NASTRAN Interfacing Modules Within the Integrated Analysis Capability (IAC) Program

by Harold P. Frisch

NASA / Goddard Space Flight Center

SUMMARY

The IAC program provides the framework required for the development of an extensive multidisciplinary analysis capability. Several NASTRAN related capabilities have been developed which can all be expanded in a routine manner to meet in-house unique needs. Plans are to complete the work discussed herein and to provide it to the engineering community through COSMIC in early 1987. Release is to be after the current IAC Level 2 contract work on the IAC executive system by Boeing Aerospace Company is completed and meshed with the interfacing modules and analysis capabilities under development at the GSFC.

INTRODUCTION

The Integrated Analysis Capability (IAC) program (ref. 1) has been under development at Boeing Aerospace Corporation (BAC) and the Goddard Space Flight Center (GSFC) since July 1979. During the development phase, several papers and seminars have been presented which define plans and usage at BAC and the GSFC, refs. 2-9. The first public release of the IAC through COSMIC was in July 1985 and is referred to as IAC Level 1.5. It contains most of the interfacing modules discussed herein, the others will be included in the next major update planned for spring 1987, IAC Level 2.0. The primary objective of the development team has been to create a computational environment in which multidisciplinary analysis can be carried out in a manner which makes optimum use of existing tried and true analysis procedures. Rather than create a family of new untested multidiscipline analysis programs, the IAC attempts to provide users with the computational tools required to generate, catalog, manipulate, query, and process data, and then to pass it from one program to another. This approach allows analysts to make use of well tested programs that they understand and trust. They must only convince themselves that the IAC data transfer and processing capabilities work to their satisfaction.

The initial thrust of the IAC development effort has been in the areas of structures, controls, thermal, and system dynamics. NASTRAN in its generic form has always been thought of as the prime analysis module to be used for structural analysis. For thermal analysis, NASTRAN is thought of as an option to be used if desired.

The objective of this paper is threefold: first to define what an IAC database and datastructure is and how they play a pivotal role in the ability to store, query, and process data; second to define purpose, function, and outline background theory for the DMAP programs and DMAP alters used to instruct NASTRAN to pass statics, dynamics, and thermal data to the outside world; lastly to define purpose, function, and outline background theory for the interfacing programs which read NASTRAN generated binary files, create the formal IAC datastructures, and then put them into an IAC database for follow-on multidiscipline analysis.

IAC DATASTRUCTURES AND DATABASES

Multidiscipline analysis to a major extent relies on the ability to pass data from one analysis module to another in a manner which is compatible with the output generation limitations of the first and the input requirements of the second. To the uninitiated, this is usually passed off as a trivial programming task; real workers know that the road to task completion can be exceedingly difficult and time-consuming. To ease the task, the IAC provides two formal datastructures: the RELATION and the ARRAY. The RELATION datastructure is used to store all data which is most naturally defined in a tabular format, the ARRAY datastructure is used to store all data which is most naturally defined as either a 2 or higher dimensional array. Within each datastructure, no restrictions are placed on the elements; they may be dimensioned quantities of any type. The datastructures are supported by an assortment of utilities which provide creating, loading, cataloguing, querying, getting, and putting capabilities. Getting and putting capabilities may be used in a variety of ways. At the highest level, interactive IAC commands are used; on a lower level, calls may be inserted into programs or subroutines which get datastructures from and put datastructures into an IAC database.

Physically, an IAC database is nothing more than a collection of user created IAC datastructures residing as binary files in a particular system subdirectory. In addition to these files, the subdirectory also contains the catalogue file IACCAT.IAC and the activities file IACACT.IAC of the database. The catalogue file is a RELATION datastructure which may be queried via IAC interactive commands. It contains the table of contents for the database along with other information such as IAC file name, title, keywords, creator, date of creation, etc. The activities file is created by the IAC executive and is used by it to implement the IAC's concurrent access capability, i.e., the ability for several users to simultaneously open the same database and access datastructures within it. There is no limit to the number of IAC databases that may reside in a user's account or across the entire system. The only restriction is that each database must reside by itself within a particular system subdirectory. Users are responsible for keeping track of use and purpose of each IAC database that they use and create. If project level IAC databases are established, users are responsible for knowing their location and purpose. The problem of keeping track of many databases in 1 or many systems is currently an active area of database management research and development. We are currently not adding that type of capability to the IAC; we are focusing on developing

software required to support computational analysis in a multidisciplinary project support environment. Conversely, we are cognizant of distributed database management systems and their ability to find, retrieve, and display data spread over many systems; we are of the opinion that the IAC can provide the high level analysis capability that these normally do not have..

Formal datastructures within the IAC are something more than just an ordered sequence of numeric and character data. Four examples are provided to illustrate the fact that they are designed in such a manner that they contain all information normally available from an annotated line printer listing. In addition to simply containing raw data, each datastructure has an accompanying descriptor. It contains all information needed by both system and user to query and then to output datastructure data to a user specified data file in a format compatible with user unique needs.

As an illustration, let TAPROD.NID be a bulk data file containing PARAM cards for the DMAP program NASDS. The IAC command

```
RUN CNASTRAN(F=TAPROD,RFA=NASDS)
```

is used to run the COSMIC/NASTRAN job. The output of the NASTRAN job is the standard .F06 file along with an OUTPUT2 file which is automatically placed in the file TAPROD.IO2. This OUTPUT2 file is to be read and processed by the interface module CINDA2. The IAC command

```
RUN CINDA2(FN=TAPROD.IO2,D=TAPROD:3)
```

will run the job. The program CINDA2 reads the OUTPUT2 file and automatically creates IAC datastructures for all recognized data blocks in the OUTPUT2 file TAPROD.IO2. Each datastructure created is given the file name TAPROD:3 when it is put into the database and catalogued. Once datastructures are in the database, users may query them and print all or part of their contents. They can also be used for follow-on analysis by any program containing the appropriate software commands. It should be noted that all IAC modules are run in a similar manner. RUN says run an IAC analysis module, the module name follows with a list of what we refer to as "run parameters." Run parameters allow users to communicate with either the module's command procedure or its executable. This approach is used to insulate users from all job control command procedures.

The following 2 examples of a RELATION datastructure descriptor are created by the interactive IAC command:

```
SHOW FILE TAPROD:3.LAMA, TAPROD:3.GPWG

DATASTRUCTURE=TAPROD:3.LAMA;1, CLASS=RELATION
TUPLES     NAME        DIMENSIONS TYPE   FORMAT
      75 EIGVAL                    R1     1X,E15.5
         RAD_SEC                   R1     1X,E15.5
         HZ                        R1     1X,E15.5
         GEN_MASS                  R1     1X,E15.5
         GEN_STIFF                 R1     1X,E15.5
```

```
DATASTRUCTURE=TAPROD:3.GPWG;1, CLASS=RELATION
TUPLES      NAME        DIMENSIONS TYPE  FORMAT
          1 REF_LOC     3          R1    1X,E11.4
            BODY_INER   6,6        R1    1X,E11.4
            S_TRANS     3,3        R1    1X,E11.4
            MASS_CG     4,3        R1    1X,E11.4
            CG_INER     3,3        R1    1X,E11.4
            PRN_INER    3          R1    1X,E11.4
            Q_TRANS     3,3        R1    1X,E11.4
```

Both examples were created automatically by the IAC interfacing program CINDA2. This program is designed to read an OUTPUT2 file containing the standard NASTRAN real eigenvalue table LAMA, the grid point weight generator table OGPWG, and many other tables and matrices generated during modal synthesis analysis via the DMAP program NASDS. The same datastructures can also be created by the module CINSA2 which is designed to read all data blocks that can be created by case control for rigid formats 1 and 3.

RELATIONs (tables) are characterized by tuples (rows) and attributes (columns). From the above examples, it should be obvious that the descriptor contains both the DATASTRUCTURE file name and its CLASSification (RELATION, ARRAY). To support more fully the needs associated with day-to-day parameter variation studies, we have found it most useful to introduce ":number" and to allow the IAC datastructure file name to be of the form

```
name:number.type;version
```

where:

```
number  - 10 digit integer (optional, default is 1)
type    - 10 character alphanumeric
version - 10 digit integer (optional)
```

The other data items contained in the descriptor are:

TUPLES - total number of tuples (rows) in the relation.

NAME - descriptive name to be associated with each attribute. It is convenient to utilize easily recognizable acronyms; in particular, one can notice that we have consistently tried to use NASTRAN .F06 file labels.

DIMENSIONS - actual dimension of data item. The above examples show scalar (blank) and multidimensional arrays. Variable single and multidimensional arrays are also allowed; these are symbolized by the * character.

TYPE - data type (I1-integer, R1-single precision real, R2-double precision, Z2-single precision complex, Z4-double precision complex, C*-variable length character string, Cn-character string of length n, L1-logical).

FORMAT - default output display format.

The following 2 examples of an ARRAY datastructure descriptor are created by the interactive IAC command:

```
SHOW FILE TAPROD.PHIG, TAPROD:4.EIGV
```

```
DATASTRUCTURE=TAPROD:1.PHIG;1, CLASS=ARRAY
INDEX  TUPLES     NAME        DIMENSIONS TYPE  FORMAT
0             5940 PHIG                    R2    1PD15.5
1              396 IN_DOF                  I1    1X,I10
                   EXT_GP                  I1    1X,I10
                   INT_GP                  I1    1X,I10
                   DOF                     C2    8X,A
2               15 CMID                    I1    1X,I10

DATASTRUCTURE=TAPROD:4.EIGV;1, CLASS=ARRAY
INDEX  TUPLES     NAME        DIMENSIONS TYPE  FORMAT
0             2100 MODES                   R2    1X,E15.5
1              140 COMPONENT               C8    3X,A8
                   EX_GRID_ID              I1    6X,I5
                   DOF                     C2    9X,A2
                   SIL_ID                  I1    6X,I5
2               15 MODE_ID                 I1    6X,I5
                   EIGVAL                  R2    1X,E15.5
                   RAD_SEC                 R2    1X,E15.5
                   HZ                      R2    1X,E15.5
```

The above 2 examples of an ARRAY datastructure descriptor were created to store eigenvector data. In the first example, the objective is to store the data block PHIG from the functional module SDR1, g-set eigenvectors along with sufficient information to associate rows with both internal/external grid point and DOF numbering sequences and columns with mode identification numbers. The objective of the second is to store eigenanalysis data obtained outside of NASTRAN using standard a-set mass and stiffness matrices or using the reduced order mass and stiffness matrices contained in a substructure operating file (SOF). In the latter case, eigenvector data associated with a substructure may be composed of physical and modal degrees of freedom which may span several components. INDEX 1 attributes provide both component and internal/external number sequence information. INDEX 2 attributes provide eigenvalue information. The name:number.EIGV datastructure is created by the module CIDRMX which reads all necessary input data directly from user specified datastructures. Again, note how easily recognized acronyms are used to define attributes. The only other data item contained in the ARRAY descriptor not found in that of the RELATION is:

INDEX - The ARRAY may be multidimensional; this is the identifier associate with each dimension. It should be noted that, in effect, a RELATION is associated with each index. In the above example, INDEX 0 is the core; the number of tuples here defines the total number of elements in the eigenvector matrix. INDEX 1 is asso-

ciated with the rows; a relation is provided to give more meaning to each row. INDEX 2 is associated with the columns; a relation is provided to give more meaning to each column. In general, there is no restriction on the number of INDEX's used. For each INDEX, TUPLES defines the number of tuples (rows) in the INDEX's associated relation.

For both the RELATION and ARRAY datastructure, the IAC provides an assortment of query and data manipulation utilities. Queries may be performed by using an assortment of relation operators (LT,LE,EQ,GE,GT,...) along with datastructure attribute names. We currently have a rather basic X,Y plot capability and a database management system (DBMS) designed primarily to support multidiscipline analysis needs. Our intent is to allow users to make use of the extensive plot and distributed DBMS capabilities currently available elsewhere. We provide the query capability to get the resultant analysis data needed to plot or store. At the GSFC, plot and DBMS programs are viewed as just other analysis modules which can be used via the IAC interactive command RUN.

INTERFACING CAPABILITY

NASTRAN users have at their disposal an extremely versatile program which can be used to solve standard problems via a host of rigid formats and non-standard problems via specially designed Rigid Format alters and DMAP programs. The interfacing capability within the IAC is designed to solve an equally broad range of data transfer problems (from NASTRAN OUTPUT2 binary file, to IAC datastructure, to an IAC database) in a manner which can be tailored by different user groups to meet their own unique in-house requirements.

IAC Level 2.0 will contain several NASTRAN to IAC interfacing capabilities. These are designed to read the binary files generated by the OUTPUT2 functional module of COSMIC and MSC NASTRAN. The objective is to read NASTRAN table and matrix data blocks generated via standard functional modules, place the data into an IAC datastructure, and then automatically place it into a user specified IAC database.

The driving force behind the interfacing capabilities currently contained in the IAC has been the controls/structure interaction analysis needs of the Guidance and Control Branch at the GSFC. Briefly, these are summarized as:

- Read any data block associated with statics or normal modes analysis which can be generated via standard case control commands.
- Do above in a manner compatible with extension to thermal and other NASTRAN analysis capabilities.
- Obtain all data required for the follow-on controls and system dynamics analysis programs DISCOS (refs. 10, 11), SAMSAN (ref. 12), INCA (refs. 5, 13), and DADS (refs. 14, 15, 16).

- o Place NASTRAN output data into IAC database. Use standard NASTRAN .F06 file acronyms as attribute names in datastructure descriptor.

- o Develop MSC/NASTRAN to COSMIC/NASTRAN interface. Provide capability to use finite elements of MSC/NASTRAN and substructure analysis capability of COSMIC/NASTRAN. Implement via an interface module designed to read MSC written OUTPUT2 files and then write COSMIC compatible INPUTT2 files.

- o Extract sufficient structure or substructure grid point location and analysis set mass and stiffness matrix information to enable one to perform eigenanalysis, modal synthesis, and post-processing analysis outside of NASTRAN in accordance with user unique needs.

INTERFACE MODULE CINSA2 FOR STRUCTURAL ANALYSIS RIGID FORMATS

The most common means of using NASTRAN is via use of one of its rigid formats. For example, rigid format 1 is used for statics analysis, and rigid format 3 is used for normal modes analysis. The following trivial DMAP alters may be used to write an OUTPUT2 file containing all data blocks generated via case control command; if the data block is not requested via case control, the OUTPUT2 functional module will ignore it. Both alters are to be placed just after the LABEL FINIS $ statement in their respective rigid formats.

O2S01.RFA

```
$  COSMIC/NASTRAN OUTPUT2 ALTER FOR RIGID FORMAT 01
$            STATIC ANALYSIS
$
$  CREATE OUTPUT2 FILE TO BE PROCESSED BY
$              IAC PROGRAM CINSA2
$
ALTER 156
OUTPUT2  GPL,BGPDT,USET,ECT,EST//C,N,0/C,N,11 $
OUTPUT2  OGPWG,MGG,UGV,PGG,QG//C,N,0/C,N,11 $
OUTPUT2  OPG1,OQG1,OUGV1,OES1,//C,N,0/C,N,11 $
OUTPUT2  OEF1,ONRGY1,OGPFB1,//C,N,0/C,N,11 $
ENDALTER
```

O2S03.RFA

```
$  COSMIC/NASTRAN OUTPUT2 ALTER FOR RIGID FORMAT 3
$            NORMAL MODES ANALYSIS
$
$  CREATE OUTPUT2 FILE TO BE PROCESSED BY
$              IAC PROGRAM CINSA2
$
ALTER 98
OUTPUT2  GPL,BGPDT,USET,MGG,ECT//C,N,0/C,N,11 $
OUTPUT2  OGPWG,LAMA,PHIG,OPHIG,//C,N,0/C,N,11 $
ENDALTER
```

These 2 DMAP alters write binary OUTPUT2 files which contain all data requested via case control. The program CINSA2 will read the OUTPUT2 file, create IAC datastructures, and automatically place them into the database that the user has opened via the IAC interactive command OPEN.

The program CINSA2 has a general modular type framework. It is able to read an arbitrary OUTPUT2 file of COSMIC/NASTRAN data blocks. A companion program INSA2 is available to read an arbitrary OUTPUT2 file of MSC/NASTRAN data blocks. After each data block is read, a search for known data block names is made. If the data block name is recognized, the data block is processed accordingly. If it is not recognized, a message is provided and the program proceeds to read the next data block on the OUTPUT2 file.

The following data blocks are currently recognized by the program CINSA2:

*** TABLES (INTEGER & REAL*4) ***

GPL - GRID POINT LIST
BGPDT - BASIC GRIP POINT DEFINITION TABLE
USET - DISPLACEMENT SET DEFINITION TABLE
OGPWG - GRID POINT WEIGHT GENERATOR OUTPUT TABLE
LAMA - REAL EIGENVALUE TABLE
OPHIG - OUTPUT EIGENVECTOR REQUESTS TABLE
ECT - ELEMENT CONNECTION TABLE *
EST - ELEMENT SUMMARY TABLE *
OQG1 - SINGLE-POINT CONSTRAINT FORCE REQUESTS
OPG1 - LOAD VECTOR REQUESTS
OUGV1 - DISPLACEMENT VECTOR REQUESTS
OEF1 - ELEMENT FORCE REQUESTS *
OES1 - ELEMENT STRESS REQUESTS *

* IAC datastructure for associated table data not yet defined.
These will be developed in tandem with need at the GSFC.

*** MATRICES (REAL*4 & REAL*8) ***

MGG - MASS MATRIX, G-SET
PHIG - MATRIX OF EIGENVECTORS, G-SET
UGV - DISPLACEMENT VECTOR MATRIX, G-SET
QG - SINGLE-POINT CONSTRAINT FORCES, G-SET
PGG - STATIC LOAD, G-SET

The data contained in the above datablocks is used to construct the following set of IAC datastructures with user specified [name:number]:

DS_1 DATASTRUCTURE = name:number.GRID, CLASS = RELATION
CONTENTS: Grid point numbering, location, and lumped inertia information.

```
DS_2 DATASTRUCTURE = name:number.DOF,      CLASS = RELATION
     CONTENTS:  Degree of freedom related information

DS_3 DATASTRUCTURE = name:number.GPWG,     CLASS = RELATION
     CONTENTS:  Grid Point Weight Generator table, rigid body mass
                and inertia properties

DS_4 DATASTRUCTURE = name:number.LAMA,     CLASS = RELATION
     CONTENTS:  Real Eigenvalue Table

DS_5 DATASTRUCTURE = name:number.OPHIG,    CLASS = ARRAY
     CONTENTS:  g-set single precision output eigenvector data

DS_6 DATASTRUCTURE = name:number.ODISP,    CLASS = ARRAY
     CONTENTS:  g-set single precision output displacement vector data

DS_7 DATASTRUCTURE = name:number.OLOAD,    CLASS = ARRAY
     CONTENTS:  g-set single precision output load vector data

DS_8 DATASTRUCTURE = name:number.OSPC,     CLASS = ARRAY
     CONTENTS:  g-set single precision output forces of single point
                constraint data

DS_9  DATASTRUCTURE = name:number.PHIG,    CLASS = ARRAY
      CONTENTS:  g-set double precision output eigenvector data

DS_10 DATASTRUCTURE = name:number.DISP,    CLASS = ARRAY
      CONTENTS:  g-set single double precision output displacement
                 vector data

DS_11 DATASTRUCTURE = name:number.LOAD,    CLASS = ARRAY
      CONTENTS:  g-set double precision output static load vector data

DS_12 DATASTRUCTURE = name:number.SPCF,    CLASS = ARRAY
      CONTENTS:  g-set double precision output forces of single point
                 constraint data
```

NASTRAN/SAMSAN/DISCOS DMAP PROGRAM NASDS AND INTERFACING MODULE CINDA2

Control/structure interaction analysis often requires that the system to be controlled must be modeled as a system of interconnected flexible bodies. In these situations, multibody analysis programs such as DISCOS or DADS must be used. If either of these programs are used, flexible body input data must be obtained in a format compatible with the program's requirements.

General multi-flexible body theory requires that several resultant mode dependent parameters be created by a preprocessor. This is true for either DISCOS, DADS, or any other multi-flexible body program.

Irrespective of program, the parameters necessary for the definition of the effects of flexibility on total system rigid and flexible body dynamics are definite integrals which depend upon mass distribution, grid point location, and eigenvector displacement. Furthermore, if any of the flexible bodies in the multibody system have 2 or more bodies attached, modal synthesis methods are required to generate the modes needed for follow-on multibody dynamics analysis. If modal synthesis methods are not used, an unreasonable number of free-free modes will usually be required and numerical precision and computational speed problems will yield a computationally impractical simulation. NASDS is designed to provide a variety of modal synthesis options and then to write an OUTPUT2 file with all data required for follow-on control structure interaction analysis. Users direct computation flow through NASDS with PARAM cards. The OUTPUT2 file created by NASDS is processed by the IAC interfacing module CINDA2.

NASDS is applicable for any structure for which a full g-set model is practical to obtain. If the model is extremely large or substructure analysis methods have been used, interfacing modules CINMSC, CINSOF, and CIDRMX may have to be utilized. These modules are designed to provide the interface between reduced order a-set or substructure analysis data and user post-processing needs. The objective of interface module CIDRMX is to compute necessary mode dependent parameters from reduced order mass, stiffness, and grid point location information.

Users of the DMAP program NASDS and the interface module CINDA2 may obtain the following NASTRAN data blocks and associated datastructures:

o GRID POINT LOCATION AND DISPLACEMENT SET DATA

All ARRAY datastructures containing displacement data or state vector coefficient matrices have either grid point or degree of freedom numbering information provided as INDEX (row/column) attributes. Unless specifically directed otherwise, the following data blocks are always written into the OUTPUT2 data file:

```
BGPDT  - BASIC GRID POINT DEFINITION TABLE
GPL    - GRID POINT LIST
USET   - DISPLACEMENT SET DEFINITION TABLE
```

If these data blocks are available, interface module CINDA2 will write the following datastructure:

```
DATASTRUCTURE = name:number.DOF,     CLASS = RELATION
CONTENTS: All degree of freedom related information, same as
          DS_2 generated by CINSA2.
```

To inhibit the above, use the PARAM card:

```
PARAM   PCHGD   -1
```

o GRID POINT WEIGHT GENERATOR AND EIGENANALYSIS TABLE DATA

If normal modes are generated, users normally call for NASTRAN to print the grid point weight generator table, the real eigenvalue table, and at times the real eigenvector table. For follow-on controls analysis, it is almost always necessary to obtain rigid body mass properties and to have direct access to the single precision eigenanalysis data in the associated data block tables. If the following PARAM cards are placed into the bulk data file:

```
PARAM   PCHGPWG   1
PARAM   PCHLAMA   1
PARAM   PCHPHIG   1
```

NASDS will write the following data blocks into the OUTPUT2 file:

```
OGPWG  - GRID POINT WEIGHT GENERATOR TABLE
LAMA   - REAL EIGENVALUE TABLE
OPHIG  - REAL EIGENVECTOR TABLE
```

If these data blocks are available, interface module CINDA2 will write the following datastructure:

```
DATASTRUCTURE = name:number.GPWG,     CLASS = RELATION
CONTENTS: Grid Point Weight Generator table, same as DS_3
          generated by CINSA2

DATASTRUCTURE = name:number.LAMA,     CLASS = RELATION
CONTENTS: Real Eigenvalue Table, same as DS_4 by CINSA2.

DATASTRUCTURE = name:number.OPHIG,    CLASS = ARRAY
CONTENTS: g-set single precision output eigenvector data,
          same as DS_5 generated by CINSA2.
```

o STANDARD NORMAL MODES ANALYSIS

This is the default computation path through NASDS. If the user does nothing other than provide a standard normal modes bulk data file, NASDS will, in addition to the above, write the following data blocks to the OUTPUT2 file:

```
MGG    - LUMPED MASS MATRIX, G-SET
PHIG   - MATRIX OF EIGENVECTORS, G -SET
BHH    - MODAL DAMPING MATRIX
KHH    - MODAL STIFFNESS MATRIX
MHH    - MODAL MASS MATRIX
```

Diagonal modal mass and stiffness matrices are constructed from known generalized mass and stiffness information within the system. Modal damping is obtained by the matrix triple product of a-set eigenvectors (PHIA), its transpose, and the a-set dynamic damping matrix which includes both viscous and the viscous equivalent of structural damping effects.

Users may inhibit writing the above matrices of double precision data to the OUTPUT2 file by using the following PARAM cards in their bulk data file:

```
PARAM   PCHGD   -1  DO NOT OUTPUT2 GRID POINT LOCATION AND DISPLACEMENT
                    SET DATA
PARAM   PCHMG   -1  DO NOT OUTPUT2 LUMPED MASS DATA
PARAM   PCHMD   -1  DO NOT OUTPUT2 (EIGENVECTOR) MODE SHAPE DATA
PARAM   PCHMM   -1  DO NOT OUTPUT2 MODAL MASS, STIFFNESS, AND DAMPING
                    MATRICES
```

If these data blocks are available, interface module CINDA2 will write the following datastructures:

```
      DATASTRUCTURE = name:number.GRID,     CLASS = RELATION
      CONTENTS:  Grid point numbering, location, and lumped inertia
                 information, same as DS_1 generated by CINSA2.

      DATASTRUCTURE = name:number.PHIG,     CLASS = ARRAY
      CONTENTS:  g-set double precision output eigenvector displacement
                 data, same as DS_9 generated by CINSA2.

DS_13 DATASTRUCTURE = name:number.MGG,      CLASS = ARRAY
      CONTENTS:  The g-set mass matrix

DS_14 DATASTRUCTURE = name:number.MMASS,    CLASS = ARRAY
      CONTENTS:  The modal mass matrix

DS_15 DATASTRUCTURE = name:number.MSTIF,    CLASS = ARRAY
      CONTENTS:  The modal stiffness matrix

DS_16 DATASTRUCTURE = name:number.MDAMP,    CLASS = ARRAY
      CONTENTS:  The modal damping matrix
```

o MASS, STIFFNESS, DAMPING, AND CONSTRAINT MATRICES

If follow-on application requires use of g-set coefficient matrices, these may be written to an OUTPUT2 file by the inclusion of the PARAM card

```
PARAM   MKMAT   1
```

in the bulk data file. If this card is used, computation within NASTRAN will terminate after the processing of all multipoint constraint data. The following data blocks will, in addition to GPL, BGPDT, and USET, be written to the OUTPUT2 file:

```
MGG - MASS MATRIX, G-SET
KGG - STIFFNESS MATRIX, G-SET
BGG - VISCOUS DAMPING MATRIX, G-SET
K4GG - STRUCTURAL DAMPING MATRIX, G-SET
RG - MULTIPOINT AND RIGID ELEMENT CONSTRAINT EQUATION
     MATRIX, G-SET TO M-SET
```

GM - MULTIPOINT AND RIGID ELEMENT TRANSFORMATION MATRIX
M-SET TO N-SET
GTOMN - PARTITIONING VECTOR, 1.0 IMPLIES N-SET WHILE
0.0 IMPLIES M-SET. UNION IS G-SET.

If these data blocks are available, interface program CINDA2 will write the following data structures:

DATASTRUCTURE = name:number.MGG, CLASS = ARRAY
CONTENTS: The g-set mass matrix, same as DS_13 defined above.

DS_17 DATASTRUCTURE = name:number.KGG, CLASS = ARRAY
CONTENTS: The g-set stiffness matrix

DS_18 DATASTRUCTURE = name:number.BGG, CLASS = ARRAY
CONTENTS: The g-set damping matrix

DS_19 DATASTRUCTURE = name:number.K4GG, CLASS = ARRAY
CONTENTS: The g-set structural damping matrix

DS_20 DATASTRUCTURE = name:number.RG, CLASS = ARRAY
CONTENTS: The multipoint and rigid element constraint equation matrix g-set to m-set

DS_21 DATASTRUCTURE = name:number.GM, CLASS = ARRAY
CONTENTS: The multipoint and rigid element transformation matrix m-set to n-set

o REDUCED MASS AND STIFFNESS MATRIX DATA

If the follow-on analysis activity is modal synthesis outside of NASTRAN, it is frequently necessary to work with reduced order mass and stiffness matrices. To obtain a-set mass and stiffness matrices, users of NASDS need only include the PARAM card

PARAM RMX 1

in the bulk data file. If this card is used, computation within NASTRAN will terminate after processing a-set mass and stiffness data blocks. The following data blocks will be written to the OUTPUT2 file:

MAA - MASS MATRIX, A-SET
KAA - STIFFNESS MATRIX, A-SET

If both MAA and KAA are on the OUTPUT2 file, the program CINDA2 assumes that the user desires to proceed with follow-on analysis via the interface program CIDRMX. The program CIDRMX was designed to read datastructures containing substructure operating file (SOF) data items and then proceed with necessary analysis outside of NASTRAN. Equivalent datastructures can be constructed from a-set information obtainable from NASDS. The following datastructures are identical to those produced by the interface program CINSOF. They are compatible with the input data requirements of CIDRMX.

DS_22 DATASTRUCTURE = name:number.KMTX, CLASS = ARRAY
CONTENTS: The a-set stiffness matrix

DS_23 DATASTRUCTURE = name:number.MMTX, CLASS = ARRAY
CONTENTS: The a-set mass matrix

DS_24 DATASTRUCTURE = name:number.EQSS, CLASS = RELATION
CONTENTS: External/internal grid point equivalence data identical to that associated with substructure analysis

DS_25 DATASTRUCTURE = name:number.BGSS, CLASS = RELATION
CONTENTS: Basic grid point coordinates data identical to that associated with substructure analysis

o FINE TO COARSE MESH MODELING

All multibody programs which accept flexible bodies require that resultant mode dependent parameters be obtained. These parameters are to be computed via the evaluation of a series of definite integrals. The integrals involve mass distribution, grid point location, and modal amplitude over the entire volume. For very large finite element models, it is computationally impractical to backtransform all eigenvectors from the a-set to the g-set. This capability was an attempt to provide sufficient data for a mass distribution interpolation program (never written) which could be used to reduce problem order. Data blocks written and datastructures created have rarely been needed; details are in the comment cards of the programs NASDS and CINDA2.

Recent theoretical developments have uncovered a means for obtaining all required mode dependent parameters directly from the full a-set mass matrix, grid point locations, and modal amplitudes. This work is implemented in the interfacing module CIDRMX.

o MODAL OBSERVABILITY AND CONTROLLABILITY

If follow-on modal analysis is the objective, one must always face the problem of deciding which modes to retain and which to discard. Both modal observability and controllability matrices may be computed within NASDS. Relative to the g-set, NASDS tells NASTRAN to set up the following matrix equation:

$$MGG * X'' + KGG * X = BMAT * U$$

$$Y = CMAT * X$$

where X is the state vector, U is the input vector, and Y is the output vector. If NASDS users insert the PARAM card

PARAM MODOBCL 1

the program will accept both matrices BMAT and CMAT via user supplied DMIG bulk data input. Normally, they are composed of simply 1's and 0's. The 1's define which DOF's are controllable and which are observable; the 0's define which are not. If this card is used, datablocks are written to the OUTPUT2 file which CINDA2 then reads and uses to create the following datastructures:

```
DS_26 DATASTRUCTURE = name:number.BMATG,     CLASS = ARRAY
      CONTENTS:  Input coefficient matrix BMAT compatible with g-set
                 eigenvector matrix PHIG

DS_27 DATASTRUCTURE = name:number.CMATG,     CLASS = ARRAY
      CONTENTS:  Output coefficient matrix CMAT compatible with g-set
                 eigenvector matrix PHIG

DS_28 DATASTRUCTURE = name:number.MODCTL,    CLASS = ARRAY
      CONTENTS:  Modal controllability matrix
            MODCTL = BMATG**T * PHIG      (**T - matrix transpose)

DS_29 DATASTRUCTURE = name:number.MODCSS,    CLASS = ARRAY
      CONTENTS:  Steady state modal controllability matrix
            MODCSS = MODCTL * diag(square root generalized
                                   stiffness inverse)

DS_30 DATASTRUCTURE = name:number.MODOBS,    CLASS = ARRAY
      CONTENTS:  The modal observability matrix
            MODOBS = CMATG * PHIG

DS_31 DATASTRUCTURE = name:number.MODOSS,    CLASS = ARRAY
      CONTENTS:  The steady state modal observability matrix
            MODOSS = MODOBS * diag(square root generalized
                                   stiffness inverse)
```

o AUGMENTED BODY MODES

If the flexible bodies in a multibody system have 2 or more contiguous bodies; that is, if the bodies have more than 1 hinge point, modal synthesis techniques are normally required to obtain a best set of modes. One approach is to create augmented body modes. Two augmented body options are available; the user may, via DMIG cards in the bulk data, add either lumped mass and/or lumped stiffness at each of the hinge points. The resultant modes are referred to as augmented body modes within NASDS, see ref. 17. Stiffness may be augmented without need of a PARAM card. To augment mass, the PARAM card

```
PARAM   AUGMOD    1
```

must be used. If this PARAM card is used, the program will compute all eigenvectors with the augmented mass matrix and modal mass with the non-augmented mass matrix. Modal mass will be non-diagonal. If the PARAM card is not used, modal mass will be computed with respect to the augmented

mass matrix and be diagonal. Modal stiffness is always computed with respect to the augmented stiffness matrix and hence is always diagonal. Datastructures created are the same as those created via standard modal analysis procedures. The above PARAM card is also needed to tell the system to write the non-augmented mass matrix into the OUTPUT2 file; follow-on multibody dynamics analysis requires this.

o FIXED-INTERFACE METHOD, CRAIG BAMPTON MODAL DATA

If the flexible bodies in a multibody system have 2 or more contiguous bodies, that is, if the bodies have more than 1 hinge point, modal synthesis techniques are normally required to obtain a best set of modes. Another modal synthesis technique commonly used is the Craig-Bampton fixed-interface method. Within NASDS, there is the capability to compute what we refer to as stage 1, stage 2, and stage 3 Craig-Bampton modes. The 3 stages are an outgrowth of the need to isolate the 6 rigid body modes within the set of constraint modes. Most multibody formulisms require only deformation modes; in fact, they require that rigid body modes be removed from the set of input modes. This is a problem with standard Craig-Bampton modes since rigid body modes cannot be simply partitioned out of the set of constraint modes. Stage 1 modes, within the context of NASDS, are standard Craig-Bampton modes; Stage 2 modes are obtained by a transformation which isolates the 6 rigid body modes; and Stage 3 modes are obtained via the setup and solution of a new eigenproblem which yields diagonal modal mass and stiffness matrices. The original multistage theory as used within NASDS was developed by Bodley and Park at Martin Marrietta, ref. 18. Reference 19 by Craig and Chang contains background ideas for Stage 2 and 3 modes in their sections devoted to the "Guyan reduction of junction (hinge) coordinates" and "Modal reduction of junction coordinates." Comment cards within NASDS, ref. 20, contain all necessary theory.

Users of this capability must adhere to certain setup rules defined completely in NASDS source code and explained by example in the programs accompanying documentation. The following PARAM cards are required if this capability is desired

```
PARAM   CB1MODS  1  Craig-Bampton stage 1, 2, or 3 desired
PARAM   CB2MODS  2  Craig-Bampton stage 2 or 3 desired
PARAM   CB3MODS  3  Craig-Bampton stage 3 desired
```

In addition to the PARAM cards, additional data is required to define which grid points are to be designated to be hinge (boundary) points. All points defined on bulk data SUPORT cards are by defintion boundary points; all others are by definition interior points. For stage 2 and 3 modes, one of the boundary points must be designated as a reference point; this is done by user specification of partitioning vectors via DMI bulk data input. The introduction of the reference point within the set of boundary points allows for the isolation of the 6 rigid body modes via a Guyan reduction step. If these cards are used, datablocks are written to the OUTPUT2 file which CINDA2 then reads and uses to create the following datastructures:

```
DS_32 DATASTRUCTURE = name:number.PHIGi,     CLASS = ARRAY
      CONTENTS:  g-set stage i (i=1,2,3) Craig-Bampton modal data.

DS_33 DATASTRUCTURE = name:number.MCBi,      CLASS = ARRAY
      CONTENTS:  Modal mass matrix for stage i (i=1,2,3) Craig-
                 Bampton modes

DS_34 DATASTRUCTURE = name:number.BCBi,      CLASS = ARRAY
      CONTENTS:  Modal damping matrix for stage i (i=1,2,3) Craig-
                 Bampton modes

DS_35 DATASTRUCTURE = name:number.KCBi,      CLASS = ARRAY
      CONTENTS:  Modal stiffness matrix for stage i (i=1,2,3)
                 Craig-Bampton modes
```

o FREE-INTERFACE METHODS

NASDS does not currently contain code to generate modal data associated with any of the free-interface methods discussed in the literature, see ref. 19. On the surface, the inclusion of the required DMAP statements to provided a free-interface method option, appears straight-forward.

MSC TO COSMIC/NASTRAN INTERFACE MODULE CINMSC

At the GSFC, there is a definite need to translate contractor supplied MSC/NASTRAN bulk data files to COSMIC/NASTRAN compatible format. The capability is primarily needed to mesh the work of various contractors via COSMIC/NASTRAN's substructure analysis capability into a complete system structural model. Rather than attempt the automated translation of bulk data files, the program CINMSC is designed to read a MSC/NASTRAN written OUTPUT2 file and to then write a COSMIC/NASTRAN compatible INPUTT2 file.

We have circumvented the bulk data translation problem by inserting OUTPUT2 statements within MSC/NASTRAN rigid formats and the appropriate INPUTT2 statements within the analogous COSMIC/NASTRAN rigid format. The module CINMSC accounts for unformatted file differences between the 2 versions of NASTRAN. It is designed to translate both table and matrix data blocks. It is not currently coded to translate complex matrix data blocks. This procedure allows the COSMIC job to be run with all MSC element cards removed from the bulk data file; they need only be replaced by a dummy mass and elastic element card. The dummy elements create the data blocks which will be filled at the appropriate time in the DMAP sequence by INPUTT2.

CAUTION!!! Lack of a sparse matrix OUTPUT2 capability within COSMIC can lead to extremely large files. It is common for COSMIC files to be 10 to 100 times larger than MSC files. Moderate size MSC files (5000 blocks) frequently translate into files which exceed total system free storage limits. This problem has limited the usefulness of interface module CINMSC.

INTERFACE MODULE CINSOF, READ SUBSTRUCTURE OPERATING FILE (SOF) DATA

Control/structure interaction analysis needs for space station and other very large systems will require use of substructure analysis techniques. Experience has shown that there is no guarantee that the structural analyst and the controls analyst will take the same modeling view of a large space structure; furthermore, there is no guarantee that a structural analyst will be available to merrily write DMAP to keep the controls analyst happy. This capability has been developed to allow follow-on controls analysis groups the ability to extract the basic data that they need from the SOF file and then process it as they desire outside of NASTRAN. The need for this capability occurs when the controls analyst attempts to develop a controller for large angle relative orientation of a pair of hinged flexible bodies. The SOF file contains mass and stiffness matrices for each component and eigenanalysis data associated with the analysis process used to obtain composite system modes and frequencies. The controls analyst needs eigenanalysis data for each body alone. The interface module CINSOF provides the ability to retrieve data for each component or substructure of components, while the interface module CIDRMX provides the ability to process it as needed.

Data for the program CINSOF is most effectively prepared via the submission of a stand alone substructure phase 2 job of the following form:

```
NASTRAN FILES=INPT
$
$  FILES=INPT establishes the specific NASTRAN permanent file
$  INPT as an executive file.  This file has the FORTRAN logical
$  unit FOR014.  Use CNASTRAN RUN parameter ASG to point to a
$  file containing the DCL statement
$
$             $ ASSIGN 'F'.EIO FOR014
$
$  This assigns a user specified file name to the NASTRAN permanent
$  file INPT and overrides its automatic deletion at job end.
$
ID READ,SOF
APP DMAP,SUBS
BEGIN  $ DMAP PROGRAM TO READ SOF FILE AND OUTPUT SELECTED ITEMS
$
$   Initialize error code DRY
PARAM    //*ADD*/DRY/1  /0  $
$
$   PRINT SOF TABLE OF CONTENTS (TOC)
SOFUT    //DRY/*TOC     */*SOFP*/0  /* */* */* */* */
               * */* */* * $
$
$  Use functional module EXIO to copy selected items from SOF
$  external file
$
```

```
$    FUNCTIONAL MODULE EXIO (EXTERNAL INPUT/OUTPUT FOR SOF)
$
$  PARAMETER #  USE      EXPLANATION
$       1       DRY      Integer code for error occurrence check
$       2       780      VAX 11/780 machine in use
$       3       DISK     EXIO file to be written will be on disk
$       4       INPT     Unit where EXIO file is to be located
$       5       INTERNAL  Internal file written with GINO
$       6       SOFOUT   Copy from the SOF to the external file
$       7       REWIND   Use in first EXIO statement
$               EOF      Use thereafter
$       8       xxxx     Data items on SOF to be copied
$       9       aaaa     Name of subucture whose items copied
$      10       bbbb     Name of substructure whose items copied
$      11       cccc     Name of substructure whose items copied
$      12       dddd     Name of substructure whose items copied
$      13       eeee     Name of substructure whose items copied
$
$       See programmer's manual page 4.130-1 for more detail
$
EXIO      //DRY/C,N,780/C,N,DISK/C,N,INPT/C,N,INTERNAL/
          C,N,SOFOUT/C,N,REWIND/*EQSS*/*WHOLESOF*/* */* */* */* * $
EXIO      //DRY/C,N,780/C,N,DISK/C,N,INPT/C,N,INTERNAL/
          C,N,SOFOUT/C,N,EOF/*BGSS*/*WHOLESOF*/* */* */* */* * $
$
$              etc.
$
END       $
TIME 30
CEND
SUBSTRUCTURE PHASE2
SOF(1)=FT20,500
PASSWORD=ABCD
$
ENDSUBS
$
TITLE = COPY SELECTED ITEMS FROM SOF TO USER FILE
BEGIN BULK
GRID,1
ENDDATA
```

The interface program CINSOF provides the ability to read the data contained in the unformatted data file produced by the functional module EXIO. The output of this program is intended for the use of experienced NASTRAN users. Users are expected to know what data is actually stored within the numerous SOF data items that are processed as standalone data blocks. Extremely large matrices will not be processed. The IAC currently requires all arrays to be stored in a dense format. A sparse format is currently under development.

The following substructure operating file data items are recognized and processed by interface program CINSOF:

```
TOC  - Substructure operating file table of contents
EQSS - External grid point and internal point equivalence data
BGSS - Basic grid point coordinates
CSTM - Local coordinate system transformation matrices
LODS - Load set identification numbers
KMTX - Stiffness matrix
MMTX - Mass matrix
PVEC - Load vectors
POVE - Load vectors on points omitted during matrix reduction
UPRT - Partitioning vector used in matrix reduction
HORG - H or G transformation matrix
UVEC - Displacement vectors or eigenvectors
QVEC - Reaction force vectors
SOLN - Load factor data or eigenvalues used in solution
       *WARNING* load factor data from statics not processed,
                 attributes refer to eigenanalysis data.
                 Program may error out for statics analysis data.
PAPP - Appended load vectors
POAP - Appended load vectors on omitted points
LOAP - Load set identification numbers for appended load vectors
LMTX - Decomposition product of REDUCE operation
GIMS - G transformation matrix for interior points in modal reduction
PHIS - Eigenvector matrix
LAMS - Eigenvalue data for modal reduction operation
K4MX - Structural damping matrix
BMTX - Viscous damping matrix
```

Data contained in the above data blocks are used to construct the following set of IAC datastructures with user specified [name:number]:

```
      DATASTRUCTURE = name:number.EQSS,     CLASS = RELATION
      CONTENTS:  External grid point and internal point equivalence
                 data, same as DS_24 generated by CINDA2

      DATASTRUCTURE = name:number.BGSS,     CLASS = RELATION
      CONTENTS:  Basic grid point coordinates data, same as DS_25
                 generated by CINDA2

      DATASTRUCTURE = name:number.KMTX,     CLASS = ARRAY
      CONTENTS:  a-set stiffness matrix, same as DS_22 generated by
                 CINDA2

      DATASTRUCTURE = name:number.MMTX,     CLASS = ARRAY
      CONTENTS:  a-set mass matrix, same as DS_23 generated by CINDA2

DS_36 DATASTRUCTURE = name:number.XXXX,     CLASS = ARRAY
      CONTENTS:  The SOF file matrix data item XXXX,
                 XXXX = K4MX,BMTX,GIMS,UPRT,PHIS,LMTX,PAPP,LMTX,
                        PAPP,PVEC,POAP,POVE,QVEC,UVEC,HORG
```

DS_37 DATASTRUCTURE = name:number.XXXX, CLASS = RELATION
CONTENTS: The SOF file data item XXXX,
XXXX = SOLN,LAMS,CSTM,LODS,LOAP

INTERFACE MODULE CIDRMX, NASTRAN - DISCOS/DADS INTERFACE PROGRAM

The interface program CIDRMX is designed to accept as input data the information contained in the following data structures:

DATASTRUCTURE = name:number.KMTX, DS_22
DATASTRUCTURE = name:number.MMTX, DS_23
DATASTRUCTURE = name:number.EQSS, DS_24
DATASTRUCTURE = name:number.BGSS, DS_25

The origin of these datastructures is of no concern to CIDRMX; currently they may be written by either CINDA2 or CINSOF. The objective of CIDRMX is to obtain all mode dependent parameters required by DISCOS, DADS, or any other multi-flexible body program from reduced order models. The starting point is a-set mass and stiffness matrices obtained via the DMAP program NASDS or from the reduced order mass and stiffness matrices of any substructure defined on an SOF file along with all grid point location and numbering information associated with all degrees of freedom in the associated state vector.

The capability contained within CIDRMX contains a fixed dimension for matrix order. There is a PARAMETER card in the source code of CIDRMX which requires all reduced order matrices to be less than order 300. This limit corresponds to the general rule of thumb within the GSFC and most other NASTRAN groups that in real world application eigenanalysis should never be done with matrices of order greater than about 250. CIDRMX contains 5 different eigenanalysis options; these correspond to the 5 capabilities provided in the EISPACK library, ref. 21. A variety of options are provided to give users some computational options. These are:

1. Default, compute all eigenvalues and eigenvectors by the QL method. Use if more than 25 percent of eigenvalues/vectors required.
2. Some eigenvalues/vectors. Determined by the method of bisection applied to the Strum sequence. Recommended if less than 25 percent of all eigenvalues/vectors required. User must specify the upper eigenvalue bound; the lower limit is set slightly negative to pick up all 0.0's.
3. Some eigenvalues/vectors. Determined by the method of bisection applied to the Strum sequence. Recommend if less than 25 percent of all eigenvalues/vectors required. User must specify index of the upper eigenvalue bound; lower bound is 1.
4. All eigenvalues and some eigenvectors. Determines all eigenvalues by using the implicit QL-method. User must specify index of the upper eigenvalue bound for which an eigenvector is desired. Lower limit is 1.

5. All eigenvalues and eigenvectors. To be used when matrix "B" is not positive definite. Results obtained via use of the general QZ algorithm applied to real symmetric matrices.

All modal data is currently obtained via a standard application of eigenanalysis procedures. Plans are currently underway to include various modal synthesis procedures in CIDRMX.

In multibody theory, the body fixed reference translates and rotates relative to the inertial reference frame at a rate that cannot be ignored. The mode dependent parameters required for follow-on multi-flexible body dynamics stem from the need to express state vector derivative in the kinetic energy expression of each flexible body relative to an inertially fixed reference frame. That is,

$$T = 1/2 \; [\dot{Na}] * [MAA] * (\dot{Na})$$

where

T - flexible body kinetic energy
MAA - reduced order mass matrix
Na - reduced order state vector, it may contain both physical and modal degrees of freedom
. - derivative relative to inertial frame

Let

R - position vector from inertial reference to body reference point
W - inertial angular velocity vector of body reference frame
Ni - deformed position and orientation of reference frame at grid point i
Pi - position vector from body reference point to undeformed position of grid point i
Di - elastic translational deformation at grid point i
Mj - j-th modal degree of freedom
o - derivative relative to body fixed reference frame

Then for each grid point associated with physical degrees of freedom

$$\dot{Ni} = \dot{R} + W \times (Pi + Di) + \overset{o}{Ni}$$

and for each modal degree of freedom

$$\dot{Mj} = \overset{o}{Mj}$$

Construct for each grid point i a 6x12 transformation matrix which can be used to express the inertial derivative of each grid point position vector Ni in terms of the body reference frame inertial rate and the grid point frame relative rate. This transformation, when accumulated for all grid points associated with all physical degrees of freedom, provides all information needed to define the transformation from inertial derivative of reduced state vector Na to inertial rate of the body fixed frame and relative rate of each grid point frame. To arrive at the actual transformation to be used, simply remove all rows associated with all DOF's

not in the set of physical DOF's in the reduced state vector, then add rows to account for modal DOF's. Eigenanalysis provides the final step, the transformation from relative deformation at grid points to modal coordinates. The fully assembled transformation is then substituted into the kinetic energy expression.

The above steps provide the transformation to a state vector composed of rigid body and modal degrees of freedom. The next step is to analytically form the matrix triple product between transformation matrix, its transpose, and the reduced mass matrix. Further manipulation leads to the time dependent reduced mass matrix expressed as a summation of time independent coefficient matrices and generalized displacement coordinates. The summation includes terms which are independent, linear, and quadratic in the generalized displacement coordinates. The Lagrange solution to the multibody equations of motion used in both DISCOS and DADS require this so that partial derivatives of the mass matrix can be formed with respect to generalized displacement coordinates. Rather than store large coefficient matrices, both formulations require mode dependent vectors and tensors. The programs use these in a computationally efficient manner to define only non-zero terms. The definition of each of the mode dependent parameters can be found in refs. 11 and 15. The equations differ only by notation. Complete specification of the unraveling steps and associated equations is planned for inclusion in the program documentation of IAC Level 2.0.

The following datastructures are created by CIDRMX:

DATASTRUCTURE=name:number.GPWG, CLASS = RELATION
CONTENTS: Grid Point Weight Generator table; standard data provided in the NASTRAN GPWG table. All data provided here is derived directly from the reduced order mass matrix MMTX and the gird point location information in EQSS & BGSS. It should be noted that within NASTRAN, GPWG data is not available beyond the phase 1 component definition step in substructure analysis. The same as DS_3.

DATASTRUCTURE=name:number.LAMA, CLASS = RELATION
CONTENTS: Real eigenvalue table; eigenvalues, natural frequencies generalized mass and stiffness data. The same as DS_4

DS_38 DATASTRUCTURE=name:number.EIGV, CLASS = ARRAY
CONTENTS: Real eigenvector data. Contains modes used for computation of all resultant mode dependent data. Modes may be standard orthogonal modes from unaltered eigenanalysis or non-orthogonal modes associated modal synthesis techniques used for interior flexible bodies. INDEX 1 attributes associate internal/external sequences of DOF's. INDEX 2 attributes associate eigenvalues.

DATASTRUCTURE=name:number.MMASS, CLASS = ARRAY
CONTENTS: Modal mass matrix. Same as DS_14.
MODES**T * MASS(MATRIX) * MODES.

DATASTRUCTURE=name:number.MSTIFF, CLASS = ARRAY
CONTENTS: Modal stiffness matrix. Same as DS_15.
MODES**T * STIFFNESS(MATRIX) * MODES.

For follow-on multibody analysis, one is expected to obtain all rigid body mass and inertia from the grid point weight generator datastructure DS3, name:number.GPWG. All mode dependent data is contained in the datastructures name:number.FLMX1 and name:number.FLMX2.

DS_39 DATASTRUCTURE=name:number.FLMX1, CLASS = RELATION
CONTENTS: The following mode dependent parameters dependent upon 1 mode:

A0 - Coupling terms between rigid body translational motion and modal velocity for the component of the full mass matrix which is independent of deformation. These are modal masses formed by the integral of modal deflection times mass distribution d-volume. One vector per mode.

D0 - Coupling terms between rigid body rotational motion and modal velocity for the component of the time dependent mass matrix which is independent of deformation. These are modal mass moments formed by the integral of grid point position vector cross grid point modal deflection vector times mass distribution d-volume. One vector per mode.

B1 - Component of the deformed body 3x3 moment of inertia dyadic which is linearly dependent upon modal deformation. These are formed by an integral of dyads formed by grid point position vector and modal deflection vector times mass distribution d-volume. One dyadic per mode.

A1 - Component of the deformed body 3x3 mass moment dyadic which is linearly dependent upon modal deformation. One dyadic per mode.

DS_40 DATASTRUCTURE=name:number.FLMX2, CLASS = RELATION
CONTENTS: The following mode dependent parameters dependent upon 2 modes:

C1 - Coupling terms between rigid body translational motion and modal velocity for the component of the full mass matrix which are linearly dependent upon modal deformation. One vector per mode pair.

C2 - Component of the deformed 3x3 body moment of inertia tensor which has a quadratic dependence on modal deformation. One dyad per mode pair.

Preliminary investigations with the interface module CIDRMX have convinced the author that the question of whether or not reduced order models contain sufficient fidelity to accurately produce all required mode dependent cross coupling terms must be more fully investigated. Early work

leads the author to believe that rigid body properties can be accurately obtained; however, mode dependent term accuracy must be more fully investigated. Current plans are to more fully understand this problem and to then include modal synthesis procedures before IAC Level 2 is released in early 1987.

INTERFACE MODULE INSAT
INTERFACE NASTRAN STATICS ANALYZER FOR THERMAL DATA

The program INSAT is designed to read a MSC/NASTRAN bulk data file which has been set up for a structural statics run and an array of nodal temperature data from a datastructure in an IAC database. INSAT then creates an enhanced bulk data file by generating cards which define thermal load sets and a table which correlates time values with solution subcases id's. This module was written by BAC for MSC/NASTRAN and has not yet been modified to work with COSMIC/NASTRAN.

INTERFACE MODULE INSAM
INTERFACE NASTRAN STATICS ANALYZER FOR MODAL DATA

The program INSAM is designed to read a MSC/NASTRAN bulk data file which has been set up for a structural statics run and an array of mode shape definitions. INSAM then creates an enhanced bulk data file by automatically generating cards which define nodal-to-modal conversion. The net result is the ability to compute nodal and modal displacements, stresses, etc. This module was written by BAC for MSC/NASTRAN and has not yet been modified to work with COSMIC/NASTRAN.

REFERENCES:

1. Vos, R. G., Beste, D. L., Gregg, J., Frisch, H. P., and Sanborn, J.A., "Integrated Analysis Capability (IAC) Program Level 1.5," COSMIC Program GSC-12992, COSMIC, Barrow Hall, Suite 112, University of Georgia, Athens, GA 30602.

2. Young, J. P., Vos, R. G., Frisch, H. P., and Jones, G. K., "Integrated Analysis Capability (IAC) Activity," SECOND CHAUTAUQUA on productivity in Engineering and Design THE CAD REVOLUTION, Nov 15-17, 1982.

3. Vos, R. G., Walker, W. J., Beste, D. L., Price, G. A., Young, J. P., and Frisch, H. P., "Development and Use of an Integrated Analysis Capability," AIAA/ASME/ASCE/AHS Structures, Structural Dynamics and Materials Conference, May 1983, Lake Tahoe, NV, AIAA Paper 83-1017.

4. Bossi, J. A., Price, G. A., and Winkleblack, S. A., " Flexible Spacecraft Controller Design Using the Integrated Analysis Capability (IAC)," AIAA Guidance and Control Conference, August 1984, Also IEEE Control Systems magazine, November 1985.

5. Bauer, F. H., Frisch, H. P., and Downing, J. P., "Integrated Control System Design Capabilities at the Goddard Space Flight Center," Pro- ceedings of the 2nd IEEE Control Systems Society Symposium on Computer- Aided Control System Design (CACSD), Santa Barbara, California, March, 13-15 1985.

6. Frisch, H. P. "Integrated Analysis Capability (IAC) for Structures, Controls, and Thermal Analysis," Seminar presented at the 13th NASTRAN Users' Colloquium, May 6-10, 1985.

7. Vos, R. G. "IAC level 0 Program Development," Large Space Systems Technology 1981, NASA Conference Publication 2215, Nov 16-19, 1981.

8. Bossi, J. A., Price, G. A., and Winkleblack, S. A., "Design of Multivariable Controllers Using the Integrated Analysis Capability," Proceedings of JPL Workshop on Identification and Control of Flexible Space Structures, San Diego, June 1984.

9. Walker, W. J., Vos, R. G., Price, G. A, and Brogren, E. W. "IAC Executive Summary," NASA Contractor Report 175196, May 1984.

10. Bodley, C. S., Devers, A. D., Park, A. C., and Frisch, H. P. "A Digital Computer Program for the Dynamic Interaction Simulation of Controls and Structure (DISCOS)," COSMIC Program GSC-12810.

11. Bodley, C. S., Devers, A. D., Park, A. C., and Frisch, H. P. "A Digital Computer Program for the Dynamic Interaction Simulation of Controls and Structure (DISCOS)," NASA Technical Paper 1219, Vols 1 and 2, May 1978.

12. Frisch, H. P. and Bauer, F. H. "Numerical Methods for Classical Sampled- System Analysis," COSMIC Program GSC-12827.

13. Bauer, F. H. and Downing, J. P. "INCA - Interactive Controls Analysis," COSMIC Program GSC-12998.

14. For information on "DADS - Dynamics Analysis and Design Systems" contact CADSI, P.O.Box 203, Oakdale, Iowa 52319.

15. Yoo, W. S. and Haug, E. J., "Dynamics of Articulated Structures, Part I: Theory," Journal of Structural Mechanics, to appear.

16. Yoo, W. S. and Haug, E. J., "Dynamics of Articulated Structures, Part II: Computer Implementation and Applications," Journal of Structural Mechanics, to appear.

17. Macala, G. A., "A Modal Reduction Method for use with Nonlinear Simulations of Flexible Multibody Spacecraft," AIAA/ASS Astrodynamics Conference, Aug 20-22, 1984, Seattle, WA, AIAA Paper 84-1989.

18. Gehling, R. N. "Model Reduction Technique for use with the Dynamic Interaction Simulation Controls and Structures (DISCOS) Computer Program," Martin Marietta Report MCR-85-534, Feb 1985.

19. Craig, R. R and Chang, C. "Substructure Coupling for Dynamic Analysis and Testing," NASA Contractor Report NASA CR-2781, Feb. 1975.

20. Frisch, H. P. "NASTRAN/DISCOS/SAMSAN DMAP Bridging Program," COSMIC Program GSC-12902.

21. Smith, B. T., Boyle, J. M., Dongarra, J. J, Garbow, B. S., Ikebe, Y., Klema, V. C., and Moler, C. B, "Matrix Eigensystem Routines - EISPACK Guide," Vol 6, Springer-Verlag, 1976.