3 1176 00161 7928

NASA CR-165,653

NASA Contractor Report 165653

NASA-CR-165653

19810007976

FRANOPP - FRAMEWORK FOR ANALYSIS AND OPTIMIZATION PROBLEMS - USER'S GUIDE

Kathleen M. Riley

KENTRON INTERNATIONAL, INC.

Hampton Technical Center

an LTV Company

Hampton, Virginia 23666

LIBRARY COPY

FEB 3 1981

LANGLEY RESEARCH CENTER
LIBRARY, NASA
HAMPTON, VIRGINIA

Contract NAS1-16000
January 1981

NF02200

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

FRANOPP, FRamework for ANalysis and OPtimization Problems, is a software aid for the study and solution of design (optimization) problems. FRANOPP provides the driving program and plotting capability of a user generated programming system. In addition to FRANOPP, the programming system also requires the optimization code CONMIN (created and maintained at the NASA/Ames Research Center), and two user supplied codes, one for analysis and one for output. Since the user supplies the analysis module which defines the design problem (more specifically, evaluates the objective function, constraints, and optionally gradients for given values of the design variables), the generated system is extremely flexible.

FRANOPP provides the user with five options for studying a design problem. Three of the available options utilize the plotting capability supplied by FRANOPP and provide an indepth study of the design problem at hand. The study can be focused on a history of the optimization process or on the interaction of variables within the design problem.

The optimization module, CONMIN, is a FORTRAN module for the minimization of a multi-variable function subject to a set of inequality constraints. The optimization problem may be linear or non-linear, constrained or unconstrained. The basic algorithm used for the constrained problem is the Method of Feasible Directions. For unconstrained problems, the Conjugate Direction Method of Fletcher and Reeves is used.

This report describes FRANOPP's design and capabilities, and how to build and execute a programming system which incorporates FRANOPP. It is not intended as a stand-alone user's guide. A familiarity with CONMIN and the Control Data Corporation Network Operating System (NOS) is assumed. FRANOPP was developed and tested on the NASA/Langley Research Center (LaRC) computers using available software utilities.

THIS PAGE IS BLANK

INTRODUCTION

Various algorithms exist for the solution of design (optimization) problems, and, correspondingly, many computer codes have arisen. In general, the problem is to determine the values for a vector of variables, termed design variables, that minimizes (or maximizes) some function of those variables, termed the objective function, while satisfying various linear and non-linear constraint equations. When using a generalized code, the user requires some means for defining a particular problem. One method is a user supplied code which contains the problem dependent formulations; typically the objective function, constraints and, optionally, their gradients.

FRANOPP, FRamework for ANalysis and OPtimization Problems, is a software aid for the study and solution of design problems. FRANOPP provides the driving program and plotting capability for a user generated software system which also includes the general purpose optimization code CONMIN (created and maintained by NASA/Ames Research Center) and two user supplied codes, one for analysis (definition of the design problem) and one for output.

FRANOPP provides the user with five options for studying a design problem. The user may soley execute the analysis module, request an optimization run, or utilize one of the three plotting capabilities supplied by FRANOPP. The plotting options provide an indepth study of the problem and can be focused on a history of the optimization process or on the interaction of variables within the design problem.

A software system incorporating FRANOPP is easy to build and use. The user only needs to be concerned with the problem dependent aspects of the generated programming system. The modular design of the user-generated system and the code within FRANOPP make the system flexible for modifications and additions. FRANOPP may be used as an initial testbed for large problems before proceeding with a more complex optimization system.

CAPABILITIES AND OPTIONS

FRANOPP has five options available to the user. These are:

ANALIZE
PLOT LINEAR
PLOT CONTOUR
OPTIMIZE
OPTIMIZE WITH PLOTS

A general description of the capabilities of each is given below.

Option: ANALIZE

This option executes the user's analysis routine for a single set of design variable values and outputs the objective function and constraint values.

Option: PLOT LINEAR

The PLOT LINEAR option allows the user to examine the dependence of a specified output quantity on a specified input variable. In the general case, the user defines a design variable as the input parameter to vary over a given scale while all other input parameters remain unchanged. Also specified is the output quantity which the user wishes to study, usually the objective function or a constraint. This option loops through the analysis routine varying the specified parameter and saving both the input values and the specified output values to be plotted in a linear, two-variable plot.

Option: PLOT CONTOUR

PLOT CONTOUR, similar to the PLOT LINEAR option, allows the user to select an output quantity for observation while varying any two input parameters. The results are displayed in a contour plot.

Option: OPTIMIZE

The OPTIMIZE option uses the general purpose optimization code, CONMIN (ref. 1) to optimize the design problem defined in the user supplied analysis routine. The user begins the process by supplying to CONMIN an initial value for each design variable and some other required parameters. Using these initial values, the analysis routine evaluates the constraints and objective function. CONMIN is entered again and produces new values for the design variables. These new values are passed to the analysis routine and the model is re-evaluated. The process continues until either a feasible design is obtained with the objective function minimized or the user supplied value for the limit of iterations is reached.

Option: OPTIMIZE WITH PLOTS

This option is an extension of the OPTIMIZE option. In addition to providing the results from the optimization process, this option saves the values of the objective function, design variables[1] and constraints at each iteration, thereby providing a history of the process. The results are then displayed in a series of plots. There are two user defined input values which determine whether or not the objective function and design variable plots are plotted as they appear, or are normalized by the initial values before plotting. The user has no option for the constraint plots. These are always plotted in the normalized form $G_{act}/G_{all} - 1$, where G_{act} is the actual behavior variable value and G_{all} is the allowable behavior variable value.[2]

---

[1] When CONMIN computes the necessary gradients for the optimization process (instead of the user supplied analysis module), the values saved for the first design variable, except at the initial and final data points, will include an increment for the finite difference method. Since the increments are normally small, they do not change the character of the plot.

[2] A behavior variable is any output from the analysis, such as stresses or forces.

PROGRAMMING DESIGN

FRANOPP provides the framework for a programming system consisting of two programs. The first calculates the numerical results for the five options and the second handles the generation of plots. Since the plotting capability does not apply to all of the options, the second program is not always executed. The input/output files and the programs are described below. The flow of execution through the system for all options is shown in figures 1 and 2. A more detailed description and a listing of each FRANOPP supplied subroutine is provided in Appendix A.

Input/Output Files

The input/output files accessed by FRANOPP's programming system are listed with their major attributes. The order of their appearance reflects their position on the program header card.

(1) INPUT: equated to TAPE5; input to program one; input to program two; contains all the user defined input to execute FRANOPP; a detailed description of contents is presented in the section on execution.

(2) TAPE3: output from program one; input to program two; contains information regarding the type of plots requested and other required items.

(3) TAPE7: output from program one; input to program two; contains the data arrays to be plotted.

(4) OUTPUT: equated to TAPE6; output from program one; output from program two; contains the results of the option executed.

Program One

Program one consists of four modules. It contains a driving module supplied by FRANOPP, the optimization code CONMIN, and two user supplied modules, one for analysis (definition of design problem) and one for documentation of the output. An explanation of each module follows:

Module FAOPS. - This module drives the entire system. It contains the logic for all the options and provides the interface between the modules. The main routine FAOP reads an input command and calls the appropriate subroutine to execute the requested option. The subroutines are listed below with the corresponding option(s) in parenthesis.

Subroutine SUBANAL (ANALIZE)
Subroutine PLTLIN (PLOT LINEAR)
Subroutine PLTCON (PLOT CONTOUR)
Subroutine OPT (OPTIMIZE and OPTIMIZE WITH PLOTS)

The subroutine layout of this module reflects the independence and separation of the options. The separation allows for ease in modifying and extending FRANOPP's capabilities. A new option, for example, can be added without regard for those that already exist.

Module CONMIN. - CONMIN is a group of FORTRAN subroutines for the minimization of a multi-variable function subject to a set of inequality constraints. CONMIN solves both linear and nonlinear, and constrained or unconstrained problems. The basic algorithm used for the constrained problem is the Method of Feasible Directions. For unconstrained problems, the Conjugate Direction Method of Fletcher and Reeves is used. For information concerning CONMIN's organization and parameter definitions see reference 1.[3]

Module ANALYS. - ANALYS is a user supplied module that evaluates the objective function, constraints and, optionally, their gradients, given a set

---

[3] CONMIN has been designed so that the user can supply the gradients of the objective function and constraints in the analysis module or CONMIN will compute the necessary gradients using finite difference. When CONMIN computes the gradients, the design variables will be incremented by either absolute or relative values which are governed by the input parameters FDCHM and FDCH, respectively. The larger increment prevails. It is advisable to adjust these parameters so that the relative increment prevails in the anticipated range of the design.

of values for the design variables. For compatibility with CONMIN, the design problem must be defined as a minimization problem with the constraints computed as (actual behavior variable value/allowable behavior variable value -1).

This module may vary slightly depending on the option executed. For example, when executing the PLOT LINEAR and PLOT CONTOUR options, the user may choose to plot variables other than the design variables, objective function or constraints. In this case, an additional common block must be added to the analysis module so FRANOPP has access to the user defined variables. The implementation of these variations are explained in the execution section.

Module DEFINES. - DEFINES is a user supplied module for documentation of the computer printout. It allows the user to define a particular design problem in the output by printing appropriate explanations and definitions. This module should output text definitions of the objective function, design variables, constraints and any other information necessary for understanding the problem. The output from this module is placed directly under the title, before any results are printed.

Program Two

The second program, entirely supplied by FRANOPP, produces the plot vector file[4] for the plotter and therefore only needs to be executed when a plotting option is selected by the user. Two of the output files generated by the first program (TAPE3 and TAPE7) are used as input to program two. A brief description of the main program and major subroutines is given below.

Main Program PLOT. - The main program reads the parameter IPLOT from TAPE3 to determine which set of plots was requested and calls the necessary subroutine to do the plotting.

Subroutine LINPLOT. - This subroutine retrieves the plotting data from TAPE7 and calls the necessary library routines to produce the two-variable linear plot specified by the PLOT LINEAR option.

---

[4] The plot vector file contains the plotting commands used to control the plotter. It is created by FORTRAN calls to the LaRC graphics library.

Subroutine CONPLOT. - Similar to LINPLOT, this subroutine retrieves the plotting data from TAPE7 and produces a contour plot of any output quantity the user chooses, with respect to any two input variables.

Subroutine HISPLOT. - This subroutine plots the history of the optimization process resulting from the OPTIMIZE WITH PLOTS option. For the objective function and design variables, the user can request that the plots be normalized by the respective initial values.

EXECUTION

To build and execute a system which utilizes FRANOPP, the user must perform the following steps:

(1) Code the user supplied ANALYS and DEFINES modules.

(2) Modify the dimension statements in the code supplied by FRANOPP to the actual problem dimensions.

(3) Create the input file.

(4) Create the job stream.

Each step is described in detail below and an example is presented in Appendix B.

Step 1: User Supplied Modules

ANALYS. - The analysis module, ANALYS, consists of a user supplied subroutine(s) which evaluates the objective function, constraints and, optionally, their gradients. Its design and complexity depends almost entirely on the problem at hand and the user. To integrate the analysis module into FRANOPP's programming system, the user must follow the seven guidelines listed below:

(1) The analysis module must contain a subroutine named ANALYS, which, if desired, can call other user supplied subroutines.

(2) There are two required common blocks which are used to input and output the required data. The design variables are input to the module through the common block /VARABLE/. The output consists

of the objective function and constraints contained in the common block /CONSTRT/. These common blocks are defined below.

COMMON /VARABLE/ NDV parameters representing design variables.

COMMON /CONSTRT/ objective function, NCON parameters representing constraints.

(3) The output, stored in common blocks, passes from the ANALYS module to CONMIN without any intermediate changes. Therefore, the ANALYS module must compute the values in a form acceptable to CONMIN. CONMIN requires the objective function to be a minimization function. In addition, the constraints must be computed as

$$G = G_{act}/G_{all} - 1$$

where G_{act} is the actual behavior variable value and G_{all} is the allowable behavior variable value. In this form, the constraints are easily classified into three groups.

If $G > 0$, the constraint is violated,
if $G = 0$, the constraint is critical,
and if $G < 0$, the constraint is satisfied.

(4) If the analysis module requires any user defined input, the user can supply the necessary READs in the analysis subroutine(s) and adjust the input file appropriately. The user should incorporate a test to insure the input is only read the first time the analysis module is entered and then stored in a common block. To facilitate this, a common block /ANDATA/ which contains one variable, LOOPCNT, supplies the current number of calls to ANALYS. For example, on the first call to ANALYS, LOOPCNT is equal to one; the second time the analysis module is entered, LOOPCNT equals two. The logic is outlined below:

```
SUBROUTINE ANALYS
   .
   .  required common blocks
   .
COMMON /ANDATA/ LOOPCNT
COMMON /any-name/ all variables appearing in the
                  read statement(s)
   .
   .
   .
IF (LOOPCNT.NE.1) GO TO 101
   .
   .  read statement(s)
   .
101 CONTINUE
   .
   .
   .
RETURN
END
```

The read statement(s) should accept data from TAPE5. The location of this data within the input file will be described in the input file section.

(5) If additional output to that given by any option is desired, the user can supply write statements using TAPE6. Since the analysis module is executed several times for each iteration, the user may want to incorporate a test so that the output is written once for each iteration, or only for the final results. The variable ITER in the common block /CMNN1/ (listed in guideline 7) contains the current iteration number. To insure that the output is written once for each iteration, the user will need to save the value of ITER in a separate common block and compare the saved value with ITER until they are unequal. For the final results, the variable LOOPCNT (see guideline 4) has the value -999 on the last call to the analysis module.

(6) If the user wishes to execute either the PLOT LINEAR or PLOT CONTOUR options and specify plotting variables other than the design variables, objective function or constraints, an additional common block, /VINOUT/, must appear in the analysis module.

```
COMMON /VINOUT/ vin(2),vout
```

where vin(1) - if the user is supplying the input variable for the x-axis (INX < 0), the variable name should replace vin(1); otherwise, this is a dummy variable.

vin(2) - only applies to the PLOT CONTOUR option, it is a dummy variable for the PLOT LINEAR option. If the user is supplying the input variable for the y-axis (INY < 0), the variable name should replace vin(2); otherwise, this is a dummy variable.

vout - if the user is supplying the output variable (INY < 0 for PLOT LINEAR option, INZ < 0 for PLOT CONTOUR option) which corresponds to the y-axis or contours depending on the option, the variable name should replace vout; otherwise, vout is a dummy variable.

(7) When executing the OPTIMIZE or OPTIMIZE WITH PLOTS option, the analysis module may need to access or supply information other than that stored in the required common blocks /VARABLE/ and /CONSTRT/. For instance, a user may want to compute the required gradients in the analysis module instead of having CONMIN compute them. Two additional common blocks which facilitate the data transfer are shown below. The variables are defined in the CONMIN manual and the dimensions are defined in STEP2, Problem Dependent Dimension Statements.

```
Common /CNMN1/ DELFUN,DABFUN,FDCH,FDCHM,CT,CTMIN,CTL,
  CTLMIN,ALPHAX,ABOBJ1,THETA,OBJ,NDV,NCON,NSIDE,IPRINT,
  NFDG,NSCAL,LINOBJ,ITMAX,ITRM,ICNDIR,IGOTO,NAC,INFO,
  INFOG,ITER
Common /ACON/ VLB(N1),VUB(N1),SCAL(N1),DF(N1),A(N1,N3),
  ISC(N2),IC(N3)
```

DEFINES. - The DEFINES module consists of a user supplied subroutine called DEFINE. The sole purpose of this module is to document the problem in the output listing. The output produced should define the design problem by including definitions of the objective function, design variables, and constraints. The output must be written to TAPE6.

Step 2: Problem Dependent Dimension Statements

Both FRANOPP supplied source files, FAOPS and PLOTS, contain subroutines with problem dependent dimension statements. The dimension statements have been coded with undefined variables for the array sizes which must be replaced by the actual values before compilation of the system. The following table gives the location of these dimension statements within FRANOPP.

FILE	ROUTINE	VARIABLES USED IN DIMENSIONS
FAOPS	SUBANAL	N1, N2
	PLTLIN	N1, N2, ITMAX
	PLTCON	N1, N2
	OPT	N1, N2, N3, N4, N5
PLOTS	LINPLOT	ITMAX4
	HISPLOT	NDV, NCON, N1, N2, ITMAX2, ITMAX4

The variables are defined below. It is important to note that when the OPTIMIZE or OPTIMIZE WITH PLOTS option is executed, variables N1 through N5, and ITMAX correspond to the same input values used for CONMIN. For the PLOT LINEAR option, ITMAX, the maximum number of iterations used in the dimension statements, must be greater than or equal to NX, which is the number of iterations input to FRANOPP.

VARIABLE		DEFINITION
NDV	=	number of design variables
NCON	=	number of constraints
N1	=	NDV + 2
N2	=	NCON + 2 * NDV
N3	=	1 + user's best estimate of the maximum number of constraints (including side constraints) which will be active at any given time in the minimization process
N4	=	maximum of NDV and N3
N5	=	2 * N4
ITMAX	=	maximum number of iterations the user allows
ITMAX2	=	ITMAX + 2
ITMAX4	=	ITMAX + 4

To modify the variables in the dimension statements for each new problem, the user can either edit the source code directly, or execute a preprocessor which does the equivalent. The preprocessor route is favorable in that only the input to the preprocessor needs to be modified for each new problem instead of the source code. Since the editing option is self explanatory, the discussion will continue with the execution of the preprocessor available at NASA/LaRC called PRE.[5]

There are four files associated with PRE. These are INPUT, OUTPUT, INFILE, and PREFIL. The user must supply the preprocessor control cards containing the variables to be changed and their corresponding values on the input file, and the source code on the file INFILE. The program PRE generates a new source file called PREFIL, which contains constant dimensions and a group of assignment statements that set the variables used in the dimensions to the constants supplied. The listable output from the preprocessor verifying the success of the run appears on the output file.

[5] Documentation for NASA/LaRC implementation of PRE is available from the LaRC User Support Office.

The preprocessor control cards are of the form:

routine name.(variable 1 = constant, variable 2 = constant, . . .,
variable n = constant).

The job stream for execution of PRE along with an example of the input is shown in the section on the job control stream.

Step 3: Input File

The input file for FRANOPP consists of the input for both programs, FAOPS and PLOTS, separated by an end-of-record mark. The file contains a command specifying which option the user wishes to execute and the necessary input for that option. A table for each command showing the required input follows.

OPTION: ANALIZE-INPUT

Card Input & Format	Description & Comments	Routine
ANALIZE	command to execute option ANALIZE	FAOP
<title> Format(4A10)	title used in output	FAOP
$INPUT DV=_,_,...,_$ Namelist format	design variable values	SUBANAL
<user specified> (optional)	if the user has input into analysis routine from TAPE5, the data is placed here	ANALYS

OPTION: PLOT LINEAR-INPUT

Card Input & Format	Description & Comments	Routine
PLOT LINEAR	command to execute option PLOT LINEAR	FAOP
<title> Format(4A10)	title used in output	FAOP
$INPUT DV=_,_..._, VIN=_,INX=_,XINC=_, NX=_,INY_$ Namelist format	variables described on next page	PLTLIN
<user specified> (optional)	if the user has input into analysis routine from TAPE5, the data is placed here	ANALYS
<end-of-record>	denotes end of input to program one; beginning of input to program two	
<labelx,labely> Format(A10,5X,A10)	labels for the x and y axes of the generated plot	LINPLOT

OPTION: PLOT LINEAR-VARIABLES

VARIABLE	DESCRIPTION
DV	array containing design variable values
VIN(1)	if INX < 0, initial value for the user defined input; if INX > 0, unncessary
INX	> 0, index into DV array to specify which design variable to use for plotting; < 0, user defines the input variable for plotting in the common block /VINOUT/ in the analysis routine; = 0, error condition
XINC	the increment value for the input variable to be used in plotting
NX	the number of times to increment the input variable.
INY	> 0, index into array to specify which constraint value to use for the y-axis of the plot; = 0, the objective function is used as the output variable in the plot; < 0, user defines the output variable for plotting in the common block /VINOUT/ in the user supplied analysis routine

OPTION: PLOT CONTOUR-INPUT

Card Input & Format	Description & Comments	Routine
PLOT CONTOUR	command to execute option PLOT CONTOUR	FAOP
<title> Format(4A10)	title used in input	FAOP
$INPUT DV=_,_,..._, VIN=_,_,INX=_,XINC=_, NX=_,INY=_,YINC=_,NY_, INZ=_,NCONINT=_$ Namelist format	variables described on next page	PLTCON
<user specified> (optional)	if the user has input into analysis routine from TAPE5, the data is placed here	ANALYS
<end-of-record>	denotes end of input to program one; beginning of input to program two	
<labelx,labely,labelz> Format(A10,5X,A10,5X,A10)	labels for the x-axis, y-axis and contours of the generated plot	CONPLOT

OPTION: PLOT CONTOUR-VARIABLES

VARIABLE	DESCRIPTION
DV	array containing design variable values
VIN(1)	if INX < 0, initial value for the user defined input variable for the x-axis; if INX > 0, unnecessary;
VIN(2)	if INY < 0, initial value for the user defined input variable for y-axis; if INY > 0, unnecessary
INX	> 0, index into DV array to specify which design variable to use for the x-axis of the plot; < 0, user defines the input variable for the x-axis in the common block /VINOUT/ in the analysis routine; = 0, error condition
XINC	the increment value for the input variable for the x-axis
NX	the number of times to increment the input variable for the x-axis (must be $\leq$ 40)
INY	> 0, index into DV array to specify which design variable to use for the y-axis of plot; < 0, user defines the input variable for the y-axis in the common block /VINOUT/ in the analysis routine; = 0, error condition
YINC	the increment value for the y-axis input variable
NY	the number of times to increment the input variable for the y-axis (must be $\leq$ 40)

OPTION: PLOT CONTOUR-VARIABLES (cont'd)

VARIABLE	DESCRIPTION
INZ	> 0, index into array G to specify which constraint to use for the contours; = 0, the objective function is used as the output variable in the plot; < 0, user defines the output variable for plotting in common block /VINOUT/ in the user supplied analysis routine
NCONINT	the number of contour intervals (default = 20)

OPTION: OPTIMIZE or OPTIMIZE WITH PLOTS-INPUT

Card Input & Format	Description & Comments	Routine
OPTIMIZE or OPTIMIZE WITH PLOTS	commands to execute options OPTIMIZE and OPTIMIZE WITH PLOTS	FAOP
<title> Format(4A10)	title used in output	FAOP
$CONPAR <CONMIN parameters> $	for description of CONMIN parameters, see ref. 1	OPT
<user specified> (optional)	if the user has input into analysis routine from TAPE5, the data is placed here	ANALYS
<end-of-record>	denotes end of input to program one; beginning of input to program two	
<normobj,normdv> Format(2I2)	input only for OPTIMIZE WITH PLOTS option; keys to determine whether or not to normalize the objective function and design variable plots; described on next page	HISPLOT

OPTION: OPTIMIZE or OPTIMIZE WITH PLOTS-VARIABLES

VARIABLE	DESCRIPTION
NORMOBJ	= 1, normalize the objective function values by the initial value before plotting; = 0, plot the objective function values without normalization NOTE: 1 - Any other input defaults to 0. 2 - If the initial value of the objective function is 0, NORMOBJ will be disregarded and the plots will not be normalized.
NORMDV	= 1, normalize each design variable array by the starting value before plotting; = 0, plot the design variable arrays without normalization NOTE: 1 - Any other input defaults to 0. 2 - If the starting value of any design variable is 0, NORMDV will be disregarded and all design variables will be plotted without normalization.

Step 4: Job Control Stream

The CDC job control stream for the execution of FRANOPP is available in a procedure file, FAOPPRC. This procedure file uses the preprocessor PRE to change the variable dimensions to constants in the FRANOPP supplied programs. FAOPPRC is designed to check the value of register one to determine if the plotting program needs to be executed. If plotting is requested (options PLOT LINEAR, PLOT CONTOUR, or OPTIMIZE WITH PLOTS) register one should be assigned the integer value one and the preprocessor input for both programs must be supplied in the control stream. Otherwise, register one is set to zero, and only the preprocessor cards for program FAOPS should be supplied.

The procedure file has four parameters which the user must define in the call statement. These are:

A1 = binary for DEFINES module

B1 = binary for analysis module

C1 = input file for FRANOPP

UN1 = user number where A1, B1, and C1 reside

An example of a user's job control stream is given below, followed by a listing of the procedure file FAOPPRC.

User's job stream

```
/JOB
FRANOP,T100,CM110000.                  <DELIVERY INFOMATION>
USER,<USER NO.>.
CHARGE,<CHARGE NO.>,LRC.
DELIVER.<DELIVERY INFOMATION>
RFL(110000)
REDUCE(-)
SET(R1=1)----------> Set register 1 to 1 for plotting
GET,FAOPPRC/UN=<USER NO.>.
CALL,FAOPPRC(A1=EXDEFB,B1=EXANAB,C1=EXINPT,UN1=<USER NO.>)
REPLACE,PLTINPT,PLTDATA.
PLOT.VARIAN--------> Plotting instructions
EXIT.
/EOR
SUBANAL(N1=6,N2=11)                 }
PLTLIN(N1=6,N2=11,ITMAX=40)         } Input to PRE for FAOPS
PLTCON(N1=6,N2=11)                  }
OPT(N1=6,N2=11,N3=4,N4=4,N5=8)      }
/EOR
LINPLOT(ITMAX4=44)                                         } Input to PRE for PLOTS
HISPLOT(N1=6,N2=11,NDV=4,NCON=3,ITMAX2=42,ITMAX4=44)       }
```

FAOPPRC - procedure file

```
COMMENT.***** BEGIN PROC FAOPPRC *****
COMMENT. TO CALL:  CALL,FAOPPRC(A1=---,B1=---,C1=---,UN1=---)
COMMENT. WHERE:
COMMENT.  A1 = DEFINE MODULE BINARY FILE
COMMENT.  B1 = ANALYSIS MODULE BINARY FILE
COMMENT.  C1 = INPUT FILE TO FRANOP
COMMENT.  UN1 = ACCOUNT NUMBER WHERE
COMMENT.        A1,B1, AND C1 ARE
COMMENT.        LOCATED.
MAP,OFF.
GET,PRE/UN=738625C.
GET,FAOPS/UN=743575C.
PRE(,,FAOPS,PREFAOP)
REWIND,PREFAOP.
FTN(I=PREFAOP,B=FAOPB,L=0)
RETURN,FAOPS,PREFAOP. ————————————→Execution of PRE for FAOPS
IF(R1=0)GOTO,1.
GET,PLOTS/UN=743575C.
PRE(,,PLOTS,PREPLOT)
REWIND,PREPLOT.
FTN(I=PREPLOT,B=PLOTB,L=0)
RETURN,PLOTS,PREPLOT. ————————————→Execution of PRE for PLOTS
1,GET,A1,B1,C1/UN=UN1.
GET,BNEWC/UN=012686N.
ATTACH,FTNMLIB/UN=LIBRARY.
LDSET,LIB=FTNMLIB,PRESET=ZERO.
LOAD(FAOPB,A1,B1,BNEWC)
EXECUTE(,C1,PLTINPT,PLTDATA) ——————————→Execution of FAOPS
RETURN,A1,B1,BNEWC.
IF(R1=0)GOTO,2.
REWIND,PLTINPT,PLTDATA.
GET,PLOT3D/UN=159137N.
ATTACH(LRCGOSF/UN=LIBRARY)
LDSET(LIB=PLOT3D/LRCGOSF,PRESET=INDEF)
PLOTB,C1,PLTINPT,PLTDATA.——————————→ Execution of PLOTS
2,COMMENT.***** END PROC FAOPPRC *****
```

CONCLUDING REMARKS

FRANOPP, FRamework for ANalysis and OPtimization Problems, is a software aid for the study and solution of design (optimization) problems. It provides the framework for a user generated programming system in which the user needs to be concerned with the problem dependent aspects of the system. The modular design, at both the system and code levels, makes the system flexible for modifications and additions. FRANOPP may be used as an initial testbed for large problems before preceding with a more complex optimization system.

FRANOPP was developed and tested on the NASA/LaRC computers using the available software packages. Usage at another installation may require significant modifications. In addition, the optimization code incorporated into FRANOPP uses the Method of Feasible Directions for constrained problems and the Conjugate Direction Method of Fletchers and Reeves for unconstrained problems. If other optimization techniques are desired, revisions to FRANOPP will be necessary.

APPENDIX A: SOURCE CODE AND DESCRIPTIONS

This appendix contains the source code for the FRANOPP supplied programs. Each routine is presented separately with a descriptive outline preceding it.

ROUTINE NAME: FAOP

FILE NAME: FAOPS(source), FAOPB(binary)

PROGRAM 1

PROBLEM INDEPENDENT

OPTION(S): all

INPUT: INPUT(TAPE5) - user supplied
CARD(1)-CARD(5) - command to specify the option to be executed
TITLE(1)-TITLE(4) - title for output

OUTPUT: OUTPUT(TAPE6)
TITLE(1)-TITLE(4)
if an unrecognizable command is read, an error message is generated

COMMON: /PLT/ IPLOT,TITLE(4)

SUBROUTINE CALLS(S): SUBANAL,PLTLIN,PLTCON,OPT

DESCRIPTION: FAOP is the main routine in program one. This routine reads an input command and calls the appropriate subroutine to execute the option requested.

```
      PROGRAM FAOP(INPUT,TAPE3,TAPE7,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
      DIMENSION CARD(5)
      COMMON /PLT/ IPLOT,TITLE(4)
C
C     READ INPUT COMMAND AND PROBLEM TITLE
C
      READ(5,901) (CARD(I),I=1,5)
 901  FORMAT(5A4)
      READ(5,1010) TITLE
 1010 FORMAT(4A10)
      WRITE(6,1020) TITLE
 1020 FORMAT(1H1,6H **** ,4A10,5H ****,//)
C
C     DEFINE DESIGN VARIABLES AND CONSTRAINTS IN OUTPUT
C
      CALL DEFINE
C
C     DETERMINE WHICH OPTION IS REQUESTED
C     AND CALL THE APPROPRIATE SUBROUTINE
C
      IF (CARD(1).NE.4HOPTI) GO TO 101
      IPLOT=0
      IF (CARD(4).EQ.4HH PL) IPLOT=3
      CALL OPT
      GO TO 1000
 101  IF (CARD(1).NE.4HANAL) GO TO 200
      CALL SUBANAL
      GO TO 1000
 200  IF (CARD(1).NE.4HPLOT) GO TO 801
      IF (CARD(2).EQ.4H LIN) CALL PLTLIN
      IF (CARD(2).EQ.4H CON) CALL PLTCON
      GO TO 1000
 801  WRITE(6,951) (CARD(I),I=1,5)
 951  FORMAT(1H1,*THE FOLLOWING CARD IS NOT VALID INPUT:*,/,5X,
     2 5A4)
 1000 STOP
      END
```

ROUTINE NAME: SUBANAL

FILE NAME: FAOPS(source), FAOPB(binary)

PROGRAM 1

PROBLEM DEPENDENT: dimension statements

OPTION(S): ANALIZE

INPUT: INPUT(TAPE5) - user supplied
Namelist /INPUT/ DV - array of design variable values

OUTPUT: OUTPUT(TAPE6) - the results of the analysis

COMMON: /ANDATA/ LOOPCNT
/CONSTRT/ OBJ,G
/VARABLE/ DV

SUBROUTINE CALL(S): ANALYS

DESCRIPTION: This routine contains the necessary logic for the ANALIZE option. It reads in the values of the design variables, performs an analysis, and outputs the results.

```
      SUBROUTINE SUBANAL
      COMMON /VARABLE/ DV(N1)
      COMMON /CONSTRT/ OBJ,G(N2)
      COMMON /ANDATA/ LOOPCNT
      NAMELIST /INPUT/ DV
C
C     READ VALUES OF DESIGN VARABLES
C
      READ(5,INPUT)
C
C     ANALYSIS
C
      LOOPCNT = 1
      CALL ANALYS
C
C     WRITE RESULTS
C
      WRITE(6,901)
  901 FORMAT(1H1,*DESIGN VARABLES*)
      WRITE(6,902) DV
  902 FORMAT(4X,E12.4)
      WRITE(6,903)
  903 FORMAT(1H0,*OBJECTIVE FUNCTION*)
      WRITE(6,902) OBJ
      WRITE(6,904)
  904 FORMAT(1H0,*CONSTRAINTS*)
      WRITE(6,902) G
      RETURN
      END
```

ROUTINE NAME: PLTLIN

FILE NAME: FAOPS(source), FAOPB(binary)

PROGRAM 1

PPROBLEM DEPENDENT: dimension statements

OPTION(S): PLOT LINEAR

INPUT: INPUT(TAPE5) - user supplied

Namelist /INPUT/ DV,VIN(1),INX,XINC,NX,INY - includes values for the design variables, which two variables (one input and one output) are to be plotted, and parameters that determine the scale of the input variable to be plotted.

OUTPUT: OUTPUT(TAPE6) - a listing of the values to be plotted

TAPE3 and TAPE7 - all the necessary data for the plotting

COMMON: /ANDATA/ LOOPCNT

/CONSTRT/ OBJ,G

/PLT/ IPLOT,TITLE(4)

/VARABLE/ DV

/VINOUT/ VIN(2),VOUT

SUBROUTINE CALL(S): ANALYS

DESCRIPTION: This routine contains the necessary logic for the PLOT LINEAR option. The input consists of values for the design variables, which two variables (one input and one output) are to be plotted in a linear plot, and parameters that determine the scale of the input variable to be plotted. The input variable is incremented by a user defined amount in a loop containing a call to the analysis routine. All necessary plotting information is written to the data files used by the second program.

```
      SUBROUTINE PLTLIN
      DIMENSION XARRAY(ITMAX),YARRAY(ITMAX)
      COMMON /VARABLE/ DV(N1)
      COMMON /CONSTRT/ OBJ,G(N2)
      COMMON /VINOUT/ VIN(2),VOUT
      COMMON /ANDATA/ LOOPCNT
      COMMON /PLT/ IPLOT,TITLE(4)
      NAMELIST /INPUT/ DV,INX,VIN,XINC,NX,INY,VOUT
      NAMELIST /PLOT/ NX,XARRAY,YARRAY
      NAMELIST /IOPT/ IPLOT
C
C     READ DATA
C
      READ(5,INPUT)
      WRITE(6,906) TITLE
  906 FORMAT(1H1,4A10)
      WRITE(6,907)
  907 FORMAT(/,5X,*XARRAY*,10X,*YARRAY*,/)
C
C     PERFORM ANALYSIS NX TIMES
C     STORE SPECIFIED RESULTS FOR PLOTTING
C       IN ARRAYS XARRAY AND YARRAY
C
      DO 101 I=1,NX
      LOOPCNT=I
      CALL ANALYS
      IF (INX.GT.0) GO TO 103
      IF (INX.LT.0) GO TO 102
      WRITE(6,909)
  909 FORMAT(1H1,*INVALID INDEX - INX = 0*,/,
     . *NO PLOTTING ATTEMPTED*)
      GO TO 105
  102 XARRAY(I)=VIN(1)
      VIN(1)=VIN(1)+XINC
      GO TO 106
  103 XARRAY(I)=DV(INX)
      DV(INX)=DV(INX)+XINC
  106 YARRAY(I)=OBJ
      IF (INY.GT.0) YARRAY(I)=G(INY)
      IF (INY.LT.0) YARRAY(I)=VOUT
      WRITE(6,908) XARRAY(I),YARRAY(I)
  908 FORMAT(1H ,E12.4,4X,E12.4)
  101 CONTINUE
C
C     WRITE PLOTTING FILES FOR PROGRAM TWO
C
      IPLOT=1
      WRITE(3,IOPT)
      WRITE(3,932) TITLE
  932 FORMAT(4A10)
      WRITE(7,PLOT)
  105 CONTINUE
      RETURN
      END
```

ROUTINE NAME: PLTCON

FILE NAME: FAOPS(souce), FAOPB(binary)

PROGRAM 1

PROBLEM DEPENDENT: dimension statements

OPTION(S): PLOT CONTOUR

INPUT: INPUT(TAPE5) - user supplied

Namelist /INPUT/ DV,VIN,INX,XINC,NX,INY,YINC,NY,INZ,NCONINT - includes values for the design variables, which three variables (two input and one output) are to be plotted, and parameters determining the scales of the plot

OUTPUT: OUTPUT(TAPE6) - a listing of the values to be plotted
TAPE3 and TAPE7 - all the necessary data for the plotting

COMMON: /ANDATA/ LOOPCNT
/CONSTRT/ OBJ,G
/PLT/ IPLOT,TITLE(4)
/VARABLE/ DV
/VINOUT/ VIN(2),VOUT

SUBROUTINE CALL(S): ANALYS

DESCRIPTION: This subroutine contains the code for the PLOT CONTOUR option. The logic is the same as that for PLTLIN except that two input variables are defined by the user and varied in the loop. The output files store the necessary data for the second program, which produces a contour plot.

```
      SUBROUTINE PLTCON
      DIMENSION Z(40,40)
      COMMON /VARABLE/ DV(N1)
      COMMON /CONSTRT/ OBJ,G(N2)
      COMMON /VINOUT/ VIN(2), VOUT
      COMMON /ANDATA/ LOOPCNT
      COMMON /PLT/ IPLOT,TITLE(4)
      NAMELIST /INPUT/ DV,VIN,INX,XINC,NX,INY,YINC,NY,VOUT,INZ,NCONINT
      NAMELIST /PLOT/ XMIN,XMAX,XINC,YMIN,YMAX,YINC,Z,NCONINT
      NAMELIST /IOPT/ IPLOT
C
C     READ DATA
C
      NCONINT=20
      VIN(1)=VIN(2)=0.
      VOUT=0.
      READ(5,INPUT)
      WRITE(6,907)
 907  FORMAT(/,5X,*XARRAY*,10X,*YARRAY*,10X,*ZARRAY*,/)
      IF (INX.NE.0) GO TO 106
      WRITE(6,909)
 909  FORMAT(1H1,*INVALID INDEX - INX = 0*,/,
     . * NO PLOTTING ATTEMPTED*)
      GO TO 105
 106  XMIN=VIN(1)
      IF (INX.GT.0) XMIN=DV(INX)
      IF (INY.NE.0) GO TO 107
      WRITE(6,910)
 910  FORMAT(1H1,*INVALID INDEX - INY = 0*,/,
     . * NO PLOTTING ATTEMPTED*)
      GO TO 105
 107  YMIN=VIN(2)
      IF (INY.GT.0) YMIN=DV(INY)
C
C     LOOP FOR X-AXIS
C
      DO 102 I=1,NX
      LOOPCNT=I
      IF (INX.GT.0) VIN(1)=DV(INX)
      IF (I.EQ.NX) XMAX=VIN(1)
      VIN(2)=YMIN
      IF (INY.GT.0) DV(INY)=YMIN
C
C     LOOP FOR Y-AXIS
C
      DO 101 J=1,NY
      IF (INY.GT.0) VIN(2)=DV(INY)
      IF (J.EQ.NY) YMAX=VIN(2)
C
C     PERFORM ANALYSIS NX*NY TIMES
C
      CALL ANALYS
      Z(I,J)=OBJ
      IF (INZ.GT.0) Z(I,J)=G(INZ)
      IF (INZ.LT.0) Z(I,J)=VOUT
      WRITE(6,908) VIN(1),VIN(2),Z(I,J)
 908  FORMAT(1H ,E12.4,3X,E12.4,3X,E12.4)
      VIN(2)=VIN(2)+YINC
      IF (INY.GT.0) DV(INY)=VIN(2)
 101  CONTINUE
      VIN(1)=VIN(1)+XINC
      IF (INX.GT.0) DV(INX)=VIN(1)
 102  CONTINUE
C
C     WRITE PLOTTING FILES FOR PROGRAM TWO
C
      IPLOT=2
      WRITE(3,IOPT)
      WRITE(3,902) TITLE
 902  FORMAT(4A10)
      WRITE(7,PLOT)
 105  CONTINUE
      RETURN
      END
```

ROUTINE NAME: OPT

FILE NAME: FAOPS(source), FAOPB(binary)

PROGRAM 1

PROBLEM DEPENDENT: dimension statements

OPTION(S): OPTIMIZE and OPTIMIZE WITH PLOTS

INPUT: INPUT(TAPE5) - user supplied
Namelist /CONPAR/ CONMIN parameters

OUTPUT: OUTPUT(TAPE6) - CONMIN output
TAPE3 and TAPE7 - if plots requested, these files contain all the necessary data for the plotting

COMMON: /ACON/ VLB(N1),VUB(N1),SCAL(N1),DF(N1),A(N1,N3),ISC(N2),IC(N3)
/ANDATA/ LOOPCNT
/CNMN1/ CONMIN parameters
/CONSTRT/ OBJ,G
/PLT/ IPLOT,TITLE(4)
/VARABLE/ DV

SUBROUTINE CALL(S): CONMIN, ANALYS

DESCRIPTION: This subroutine contains the logic for both optimization options (with and without plots). It reads in the CONMIN parameters and makes the necessary data connections with CONMIN. The optimization is performed in an optimization loop which contains a call to both CONMIN and the analysis routine. If plotting is requested, it writes the necessary data files for the second program.

```
      SUBROUTINE OPT
      DIMENSION S(N1),G1(N2),G2(N2),B(N3,N3),C(N4),MS1(N5)
      DIMENSION SAVX(N1),SAVG(N2)
      INTEGER SAVITER
      COMMON /ACON/ VLB(N1),VUB(N1),SCAL(N1),DF(N1),A(N1,N3),
     .              ISC(N2),IC(N3)
      COMMON /CNMN1/ DELFUN,DABFUN,FDCH,FDCHM,CT,CTMIN,CTL,CTLMIN,
     .               ALPHAX,ABOBJ1,THETA,OBJ,NDV,NCON,NSIDE,IPRINT,
     .               NFDG,NSCAL,LINOBJ,ITMAX,ITRM,ICNDIR,IGOTO,NAC,
     .               INFO,INFOG,ITER
      COMMON /VARABLE/ X(N1)
      COMMON /CONSTRT/ AOBJ,G(N2)
      COMMON /ANDATA/ LOOPCNT
      COMMON /PLT/ IPLOT,TITLE(4)
      NAMELIST /CONPAR/ INFOG,INFO,NFDG,IPRINT,NDV,ITMAX,NCON,NSIDE,
     .                  ICNDIR,NSCAL,FDCH,FDCHM,CT,CTMIN,CTLMIN,THETA,
     .                  PHI,DELFUN,DABFUN,LINOBJ,ITRM,X,VLB,VUB,
     .                  N1,N2,N3,N4,N5,ALPHAX,ABOBJ1,CTL,ISC,SCAL
      NAMELIST /PLOT/ SAVITER,SAVOBJ,SAVX,SAVG
      NAMELIST /PLTINPT/ NOITER,ITMAX,NDV,NCON
      NAMELIST /IOPT/ IPLOT
C
C     READ THE PARAMETERS FOR CONMIN
C
      READ(5,CONPAR)
      WRITE(6,CONPAR)
      NLIM=ITMAX*(NDV+5)
C
C     NON-ITERATIVE PART OF ANALYSIS
C
      IGOTO = 0
      SAVITER=-1
C
C     ITERATIVE PART OF ANALYSIS
C
      DO 1000 I = 1,NLIM
      LOOPCNT=I
C
C     CALL THE OPTIMIZATION ROUTINE CONMIN
C
      CALL CONMIN(X,VLB,VUB,G,SCAL,DF,A,S,G1,G2,B,C,ISC,IC,MS1,N1,N2,
     .            N3,N4,N5)
C
      IF(IGOTO.EQ.0) LOOPCNT=-999
C
C     ANALYSIS MODULE
C
      CALL ANALYS
      OBJ=AOBJ
      IF (IPLOT.EQ.0) GO TO 1001
C
C     WRITE PLOTTING INFORMATION TO TAPE7
C       IF REQUESTED
C
  899 IF (ITER.EQ.SAVITER) GO TO 999
      IF (I.NE.1) WRITE(7,PLOT)
      SAVITER=ITER
  999 SAVOBJ=OBJ
      DO 101 II=1,N1
      SAVX(II)=X(II)
  101 CONTINUE
      DO 102 II=1,N2
      SAVG(II)=G(II)
  102 CONTINUE
 1001 IF (IGOTO.EQ.0) GO TO 1100
 1000 CONTINUE
C
C     WRITE PLOTTING INFORMATION TO TAPE3
C       IF REQUESTED
C
 1100 IF (IPLOT.EQ.0) GO TO 1200
      WRITE(7,PLOT)
      NOITER=ITER+1
```

```
      WRITE(3,IOPT)
      WRITE(3,907) TITLE
907   FORMAT(4A10)
      WRITE(3,PLTINPT)
      WRITE(6,906)
906   FORMAT(1H1,* PLOTTING INFORMATION WRITTEN TO TAPE.*)
1200  RETURN
      END
```

ROUTINE NAME: PLOT

FILE NAME: PLOTS(source), PLOTB(binary)

PROGRAM 2

PROBLEM INDEPENDENT

OPTION(S): PLOT LINEAR, PLOT CONTOUR, OPTIMIZE WITH PLOTS

INPUT: TAPE3 - generated by program one

Namelist /IOPT/ IPLOT - determines which plotting option was selected

TITLE(1)-TITLE(4) - title for plots

OUTPUT: OUTPUT(TAPE6) - if IPLOT=0, a message "NO PLOTTING" is printed

COMMON: /PLT/ IPLOT,TITLE(4)

SUBROUTINE CALL(S): LINPLOT, CONPLOT, HISPLOT

DESCRIPTION: PLOT is the main routine in program two. It reads in the value of IPLOT to determine which plots were requested and executes the appropriate subroutine.

```
      PROGRAM PLOT(INPUT,TAPE3,TAPE7,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
      COMMON /PLT/ IPLOT,TITLE(4)
      NAMELIST /IOPT/ IPLOT
C
C     READ IPLOT TO DETERMINE WHICH OPTION
C       IS REQUESTED
C     READ TITLE FOR PLOTS
C
      READ(3,IOPT)
      READ(3,902) TITLE
  902 FORMAT(4A10)
C
C     CALL APPROPRIATE SUBROUTINE TO PERFORM
C       THE PLOTTING
C
      IF (IPLOT.EQ.1) CALL LINPLOT
      IF (IPLOT.EQ.2) CALL CONPLOT
      IF (IPLOT.EQ.3) CALL HISPLOT
      IF (IPLOT.EQ.0) WRITE(6,901)
  901 FORMAT(1H1,* NO PLOTTING*)
      STOP
      END
```

ROUTINE NAME: LINPLOT

FILE NAME: PLOTS(source), PLOTB(binary)

PROGRAM 2

PROBLEM DEPENDENT: dimension statements

OPTION(S): PLOT LINEAR

INPUT: INPUT(TAPE5) - user supplied

LABELX,LABELY - labels for axes of plot

TAPE7 - generated by program one

Namelist /PLOT/ NX,XARRAY,YARRAY

NX - number of data pairs

XARRAY - the x values of the data pairs

YARRAY - the y values of the data pairs

OUTPUT: OUTPUT(TAPE6) - confirms the creation of the plotting file which gets sent to the plotter

COMMON: /PLT/ IPLOT,TITLE(4)

SUBROUTINE CALL(S): LRCGOSF Library

DESCRIPTION: This routine creates the plotting file for the two variable linear plots generated by the PLOT LINEAR option. It is designed to allow the user to specify any input variable for the x-axis and any output variable for the y-axis.

```
      SUBROUTINE LINPLOT
      DIMENSION XARRAY(ITMAX4),YARRAY(ITMAX4)
      COMMON /PLT/ IPLOT,TITLE(4)
      NAMELIST /PLOT/ NX,XARRAY,YARRAY
      CALL PSEUDO
C
C     READ DATA
C
      READ(7,PLOT)
      READ(5,901) LABELX, LABELY
  901 FORMAT(A10,5X,A10)
      NX1=NX+1
      NX2=NX+2
C
C     PLOT
C
      CALL CALPLT(2.,2.,-3)
      CALL ASCALE(XARRAY,8.,NX,1,10.0)
      CALL ASCALE(YARRAY,8.,NX,1,10.0)
      CALL AXES(0.,0.,0.,8.,XARRAY(NX1),XARRAY(NX2),1.0,0.,LABELX,.15,
     2 -10,3)
      CALL AXES(0.,0.,90.,8.,YARRAY(NX1),YARRAY(NX2),1.0,0.,LABELY,.15,
     2 10,3)
      CALL NOTATE(1.75,8.5,.15,TITLE,0.,40)
      CALL LINPLT(XARRAY,YARRAY,NX,1,1,1,1)
      CALL NFRAME
C
C     WRITE VERIFICATION OF PLOT
C
      WRITE(6,905) TITLE
  905 FORMAT(1H1,4A10)
      WRITE(6,906) LABELX, LABELY
  906 FORMAT(1H0,*TWO VARIABLE PLOT FOR *,A10,* AND *,A10,
     2 *WRITTEN TO PLOT FILE*)
      RETURN
      END
```

ROUTINE NAME: CONPLOT

FILE NAME: PLOTS(source), PLOTB(binary)

PROGRAM 2

OPTION(S): PLOT CONTOUR

INPUT: INPUT(TAPE5) - user supplied

LABELX,LABELY,LABELZ - labels for the x and y axes and definition of contours

TAPE7 - generated by program one

Namelist /PLOT/ XMIN,XMAX,XINC,YMIN,YMAX,YINC,Z,NCONINT

XMIN - minimum x value

XMAX - maximum x value

XINC - increment for x

YMIN - minimum y value

YMAX - maximum y value

YINC - increment for y

Z - array of values that corresponds to the (x,y) pairs

NCONINT - the number of contours to plot

OUTPUT: OUTPUT(TAPE6) - confirms the creation of the plotting file

COMMON: /PLT/ IPLOT,TITLE(4)

SUBROUTINE CALL(S): PLOT3D Library

DESCRIPTION: Similar to LINPLOT, this routine produces a contour plot of any output quantity the user chooses, with respect to any two input variables.

```
      SUBROUTINE CONPLOT
      REAL Z(40,40),XY(9),MM(4)
      COMMON /PLT/ IPLOT,TITLE(4)
      NAMELIST /PLOT/ XMIN,XMAX,XINC,YMIN,YMAX,YINC,Z,NCONINT
      CALL PSEUDO
      CALL CALPLT(2.,2.,-3)
C
C     READ DATA
C
      NC=40
      READ(7,PLOT)
      READ(5,903) LABELX, LABELY, LABELZ
  903 FORMAT(A10,5X,A10,5X,A10)
C
C     PLOT
C
      M=(XMAX-XMIN)/XINC+1
      N=(YMAX-YMIN)/YINC+1
      L=40
      CALL MAXMIN(Z,L,M,N,MM)
      FLO=MM(1)-.0000001
      HI=MM(2)
      FINC=(MM(2)-MM(1))/NCONINT
      I=FINC
      IF (FINC.GT.1.0.AND.FINC.NE.I) FINC=I+1.0
      NSET=NHI=0
      NDOT=682
      TSN=0.0
      XY(1)=XMIN
      XY(2)=YMIN
      XY(3)=XMAX
      XY(4)=YMAX
      XY(5)=0.
      XY(6)=0.
      XY(7)=(M-1)/8.
      XY(8)=(N-1)/8.
      XY(9)=1.
      CALL USCONTR(Z,L,M,N,FLO,HI,FINC,NSET,NHI,NDOT,XY,TSN,
     1XS,YS)
C
C     LABEL PLOT WITH PARAMETERS
C
      IS=3
      LT=3
      VD=1.5
      HD=4.
      AD=-1.
      IC=1
      CALL LABELER(TITLE,NC,IS,LT,VD,HD,RD,AO,AD,IC)
      VD=1.
      HD=2.
      AD=-1.
      IC=0
      CALL LABELER(11HCONTOURS OF,11,IS,LT,VD,HD,RD,AO,AD,IC)
      HD=4.
      NC=10
      CALL LABELER(LABELZ,NC,IS,LT,VD,HD,RD,AO,AD,IC)
      LT=1
      VD=1.
      HD=4.
      AD=-1.
      IC=1
      NC=10
      CALL LABELER(LABELX,NC,IS,LT,VD,HD,RD,AO,AD,IC)
      LT=4
      VD=1.
      HD=4.
      AD=-1.
      IC=1
      CALL LABELER(LABELY,NC,IS,LT,VD,HD,RD,AO,AD,IC)
      CALL NFRAME
      CALL CALPLT(1.,1.,999)
```

```
C
C     WRITE VERIFICATION OF PLOT
C
      WRITE(6,906) TITLE
 906  FORMAT(1H1,4A10)
      WRITE(6,907) LABELX, LABELY, LABELZ
 907  FORMAT(1H0,*CONTOUR PLOT*,/,4X,*X AXIS = *,A10,/,4X,
     . *Y AXIS = *,A10,/,4X,*CONTOURS = *,A10)
      RETURN
      END
```

ROUTINE NAME: HISPLOT

FILE NAME: PLOTS(source), PLOTB(binary)

PROGRAM 2

PROBLEM DEPENDENT - dimension statements

OPTION(S): PLOT CONTOUR

INPUT: INPUT(TAPE5) - user supplied

NORMOBJ,NORMDV - keys to determine whether or not to normalize the objective function and design variable plots.

TAPE3 - generated by program one

Namelist /PLTINPT/ NOITER,ITMAX,NDV,NCON

NOITER - the number of iterations for the optimization loop

ITMAX - iteration maximum for CONMIN

NDV - number of design variables

NCON - number of constraints

TAPE7 - generated by program one

Namelist /PLOT/ ITER,OBJ,X,G - this namelist appears NOITER times on this file (once for each iteration)

ITER - iteration number

OBJ - objective function value

X - an array of values for the design variables

G - an array of constraint values

OUTPUT: OUTPUT(TAPE6) - confirms the creation of the plotting file

COMMON: /PLT/ IPLOT,TITLE(4)

SUBROUTINE CALL(S): MAXMIN, LINZERO, LRCGOSF Library

ROUTINE NAME: HISPLOT (cont'd)

DESCRIPTION: This routine plots the history of the optimization process, the result from the OPTIMIZATION WITH PLOTS option. The plots can be divided into three sets:

Set 1: Iterations versus Objective Function

Set 2: Iterations versus Design Variables

Set 3: Iterations versus Constraints

Because a design problem may contain both a large number of design variables and/or constraints, a maximum of four quantities are placed on a single plot in sets 2 and 3.

```
      SUBROUTINE HISPLOT
      DIMENSION SAVX(N1),SAVG(N2),XARRAY(ITMAX4),YARRAY(ITMAX4)
      DIMENSION X2(ITMAX2,NDV),G2(ITMAX2,NCON),X2NORM(NDV)
      INTEGER SAVITER
      REAL MM(4)
      COMMON /PLT/ IPLOT,TITLE(4)
      NAMELIST /PLTINPT/ NOITER,ITMAX,NDV,NCON
      NAMELIST /PLOT/ SAVITER,SAVOBJ,SAVX,SAVG
      CALL PSEUDO
C
C     READ IN DATA
C
      READ(5,899) NORMOBJ,NORMDV
 899  FORMAT(2I2)
      READ(3,PLTINPT)
      DO 1000 I=1,NOITER
      READ(7,PLOT)
      XARRAY(I)=SAVITER-1
      YARRAY(I)=SAVOBJ
      DO 101 J=1,NDV
      X2(I,J)=SAVX(J)
 101  CONTINUE
      DO 102 J=1,NCON
      G2(I,J)=SAVG(J)
 102  CONTINUE
 1C00 CONTINUE
      NOIT1=NOITER+1
      NOIT2=NOITER+2
C
C     FIX SCALES FOR X-AXIS
C
      XARRAY(NOIT1)=0.0
      XARRAY(NOIT2)=NOITER/8.
      XMAJ=8./NOITER
C
C     OBJECTIVE FUNCTION - NORMALIZE, THEN PLOT
C
      OBJNORM=YARRAY(1)
      IF (OBJNORM.EQ.0..OR.NORMOBJ.NE.1) GO TO 105
      DO 106 I=1,NOITER
      YARRAY(I)=YARRAY(I)/OBJNORM
 106  CONTINUE
 105  CONTINUE
      CALL MAXMIN(YARRAY,ITMAX4,NOITER,1,MM)
      CALL ASCALE(MM,8.,2,1,10.0)
      YARRAY(NOIT1)=MM(3)
      YARRAY(NOIT2)=MM(4)
      CALL CALPLT(2.,2.,-3)
      CALL AXES(0.,0.,0.,8.,XARRAY(NOIT1),XARRAY(NOIT2),XMAJ,0.,
     .          10HITERATIONS,.15,-10)
      CALL AXES(0.,0.,90.,8.,YARRAY(NOIT1),YARRAY(NOIT2),1.0,0.,
     .          12HOBJ FUNCTION,.15,12)
      CALL LINPLT(XARRAY,YARRAY,NOITER,1,1,1,1)
      IF (YARRAY(NOIT1).NE.0.) CALL LINZERO(XARRAY,YARRAY,ITMAX4,NOITER)
      CALL NOTATE(1.75,8.5,.15,TITLE,0.,40)
      IF (OBJNORM.EQ.0..OR.NORMOBJ.NE.1) GO TO 107
      CALL NOTATE(9.0,6.75,.15,13HNORMALIZATION,0.,13)
      CALL NOTATE(9.0,6.5,.15,14HFACTOR AND KEY,0.,14)
      CALL NOTATE(9.0,6.0,.15,4HOBJ=,0.,4)
      CALL NUMBER(9.65,6.0,.15,OBJNORM,0.,4)
      CALL PNTPLT(11.0,6.08,1,1)
      GO TO 108
 107  CONTINUE
      CALL NOTATE(9.0,6.5,.15,3HKEY,0.,3)
      CALL NOTATE(9.0,6.0,.15,3HOBJ,0.,3)
      CALL PNTPLT(9.9,6.08,1,1)
 108  CONTINUE
      CALL NFRAME
      WRITE(6,900) TITLE
 900  FORMAT(1H1,4A10)
      WRITE(6,901)
 901  FORMAT(1H0,*OBJECTIVE FUNCTION*,/,* --------- --------*)
      IF (OBJNORM.NE.0..AND.NORMOBJ.EQ.1) WRITE(6,902) OBJNORM
 902  FORMAT(1H ,*NORMALIZED BY STARTING VALUE*,E16.8)
      DO 109 K=1,NOITER,7
      K2=K+6
```

```
      IF (NOITER.LT.K2) K2=NOITER
      WRITE(6,903) (KK-1,KK=K,K2)
  903 FORMAT(1H ,///,* ITER *,8X,I2,6(14X,I2),/)
      WRITE(6,904)
  904 FORMAT(1H ,120(*-*),/,*  OBJ  *)
      WRITE(6,905) (YARRAY(KK),KK=K,K2)
  995 FORMAT(1H ,* FUN  *,7E16.8)
  109 CONTINUE
C
C     DESIGN VARIABLES - NORMALIZE, THEN PLOT
C
      X2NORM(1)=X2(1,1)
      IF (NORMDV.NE.1) GO TO 118
      DO 110 J=2,NDV
      IF (X2(1,J).EQ.0.) X2NORM(1)=0.
  110 CONTINUE
      IF (X2NORM(1).EQ.0.) GO TO 118
      DO 111 J=1,NDV
      X2NORM(J)=X2(1,J)
      DO 111 I=1,NOITER
      X2(I,J)=X2(I,J)/X2NORM(J)
  111 CONTINUE
C
C     FIND MAX AND MIN OF ALL ARRAYS TO BE PLOTTED
C
  118 CONTINUE
      CALL MAXMIN(X2,ITMAX2,NOITER,NDV,MM)
      CALL ASCALE(MM,8.,2,1,10.0)
      YARRAY(NOIT1)=MM(3)
      YARRAY(NOIT2)=MM(4)
      DO 112 I=1,NDV,4
      Y=6.5
      DO 113 K=1,4
      II=I+K-1
      IF (II.GT.NDV) GO TO 117
      DO 114 J=1,NOITER
      YARRAY(J)=X2(J,II)
  114 CONTINUE
      IF (K.NE.1) GO TO 115
      CALL CALPLT(2.,2.,-3)
      CALL AXES(0.,0.,0.,8.,XARRAY(NOIT1),XARRAY(NOIT2),XMAJ,0.,
     .          10HITERATIONS,.15,-10)
      CALL AXES(0.,0.,90.,8.,YARRAY(NOIT1),YARRAY(NOIT2),1.0,0.,
     .          16HDESIGN VARIABLES,.15,16)
  115 CALL LINPLT(XARRAY,YARRAY,NOITER,1,1,K,1)
      Y=Y-.5
      Y2=Y+.08
      IF (K.NE.1) GO TO 116
      CALL NOTATE(1.75,8.5,.15,TITLE,0.,40)
      IF (X2NORM(1).EQ.0..OR.NORMDV.NE.1) GO TO 119
      CALL NOTATE(9.0,6.75,.15,13HNORMALIZATION,0.,13)
      CALL NOTATE(9.0,6.5,.15,14HFACTOR AND KEY,0.,14)
  116 CONTINUE
      IF (X2NORM(1).EQ.0..OR.NORMDV.NE.1) GO TO 119
      CALL NOTATE(9.0,Y,.15,1HX,0.,1)
      R=II
      CALL NUMBER(9.15,Y,.15,R,0.,-1)
      CALL NOTATE(9.3,Y,.15,1H=,0.,1)
      CALL NUMBER(9.5,Y,.15,X2NORM(II),0.,4)
      CALL PNTPLT(11.0,Y2,K,1)
      GO TO 113
  119 CONTINUE
      IF (K.EQ.1) CALL NOTATE(9.0,6.5,.15,3HKEY,0.,3)
      CALL NOTATE(9.0,Y,.15,1HX,0.,1)
      R=II
      CALL NUMBER(9.15,Y,.15,R,0.,-1)
      CALL PNTPLT(9.75,Y2,K,1)
  113 CONTINUE
  117 IF (YARRAY(NOIT1).NE.0.) CALL LINZERO(XARRAY,YARRAY,ITMAX4,NOITER)
      CALL NFRAME
  112 CONTINUE
```

```
C
C     WRITE RESULTS
C
      WRITE(6,906)
 906  FORMAT(1H1,*DESIGN VARIABLES*,/,* ------ ---------*)
      IF (X2NORM(1).EQ.0..OR.NORMDV.NE.1) GO TO 135
      WRITE(6,907)
 907  FORMAT(1H ,*NORMALIZED BY STARTING VALUES*)
      WRITE(6,910) (J,X2NORM(J),J=1,NDV)
 910  FORMAT(1H ,*    X*,I2,* =*,E16.8)
 135  DO 121 K=1,NOITER,7
      K2=K+6
      IF (NOITER.LT.K2) K2=NOITER
      WRITE(6,903) (KK-1,KK=K,K2)
      WRITE(6,908)
 908  FORMAT(1H ,120(*-*),/,* D VAR*)
      DO 122 J=1,NDV
      WRITE(6,909) J,(X2(KK,J),KK=K,K2)
 909  FORMAT(1H ,*    X*,I2,2X,7E16.8)
 122  CONTINUE
 121  CONTINUE
C
C     CONSTRAINTS - PLOT
C
C     FIND MAX AND MIN OF ALL ARRAYS TO BE PLOTTED
C
      CALL MAXMIN(G2,ITMAX2,NOITER,NCON,MM)
      CALL ASCALE(MM,8.,2,1,10.0)
      YARRAY(NOIT1)=MM(3)
      YARRAY(NOIT2)=MM(4)
      DO 212 I=1,NCON,4
      Y=6.5
      DO 213 K=1,4
      II=I+K-1
      IF (II.GT.NCON) GO TO 217
      DO 214 J=1,NOITER
      YARRAY(J)=G2(J,II)
 214  CONTINUE
      IF (K.NE.1) GO TO 215
      CALL CALPLT(2.,2.,-3)
      CALL AXES(0.,0.,0.,8.,XARRAY(NOIT1),XARRAY(NOIT2),XMAJ,0.,
     .          10HITERATIONS,.15,-10)
      CALL AXES(0.,0.,90.,8.,YARRAY(NOIT1),YARRAY(NOIT2),1.0,0.,
     .          11HCONSTRAINTS,.15,11)
      CALL NOTATE(1.75,8.5,.15,TITLE,0.,40)
      CALL NOTATE(9.0,6.5,.15,3HKEY,0.,3)
 215  CALL LINPLT(XARRAY,YARRAY,NOITER,1,1,K,1)
      Y=Y-.5
      Y2=Y+.08
      CALL NOTATE(9.0,Y,.15,1HG,0.,1)
      R=II
      CALL NUMBER(9.15,Y,.15,R,0.,-1)
      CALL PNTPLT(9.75,Y2,K,1)
 213  CONTINUE
 217  IF (YARRAY(NOIT1).NE.0.) CALL LINZERO(XARRAY,YARRAY,ITMAX4,NOITER)
      CALL NFRAME
 212  CONTINUE
C
C     WRITE RESULTS
C
      WRITE(6,911)
 911  FORMAT(1H1,*CONSTRAINTS*,/,* -----------*)
      DO 221 K=1,NOITER,7
      K2=K+6
      IF (NOITER.LT.K2) K2=NOITER
      WRITE(6,903) (KK-1,KK=K,K2)
      WRITE(6,912)
 912  FORMAT(1H ,120(*-*),/,* CONST*)
      DO 222 J=1,NCON
      WRITE(6,913) J,(G2(KK,J),KK=K,K2)
 913  FORMAT(1H ,*    G*,I2,2X,7E16.8)
 222  CONTINUE
 221  CONTINUE
      RETURN
      END
```

Additional subroutines accessed by HISPLOT, MAXMIN and LINZERO, are listed below. Subroutine MAXMIN finds the maximum and minimum values in a two-dimensional array. Subroutine LINZERO draws the zero line on a plot.

```
      SUBROUTINE MAXMIN(X,NDIM1,N,NDIM2,MM)
      PEAL X(NDIM1,NDIM2),MM(4)
      XMIN=XMAX=0
      DO 101 I=1,NDIM2
      DO 101 J=1,N
      IF (X(J,I).LT.XMIN) XMIN=X(J,I)
      IF (X(J,I).GT.XMAX) XMAX=X(J,I)
101   CONTINUE
      MM(1)=XMIN
      MM(2)=XMAX
      RETURN
      END
```

```
      SUBROUTINE LINZERO(XARRAY,YARRAY,MAX2,NO)
      DIMENSION XARRAY(MAX2),YARRAY(MAX2)
      DIMENSION X(4),Y(4)
      NO1=NO+1
      NO2=NO+2
      X(1)=Y(1)=Y(2)=0.
      X(2)=XARRAY(NO)+1.
      X(3)=XARRAY(NO1)
      X(4)=XARRAY(NO2)
      Y(3)=YARRAY(NO1)
      Y(4)=YARRAY(NO2)
      CALL LINE(X,Y,2,1,0,0,0.)
      RETURN
      END
```

APPENDIX B: EXAMPLE

This appendix contains a sample execution of the OPTIMIZE WITH PLOTS option. The example chosen is the Constrained Rosen-Suzuki Function. This example is also presented as the first example in the CONMIN manual and was chosen for this reason. A listing of all the user defined files is provided, followed by the output.

Analysis module

```
      SUBROUTINE ANALYS
C
      COMMON /VARABLE/ X(4)
      COMMON /CONSTRT/ OBJ,G(3)
C
C     ANALYSIS
C
      OBJ=X(1)**2-5.*X(1)+X(2)**2-5.*X(2)+2.*X(3)**2-21.*X(3)+
     1X(4)**2+7.*X(4)+50.
      G(1)=X(1)**2+X(1)+X(2)**2-X(2)+X(3)**2+X(3)+X(4)**2-X(4)-8.
      G(2)=X(1)**2-X(1)+2.*X(2)**2+X(3)**2+2.*X(4)**2-X(4)-10.
      G(3)=2.*X(1)**2+2.*X(1)+X(2)**2-X(2)+X(3)**2-X(4)-5.
C
      RETURN
      END
```

Defines module

```
      SUBROUTINE DEFINE
      WRITE(6,1)
1     FORMAT(1H0,16HDESIGN VARIABLES,/,
     .5H X(1),/,
     .5H X(2),/,
     .5H X(3),/,
     .5H X(4))
      WRITE(6,2)
2     FORMAT(1H0,18HOBJECTIVE FUNCTION,/,
     .59H OBJ  =   X(1)**2 - 5*X(1) +   X(2)**2 - 5*X(2) + 2*X(3)**2,
     .36H   21*X(3) +   X(4)**2 + 7*X(4) + 50)
      WRITE(6,3)
3     FORMAT(1H0,11HCONSTRAINTS,/,
     .59H G(1) =   X(1)**2 +   X(1) +   X(2)**2 -   X(2) +   X(3)**2,
     .36H +    X(3) +   X(4)**2 -   X(4) -  8,/,
     .59H G(2) =   X(1)**2 -   X(1) + 2*X(2)**2         +   X(3)**2,
     .36H           + 2*X(4)**2 -   X(4) - 10,/,
     .59H G(3) = 2*X(1)**2 + 2*X(1) +   X(2)**2 -   X(2) +   X(3)**2,
     .36H                         -   X(4) -  5)
      RETURN
      END
```

Input file

```
OPTIMIZE WITH PLOTS
TEST OF OPTIMIZE WITH PLOTS OPTION
 $CONPAR
 X(1)=1.,
 X(2)=1.,
 X(3)=1.,
 X(4)=1.,
 N1=6,
 N2=11,
 N3=4,
 N4=4,
 N5=8,
 NDV=4,
 NCON=3,
 NSIDE=0,
 IPRINT=2,
 NFDG=0,
 NSCAL=0,
 DELFUN=0.,
 ITMAX=11,
 $END
 1 1 ———→ End of record
```

Job stream

```
/JOB
FRANOP,T100,CM110000.                          <DELIVERY INFOMATION>
USER,<USER NO.>.
CHARGE,<CHARGE NO.>,LRC.
DELIVER.<DELIVERY INFOMATION>
RFL(110000)
REDUCE(-)
SET(R1=1)
GET,FAOPPRC/UN=<USER NO.>.
CALL,FAOPPRC(A1=EXDEFB,B1=EXANAB,C1=EXINPT,UN1=<USER NO.>)
REPLACE,PLTINPT,PLTDATA.
PLOT.VARIAN
EXIT.
/EOR
SUBANAL(N1=6,N2=11)
PLTLIN(N1=6,N2=11,ITMAX=40)
PLTCON(N1=6,N2=11)
OPT(N1=6,N2=11,N3=4,N4=4,N5=8)
/EOR
LINPLOT(ITMAX4=44)
HISPLOT(N1=6,N2=11,NDV=4,NCON=3,ITMAX2=42,ITMAX4=44)
```

FRANOPP OUTPUT

PREPROCESSOR CONTROL CARDS

```
SUBANAL(N1=6,N2=11)
PLTLIN(N1=6,N2=11,ITMAX=40)
PLTCON(N1=6,N2=11)
OPT(N1=6,N2=11,N3=4,N4=4,N5=8)
```

PREPROCESSOR CONTROL CARDS

```
LINPLOT(ITMAX2=42)
HISPLOT(N1=6,N2=11,NDV=4,NCON=3,ITMAX=40,ITMAX2=42)
```

```
**** TEST OF OPTIMIZE WITH PLOTS OPTION      ****


DESIGN VARIABLES
X(1)
X(2)
X(3)
X(4)

OBJECTIVE FUNCTION
OBJ  =   X(1)**2 - 5*X(1) +   X(2)**2 - 5*X(2) + 2*X(3)**2   21*X(3) +   X(4)**2 + 7*X(4) + 50

CONSTRAINTS
G(1) =   X(1)**2 +   X(1) +   X(2)**2 -   X(2) +   X(3)**2 +   X(3) +   X(4)**2 -   X(4) -  8
G(2) =   X(1)**2 -   X(1) + 2*X(2)**2           +   X(3)**2          + 2*X(4)**2 -   X(4) - 10
G(3) = 2*X(1)**2 + 2*X(1) +   X(2)**2 -   X(2) +   X(3)**2                       -   X(4) -  5
```

```
$CONPAR
INFOG   = 0,
INFO    = 0,
NFDG    = 0,
IPRINT  = 2,
NDV     = 4,
ITMAX   = 40,
NCON    = 3,
NSIDE   = 0,
ICNDIR  = 0,
NSCAL   = 0,
FDCH    = 0.0,
FDCHM   = 0.0,
CT      = 0.0,
CTMIN   = 0.0,
CTLMIN  = 0.0,
THETA   = 0.0,
PHI     = 0.0,
DELFUN  = 0.0,
DABFUN  = 0.0,
LINOBJ  = 0,
ITRM    = 0,
X       = .1E+01, .1E+01, .1E+01, .1E+01, 0.0, 0.0,
VLB     = 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
VUB     = 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
```

```
N1      = 6,
N2      = 11,
N3      = 4,
N4      = 4,
N5      = 8,
ALPHAX  = 0.0,
ABOBJ1  = 0.0,
CTL     = 0.0,
$END
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                   *
*                   C O N M I N                     *
*                                                   *
*               FORTRAN PROGRAM FOR                 *
*                                                   *
*        CONSTRAINED FUNCTION MINIMIZATION          *
*                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * *


CONSTRAINED FUNCTION MINIMIZATION

CONTROL PARAMETERS

IPRINT  NDV    ITMAX    NCON     NSIDE  ICNDIR   NSCAL   NFDG
  2      4      40        3        0       5       0       0

LINOBJ  ITRM    N1       N2       N3      N4      N5
  0      3       6       11        4       4       8

    CT             CTMIN            CTL             CTLMIN
 -.10000E+00     .40000E-02     -.10000E-01      .10000E-02

   THETA            PHI            DELFUN           DABFUN
  .10000E+01     .50000E+01      .10000E-03      .31000E-01

   FDCH            FDCHM           ALPHAX           ABOBJ1
  .10000E-01     .10000E-01      .10000E+00      .10000E+00

ALL CONSTRAINTS ARE NON-LINEAR


INITIAL FUNCTION INFORMATION

OBJ =    .310000E+02

DECISION VARIABLES (X-VECTOR)
  1)     .10000E+01   .10000E+01   .10000E+01   .10000E+01

CONSTRAINT VALUES (G-VECTOR)
  1)    -.40000E+01  -.60000E+01  -.10000E+01
```

```
ITER =    1     OBJ =     .25484E+02

DECISION VARIABLES (X-VECTOR)
  1)     .10436E+01   .10436E+01   .12479E+01   .86847E+00

CONSTRAINT VALUES (G-VECTOR)
  1)    -.31307E+01  -.55788E+01  0.


ITER =    2     OBJ =     .12204E+02

DECISION VARIABLES (X-VECTOR)
  1)    -.65498E+00   .10325E+01   .23572E+01   .13804E+00

CONSTRAINT VALUES (G-VECTOR)
  1)    -.39775E+00  -.13275E+01  -.25580E-12


ITER =    3     OBJ =     .83763E+01

DECISION VARIABLES (X-VECTOR)
  1)     .22440E+00   .99268E+00   .20345E+01  -.31841E+00

CONSTRAINT VALUES (G-VECTOR)
  1)    -.11388E+01  -.35427E+01  -.56843E-13


ITER =    4     OBJ =     .69420E+01

DECISION VARIABLES (X-VECTOR)
  1)    -.34392E+00   .10043E+01   .21498E+01  -.80388E+00

CONSTRAINT VALUES (G-VECTOR)
  1)    0.           -.80266E+00  -.21613E-01
```

```
ITER =    5     OBJ =    .63271E+01

DECISION VARIABLES (X-VECTOR)
  1)    -.67566E-01   .10136E+01   .20734E+01  -.81323E+00

CONSTRAINT VALUES (G-VECTOR)
  1)    -.20225E+00  -.14382E+01  0.



ITER =    6     OBJ =    .61723E+01

DECISION VARIABLES (X-VECTOR)
  1)    -.94581E-01   .99247E+00   .20400E+01  -.96346E+00

CONSTRAINT VALUES (G-VECTOR)
  1)    0.           -.94507E+00  -.53852E-01



ITER =    7     OBJ =    .60706E+01

DECISION VARIABLES (X-VECTOR)
  1)     .74640E-01   .98928E+00   .19478E+01  -.10562E+01

CONSTRAINT VALUES (G-VECTOR)
  1)    -.16766E-01  -.10302E+01  0.



ITER =    8     OBJ =    .60218E+01

DECISION VARIABLES (X-VECTOR)
  1)    -.17653E-01   .10038E+01   .20139E+01  -.97523E+00

CONSTRAINT VALUES (G-VECTOR)
  1)    -.17726E-01  -.10338E+01   .28422E-13



ITER =    9     OBJ =    .60182E+01

DECISION VARIABLES (X-VECTOR)
  1)     .23921E-01   .99428E+00   .19869E+01  -.10102E+01

CONSTRAINT VALUES (G-VECTOR)
  1)    -.15891E-01  -.10472E+01   .11747E-02
```

```
ITER =   10     OBJ =    .60133E+01

DECISION VARIABLES (X-VECTOR)
  1)    -.17147E-01   .10055E+01    .20139E+01  -.97533E+00

CONSTRAINT VALUES (G-VECTOR)
  1)    -.15050E-01  -.10270E+01    .29211E-02


ITER =   11     OBJ =    .60098E+01

DECISION VARIABLES (X-VECTOR)
  1)     .19441E-01   .99482E+00    .19908E+01  -.10058E+01

CONSTRAINT VALUES (G-VECTOR)
  1)    -.13894E-01  -.10474E+01    .34703E-02
```

```
FINAL OPTIMIZATION INFORMATION

 OBJ =     .600982E+01

 DECISION VARIABLES (X-VECTOR)
   1)     .19441E-01   .99482E+00   .19908E+01  -.10058E+01

 CONSTRAINT VALUES (G-VECTOR)
   1)    -.13894E-01  -.10474E+01   .34703E-02

 THERE ARE    2 ACTIVE CONSTRAINTS
 CONSTRAINT NUMBERS ARE
     1    3

 THERE ARE    0 VIOLATED CONSTRAINTS

 TERMINATION CRITERION
      ABS(OBJ(I)-OBJ(I-1))   LESS THAN DABFUN FOR  3 ITERATIONS

 NUMBER OF ITERATIONS =   11

 OBJECTIVE FUNCTION WAS EVALUATED          78  TIMES

 CONSTRAINT FUNCTIONS WERE EVALUATED       78  TIMES
```

TEST OF OPTIMIZE WITH PLOTS OPTION

OBJECTIVE FUNCTION

NORMALIZED BY STARTING VALUE .31000000E+02

ITER	0	1	2	3	4	5	6
OBJ FUN	.10000000E+01	.82109838E+00	.39165394E+00	.26873788E+00	.22210476E+00	.20244614E+00	.19743440E+00

ITER	7	8	9	10	11
OBJ FUN	.19426430E+00	.19263189E+00	.19254048E+00	.19235587E+00	.19386524E+00

DESIGN VARIABLES

NORMALIZED BY STARTING VALUES

X 1 = .10000000E+01
X 2 = .10000000E+01
X 3 = .10000000E+01
X 4 = .10000000E+01

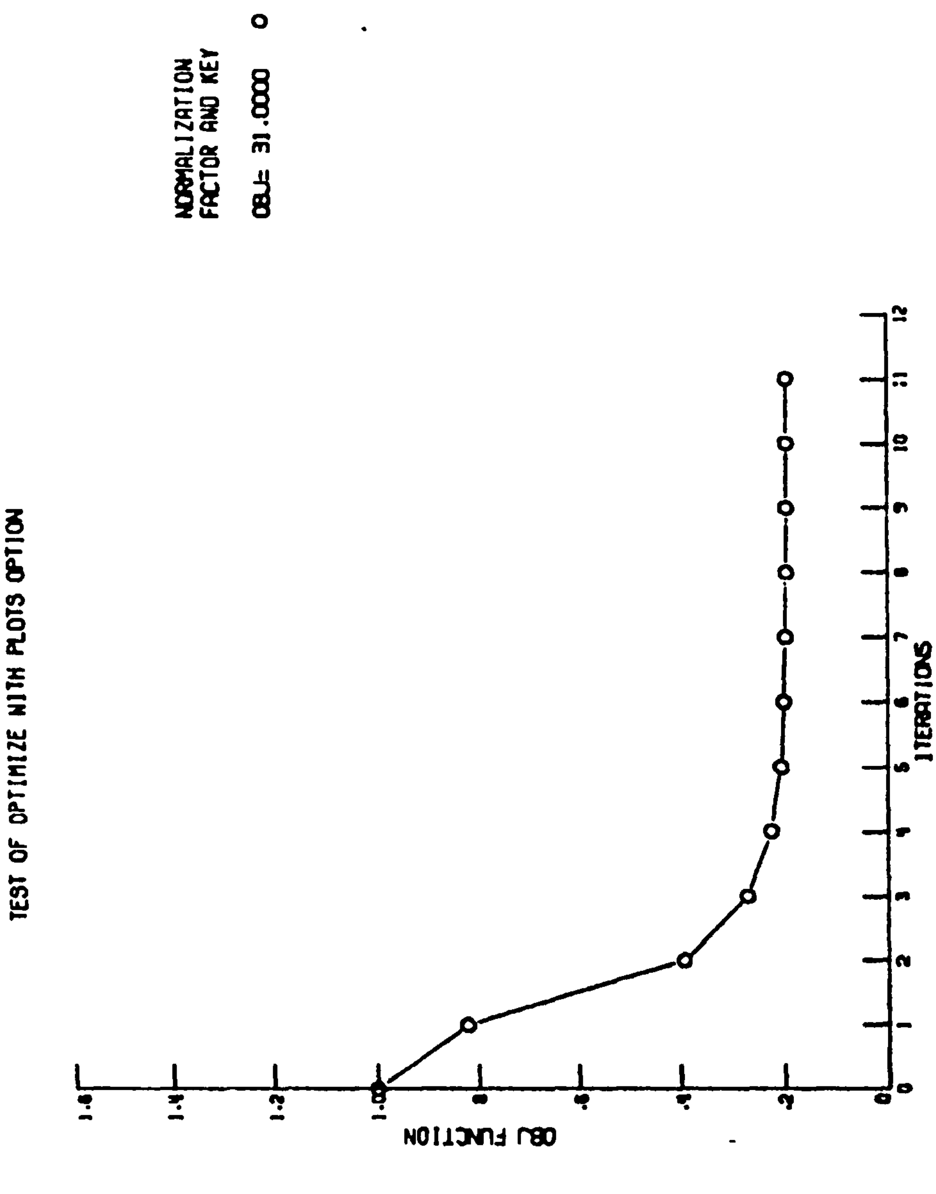
ITER	0	1	2	3	4	5	6
D VAR							
X 1	.10000000E+01	.10540858E+01	-.64498316E+00	.23440079E+00	-.33391709E+00	-.57565808E-01	-.84581011E-01
X 2	.10000000E+01	.10436493E+01	.10324695E+01	.99267962E+00	.10043380E+01	.10136010E+01	.99246925E+00
X 3	.10000000E+01	.12478812E+01	.23572179E+01	.20345385E+01	.21497500E+01	.20733985E+01	.20399587E+01
X 4	.10000000E+01	.86846822E+00	.13803935E+00	-.31840699E+00	-.80388119E+00	-.81323403E+00	-.96346139E+00

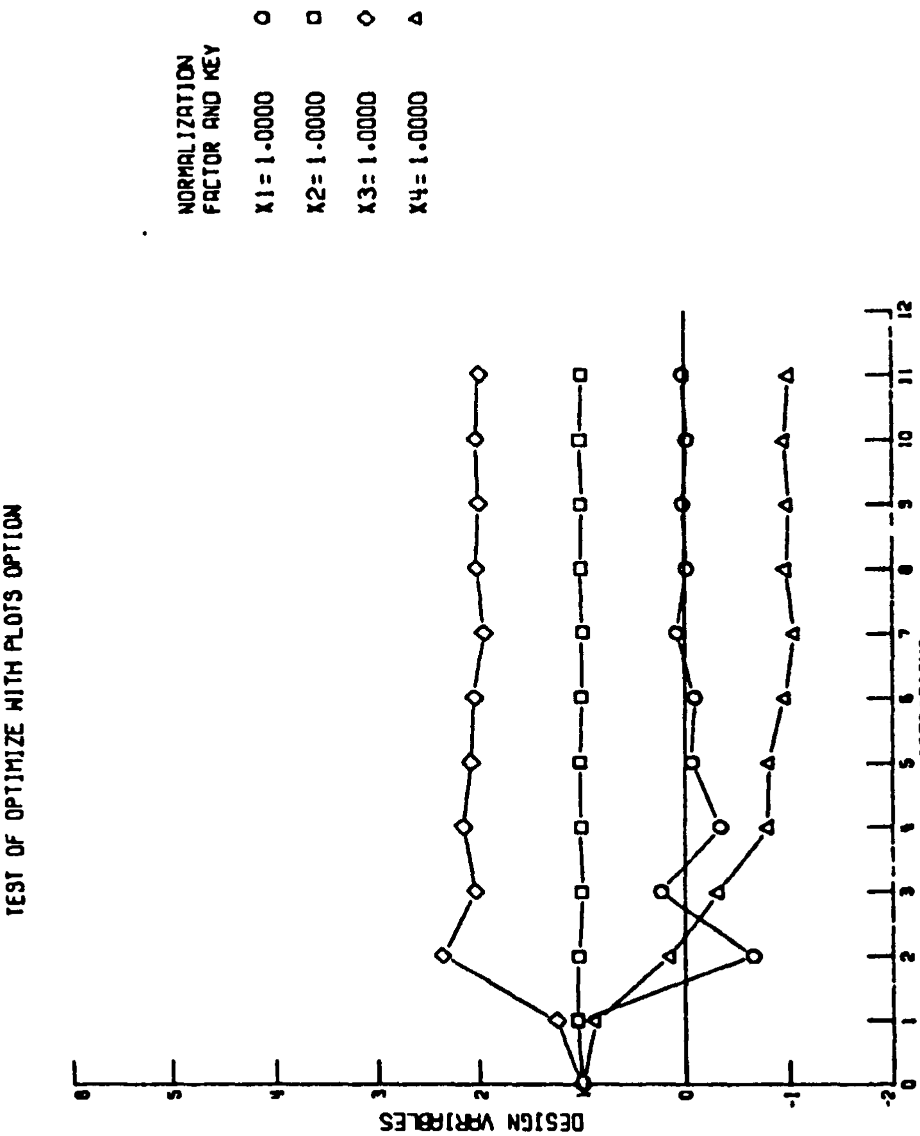
ITER	7	8	9	10	11
D VAR					
X 1	.84640201E-01	-.76531029E-02	.33920629E-01	-.71471336E-02	.19441181E-01
X 2	.98928228E+00	.10038076E+01	.99427809E+00	.10054562E+01	.99481727E+00
X 3	.19478042E+01	.20138598E+01	.19868698E+01	.20139041E+01	.19907692E+01
X 4	-.10562390E+01	-.97522950E+00	-.10102266E+01	-.97533160E+00	-.10058258E+01

CONSTRAINTS

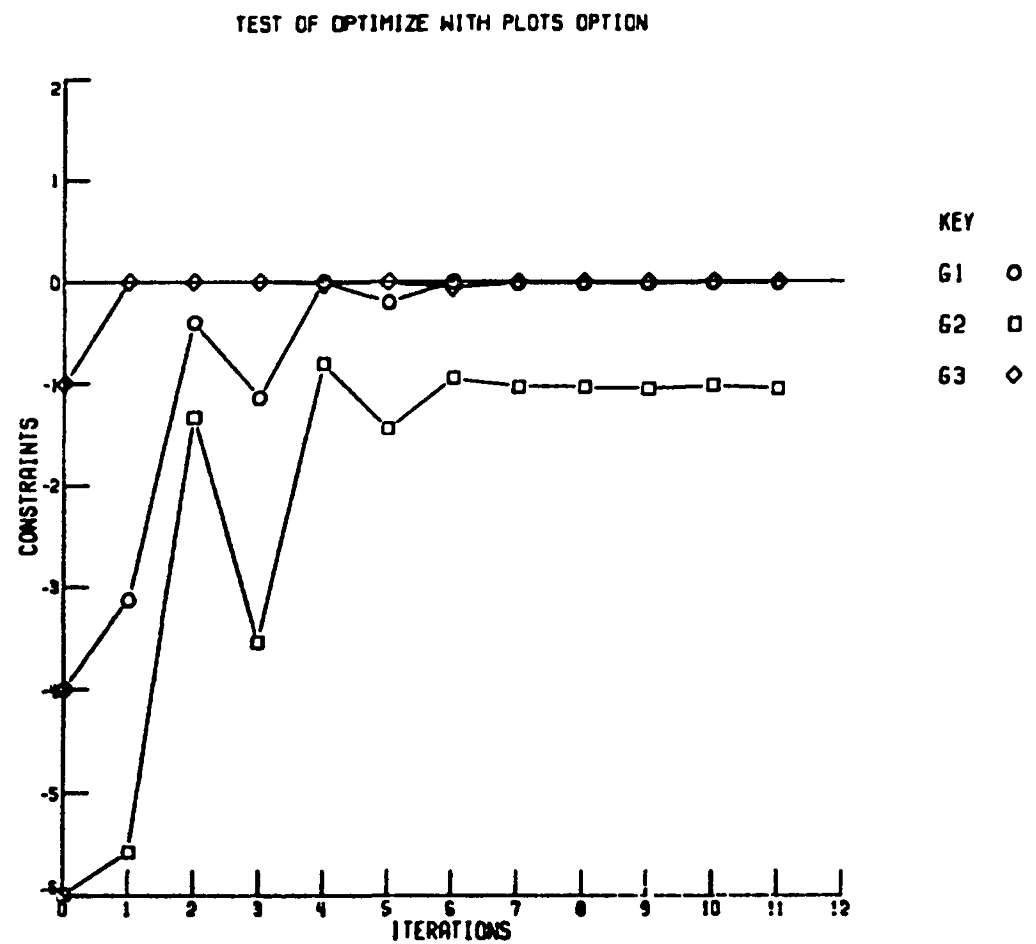
ITER	0	1	2	3	4	5	6
CONST							
G 1	-.40000000E+01	-.30984054E+01	-.40074673E+00	-.11242470E+01	.32216582E-02	-.19350256E+00	.82083798E-02
G 2	-.60000000E+01	-.55673679E+01	-.13504806E+01	-.35481116E+01	-.81943653E+00	-.14494319E+01	-.95686600E+00
G 3	-.10000000E+01	.64658979E-01	-.59993265E-02	.29176031E-01	-.15169804E-01	.17497368E-01	-.37435245E-01

ITER	7	8	9	10	11
CONST					
G 1	-.51735781E-02	-.79792078E-02	-.53121650E-02	-.52929435E-02	-.13894055E-01
G 2	-.10386557E+01	-.10440228E+01	-.10565985E+01	-.10372328E+01	-.10473814E+01
G 3	.23185608E-01	.19493876E-01	.22331508E-01	.22435165E-01	.34703007E-02


TEST OF OPTIMIZE WITH PLOTS OPTION
NORMALIZATION
FACTOR AND KEY
OBJ= 31.0000
OBJ FUNCTION
ITERATIONS
0
.2
.4
.6
.8
1.0
1.2
1.4
1.6
0
1
2
3
4
5
6
7
8
9
10
11
12



TEST OF OPTIMIZE WITH PLOTS OPTION
DESIGN VARIABLES
ITERATIONS
NORMALIZATION FACTOR AND KEY
X1= 1.0000
X2= 1.0000
X3= 1.0000
X4= 1.0000


TEST OF OPTIMIZE WITH PLOTS OPTION
CONSTRAINTS
2
1
0
-1
-2
-3
-4
-5
-6
ITERATIONS
0
1
2
3
4
5
6
7
8
9
10
11
12
KEY
G1
G2
G3

REFERENCES

1. Vanderplaats, Garret N.: CONMIN - A FORTRAN Manual for Constrained Function Minimization. User's Manual. NASA TM X-62,282, 1973.

BIBLIOGRAPHY

Control Data Corporation: FORTRAN Extended Version 4 Reference Manual (604 97800) NOS 1.3. September 1979.

NASA Langley Research Center: NOS User's Guide for LaRC Computer Complex, NOS 1.3. September 1979.

NASA Langley Research Center: XEDIT User's Guide. Level-A, Version 2.1.7, July 1978.

Sobieszczanski-Sobieski, J.; and Bhat, R. B.: Adaptable Structural Synthesis Coupled by a Computer Operating System. AIAA Paper No. 79-0723, April 1979.

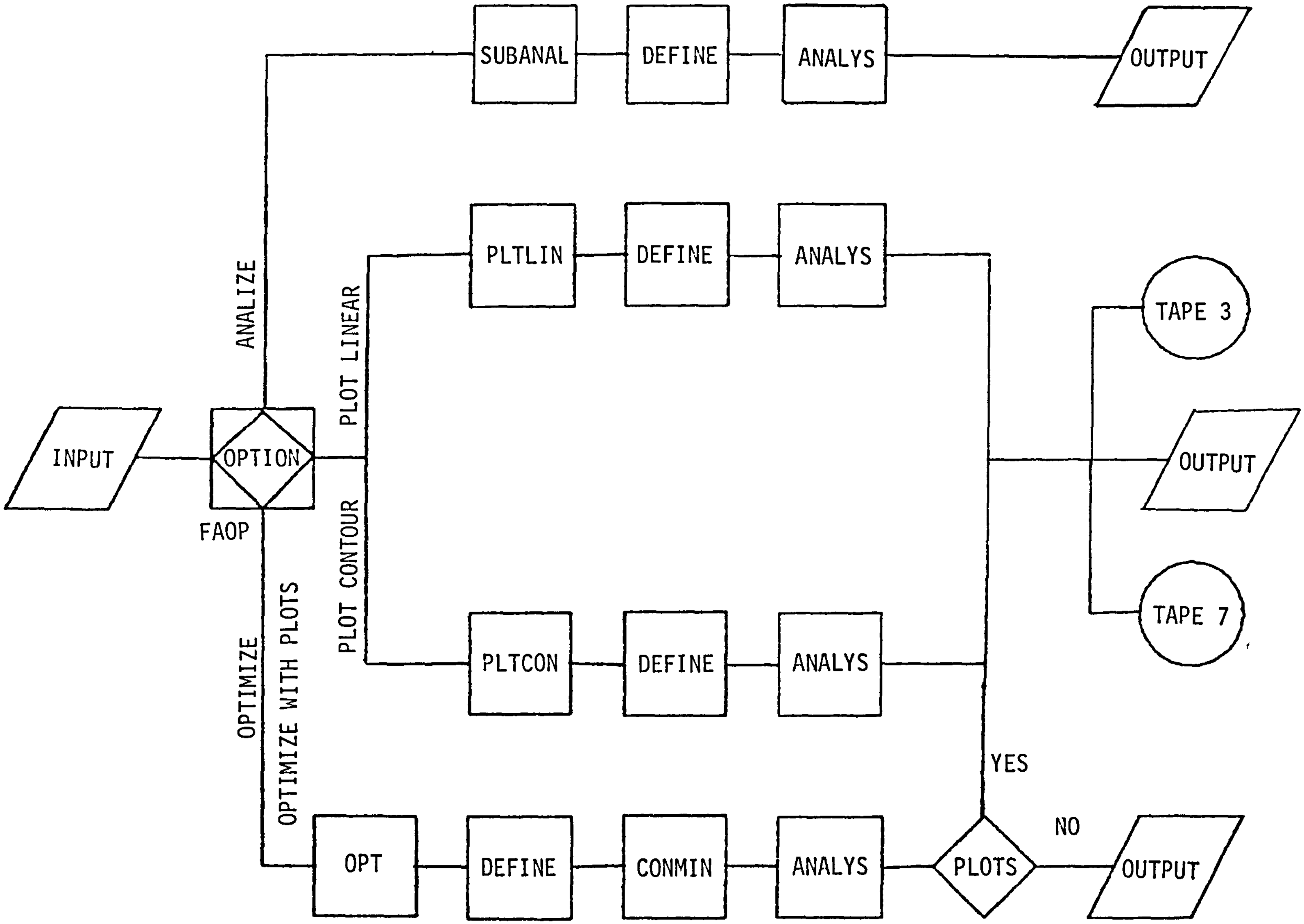


FIGURE 1: SUBROUTINE AND FILE FLOW OF PROGRAM ONE

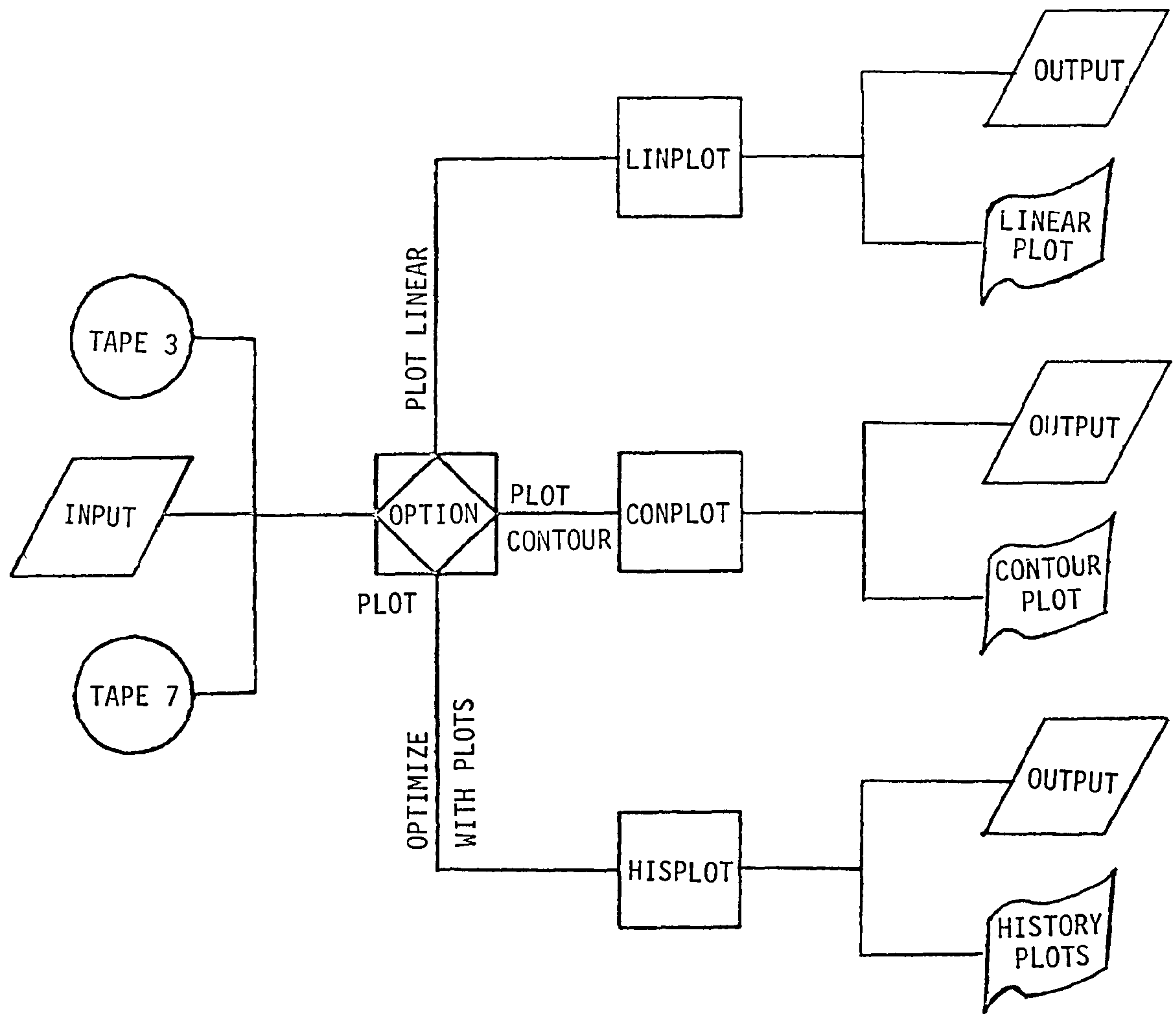


FIGURE 2: SUBROUTINE AND FILE FLOW OF PROGRAM TWO

1 Report No NASA CR-165653	2 Government Accession No	3 Recipient's Catalog No
4 Title and Subtitle FRANOPP - FRAMEWORK FOR ANALYSIS AND OPTIMIZATION PROBLEMS - USER'S GUIDE		5 Report Date January 1981 6 Performing Organization Code
7 Author(s) Kathleen M. Riley		8 Performing Organization Report No
9 Performing Organization Name and Address Kentron International, Inc. Hampton Technical Center an LTV Company Hampton, VA 23666		10 Work Unit No 505-33-63-02 11 Contract or Grant No NAS1-16000
12 Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546		13 Type of Report and Period Covered Contractor Report 14 Army Project No
15 Supplementary Notes Technical Monitor: J. Sobieski, NASA Langley Research Center, Mail Stop 243, Hampton, VA 23665		

16 Abstract

FRANOPP, FRamework for ANalysis and OPtimization Problems, is a software aid for the study and solution of design (optimization) problems. FRANOPP provides the driving program and plotting capability for a user generated programming system. In addition to FRANOPP, the programming system also contains the optimization code CONMIN (created and maintained by NASA Ames Research Center), and two user supplied codes, one for analysis and one for output.

FRANOPP provides the user with five options for studying a design problem. Three of the options utilize the plot capability and present an in-depth study of the design problem. The study can be focused on a history of the optimization process or on the interaction of variables within the design problem.

17 Key Words (Suggested by Author(s)) Programming system Analysis Optimization Plotting capability		18 Distribution Statement Unclassifed - Unlimited Subject Category 39	
19 Security Classif (of this report) Unclassified	20 Security Classif (of this page) Unclassified	21 No of Pages 73	22 Price* A04

*For sale by the National Technical Information Service, Springfield, Virginia 22161