which follow will discuss each class in the order given above. Two forms will be given for each data manipulation function: a Fortran calling sequence and a statement in a higher-level language called SDML (Scientific Data Management Language). The SDML statements are included only for their descriptive value. They may be included as comments since they begin with an * in column one.

3.01 SDMS Initialization Routine (ISDMS)

* BEGIN SDMS.

```
CALL ISDMS (fwa,lwa)
```

where

fwa = 1 if this is the first word of SDMS working storage.
= 0 if working storage is to be obtained from the system as needed.

lwa = last word of SDMS working storage, if fwa = 1 .

ISDMS initializes data areas for the use of data base processing routines described in subsequent sections.

It is called only once in a given executable step.

It must be called before any other SDMS routines are called.

In an overlay program, it must be called from the (0,0) level.

The presence of ISDMS arguments indicates that all buffer space and dynamic storage areas required by SDMS routines will be allocated within the memory limits specified. For example, the code

```
A(1) = 1
CALL ISDMS (A(1),A(10000))
```

specifies that SDMS can use A(1) through A(10000) as a dynamic storage area.

If fwa = 0, SDMS will get dynamic storage starting at the current value of field length. The amount of additional storage used will fluctuate depending on the number of data bases and sequential datasets open at one time.

The following table indicates SDMS dynamic space requirements.

For each	central memory words required (decimal) is
data base	4 + no. of datasets.
dataset map(3.3)	14 + no. of dynamic variables + no. of key set variables + 2 no. of data elements.
open element set sequence(3.5)	1045

3.1 Data Base Initialization Routine (DBOPEN)

Some DBOPEN calls require the use of a data base descriptor dbd and a master definition description mdd. Both descriptors are short arrays having the same form.

The descriptors tell SDMS where to find or create permanent files.

mdd(1)	master definition file name.
mdd(2)	master definition file user number. May be 0 if same as job.
//mdd(2)	master definition file set name. // (1 to 9 characters)
//mdd(3)	master definition file user id. // (1 to 9 characters)
dbd(1)	data base name (dbn). (1 to 6 alphanumeric characters starting with a letter)
dbd(2)	data base files user number. May be 0 if same as job.
//dbd(2)	data base files set name. // (1 to 9 characters)
//dbd(3)	data base files user id. // (1 to 9 characters)

Note: only the first six non-blank characters of mdd(1) and dbd(1) are used. // Unless a private pack is mounted for the job, mdd(2) and dbd(2) must be set to 'SYSTEM'. After a DBOPEN call using dbd, the default set name for the remainder of the job will be set to dbd(2).//

3.1.1 Data Base Creation

The first function to be performed relative to a particular data base is that of data base creation. In this step, an "empty" data base is created having a form supplied by a specified master definition.

The exercise of this function takes one of two forms. The first is for the creation of temporary data bases.

```
* OPEN DATA BASE dbn USING mdd.

        CALL DBOPEN (dbn,'USING',mdd)
```

where dbn = data base name (1 to 6 characters)
mdd = master definition descriptor.

Since a temporary data base is being created, DBOPEN creates 'local' (in the operating system sense) data base files with names dbn1, dbn2, dbn3 and dbn4 (after returning to the system any previously existing files with the same names).

The statement form for permanent data base creation is:

```
* OPEN DATA BASE dbd PERMANENT pw USING mdd.

        CALL DBOPEN (dbd, 'PERMANENT', pw, 'USING', mdd)
```

where dbd = data base descriptor. (3.1)
pw = master password (1 to 10 characters).
mdd = master definition descriptor. (3.1)

With this calling sequence variant, DBOPEN creates direct access permanent data base files with an associated master password.

3.1.2 Post Creation Access

DBOPEN is also called to connect to existing data bases. If dbn is an existing temporary data base (exists on local files), the following form is used.

```
* OPEN DATA BASE dbn.

        CALL DBOPEN (dbn)
```

If dbn is an existing permanent data base (exists on permanent files) the form used is:

```
* OPEN [SHARED] DATA BASE dbd PERMANENT pw.

        CALL DBOPEN (dbd, 'PERMANENT', pw, 'OLD', [,'SHARED',])
```

where dbd = data base descriptor. (3.1)
pw = password (1 to 10 characters).

If the parameter 'SHARED' is included, the data base files are attached in READ mode. This permits other programs to use dbd in SHARED mode at the same time. None of these programs may modify data base dbd.

If password pw is the master password assigned at data base creation, then read permission is granted for all datasets in dbd; write permission is granted for all datasets if dbd is not SHARED.

If pw is not the master password, then access to a dataset is controlled by the set of passwords included in its definition. If pw is not in the password set, no dataset elements may be accessed. If pw is in the set, it has been granted either read (R) or write (W) permission. Write permission includes read permission.

Restrictions

Attempting to open a permanent data base in SHARED mode when it is currently open in non-SHARED mode will result in a non-fatal error (see 4.0).

No more than 10 data bases can be open (active) at one time.

3.2 Data Base Termination Routine (DBCLOS)

After data base processing is complete, the using program may elect to perform a data base termination function.

```
*  CLOSE/RETURN DATA BASE dbn.

        CALL DBCLOS (dbn [,'RETURN',])
```

where x/y indicates 'x or y'.
dbn = data base name.

DBCLOS updates data base files for dbn to reflect its current state. It also releases all dynamic table space used by dbn, including its maps (3.3). Inclusion of the RETURN parameter causes data base files to be returned to the system. In the case of abnormal job termination, SDMS regains control and automatically closes any open data bases, excluding any for which the DBOPEN process was not completed.

Restrictions

Any data base which has been modified and which is required after program termination must be CLOSED before termination.

A data base must be CLOSED before it can be reOPENed.

3.3 Dataset Mapping Routines

SDMS handles communication between two different data spaces: one containing program variables like x,y,z and another containing data base elements like X-VALUE, Y-VALUE, Z-VALUE (Figure 3-1).

In order for data transfer to take place, a two-part process must occur. First, a named data map is constructed by a sequence of subroutine calls (3.3.3). The data map binds program variables to elements of a data base as specified in its master definition (Figure 3-2). Then calls to other SDMS routines move data to and from the data base by referencing the map name (3.4, 3.5).

From a mapping standpoint, program variables fall into two classes: dynamic and static. Dynamic variables have addresses which may change from reference to reference. In Fortran, dynamic variables consist of variable-subscript array references like X(I) and subroutine formal parameters. All other variables define static references.

The distinction is made because static variables can be "bound" when the map is constructed. Dynamic variables cannot be bound until data transfer is requested.

3.3.1 Static Mapping

Figure 3-3 shows a mapping in which a map M forms an association between static variable V and data element d in dataset DS. Map M is created by subroutine calls. A single map may define an arbitrary number of variable-to-data element bindings; only one is shown for clarity.

The process of data transfer between V and d is diagrammed in Figure 3-4. Transfers to and from the data base are effected by SDMS subroutines which reference map M to guide the transfer.

Suppose that DS(K) is a dataset with one key K. We add to map M a program variable V_1 which is bound to dataset key K. Figure 3- 5 shows how this map extension permits transfers between program variable V_2 and data element d in DS.

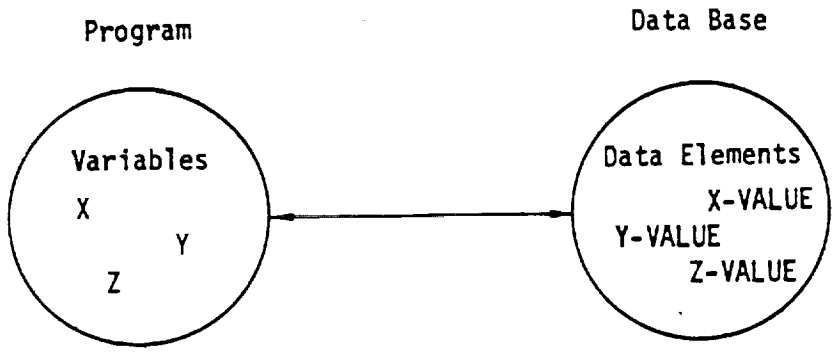


Figure 3-1: SDMS Data Spaces

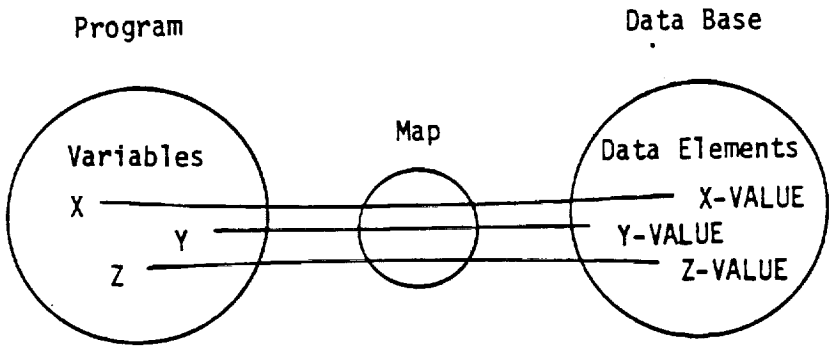


Figure 3-2: Variables-to-Data Element Mapping

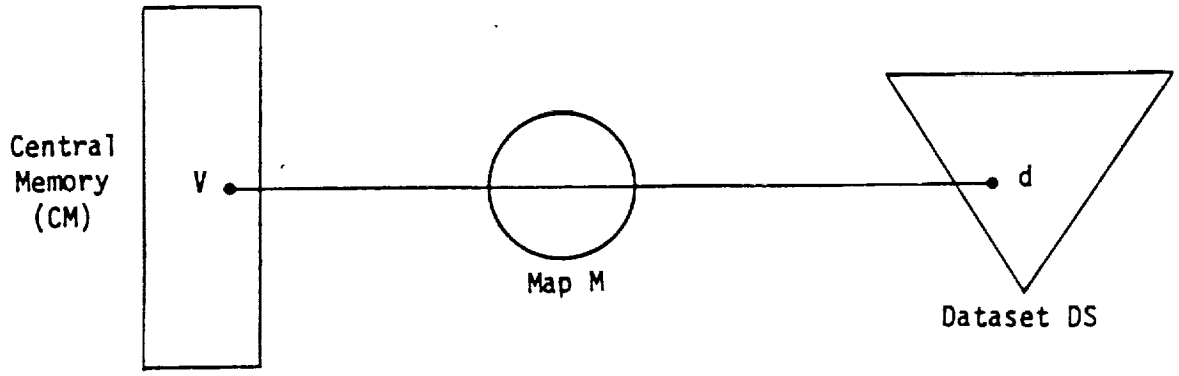


Figure 3-3: Static Variable Map

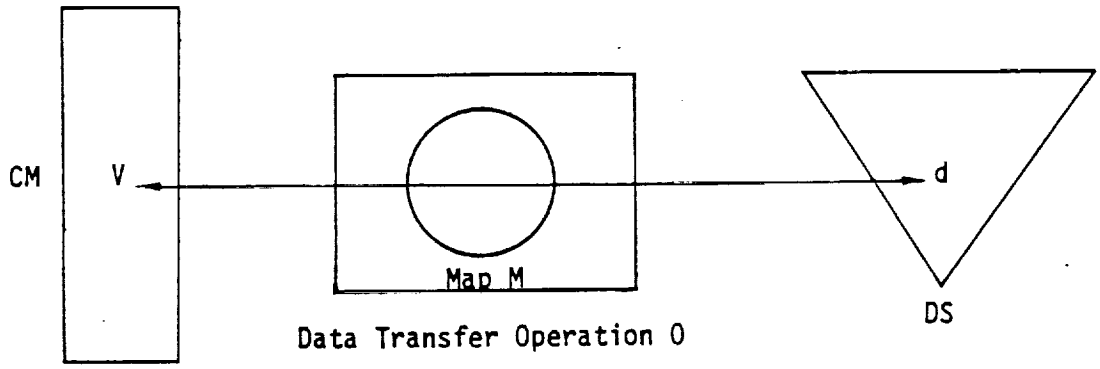


Figure 3-4: Data Transfer of Static Variable

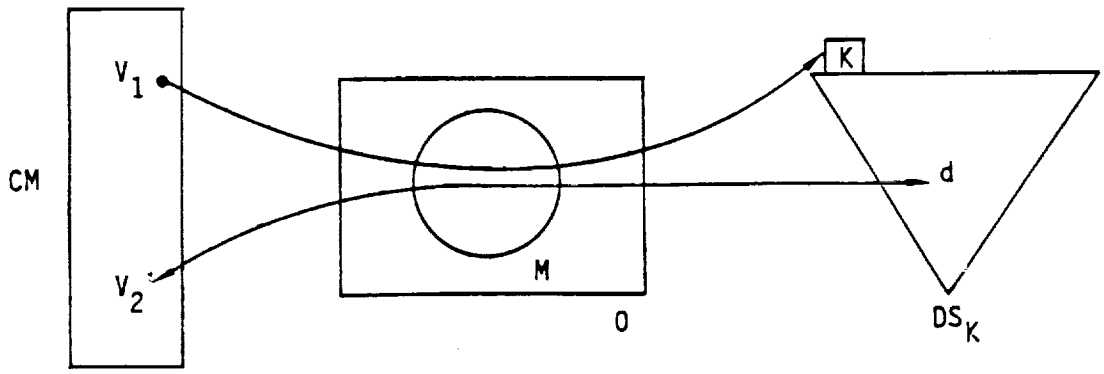


Figure 3-5: Mapping of Keyed Dataset

3.3.2 Dynamic Mapping

The use of static variables makes mapping efficient but rigid. In our previous examples, the source or destination of data element d as determined by M must be a fixed location in central memory.

The use of dynamic mapping (Figure 3-6) removes this restriction. At map creation, only the data base side of the correspondence is established. The central memory portion of the linkage is completed at data transfer time (Figure 3-7). Dynamic mapping makes it possible to reference dynamic variables such as subroutine formal parameters and variably-subscripted array references.

A map M can reflect both static and dynamic mappings, as shown in Figure 3-8.

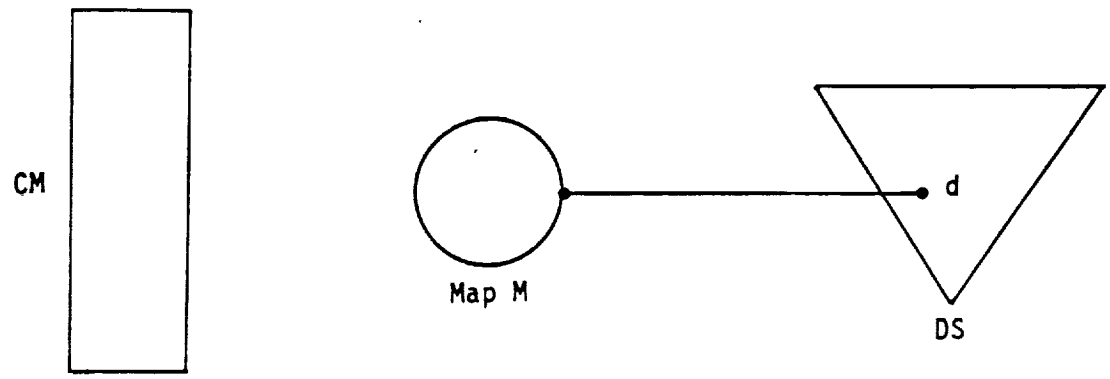


Figure 3-6: Dynamic Variable Map

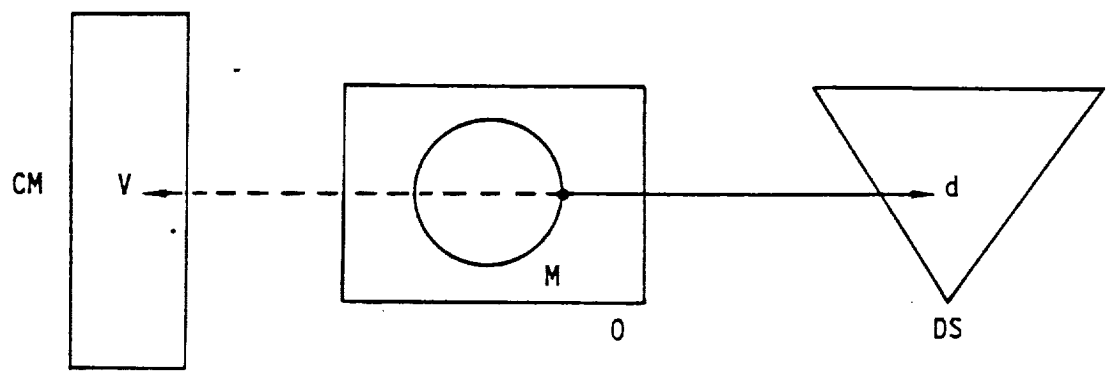


Figure 3-7: Transfer of Dynamic Variable

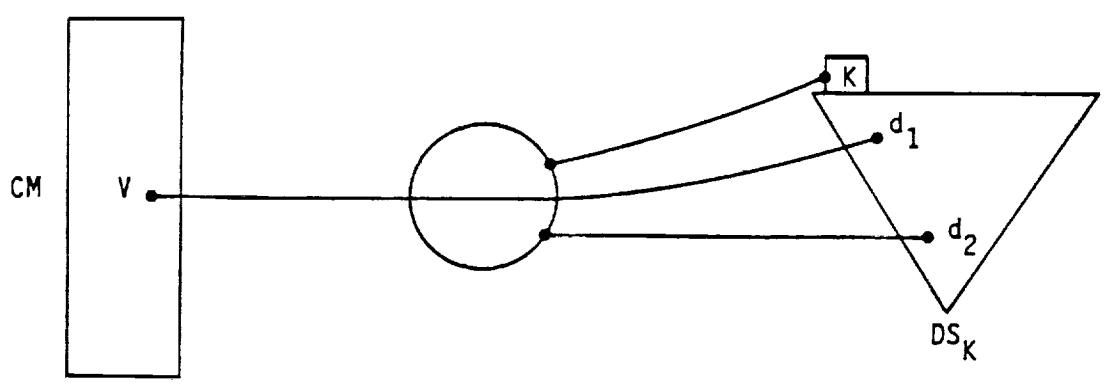


Figure 3-8: Typical Map

3.3.3 Map Creation

```
* MAP dmn FOR dsn IN dbn HAS [vn=den,]* [vden (i)]*.

          CALL DSMAP (n,dsn,dbn)
         ----------------------------------------*
         [CALL SVMAP (vn(1),den(1), . . . ,vn(n),den(n))]
         ----------------------------------------
         [CALL DVMAP (vden(1), . . . ,vden(m))]
          CALL ENDMAP
```

where
- dmn = unique map name (1 to 10 characters).
- dsn = dataset name (Hollerith literal[1]).
- dbn = name of open data base.
- * = arbitrary integer ≥ 0.
- [x]* = x,x,...,x
- vn(i) = static variable name.
- den(i) = Hollerith literal defining data element name corresponding to vn(i).
- vden(k) = Hollerith literal defining data element name associated with dynamic variables in data transfer calls (3.4.1 - 3.4.6, 3.5.4, and 3.5.5).

The map name dmn is used by data transfer functions (3.4) to determine the sources and destinations of program variables and data base elements.

Restrictions

DSMAP must be called only once for each value of dmn.

A map definition sequence must begin with a call to DSMAP and end with a call to ENDMAP.

The calls to SVMAP and DVMAP are order-independent.

No more than one DVMAP call may appear in the sequence.

No more than 10 SVMAP calls may appear in the sequence.

No more than 100 maps can be defined at one time.

1. The name of a 2-word array variable may be substituted for the Hollerith literal. If the text string has 10 characters or less, the second word must have a zero value.

3.3.4 Static Mapping Example

The most efficient method of binding program variables to data base elements is in the DSMAP call itself. Such variables must be static; they cannot be subroutine formal parameters or a dynamic array reference such as A(I),. Any other references are legal, such as local variables L or A(3,4), or COMMON variable X.

Consider the dataset shown in Table 3-1. The following DSMAP call forms an association between local program variables N,X,Y and dataset elements NC, X-VALUE and Y-VALUE respectively.

```
* MAP CRVMAP FOR 2D-CURVE IN dbn HAS N=NC, X=X-VALUE, Y=Y-VALUE.

        CALL DSMAP ('CRVMAP', '2D-CURVE', dbn)
        CALL SVMAP (N,'NC',X,'X-VALUE',y,'Y-VALUE')
        CALL ENDMAP
```

This association is specified by map name in subsequent calls to SDMS data transfer routines. After the previous map calls have been made, the statement

```
* PUT CRVMAP.

        CALL ESPUT ('CRVMAP')
```

causes a transfer of variables from the program to the data base as shown below.

Program Variable		Data Base Element
N	--->	NC
X(i),i=1,N	--->	X-VALUE
Y(i),i=1,N	--->	Y-VALUE

The use of ESPUT is fully explained in Section 3.4.1.

```
DATASET 2d-curve

        ELEMENT SET

             curve-title        5   t    $    curve title array.
             nc                     i    $    number of coordinate pairs.
             x-value           nc   r    $    x coordinate array.
             y-value           nc   r    $    y coordinate array.

        END

END DATASET
```

Table 3-1: Example Dataset

3.3.5 Dynamic Mapping Example

In the previous example, suppose that N,X and Y were formal sub-routine parameters. They would then be referenced in the ESPUT call instead of the DSMAP call, as shown below:

```
* MAP CRVMAP FOR 2D-CURVE IN dbn HAS NC, X-VALUE, Y-VALUE.

        CALL DSMAP ('CRVMAP', '2D-CURVE',dbn)
        CALL DVMAP ('NC,'X-VALUE','Y-VALUE')
        CALL ENDMAP

             .
             .
             .

* PUT CRVMAP USING N,X,Y.

        CALL ESPUT ('CRVMAP', N,X,Y)
```

3.3.6 Restrictions

DSMAP must be called only once to establish a map with a given name (dmn value). It should be considered a declarative command.

Data element names must be Hollerith literals in the calling sequence.

If a variable-length array data element is included in a map, its length-defining scalar must also be included (2.2.6).

All key data elements must be included in the map.

3.3.7 Permissible Usages

Any subset of dataset elements may be referenced in a single map.

Dynamic and static mappings may be included in the same map. For example,

```
        CALL DSMAP ('CRVMAP','2D-CURVE',dbn)
        CALL SVMAP (N,'NC')
        CALL DVMAP ('X-VALUE','Y-VALUE')
        CALL ENDMAP
```

is legal.

Data element names and static and dynamic variables may be used in more than one map.

3.3.8 Map Usage Techniques

In order to reference static map variables in more than one coding module, it is necessary to pass them through blank or labelled COMMON. It is recommended that all static variables and only those variables associated with a particular map be kept in a single labelled common block with the same name as the map. It is also recommended that the correspondence between program variables and mapped data elements be identified by comments immediately preceding or following the common block containing mapped variables.

To keep the clerical work involved in maintaining several copies of map-related text to a minimum, it is suggested that the UPDATE utility[1] be used to maintain the source program. Then each set of map comments and a COMMON declaration can be maintained as a single COMDECK within UPDATE.

The following text shows how map-related text might look for the map used in 3.3.4.

```
C.................................C
C   MAP=CRVMAP      DATASET=2D-CURVE
C   VARIABLES       DATA ELEMENTS
C        N                    NC
C        X               X-VALUE(NC)
C        Y               Y-VALUE(NC)
    COMMON/CRVMAP/N,X(100),Y(100)
C.................................C
```

When dynamic variables are used, the list position of the data element in the DVMAP call should be given in place of the variable name. For example, the map text for the map used in 3.3.5 might be as follows:

```
C.................................C
C   MAP=CRVMAP      DATASET=2D-CURVE
C   VARIABLES       DATA ELEMENTS
C        D.V. 1               NC
C        D.V. 2          X-VALUE(NC)
C        D.V. 3          Y-VALUE(NC)
C.................................C
```

Note that no COMMON declaration is required because the map includes no static variables.

1. UPDATE Reference Manual, Control Data Corp. Document No. 60449900.

3.3.9 Map Construction in Overlay Programs

To conserve variable space, it may be desirable to issue map-construction statements in overlay levels above the 0,0 level. If this is done, the programmer must arrange to have these calls executed only once no matter how many times the overlay is called. This can be done conveniently by placing a map-construction flag in a labelled COMMON block in the 0,0 overlay and testing it just prior to map creation. If the flag is false, the map does not yet exist. It is then created and the flag is set to true.

3.3.10 Preventing Mapping Error Aborts

If a data or key element name is mis-spelled during map construction, SDMS error 18 is detected during ENDMAP processing. Normally, this causes an explanatory message to be printed followed by an immediate abort of the job. If many errors are present in a set of maps, it will take many runs to find them all.

The job abort due to SDMS error 18 can be prevented by executing

```
CALL MAPCHK
```

prior to map processing. The explanatory message is still printed, but execution of the job continues.

After map processing is complete, the user program should execute

```
CALL MAPERR(err)
```

The variable err will be .TRUE. if SDMS error 18 has occurred since the MAPCHK call. It is the calling program's responsibility to terminate the job if an error condition is detected.

3.4 Random Dataset Functions

I/O operations on random datasets are carried out relative to a map of the dataset (3.3). If the dataset has a key set, all corresponding key variables must be set before an I/O operation is performed. A key variable is a program variable which corresponds to a KEY SET element in a dataset (2.1, 2.2.4). In the example of 3.4.7, NAME is the key variable associated with key element 'MATRIX-NAME' in dataset 'MATRIX-DATA'.

3.4.1 Put Element Set (ESPUT)

* PUT dmn [USING pv*].

```
CALL ESPUT (dmn [,pv(1), . . . ,pv(n)])
```

where dmn = name of non-DIRECT random dataset map.
pv(k) = dynamic Fortran variable corresponding to data element vden (k) in map dmn (3.3.3).

ESPUT uses DSMAP static variables vn(i) and ESPUT dynamic variables pv(k) (if present) to transfer data to the corresponding data elements in the selected element set.

Note: if an element set with the specified key values already exists, no transfer is made and non-fatal error 21 is returned (4.0). This error can be made fatal by calling SETAEF.

3.4.2 Put DIRECT Element Set (DESPUT)

* PUT dmn DIRECT origin lb.

```
CALL DESPUT (dmn,origin,lb [,pv(1),...,pv(n)])
```

where dmn = name of DIRECT random dataset map.
origin = start of block to be transferred.
lb = length of block.
pv(k) = dynamic Fortran variable corresponding to key element vden(k) in map dmn (3.3.3).

DESPUT transfers the block to the DIRECT dataset specified by dmn. Since DIRECT datasets have no element set structure (2.2.2), they can be moved to and from the data base more efficiently.

Note: the note in 3.4.1 also applies to DESPUT.

3.4.3 Element Set (ESGET)

* GET dmn [USING pv*].

```
CALL ESGET (dmn[,pv(1), . . . ,pv(n)])
```

where dmn = name of random dataset map.
pv(k) = dynamic Fortran variable corresponding to data element vden(k) in map dmn (3.3.3).

ESGET uses DSMAP variables vn(i) and ESGET variables pv(k) (if present) to transfer data from the corresponding data elements in the selected dataset. A non-fatal error occurs if the selected data set does not exist (4.0). Any data element with undefined value will be input as the indefinite quantity 1777 0000 0000 0000 0000 octal.

Note: if the specified element set does not exist, non-fatal error 23 is returned (4.0). This error can be made fatal by calling SETAEF.

3.4.4 Get DIRECT Element Set (DESGET)

* GET dmn DIRECT origin limit lb.

```
CALL DESGET (dmn,origin,limit,lb [,pv(1), . . . ,pv(n)])
```

where
- dmn = name of DIRECT DATASET MAP.
- origin = start of block to be received.
- limit = maximum block length permitted.
- lb = actual block length (output).
- pv(k) = dynamic Fortran variable corresponding to key element vden(k) in map dmn (3.3.3).

The element set specified by dmn is transferred into the program area indicated in the DESGET call. No error is returned if lb exceeds limit.

Note: the note in 3.4.3 also applies to DESGET.

3.4.5 Replace Element Set (ESREP)

* REPLACE dmn [USING pv*].

```
CALL ESREP (dmn[,pv(1), . . . ,pv(n)])
```

where
- dmn = name of non-DIRECT random dataset map.
- pv(k) = Fortran variable corresponding to data element vden(k) in map dmn (3.3.3)

ESREP replaces the whole element set selected through map dmn, not just the mapped data elements. If the previous element set is at least as long as its replacement, it will be overwritten. Otherwise, additional disk space will be used.

Note: the note in 3.4.3 also applies to ESREP.

3.4.6 Replace DIRECT Element Set (DESREP)

* REPLACE dmn direct origin lb.

```
CALL DESREP (dmn, origin, lb [,pv(1), . . . ,pv(n)])
```

where
- dmn = name of DIRECT dataset map.
- origin = start of block to be transmitted.
- lb = actual block length.
- pv(k) = dynamic Fortran variable corresponding to key element vden(k) in map dmn (3.3.3).

DESREP replaces the DIRECT element set selected via map dmn. If the previous element set is at least as long as its replacement, it will be overwritten. Otherwise, additional disk space will be used.

Note: the note in 3.4.3 also applies to DESREP.

3.4.7 Creating and Accessing Random Datasets

A common data representation for a matrix is shown in Figure 3-9. Each matrix X consists of mn blocks X(i,j) in a rectangular arrangement. Each block X(i,j) contains r(i) rows and c(j) columns. Table 3-2 shows an SDMS representation of block-format matrices with comments referencing the symbols of Figure 3-9.

The dataset 'matrix-data' holds global information about each matrix keyed by matrix name. Dataset 'blocks' describe the structure of an individual block. It includes r(i), c(j) and X(i,j). Its keys include i, j and the name of X.

This pair of datasets describe a whole family of matrices X(i,j), i=1,2,...,m(X),j=1,2,...,n(X). The global attributes of any matrix X are retrieved 'matrix-data' using the matrix name as key. The contents of any block X(i,j) are retrieved from 'blocks' using (i,j), and the name of X as keys.

Assuming that an empty data base MTX has been initialized, the next step is to provide a mapping for 'matrix-data'. Since there is no need for dynamic reference to its data elements, the mapping will be exclusively in terms of static variables.

```
CALL DSMAP ('M-D','MATRIX-DATA','MTX')
CALL SVMAP (NAME,'MATRIX-NAME',NRB,'N-ROW-BLOCKS',
            NCB,'NUM-COL-BLOCKS',NR,'N-ROWS',NC,
             'NUM-COLS')
CALL ENDMAP
```

In the case of dataset 'blocks', a dynamic variable will be used to reference data element ,block, since the program will be working on more than one block at a time.

```
CALL DSMAP ('BLK','BLOCKS','MTX')
CALL SVMAP (NAME,'MATRIX-NAME',RBI,'ROW-BLOCK-INDEX,
             CBI,'COL-BLOCK-INDEX',NRPB,'NUM-OF-ROWS/BLOCK')
             NCPB,'NUM-OF-COLS/BLOCK',LB,'BLOCK-LENGTH')
CALL DVMAP ('BLOCK')
CALL ENDMAP
```

After the variables NAME, NRB, NCB, NR and NC have been set, the statement

```
CALL ESPUT ('M-D')
```

causes a new element set to be created in dataset 'matrix-data'.

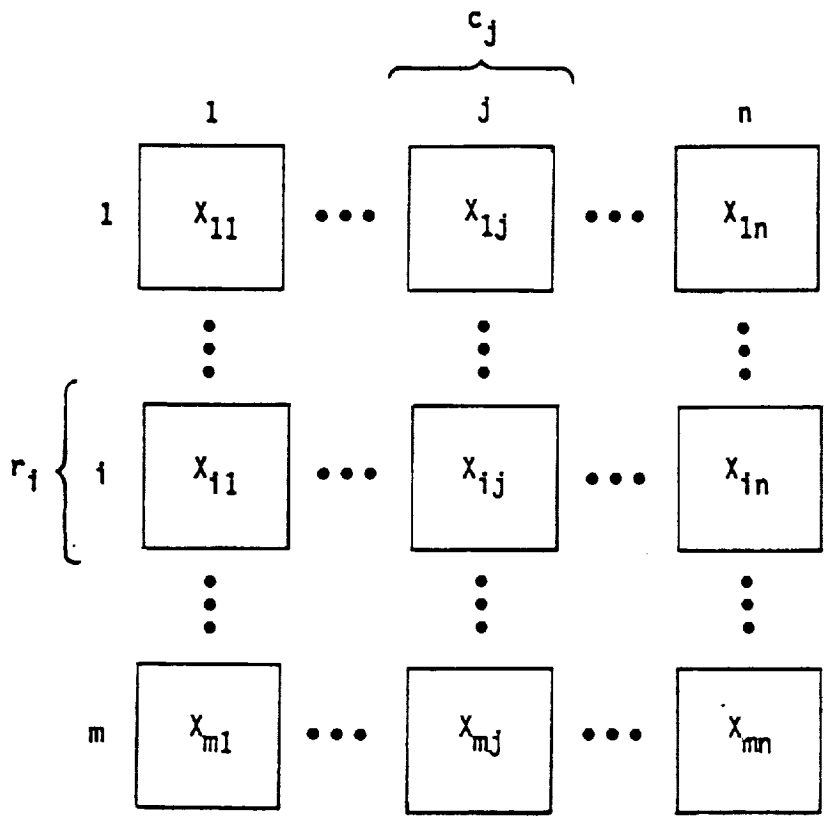


Figure 3-9: Block Format Matrix

```
DATASET matrix-data

        KEY SET

             matrix-name              t    $    name of matrix x.

        END

        ELEMENT SET

             num-row-blocks           i    $    m
             num-col-blocks           i    $    n
             num-rows                 i    $    sum of r (i), i=1,m
             num-cols                 i    $    sum of c (j), j=1,n

        END

        END DATASET

DATASET blocks

        KEY SET

             matrix-name              t    $    name of matrix X.
             row-block-index          i    $    i
             col-block-index          i    $    j

        END

        ELEMENT SET

             num-of-rows/block        i    $    r(i)
             num-of-cols/block        i    $    c(j)
             block-length             i    $    r(i) * c(j)
             block block-length       r    $    X(i,j) elements in column
                                                order.

END

END DATASET
```

Table 3-2: Random Dataset Example

Assume the dynamic variable XIJ is the origin of block X(i,j). After key variables NAME, RBI, CBI, and data variables NRPB, LB are set, the statement

```
CALL ESPUT ('BLK',XIJ)
```

transfers the specified matrix block to the data base.

To read a matrix block, the key variables NAME, RBI, CBI are set and dynamic variable XBS(I) is selected to serve as a correspondent to 'BLOCK'. The statement

```
CALL ESGET ('BLK',XBS(I))
```

causes the matrix block to be transferred to the block of memory starting at XBS(I).

3.4.8 DIRECT Dataset Usage

Table 3-3 shows a typical DIRECT dataset containing information related to a group of surface "patches". The patches themselves might be defined individually in another dataset. To analyze a surface region efficiently, a group of patches can be formed into a larger block and transferred to 'intermed-geom-data' for use in subsequent processes.

This dataset is mapped with the statement

```
CALL DSMAP ('I-G-D','INTERMED-GEOM-DATA',dbn)
CALL SVMAP (NR,'REGION-NUMBER')
CALL ENDMAP
```

Assume that the block is constructed in the array REG. After the static key variable NR is set, a block is output with the statement

```
CALL DESPUT ('I-G-D',REG,LREG)
```

where LREG is the block length.

Block input is achieved by again setting NR and executing the statement

```
CALL DESGET ('I-G-D',REG,NREG,LREG)
```

where REG and LREG have the same meaning as previously and NREG is the maximum number of words to be read.

```
DATASET intermed-geom-data DIRECT

        KEY SET

             region-number  i          $    index to a group of neigh-
                                       $    boring surface patches.

        END

END DATASET
```

Table 3-3: DIRECT Dataset.

3.5 Sequential Dataset Functions

3.5.1 Open Element Set Sequences (ESSOPN)

Before data can be transferred to or from an element set sequence, it must be opened or initialized with a call to ESSOPN.

* OPEN lsn FOR dmn FIRST/LAST .

```
CALL ESSOPN (lsn,dmn, 'FIRST'/'LAST')
```

where dmn = name of existing sequential dataset map.
lsn = local sequence name (1 to 10 characters)

If the sequence specified by map dmn and its key values (if any) exists, then it is opened at the position indicated by the table below.

argument 3	position
'FIRST'	before first element set,
'LAST'	after last element set.

If the referenced sequence does not exist, an empty sequence is created. Any sequence previously associated with lsn will be closed (3.5.3).

Cautions

Opening an element set sequence requires buffer and control table space. Dead space can be minimized by keeping inactive sequences closed.

Restrictions

All key variables in dmn must be static.

3.5.2 Position Element Set Sequence (ESSPOS)

* POSITION lsn FIRST/LAST .

```
CALL ESSPOS (lsn, 'FIRST'/'LAST')
```

where lsn = local sequence name.

If 'FIRST' is specified in the calling sequence, lsn is positioned before the first element set in the sequence. If 'LAST' is specified, lsn is positioned after the last element set in the sequence.

3.5.3 Close Element Set Sequence (ESSCLS)

* CLOSE lsn .

```
CALL ESSCLS (lsn)
```

where lsn = local sequence name.

ESSCLS updates the data base image of lsn if necessary and marks associated central memory space for release.

3.5.4 Put Into Next Element Set (ESSPUT)

* PUT INTO SEQUENCE lsn [USING pv*] .

```
CALL ESSPUT (lsn [,pv(1),pv(2), . . . ,pv(n)])
```

where lsn = local sequence name.
pv(i) = dynamic Fortran variable corresponding to data element vden(i) in the map associated with lsn (3.3.3, 3.5.1).

ESSPUT uses map variables vn(i) and variables pv(k) to transfer data to the corresponding data elements (see 3.3.6) in the next element set in the sequence. This element set becomes the last element set in the sequence.

3.5.5 Get From Next Element Set (ESSGET)

* GET FROM SEQUENCE lsn [USING pv*] .

```
CALL ESSGET (lsn [,pv(1), . . . ,pv(n)])
```

where lsn = local sequence name.
pv(i) = dynamic Fortran variable corresponding to data element vden(i) in the map associated with lsn (3.3.3, 3.5.1).

ESSGET is symmetric to ESSPUT. It uses map variables vn(i) and variables pv(k) to transfer data into the program from the corresponding data elements in the 'next' element set in the sequence. Any undefined data element values will be represented by the indefinite value 1777 0000 0000 0000 0000 octal.

3.5.6 Using Sequential Datasets

Table 3-4 contains a definition which will be used to illustrate how sequential datasets are built. The first step is to create a new temporary data base and build a map for dataset space curve.

```
* OPEN DATA BASE SHAPE USING CRVSET.

        CALL DBOPEN ('SHAPE', 'USING', 'CRVSET')

* MAP CURVE OF SPACE-CURVE IN SHAPE HAS X=X-VALUE, Y=Y-VALUE, Z=Z-VALUE,
    CN=CURVE-NAME.

        CALL DSMAP ('CURVE', 'SPACE-CURVE', 'SHAPE')
        CALL SVMAP (X,'X-VALUE', Y, 'Y-VALUE', Z, 'Z-VALUE',
                    CN, 'CURVE-NAME')
        CALL ENDMAP
```

The next step is to specify the name of a curve to be constructed and open a local sequence to hold it.

```
        CN = 'SPIRAL'

*       OPEN CRV1 FOR CURVE.

        CALL ESSOPN ('CRV1','CURVE')
```

This is followed by steps to move successive curve points into the 'SPIRAL' element set sequence.

```
        DO 10 I=1,NPOINTS (set values of x,y,and z)

*       PUT INTO CRV1.

10      CALL ESSPUT ('CRV1')
```

The next curve can be constructed by opening a 'CURVE' element sequence with a different key.

```
        CN = 'GREAT-CIRCLE'

*       OPEN CRV1 FOR CURVE.

        CALL ESSOPN ('CRV1', 'CURVE')
```

As indicated in 3.5.1, the 'SPIRAL' element sequence will be CLOSED automatically when 'GREAT-CIRCLE' is opened. After all curves have been generated, local sequence CRV1 is CLOSED.

```
* CLOSE CRV1.

        CALL ESSCLS ('CRV1')
```

```
MASTER DEFINITION crvset

        DATASET space-curve

            KEY SET

                curve-name      2       t    $ curve identifier.

        END

        ELEMENT SET sequence

                x-value                 r    $    x coordinate.
                y-value                 r    $    y coordinate.
                z-value                 r    $    z coordinate.

        END

    END DATASET

END DEFINITION
```

3.6 Miscellaneous Data Base Functions

3.6.1 Purging Data Base (DBPURG)

Permanent data bases may be completely removed from the system by use of the DBPURG routine.

* PURGE DATA BASE dbd.

```
        CALL DBPURG (dbd)
```

where dbd = data base descriptor (3.1)

Restrictions:

- The data base referenced by dbd must be closed when DBPURG is called. Since the data base is completely removed from the system, DBPURG must be used with considerable caution.

3.6.2 Deleting Key Sets (KSDEL)

Key sets may be removed from a dataset index by using the KSDEL routine.

Calling sequence:

```
        CALL KSDEL(dmn [,pv(1), . . . ,pv(n)])
```

where:

dmn = dataset map name.
pv(k) = dynamic Fortran variable corresponding to argument vden(k) in map dmn (3.3.3).

KSDEL can be used with any of the three dataset classes: random, direct and sequential.

No error condition results if the specified key set values are not found.

Note: KSDEL results in a logical rather than a physical delete of a key set and its associated data.

4.0 Error Handling

SDMS detects both fatal and non-fatal errors. Detected fatal errors cause the printout of the error number plus a traceback showing the flow of control which preceded the error. SDMS enters a recovery phase in which it closes all data bases currently open and aborts the job.

Non-fatal errors can occur after calls to ESPUT, ESGET, ESREP, ESGET, DESREP AND ESSGET. These can be detected by including the COMMON block SDMSER as shown below.

```
COMMON/SDMSER/NERR
```

If NERR is non-zero, non-fatal error with that value occurred on the preceding SDMS call.

Tables 4-1 and 4-2 list all SDMS error messages.

Non-fatal errors can be made fatal by executing the statement CALL SETAEF ('ON'). Afterwords, all non-fatal errors are treated as fatal except error 27, 'end-of-information detected on element set sequence'. The default condition can be restored with the statement CALL SETAEF ('OFF').

Table 4-1: SDMS Error Messages

Error Number	Meaning	Fatal Error
1	Second call to ISDMS in same program.	yes
2	Wrong no. of arguments to ISDMS.	yes
3	Too many data bases defined.	yes
4	ISDMS not yet called.	yes
5,n	Permanent file error. (see table 4-3 or 4-4)	yes
6,n	System error. (see Table 4-2)	yes
7	Field length limit exceeded while getting working storage from system or supplied working storage buffer exceeded (see 3.01).	yes
8	Previous map not yet complete.	yes
9	Too many maps defined.	yes
10	Duplicate map name.	yes
11	Unknown data base name.	yes
12	Too many SVMAP calls for one map.	yes
13	Too many DVMAP calls for one map.	yes
14	ENDMAP call without preceding DSMAP call.	yes
15	Unknown dataset name.	yes
16	Key element not found in map arguments.	yes
17	Array-length parameter not given for variable-length array.	yes
18	Dataset does not contain one or more elements specified in map.	yes

Error Number	Meaning	Fatal Error
19	Unknown map name.	yes
20	Wrong no. of dynamic variables.	yes
21	Element set already exists with the specified key set.	no
22	Illegal dataset access within permanent data base. Attempting to write to shared data base or attempting access which violates user password permissions.	yes
23	Key entry not found.	no
24	Too many local sequences open.	yes
25	Undefined local sequence.	yes
26	Undefined local sequence positioning operation.	yes
27	End-of-information detected reading element set sequence.	no
28	Duplicate data base name.	yes
29	Element set sequence already open.	yes
30	Bad syntax in DBOPEN call.	yes
31	matrix column length (via MATMAP call) less than submatrix column length.	yes
32	array or matrix element dimension negative.	yes
33	not used.	
34	too many simultaneous searches.	yes
35	search over unique dataset not necessary or permitted.	yes
36	0 (zero) is not a legal database name.	yes
37	not used.	
38	not used.	

Table 4-1: (continued)

Error Number	Meaning	Fatal Error
39	datamap not terminated by ENDMAP call.	yes
40	not used.	
41	unpaired SVMAP argument.	yes
42	DIRECT dataset block length .lt. 0.	yes

Table 4-1: (Continued)

System Error Number n	Meaning
1	Incorrect length of file resident table.
2	Premature end-of-information detected on data base file.
3	Index tree malformed.
4	Positioning error on file dbn2.
	Too many index levels.
6	Too many dynamic variables.
7	Not used.
8	Positioning error on file dbn4.
9	Element set length mismatch.
10	Element set length field on dbn4 has wrong parity.
11	Variable-length data element header has wrong parity.
12	Element set boundary exceeded.
13	Master definition file length does not match its directory value.
14	Output word count doesn't match buffer length.
15	Next available position doesn't match end-of-information position on data base file.

Table 4-2: SDMS System Errors.

Table 4-3: KRONOS/NOS Permanent File Errors

Perm. File Error Number n	Meaning
1	The specified direct access file is attached to another job.
2	One of the following: * The specified permanent file could not be found. * The specified account number could not be found. * The user is not allowed to access the specified file. * The user issued an indirect access file command on a direct access file. * The user issued a direct access file command on an indirect access file. If this message occurs in response to the SAVE macro, the specified local file is not attached to the control point, is a direct access file,or is an execute-only file.
3	The file specified on a SAVE macro contains no data.
4	The file to be saved is not on mass storage: the first track of the file is not recognizable.
5	The user has already saved or defined a file with the name specified.
6	The user attempted to define a file that was not a local file.
7	File name contains illegal characters.
8	The user is not validated to create direct access or indirect access files or to access auxiliary devices.
9	The device type (r parameter) specified on a request for an auxiliary device cannot be recognized or does not exist in the system. If the auxiliary device specified by the pn parameter is not the same type as the system default, the r parameter must be included; if not, this message is issued.

Perm. File Error Number n	Meaning
10	The local file specified for a SAVE, REPLACE, or APPEND command exceeds the length allowed or the direct access file specified for an ATTACH in WRITE, MODIFY, or APPEND mode exceeds the direct access file length limit for which the user is validated.
11	One of the following: * Illegal command code passed to PFM * Illegal permitmode or catalog type specified * CATLIST request has permit specified without a file name * PERMIT command attempted on a library file
12	Access to the permanent file device requested is not possible.
13	The device on which the file resides may not contain direct access files because: 1. The device is not specified as a direct access device in the catalog descriptor table. 2. The device is not specified as ON and initialized in the catalog descriptor table. 3. The device is a dedicated indirect access permanent file device.
14	Because a permanent file utility is currently active, the operarion was not attempted; the user should retry the operation.
15	An error occured in a read operation during a file transfer.
16	The number of files in the user,s catalog exceeds the limit (refer to LIMITS control statement, section 6).
17	The cummalitive size of the indirect access files in the user's catalog exceeds the limit (refer to LIMITS control statements. section 6).

Table 4-3: (Continued)

Perm. File Error Number n	Meaning
18	The number of PRUS specified via the S parameter on the DEFINE macro is not available.
19	A request was attempted on a local file that is currently active. This error can occur, for example, if the user creates two FETs for the same file and issues a second request before the first is completed.
20	The job's local file limit has been exceeded by an attempt to GET or ATTACH the file.
21	The job's mass storage PRU limit has been exceeded during preparation of a local copy of an indirect access file.
22	Permit limit has been exceeded for a private file.
24	The resource executive has detected a fatal error.
25	No allocatable tracks remain on equipment xx, where xx is the EST ordinal.
26	The length of a file does not equal the catalog length: the action taken depends on the type of command issued.

Command	Action
GET	A local file is created with length being the actual length retrieved.
SAVE	If file length is longer than TRT specification, file is truncated.
REPLACE	Same as for SAVE.

27	Permit random address error.
28	The system sector data for the file does not match the catalog data.
29	The same file was found twice during a catalog search. This error can occur for APPEND or REPLACE commands after a file is found and purged and the catalog search is continued.
30	Error flag detected at PFM control point.
31	An error was encountered in reading a portion of the permanent file catalog or permit information.

Table 4-3: (continued)

Perm. file Error Number n	Meaning
2	Local file name already in use.
3	Local file name unknown.
4	No room for new cycle.
5	Catalog full (PAM full).
6	No local file name or permanent file name, both null.
8	Blocked file not closed.
9	File not on mass storage device. File not requested for *PF.
10	File not on set. Cycle referenced does not exist on set. File purged while waiting for attach.
12	Invalid cycle. Nonnumeric character in numeric field.
13	Duplicate cycle.
14	Directory full.
15	Purge attempted on nonpermanent file. Nonpermanent file cannot be extended.
16	Catalog attempt on nonlocal file.
17	Parameter error.
18	Catalog attempt on null, INPUT or OUTPUT file.
19	Cycle incomplete on an attach. RECOVER detected error on file.
20	Duplicate attach.
22	I/O error on permanent file device. Invalid extend.
23	Illegal local file name.
29	PF name already in system.
32	Illegal setname (set not mounted).

Table 4-4: SCOPE 2.1 Permanent File Errors

5.0 Diagnostic Features

SDMS has extensive instrumentation to assist in finding stubborn problems. After execution of the statement CALL TRACER ('ON'), each SDMS call produces trace information on the OUTPUT file. This information shows when SDMS routines are called and when they return, and the contents of internal tables and important parameters. The tracing process is terminated with the statement CALL TRACER ('OFF').

6.0 Recovery Options

If a system detected error occurs (address out of range, etc.), SDMS normally initiates data base recovery procedures. Any modified element set sequences currently open are closed. All open data bases are closed, with the exception of any data base being opened when the error occurred.

At this point, the SDMS-supplied routine DBABT without argents is called to abort the job. If the user wishes to perform his own recovery operations, he may include his own DBABT routine.

To disable the recovery process and obtain a normal abort sequence, the statement CALL NORCVR must be executed before the call to ISDMS.

7.0 Access to SDMS Subroutines

At the BCS Renton Data Center, the SDMS subroutines described in Section 3 are accessed with the control card

```
ATTACH(SDMSLIB/UN=PAWAM1)
```

SDMSLIB is in user library format. Loading of required SMDS subroutines can be accomplished in several ways. One way is to execute

```
LIBRARY,SDMSLIB
```

and then carry out the necessary loading operations.

Another way is to use SDMSLIB directly in the loading operation; e.g.

```
LOAD(lfn, SDMSLIB,. . .)
```

or

```
LOADXEQ(F=lfn,U=SDMSLIB,. . .)
```

APPENDIX A

SDMS MATRIX DATA ELEMENT USAGE

Matrix Definition

A matrix data element type has been added to SDMS. This feature permits the definition and use of two-dimensional arrays. Both row and column dimensions may be fixed or variable in size.

The matrix data element syntax in the master definition is as follows:

```
elname  row-dim  col-dim  data-type
```

where

elname	=	data element name (1 to 20 char.)
row-dim	=	integer value of row dimension or name of scalar element containing row dimension value.
col-dim	=	integer value of column dimension or name of scalar element containing column dimension value.
data-type	=	TEXT, REAL, INTEGER, R, I, T.

The following dataset definition illustrates the various possible matrix data element forms.

```
DATASET MAT-DEF
ELEMENT SET
  NROW                    I
  NCOL                    I
  MAT1    NROW  NCOL      R
  MAT2    6     6         R
  MAT3    3     NCOL      R
  MAT4    NROW  4         R
END
END DATASET
```

Matrix Usage

SDMS matrices are stored in the data base in contiguous column-wise order. That is, a 5 by 7 matrix is stored as a succession of 7 columns, each 5 words long, starting with column 1.

This is also the default storage arrangement in central memory. An m by n matrix data element will occupy m*n contiguous addresses in core, in column order a la Fortran. This default arrangement is obtained by simply using the existing map creation calls to SVMAP and DVMAP. This usage is a simple extension of the techniques used for fixed and variable-length vectors.

However, the matrix data element $D_{m,n}$ to be transferred may be a sub-matrix of a larger matrix $M_{mx,nx}$ in central memory, where we assume without loss of generality that $D_{1,1}$ maps into $M_{1,1}$. If m is not equal to mx, the sequence of data map calls must include the following subroutine call:

CALL MATMAP(me_1, cl_1, me_2, cl_2, ...)

where me_i = name of matrix element D.
cl_i = name of variable containing column length of array M corresponding to D.

and $M_{1,1}$ must be associated with $D_{1,1}$ either through static or dynamic variable references.

Note: the value of column length variable cl_i is used at data transfer time and therefore may be changed dynamically as required.

Examples

Suppose we wish to map the matrix element 'MAT2' of dataset MAT-DEF (see previous page) to the array MAT2(6,6). It is only necessary to associate the array MAT2 and the data element 'MAT2' in the data map.

```
CALL DSMAP('M1', dbn, 'MAT-DEF')
CALL SVMAP(MAT2, 'MAT2')
CALL ENDMAP
```

Now suppose that we have an array BIGMAT(30,24) which consists of 20 6 by 6 partitions. We want to dynamically map the data element MAT2 into any one of these partitions. The following map will suffice for this purpose.

```
CALL DSMAP('M2', dbn, 'MAT-DEF')
CALL DVMAP('MAT2')
CALL MATMAP('MAT2', 30)
CALL ENDMAP
```

The following code would output the 6 by 6 partition starting at BIGMAT(7,7).

```
CALL ESPUT('M2', BIGMAT(7,7))
```

Using Vectors As Sub-matrices

Fixed and variable-length vector data elements may also be treated as matrix rows in central memory through the use of the MATMAP call. That is, a vector data element of length n can also be considered to be a 1 by n matrix for data transfer purposes. In the previous example, if 'MAT2' had been defined as a vector of length 6, the ESPUT call would have caused the first row of the 6 by 6 partition starting at BIGMAT(7,7) to be output.

APPENDIX B

INDEXED SEQUENTIAL DATASET SEARCH

Introduction

The search functions described in this section permit the indexed sequential retrieval of selected key sets in a dataset. That is, starting at a specified key set, key sets may be retrieved in order. In the case of random datasets, mapped data elements may be retrieved along with their key sets.

Key sets are kept in a dataset index in ascending order of composite key value. The composite key is the concatenation of key elements in KEY SET definition order, with the first key in the high order position.

In a key set with multiple keys, the composite key ordering causes all key sets with the same key 1 value to be grouped together (see Fig. B-1). Likewise, all key sets with the same key 1 and key 2 values will be grouped together, and so on.

The SMDS indexed sequential search features take advantage of this property by permitting the initial key set for retrieval to be determined by specifying values for major key combinations as shown below:

```
no keys.
key 1.
key 1, key 2.
key 1, key 2, key 3.
        .
        .
        .
key 1, key 2, key 3. ... key n.
```

The initial key set retrieved will be the first one with key values equal to or greater than those specified. In Figure 4-1, if key 1 = TAIL is given, the initial key set retrieved would be key 1 = TAIL, key 2 = 1. The initial key set would be the same if key 1 had been ENGINE, since TAIL is the next highest key 1 value. If no keys are given, the initial key set would be the first one; key 1 = BODY, key 2 = 1.

Initial index positions are established by calls to subroutine BGNDSS. Successive key sets are retrieved by calling GETNXT. Search operations are terminated by calling ENDDDS. Figure B-2 gives a scenario for performing search operations.

Some additional search features are as follows:

* Up to five dataset searches may be active at one time.

* Within the above limit, several searches may be active on the same dataset.

* All SDMS I/O operations may be carried out on any dataset during searches. However, key sets added during a search by functions like ESPUT may or may not be retrieved by GETNXT calls.

RANDOM DATASET NDD

KEY ELEMENTS		DATA ELEMENT
COMPONENT	NETWORK	NETWORK-TYPE
BODY	1	1
BODY	2	2
BODY	3	2
TAIL	1	1
TAIL	2	1
WING	1	1
WING	2	2
WING	3	8
WING	4	8

DATAMAP FOR NDD

```
CALL DSMAP( 'M1', 'NDD', don)
CALL SVMAP(COMP,'COMPONENT',NTWK,'NETWORK')
CALL SVMAP(NWTYP,'NETWORK-TYPE')
CALL ENDMAP
```

Figure B-1: Dataset and Map for Searcn Examples.

```
Start search.
        *
        *<****************
        v                *
Set key values K for     *
     dataset D.          *
        *                *
        *                *
        v                *
Position to first key set *
equal to or greater than  *
K in index on dataset D.  *
        *                *
        *<****************
        v                *
Retrieve next key set in D. *
Optionally retrieve random  *
  dataset data elements.    *
        *                *
        ******************
        v
   Terminate search.
```

Figure B-2: Flow diagram of an indexed sequential search.

Begin Dataset Search (BGNDSS)

* SEARCH dmn WITH nmk KEYS [USING dv*].

CALL BGNDSS(dmn,nmk [,dv_1, . . . ,dv_n])

where

dmn = name of static map for n-key dataset ($n \geq 1$)
nmk = number of major keys supplied ($0 \leq nmk \leq n$)
dv_i = dynamic variables if specified in map dmn.

A call to BGNDSS is used to start a search procedure. Up to five search procedures may be active at one time.

Prior to calling BGNDSS, nmk of the mapped key variables are set. BGNDSS positions a map-associated search pointer at the first key set with key values equal to or greater than those supplied.

For a better understanding of BGNDSS operation, consider Fig. B-1 and the following search operations.

1. To begin a search at COMPONENT = 'TAIL', NETWORK = 2:

```
COMP='TAIL'
NTWK=2
CALL BGNDSS('M1',2)
```

2. To begin searching at the first WING network:

```
COMP='WING'
CALL BGNDSS('M1',1)
```

3. To begin searching at the first network of the first component:

```
CALL BGNDSS('M1',0)
```

4. To search for component NACELLE (which doesn't exist):

```
COMP='NACELLE'
CALL BGNDSS('M1',1)
```

Get Next Operation (GETNXT)

* Get next from dmn.

```
CALL GETNXT(dmn,eoiflg)
```

where

```
      dmn  =  datamap name reference in a preceding BGNDSS call.
   eoiflg  =  .T. if end-of-index detected (output).
```

A call to GETNXT causes the next key values in the index of the mapped dataset to be moved to the mapped key variables. Mapped data elements will also be transferred in the case of random datasets. In the case if direct and sequential datasets, no data elements are transferred.

The following shows the results of the first execution of CALL GETNXT('MI',FLG) for the four examples of the previous section:

	COMP	NTWK	NWTYP	FLG
1.	TAIL	2	1	.F.
2.	WING	1	1	.F.
3.	BODY	1	1	.F.
4.	TAIL	1	1	.F.

In example 4, COMP=TAIL and NTWK=1 because COMPONENT 'NACELLE' was not found and the search pointed was positioned at its insertion position.

End Dataset Search (ENDDSS)

* END SEARCH using dmn.

```
CALL ENDDSS(dmn)
```

where

dmn = datamap name reference in a preceding BGNDSS call.

A call to ENDDSS terminates a search operation using the reference datamap, and releases search buffer space.

It is not necessary to call ENDDSS to initiate a new search using the same map.

APPENDIX C

Qualified Dataset Search

Introduction

A feature has been added to SDMS which extends dataset searching capabilties well beyond those available previously (Appendix B). The indexed sequential search requires the specification of 0 or more major keys to establish the initial position in the key set index. Successive key sets can then be retrieved in ascending order.

The new search capability permits the specification of one or more search ranges for each key in the key set. The basic form of the query specification is as follows: using map M, retrieve keys [and data] where key 1 is in range r_{11} or key 1 is in range r_{12} or ... etc. and key 2 is in range r_{21} or key 2 is in range r_{22} or etc. and etc. Each range r_{ij} for some key j consists of a pair of values (l_{ij}, u_{ij}). All retrieved values will satisfy $l_{ij} \leq$ key j $\leq u_{ij}$.

A wide range of queries can be made within this framework. Suppose we wish to restrict a retrieved integer key to be greater than 40. The corresponding range is $(41,2^{47})$, where 2^{47} is close to the largest integer value possible in FTN Fortran. The range for "less than 40" would be $(-2^{47},39)$. The range for "equals 23" is (23,23). The specification "equals 23 or 27" requires 2 ranges for this key; (23,23) and (27,27).

For example, consider the takeoff data dataset sketched in Figure C-1. This dataset described takeoff performance of a specific airplane as a function of several key parameters: airplane weight, wind velocity, and runway slope and altitude.

```
MASTER DEFINITION PERFMD

 DATASET TAKEOFF
  KEY SET
   WEIGHT            I $ AIRPLANE WEIGHT.
   WIND              I $ WIND VELOCITY. (KNOTS)
   SLOPE             I $ RUNWAY SLOPE. (DEGREES)
   ALTITUDE          I $ RUNWAY ELEVATION. (FEET)
  END
  ELEMENT SET
   TAKEOFF-DIST      R $ TAKEOFF DISTANCE REQUIRED. (FEET)
  END
 END DATASET

END DEFINITION
```

Figure C-1: Takeoff Data Definition.

Suppose we want to retrieve all data for runways at 5000 feet elevation or higher with +2 degree runway slope, for airplane weights of 140K and 160K lb. The range specifications needed are as follows:

Parameter	Ranges
airplane weight	(140K,140K), (160K,160K)
wind velocity	($-\infty$,$+\infty$)
runway slope	(2,2)
runway altitude	(5000,$+\infty$)

Usage

Retrievals are accomplished by making repeated calls to SDMS subroutine GETNXQ. Each successful call retrieves one qualified key set and, in the case of random datasets, its associated mapped values.

Prior to the first call to GETNXQ, range information is entered into labeled COMMON block /QSP/ for use by GETNXQ. The format is:

```
COMMON/QSP/ KEYCNT,KEY(10),LB(5,10),UB(5,10),NR(10)
INTEGER UB
```

where

```
KEYCNT = no. of keys in key set.

for J = 1 to KEYCNT:
  KEY(J) = program variable mapped to dataset key J.
  NR(J) = no. of ranges specified for key J.

  for I = 1 to NR(J):
     LB(I,J) = lower bound of Ith range for key J.
     UB(I,J) = upper bound of Ith range for key J.
```

If real keys are used, then the additional declarations

```
REAL XKEY(10), XLB(5,10), XUB(5,10)
EQUIVALENCE (XKEY,KEY), (LB,XLB), (UB,XUB)
```

will be necessary to avoid type conversion problems when accessing keys and key bounds.

It is of course necessary that LB(I,J) $\leq$ UB(I,J). An additional requirement is that ranges for a given key may not overlap and must be specified in ascending order. Mathematically,

$$UB(I,J) < LB(I + 1,J), \; I < NR(J).$$

The following table gives recommended values for $\pm\infty$ for each of the key types.

key type	$-\infty$	$+\infty$
real	-1.E322	1.E322
integer	-2^{47}	2^{47}
text	0	";;;;;;;;;;"

After the appropriate values are placed in /QSP/ and the logical variable EOI is initialized to .TRUE., a call is made to GETNXQ with the arguments

```
CALL GETNXQ(dmn,eoi)
```

where

dmn = name of previously defined datamap, in which the KEY array of COMMON block /QSP/ must be matched one-to-one to the key elements of the mapped dataset.

eoi = returned .FALSE. if a qualified key set was retrieved from the dataset.
= returned .TRUE. if no more qualified key sets are available.

Note: eoi must be set .TRUE. before the first call to GETNXQ in order to initialize the search.

If eoi = .FALSE. after calling GETNXQ, then the KEY array in /QSP/ holds the next set of qualifying key values. Repeated calls are made to GETNXQ until eoi is returned .TRUE. indicating that no more qualified values exist.

GETNXQ can be used with any of the 3 dataset types: random, direct and sequential. Mapped data element values are returned in the case of random datasets. Only key set values are returned for direct and sequential datasets.

Restrictions

No more than 5 ranges may be specified for each key. GETNXQ can be used with only one map at a time. Also, no indexed sequential search routines (BGNDSS,GETNXT,ENDDSS) should be used with this map until GETNXQ returns eoi = .TRUE. This is because GETNXQ makes use of these routines itself.

Any other SDMS operations (ESPUT, ESGET, etc.) may be carried out during a sequence of GETNXQ calls without interference.

Example

The code on the next page shows how GETNXQ would be used to answer the query used in the Introduction aganst the dataset of Figure C-1. Namely, to retrieve takeoff data for airplanes weighing 140K and 160K lb. on runways with 2 degree upslopes at an elevation of 5000 ft. or higher.

```
      COMMON/QSP/ KEYCNT,KEY(10),LB(5,10),UB(5,10),NR(10)
      INTEGER NR
      LOGICAL EOI

C     SET UP MAP FOR TAKEOFF DATA.
      CALL DSMAP ('MAP', 'TAKEOFF', 'PERF')
      CALL SVMAP (KEY(1), 'WEIGHT', KEY(2), 'WIND', KEY(3), 'SLOPE')
      CALL SVMAP (KEY(4), 'ALTITUDE', TOD, 'TAKEOFF-DIST')
      CALL ENDMAP

C     INITIALIZE /QSP/ FOR SEARCH.
      KEYCNT = 4

C     SET FIRST RANGE FOR KEY 1 (AIRPLANE WEIGHT)
      LB(1,1) = UB(1,1) = 140000

C     SET SECOND RANGE FOR KEY 1.
      LB(2,1) = UB(2,1) = 160000
      NR(1) = 2

C     SET RANGE FOR KEY 2 (WIND VELOCITY)
      LB(1,2) = -2**47 $ UB(1,2) = 2**47 $ NR(2) = 1

C     SET RANGE FOR KEY 3 (RUNWAY SLOPE)
      LB(1,3) = UB(1,3) = 2 $ NR(3) = 1

C     SET RANGE FOR KEY 4 (RUNWAY ALTITUDE)
      LB(1,4) = 5000 $ UB(1,4) = 2**47 $ NR(4) = 1

C     GET QUALIFIED KEY SETS AND ASSOCIATED DATA
      EOI = .TRUE.

10    CALL GETNXQ('MAP',EOI) $ IF(EOI) GO TO 20
      PRINT *, (KEY (I), I = 1,4), TOD
      GO TO 10

C     SEARCH COMPLETE.
20    CONTINUE.
```

Efficiency Considerations

To speed up range searching, keys with one or more of the following characteristics should be earliest in the key set definition:

1. keys with selection ranges which are likely to contain relatively few values.

2. keys which naturally have a small set of possible values to select from. E.g., eye color = (brown, blue, black).

Conversely, keys which have unique (or nearly so) values should be among the last keys mentioned in the key set. An example would be employee number or social security number.

The effect of using these guidelines will be to limit, in many cases, the portion of the index which needs to be traversed during range searches, thus reducing the amount of database I/O performed.

15.0. SDMS CONVERSION

15.1 INTRODUCTION

While much of SDMS is written in ANSI-standard FORTRAN, certain of the requirements for SDMS are not supported by standard FORTRAN. These requirements have led to very specific conversion problems. These problems are characterized by the following areas:

1. Assembly language - portions of SDMS are written in assembly level language (CAL on the CRAY and COMPASS on the CYBERS)

2. Word length - SDMS assumes a word length of 8 characters per word and makes no provisions for double length variables.

3. FORTRAN dialect - SDMS was written in ANSI standard FORTRAN wherever possible; system enhancements on the CRAY and CYBER such as "BUFFER" or "ENCODE/DECODE" statements were not used. The dialect of FORTRAN should not be a problem assuming that the destination machine supports standard FORTRAN and not a subset (For example, the DEC/VAX version of Fortran is a subset.).

4. Variable length calling sequence - SDMS as implemented in PANAIR requires that the calling sequence be variable. That is, the user can call certain routines with as many arguments as required to perform the interface; the called routine determines how many arguments exist in the calling sequence and processes them accordingly.

5. Operating system interface - SDMS requires support from the operating system for functions such as permanent file access, error recovery, etc. Such interfaces must exist on the destination system.

6. Format of calling sequence - SDMS assumes that the FORTRAN compiler assembles a list of addresses for the arguments within the calling sequence and that this list is available to SDMS. This allows the various components of SDMS to access the calling sequences of other routines.

7. Absolute central memory addressing - a practice common to much of SDMS is the concept of central memory addressing from within FORTRAN. This allows SDMS to access variables local to portions of the host program (eg: PANAIR) by knowing the central memory address of the variable.

8. Input/Output - SDMS requires an I/O scheme that can handle variable length random access records. The method must allow for the opening and closing of such files as well as input and output.

The most pervasive problem is item (6) - the assumption of a mechanism whereby subprograms can access each other's calling sequence. Solution of this problem on the CRAY was accomplished by storing the addresses of the calling sequences within central memory. A subsequently called routine accessed actual arguments from a previously called routine by first accessing the address of the previous routine's calling sequence and thereby the addresses of the actual arguments. Two assumptions are implicit in this method:

1. Addresses are storable in central memory words

2. If the address of an array, "ARRAY", is given by "INDA", and the address of the desired variable is "INDV", the desired variable may be referenced by "ARRAY(INDV-INDA+1)".

If these assumptions cannot be satisfied on the target machine, the conversion effort will be extensive. Subprograms that make use of this indirect addressing technique are shown in Table 15.1.

The next most important consideration is the variable-length calling sequence. On the CYBERs, this was handled directly in FORTRAN (As an extension to standard FORTRAN, the dialect implemented on the CYBERs allows a variable- length calling sequence.). On the CRAY, this was handled by coding a CAL interface: The variable- length argument list was handled by the CAL routine, and the FORTRAN routine was given the address of the list that contained the addresses of the argument list. Implementation of a variable length calling sequence is required to avoid extensive changes to the PANAIR program.

Both the operating system interface and the I/O package within SDMS are isolated in the code. Routines requiring an OS interface are shown in Table 15.1. System routines are required to perform the following operations :

- Access permanent files
- Open and close local files
- Control central memory field length
- Control recovery from all errors
- Perform random access, word-addressible I/O

The requirements dictated above to support access of subprogram calling sequences, variable length calling sequences, and system interfaces represent minimum requirements that must be met by the target system to ensure that conversion is feasible.

15.2 MACHINES AND OPERATING SYSTEMS TO WHICH SDMS HAS BEEN CONVERTED

The SDMS package is currently operating or was operating on the following machines and operating systems:

MACHINE	OPERATING SYSTEM
CDC 170	NOS 1
CDC 170	NOS 2
CDC 7600	SCOPE 2
CDC 170	NOS/BE
CRAY 1/S	COS 1.09
CRAY XMP	COS 1.12
DEC 11/780*	VMS
DEC 11/70*	IAS

* MSDMS - a subset of SDMS

Note that the version of SDMS operative on DEC equipment (MSDMS) was generated by recoding from the SDMS design rather than by a true conversion effort. The MSDMS system will NOT support PANAIR without extensive changes made to the PANAIR system.

15.2.1 Conversion of the Code to the CRAY

The SDMS package was converted from CDC CYBER machines to CRAY. Comments were placed in the source code to indicate changes made in the code during the conversion. The routines changed are indicated in Table 15.1. The changes made are highlighted in the source code by surrounding comments. The comment "C** CRAY DEFINED" prefaces such changes, and "C** END CRAY" follows the changes. The conversion required approximately one person-month. Note however, that estimating conversion to other machines based on these changes may be misleading - CYBER and CRAY machines are architectured very similarly; conversion between the two machines is often elementary compared to other conversion efforts. Conversion of SDMS to another machine would be extremely difficult if the minimum requirements outlined in section 15.1 were not met.

15.3 SUMMARY OF CONVERSION PROBLEMS BY SUBPROGRAM

Table 15.1 and 15.2 summarize conversion problems within SDMS by routine. Table 15.1 summarizes the SDMS library routines and Table 15.2 summarizes the Data Definition Processor (DDP). Note that columns one through eight correspond to the difficulties noted in section 15.1; column nine indicates conversion effort required during conversion to CRAY.

Some confusion has arisen from the term "MASHCII code" in the CRAY version of SDMS. The technique consists in mapping the eight-bit ASCII codes into a six-bit hash. This six-bit hash allows storage of ten characters of data into a single CRAY word. In this discussion, the use of this technique has been included under the heading "word length".

15.3 PURPOSE OF ASSEMBLY LANGUAGE ROUTINES

The following routines have been written in assembly language (CAL) to accommodate a requirement not allowed in FORTRAN

NAME	..... PURPOSE .....
BGNDSS	Accommodates variable-length calling sequence for BGNDSF
CINFO	Returns name and line of calling routine
DESGET	Accommodates variable-length calling sequence for DESGTF
DESPOR	Accommodates variable-length calling sequence for DESPRF
DESPUT	Accommodates variable-length calling sequence for DESPTF
DESREP	Accommodates variable-length calling sequence for DESRFF
DVMAP	Accommodates variable-length calling sequence for DVMAPF
ESGET	Accommodates variable-length calling sequence for ESGETF
ESPUT	Accommodates variable-length calling sequence for ESPUTF
ESREP	Accommodates variable-length calling sequence for ESREPF
ESSGET	Accommodates variable-length calling sequence for ESSGTF
ESSPUT	Accommodates variable-length calling sequence for ESSPTF
IUS	Returns user number of running job
KSDEL	Accommodates variable-length calling sequence for KSDELF
MATMAP	Accommodates variable-length calling sequence for MATMPF
SVMAP	Accommodates variable-length calling sequence for SVMAPF

TABLE 15.1 - CONVERSION TASKS GROUPED BY SUBPROGRAM

NAME	1	2	3	4	5	6	7	8	9
ADF				X					X
AKE*							X		
ARES				X			X		X
ARRES							X		
ASP		X			X				
ASPX									
ATE									
ATTACH				X	X				X
BGNDSF				X			X		X
BGDNSS	X								
BKE		X					X		X
BMKE*		X					X		X
BUFSIZ									
CALLPT*									X
CATLOG				X	X				X
CDMT		X					X		X

COLUMN LEGEND:

1. Routine written in assembly language (CAL)
2. Word length dependant
3. Non-standard FORTRAN
4. Variable length calling sequence
5. Operating System interface
6. Calling sequence format dependant
7. Absolute central memory dependant
8. Input/Output dependant
9. Conversion effort required for CYBER/CRAY conversion

* While a part of SDMS, this routine is not used by PANAIR

TABLE 15.1 - CONVERSION TASKS GROUPED BY SUBPROGRAM (Continued)

NAME	1	2	3	4	5	6	7	8	9
CINFO	X								
CLS1							X		
CMD									
CPB*							X		
DBABT*					X				
DBADJ*				X			X		X
DBCLOS					X		X		X
DBOPEN				X	X				X
DBPURG			X		X				X
DBTSET									
DDF				X					X
DESGET	X								
DESGTF				X			X		X
DESPOR	X								
DESPRF				X			X		
DESPUT	X								X

COLUMN LEGEND:

1. Routine written in assembly language (CAL)
2. Word length dependant
3. Non-standard FORTRAN
4. Variable length calling sequence
5. Operating System interface
6. Calling sequence format dependant
7. Absolute central memory dependant
8. Input/Output dependant
9. Conversion effort required for CYBER/CRAY conversion

* While a part of SDMS, this routine is not used by PANAIR

TABLE 15.1 - CONVERSION TASKS GROUPED BY SUBPROGRAM(Continued)

NAME	1	2	3	4	5	6	7	8	9
DESREP	X								X
DKE*									
DSMAP				X					X
DVMAP	X								X
DVMAPF							X		X
ENDDSS*							X		
EMDMAP									X
EREXIT		X							
ERRCLR									
ERRST				X					X
ESDG*									
ESFL				X	X		X		
ESGET	X			X					X
ESGETF				X			X		X
ESPORF				X			X		X
ESPUT	X			X					X

COLUMN LEGEND:

1. Routine written in assembly language (CAL)
2. Word length dependant
3. Non-standard FORTRAN
4. Variable length calling sequence
5. Operating System interface
6. Calling sequence format dependant
7. Absolute central memory dependant
8. Input/Output dependant
9. Conversion effort required for CYBER/CRAY conversion

* While a part of SDMS, this routine is not used by PANAIR

TABLE 15.1 - CONVERSION TASKS GROUPED BY SUBPROGRAM (Continued)

NAME	1	2	3	4	5	6	7	8	9
ESPUTF				X			X		X
ESREP	X			X					X
ESREPF				X			X		X
ESSCLS				X			X		X
ESSGET*	X			X					X
ESSGTF*				X			X		X
ESSOPN*				X			X		
ESSPOS*				X					
ESSPUT*	X								X
ESSPTF*				X			X		X
FALP		X					X		X
FARP							X		
FDOESS							X		
FDP							X		
FDVA		X		X			X		
FESD				X			X		

COLUMN LEGEND:

1. Routine written in assembly language (CAL)
2. Word length dependant
3. Non-standard FORTRAN
4. Variable length calling sequence
5. Operating System interface
6. Calling sequence format dependant
7. Absolute central memory dependant
8. Input/Output dependant
9. Conversion effort required for CYBER/CRAY conversion

* While a part of SDMS, this routine is not used by PANAIR

TABLE 15.1 - CONVERSION TASKS GROUPED BY SUBPROGRAM (Continued)

NAME	1	2	3	4	5	6	7	8	9
FILE							X		
FKBI									
FKE									
FNESL		X					X		X
FUDE		X				X	X		X
FVKS									
GETNXQ*									
GETNXT*							X		
GKB							X		
GKB1*							X		
GTS		X		X	X		X		
ICPRZ									X
IDBFD							X		
IDMT				X			X		X
IETB		X							
IFFN		X							

COLUMN LEGEND:

1. Routine written in assembly language (CAL)
2. Word length dependant
3. Non-standard FORTRAN
4. Variable length calling sequence
5. Operating System interface
6. Calling sequence format dependant
7. Absolute central memory dependant
8. Input/Output dependant
9. Conversion effort required for CYBER/CRAY conversion

* While a part of SDMS, this routine is not used by PANAIR

TABLE 15.1 - CONVERSION TASKS GROUPED BY SUBPROGRAM (Continued)

NAME	1	2	3	4	5	6	7	8	9
IKDB				X			X		
IKE									
IMES		X		X			X		X
INDB					X				X
INDXZ									
IODB					X				X
ISDMS		X			X		X		X
ISDT*				X					
ISES*		X		X			X		X
IUS	X								X
KRT*									
KSDEL*	X			X					X
KSDELF*				X			X		X
LGTHL		X							
LGTHW		X							
LOCF*					X		X		X

COLUMN LEGEND:

1. Routine written in assembly language (CAL)
2. Word length dependant
3. Non-standard FORTRAN
4. Variable length calling sequence
5. Operating System interface
6. Calling sequence format dependant
7. Absolute central memory dependant
8. Input/Output dependant
9. Conversion effort required for CYBER/CRAY conversion

* While a part of SDMS, this routine is not used by PANAIR

TABLE 15.1 - CONVERSION TASKS GROUPED BY SUBPROGRAM (Continued)

NAME	1	2	3	4	5	6	7	8	9
LSDB*					X		X		
MAPCHK*									
MAPERR*									
MATCH1									
MATMAP	X								X
MATMPF									X
MDB									
MMRU									
MONSIO*									
NORCVR									
ODBFD					X		X		
OKB					X		X		
OKBTT									
OMES		X					X		X
OPENRM					X				X

COLUMN LEGEND:

1. Routine written in assembly language (CAL)
2. Word length dependant
3. Non-standard FORTRAN
4. Variable length calling sequence
5. Operating System interface
6. Calling sequence format dependant
7. Absolute central memory dependant
8. Input/Output dependant
9. Conversion effort required for CYBER/CRAY conversion

* While a part of SDMS, this routine is not used by PANAIR

TABLE 15.1 - CONVERSION TASKS GROUPED BY SUBPROGRAM (Continued)

NAME	1	2	3	4	5	6	7	8	9
OSES*		X					X		X
PBS									
PFLET		X					X		X
PFNDB									
PKBB*									
PKCHR		X							X
PKET				X			X		
PLA		X				X	X		X
PLA1		X				X	X		
PMET		X		X			X		X
PMPT*									
PMVT*									X
PPT							X		X
PRDMT*									
PSET		X					X		X
PSPT*									

COLUMN LEGEND:

1. Routine written in assembly language (CAL)
2. Word length dependant
3. Non-standard FORTRAN
4. Variable length calling sequence
5. Operating System interface
6. Calling sequence format dependant
7. Absolute central memory dependant
8. Input/Output dependant
9. Conversion effort required for CYBER/CRAY conversion

* While a part of SDMS, this routine is not used by PANAIR

TABLE 15.1 - CONVERSION TASKS GROUPED BY SUBPROGRAM (Continued)

NAME	1.	2	3	4	5	6	7	8	9
PVLET									X
RDBF									
RDBUF		X		X	X		X	X	X
RNDB*							X		X
RNKB				X			X		X
RSDB*				X			X		
RTNPRT*									X
RTS		X							X
SETAEF									
SFKB				X					
SFP									X
SKBI									
SPB		X			X				
SVMAP	X								X
SVMAPF				X	X		X		X
SYSTEM					X				X

COLUMN LEGEND:

1. Routine written in assembly language (CAL)
2. Word length dependant
3. Non-standard FORTRAN
4. Variable length calling sequence
5. Operating System interface
6. Calling sequence format dependant
7. Absolute central memory dependant
8. Input/Output dependant
9. Conversion effort required for CYBER/CRAY conversion

* While a part of SDMS, this routine is not used by PANAIR

TABLE 15.1 - CONVERSION TASKS GROUPED BY SUBPROGRAM (Concluded)

NAME	1	2	3	4	5	6	7	8	9
TFB				X					
TFSDB*				X					
TIME					X				X
TMFSDB*				X					
TNTSDB*				X					
TRACER*									
TTB				X			X		
TTSDB*									
TYPSET		X							X
UDBFD									
UKBT									
UPKCHR		X							X
USDT*				X					
WSDB				X			X		
WTBUF		X		X	X		X	X	X

COLUMN LEGEND:

1. Routine written in assembly language (CAL)
2. Word length dependant
3. Non-standard FORTRAN
4. Variable length calling sequence
5. Operating System interface
6. Calling sequence format dependant
7. Absolute central memory dependant
8. Input/Output dependant
9. Conversion effort required for CYBER/CRAY conversion

* While a part of SDMS, this routine is not used by PANAIR

TABLE 15.2 - DDP CONVERSION TASKS GROUPED BY SUBPROGRAM

NAME	1	2	3	4	5	6	7	8	9
BEST									
BKT									
BPT									
BST									
CVED									
DDP									
ERRST									
FATAL									
FNT									
ILT									
INDEX									
INITP		X							X
INTGR									
IST							X		X
LFTCH									
ODD									

COLUMN LEGEND:

1. Routine written in assembly language (CAL)
2. Word length dependant
3. Non-standard FORTRAN
4. Variable length calling sequence
5. Operating System interface
6. Calling sequence format dependant
7. Absolute central memory dependant
8. Input/Output dependant
9. Conversion effort required for CYBER/CRAY conversion

TABLE 15.2 - DDP CONVERSION TASKS GROUPED BY SUBPROGRAM (Continued)

NAME	1	2	3	4	5	6	7	8	9
ODFD								X	X
ODT									
OSD									
PASD									
PCIT									
PMDF									
PMDN		X							X
PSD		X							X
RFD									
RNC									
SESD									
SFDDS									
SFDE									
SMLP									
SNIT									
SRSD									

COLUMN LEGEND:

1. Routine written in assembly language (CAL)
2. Word length dependant
3. Non-standard FORTRAN
4. Variable length calling sequence
5. Operating System interface
6. Calling sequence format dependant
7. Absolute central memory dependant
8. Input/Output dependant
9. Conversion effort required for CYBER/CRAY conversion

TABLE 15.2 - DDP CONVERSION TASKS GROUPED BY SUBPROGRAM (Concluded)

NAME	1	2	3	4	5	6	7	8	9
SSEP			X						X
SSME			X						X
STNC									
TOKEQ									
UFLET									
UMET									
USET									
UVLET									

COLUMN LEGEND:

1. Routine written in assembly language (CAL)
2. Word length dependant
3. Non-standard FORTRAN
4. Variable length calling sequence
5. Operating System interface
6. Calling sequence format dependant
7. Absolute central memory dependant
8. Input/Output dependant
9. Conversion effort required for CYBER/CRAY conversion

16.0 SOFTWARE GLOSSARY

Key Word	Description
Abutment	A curve where two or more network edges (exactly or approximately) meet.
Abutment Intersections	Points where several abutments meet.
Account numbers	Computing center cost accounting labels.
Address	The software identification of a word in central memory.
Array	A collection of contiguous words in central memory.
B (Outer) Spline	A matrix which gives the value of source or doublet strength at panel grid points in terms of surrounding singularity parameters.
BP-Spline	A row vector giving a flow quantity at a grid point in terms of the flow quantities at surrounding control points.
Block Partition Format	The arrangement of a coefficient matrix as a collection of rectangular submatrices.
Buffer	An area of storage which temporarily holds data that will be subsequently delivered to a processor.
CAL	The CRAY Assembly Language.
Calling relationship	The set of all subprograms invoked by a program unit.
CFT	A procedure-oriented language supported by CRAY compilers.
Closure Condition	A non standard boundary condition imposed to insure a design network edge will remain unchanged after the geometry has been relofted.
Communication Vehicle	A method of data transfer between subprograms.
Compilation	The translation of a high level source language, like CFT, into machine language.

Key Word	Description
Compressibility Direction	The direction of freestream flow in the Prandtl-Glauert equation. It is defined by the input terms "CALPHA" and "CBETA."
Compressible Inner Product	An inner product with respect to the compressibility coordinate system.
Constraints	Right-hand-side values for boundary condition equations.
Control Card Stream	A sequence of control statements.
Control Statement	A user instruction to the operating system.
Core	Semi-conductor memory which is manipulated by the central processing unit.
COS	CRAY Operating System.
CPU.(Central Processor Unit)	Elements of a data processing system that carry out a variety of essential data manipulations and controlling tasks.
CRAY 1-S, X-MP, 1-M	CRAY Research data processing systems.
Data Base Communication Chart	A tabular listing which correlates datasets and the subprograms which use them.
Data Base Directive	A user directive which may specify the file identification parameters for the PAN AIR databases and the master definitions.
Data Base Information Table	A tabular listing of the specifications made by the data base directives.
Data Base Management System	A piece of software which manages data bases in direct access storage.
Data Base status	The completeness of the information in a data base.
Data flow	The relationship of the output data of one program to the input data of another program.

Key Word	Description
Dataset	A collection of element sets and their associated key sets.
Design Code	See pseudo code.
Diagnostic message, warning message	Program identification of an abnormality detected during execution which will not result in program termination.
Disk	A computer storage medium external to the CPU.
Element	The basic informational unit of an SDMS data base.
Element Set	A well defined collection of elements.
'end of record' card	The delimiter between sections of a job input file.
Executable Code	FORTRAN statements which specify actions the program is to take.
Execution	The operation of the CPU under control of a program.
Execution time	The wall clock time at which a program is in execution.
Executive Directive	A user directive which specifies the type of PAN AIR analysis.
Executive Module	The component of a software system which controls the execution of other system components.
Fatal error	An abnormality detected by the program during execution which results in program termination.
Flow quantity	Surface potential, velocity or normal mass flux.
Formal Parameters	Arguments which appear in calling sequence of SUBROUTINE or a FUNCTION.
Free field format	The interpretation of program input by its content instead of position.

Key Word	Description
Functional Decomposition	The breakdown of a major computing task into basic computing functions.
Glossary	A section of the program preface which describes program variables.
Heterogeneous	The condition of the specified flow data set of the DIP data base when smearing has not been employed.
Homogeneous	The condition of the specified flow data set of the DIP data base after smearing has been employed.
IC Matrix	A matrix giving one or more field flow properties as a linear combination of the array of singularity parameters.
Input	Data used downstream from a given PAN AIR module.
JCL (Job Control Language)	The criteria for defining the set of all syntactically correct control statements.
Key	An element set identifier.
Key Set	A collection of keys which uniquely identify an element set.
Library	See program library.
Load	Transform a program held on some external storage medium into the main memory of the machine in a form suitable for execution.
Macro-options	A data set of the MEC data base which will inform all downstream PAN AIR modules of an IC-update, solution update or post-solution run.
Main program	A program which is not a subprogram.
Main Overlay	The overlay which is loaded initially and remains in core.
Maintainability	Resilience to internal changes
Map	A correlation between SDMS dataset elements and program variables.

Key Word	Description
Masking	A bit by bit logical operation on one or more words in central memory.
Master Definition File	A collection of data records which defines the structure of a permanent/temporary data base.
MEC Directives	Data base directives and executive directives.
Modular Code	Software which has localized the impact of changes in its operating environment.
Module	One of the ten basic programs of the PAN AIR system.
Operating System	The computer system software that assists the hardware to implement various supervisory and control functions it performs for the tasks created by the users.
Out-of-core Matrix Multiplication	The computation of the product of two out-of-core matrices (stored on SDMS data bases).
Output	Data used downstream from a given PAN AIR module.
Overlay	A portion of a program written on a file in absolute form and loaded at execution time without relocation.
PAN AIR Problem	The computation of a numerical solution to the Prandtl-Glauert equation and boundary condition equations over a surface configuration.
Permanent (Temporary) Data Base	A well defined data structure, generated by a particular PAN AIR module, which will (not) remain accessible after the job has run to completion.
Plot data file	Input data to plotting software.
Preface	Software documentation presented as comment statements at the beginning of each PAN AIR module.

Key Word	Description
Primary Overlay	An overlay which may be called into core only by the main overlay and is loaded immediately following the main overlay.
Procedure	A collection of control statements, separate from the job control statement section, that may be called by a control statement.
Procedure File	A collection of data records which may be called as a procedure.
Program	A collection of FORTRAN statements, with optional comments, terminated by an END statement.
Program Library	A collection of computer programs made available to computer users to reduce the work of programming.
Program Tree Structure	The schematic representation of calling relationships between subprograms of a module.
Pseudo Code	A user-defined, non compilable shorthand for use in defining the flow of a program segment.
Secondary Overlay	An overlay which may be called into core only by a primary overlay and is loaded immediately following the primary overlay.
Smearing	The application of a single specified flow value to a subset of control points.
Software System	An integrated collection of programs which perform a major computing task.
Solution data	Basic flow quantities associated with a particular set of right-hand-side equality constraints.
Stand-alone program	A program module which may be executed independent from other modules.
Structured Programming	Software development which has employed disciplined program organization and notation to facilitate correct and

Key Word	Description
	clear descriptions of data and control structures.
System Architecture	The construction of a computing system by assembling basic modules.
Submodule	A subprogram of a PAN AIR module.
Subprogram	A program unit that begins with a SUBROUTINE, FUNCTION or BLOCK DATA statement.
Subroutine	A subprogram unit that begins with a SUBROUTINE statement.
Symmetrize	Transform a large system of linear equations into smaller systems of linear equations, by using symmetric properties of the coefficient matrix.
Transportability	Resilience to external changes.
Tree Diagram	See program tree structure.
Unsymmetrize	Transform the solutions of symmetrized systems of linear equations into the solution of the original system.
User Directives	A collection of user specifications which define a particular PAN AIR problem and its computing environment.

17.0 REFERENCES

1. Magnus, Alfred E.; and Epton, Michael A.: PAN AIR Volume I – Theory Document (Version 1.0), NASA CR-3251, 1980.

2. Sidwell, Kenneth W.; Baruah, Pranab K.; and Bussoletti, John E.: PAN AIR Volume II – User's Manual (Version 1.0), NASA CR-3252, 1980.

3. Medan, Richard T. (Editor); Magnus, Alfred E.; Sidwell, Kenneth W.; and Epton, Michael A.: PAN AIR Volume III – Case Manual (Version 1.0), NASA CR-3253, 1981.

4. Sidwell, Kenneth W.; and Derbyshire, Thomas: PAN AIR Summary Document (Version 1.0), NASA CR-3250, 1982.

5. UPDATE Reference Manual, SR-0013, CRAY Research, Inc.

6. Businger, P.; and Golub, G. H.: "Linear Least Squares Solutions by Householder Transformations," in the "Handbook for Automatic Computation, Volume II," Springer Verlag, New York, 1971, pp. 111-118.

7. CRAY-OS Reference Manual, SR-0011, CRAY Research, Inc.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA CR-3254	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle PAN AIR–A Computer Program for Predicting Subsonic or Supersonic Linear Potential Flows about Arbitrary Configurations Using a Higher Order Panel Method Volume IV–Maintenance Document (Version 3.0)		5. Report Date January 1990
		6. Performing Organization Code
7. Author(s) David J. Purdon, Pranab K. Baruah, John E. Bussoletti, Michael A. Epton, William A. Massena, Franklin D. Nelson, and Kiyoharu Tsurusaki		8. Performing Organization Report No.
		10. Work Unit No.
9. Performing Organization Name and Address Boeing Military Airplane Company P.O. Box 3707 Seattle, Washington 98124		11. Contract or Grant No. NAS2-12036
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration, Washington, DC 20546-0001 AFWAL and ASD, Wright-Patterson, AFB, Ohio 45433 NCSC, Panama City, Florida 32407		13. Type of Report and Period Covered Contractor Report Oct. 1984–Sept. 1987
		14. Sponsoring Agency Code

15. Supplementary Notes

Point-of Contact: Larry L. Erickson, Ralph L. Carmichael, Michael D. Madson, and Alex C. Woo
Ames Research Center, MS 227-2, Moffett Field, CA 94035-1000
(415) 604-5856 or FTS 464-5856

16. Abstract

The Maintenance Document Version 3.0 is a guide to the PAN AIR software system, a system which computes the subsonic or supersonic linear potential flow about a body of nearly arbitrary shape, using a higher order panel method. The document describes the overall system and each program module of the system. Sufficient detail is given for program maintenance, updating and modification. It is assumed that the reader is familiar with programming and CRAY computer systems.

The PAN AIR system was written in FORTRAN IV language except for a few CAL language subroutines which exist in the PAN AIR library. Structured programming techniques were used to provide code documentation and maintainability. The operating systems accommodated are COS 1.11, COS 1.12, COS 1.13, and COS 1.14 on the CRAY 1S, 1M, and X-MP computing systems.

The system is comprised of a data base management system, a program library, an execution control module and nine separate FORTRAN technical modules. Each module calculates part of the posed PAN AIR problem. The data base manager is used to communicate between modules and within modules. The technical modules must be run in a prescribed fashion for each PAN AIR problem. In order to ease the problem of supplying the many JCL cards required to execute the modules, set of CRAY procedures (PAPROCS) was created to automatically supply most of the JCL cards.

Most of this document has not changed for Version 3.0. It now, however, strictly applies only to PAN AIR version 3.0. The major changes are: (1) additional sections covering the new FDP module (which calculates streamlines and offbody points), (2) a complete rewrite of the section on the MAG module, and (3) strict applicability to CRAY computing systems.

17. Key Words (Suggested by Author(s)) Aerodynamics, Potential flow, Panel methods, Influence coefficients, Data base, Functional decomposition, Splines, Tree diagram, Master definition, JCL, CRAY		18. Distribution Statement [redacted] Subject Category - 02	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 979	22. Price

NASA FORM 1626 OCT 86