57-39
190577

N94-27834

DESIGN OPTIMIZATION STUDIES USING COSMIC NASTRAN

S. M. Pitrof, G. Bharatram, V. B. Venkayya
Wright Laboratory
Wright-Patterson AFB OH 45433-7552

Summary

The purpose of this study is to create, test and document a procedure to integrate mathematical optimization algorithms with COSMIC NASTRAN. This procedure is very important to structural design engineers who wish to capitalize on optimization methods to ensure that their design is optimized for its intended application. The OPTNAST computer program was created to link NASTRAN and design optimization codes into one package. This implementation was tested using two truss structure models and optimizing their designs for minimum weight, subject to multiple loading conditions and displacement and stress constraints. However, the process is generalized so that an engineer could design other types of elements by adding to or modifying some parts of the code.

Introduction

Since the advent of NASTRAN during the early 70's, engineers have found many applications of finite element analysis in diverse fields. Its popularity, which is still growing, has spawned many commercial and research programs and they are available on just about every kind of computer available on the market. The parallel development of graphics interfaces, which started as pre- and post-processors to finite element programs, have further stimulated fascinating applications in the analysis of mechanical components, built-up structures, fluid-structure interaction problems, thermal and heat transfer analysis, acoustics and other engineering analyses. The reliability of finite element analysis is increasingly attributed to the graphical aids. They are the means for model error correction, display of analysis results such as displacements, mode shapes (including animation), color coded displays of stresses and strains, etc. With shrinking budgets and increasing competition for market share, the industry is groping for ways to cut product development costs and reduce development time from concept to market. Analysis tools such as NASTRAN offer challenging opportunities for rapid parametric studies at minimal cost. Adept use of these tools is the key to improving quality and reducing cost of new products. These two aspects are the most important ingredients for market leadership.

Having realized the many advantages of finite element analysis during the 70's, engineers have embarked upon the development of even more ambitious integrated design systems in the name of computer aided engineering (CAE). The basic elements of these multidisciplinary systems are finite element analysis and mathematical optimization (nonlinear programming) algorithms coupled by sensitivity analysis. The sensitivity analysis is an extension of finite element analysis through first order approximations. These integrated systems take full advantage of the ever improving capabilities of modern digital computers and provide significant reductions in product development costs and time. The objective of this paper is to show how COSMIC NASTRAN, which is basically an analysis tool, can be coupled to a nonlinear programming package to obtain an optimized structure. Although the single discipline analysis architecture of NASTRAN presents numerous difficulties, it is possible to achieve objectives of optimization to a limited extent. The bridge between the analysis and optimization is the sensitivity analysis and the procedure outlined in Reference 1 is used in this implementation.

The next section provides a brief introduction to optimization theory and sensitivity analysis, followed by some details of the implementation using COSMIC NASTRAN. This is followed by discussion of the results gained from this implementation as applied to simple truss problems.

Theory

The optimization problem is generally posed as follows:

Minimize an objective function:

$$F(\underset{\sim}{x}) = F(x_1, x_2, \dots x_a)$$

Subject to a set of constraints:

$$z_i(\underset{\sim}{x}) = z_i(x_1, x_2, \dots x_a) \le \bar{z}_i$$

$$z_j(\underset{\sim}{x}) = z_j(x_1, x_2, \dots x_a) = \bar{z}_j$$

$$\underset{\sim}{x}^l \le \underset{\sim}{x} \le \underset{\sim}{x}^u$$

F is the user defined objective function, while $\underset{\sim}{x}$ is the vector of design variables. The first set of constraints, z_i, is the inequality constraints. The second set z_j is the equality constraints. The third set is the constraints on the variables (upper and lower bounds) themselves. The weight of the structure is the objective function addressed in this paper while the constraints are on the displacements and stresses. The variables in the structural optimization problem described in this paper are the crosssectional areas of the rods, but could instead be thicknesses of the plates or some other design parameter.

The constraints are non-linear functions of the variables and thus the problem comes under the category of nonlinear programming. The iterative solution of the linear or nonlinear programming problems can be written as:

$$\underset{\sim}{x}^{v+1} = \underset{\sim}{x}^{v} + \tau \underset{\sim}{D}$$

where $\underset{\sim}{x}^{v}$ and $\underset{\sim}{x}^{v+1}$ are the variable vectors in two consecutive cycles, $\underset{\sim}{D}$ (∇F, ∇Z) is the travel direction or perturbation, and τ is the step size. The travel direction in most gradient-based solutions is based on the objective and constraint function gradients (∇F and ∇Z).

So basically, the steps involved in the solution of the nonlinear programming problem are as follows:

1. Initial solution $\underset{\sim}{x}$
2. Function evaluation
3. Selection of active constraints
4. Gradient evaluation
5. Determine the travel direction $\underset{\sim}{D}$
6. Determine the step size τ .
7. Check for the optimality conditions.
8. Repeat the steps until the conditions are satisfied.

Gradient computations are as outlined in Reference 1. CONMIN, a nonlinear programming package based on the modified method of feasible directions, is used as the optimizer (Reference 2).

Implementation

As previously discussed, there is potential for considerable benefit in performing structural design optimization studies using NASTRAN. However, integrating optimization algorithms with NASTRAN has been a daunting proposition. The effort required to develop a fully integrated structural design optimization package is so extensive that only through intensive, dedicated efforts such as the Air Force's Automated STRuctural

Optimization Program (ASTROS) program can finite element analysis codes and mathematical optimization algorithms be interfaced into a system capable of performing structural design. A true integrated package such as ASTROS consists of one executable program, with all capabilities built into it. Another approach, which we will discuss in this paper, is to synthesize separate executable files with a shell script program run by the computer's operating system. The script program calls multiple executable files and performs some rudimentary computations and data processing activities. In the past few years two phenomena have emerged to make our task of implementing optimization in NASTRAN far more realizable.

The first is the emergence of code written in subroutine form to compute values needed as inputs to optimization algorithms such as constraint values and constraint sensitivities. Optimization algorithms need to specify a design problem as an objective function to be maximized or minimized. As design variable values change, the objective function value changes. The algorithm also requires that bounds on the problem are placed. These bounds take form as constraint values and design variable upper and lower bounds. Much of the information required by optimization algorithms is very simple and straightforward to compute. Some values such as initial design variable values, design variable value upper and lower bounds, and constraint limits are left to the user to define. Other values such as objective function values and constraint values are fairly simple to compute but require information about the structure such as geometry and response to loading. Of significantly greater difficulty to compute are objective function and constraint sensitivities. Sensitivity values, which are defined as the first derivative of the objective and constraint functions, tell the optimizer which direction in design space to move. Recently, programs in subroutine form to compute such values have become more available (exemplified in Reference 1) to calculate constraint sensitivities for NASTRAN elements.

The second phenomenon is the emergence of open computer operating architectures. Cosmic NASTRAN has in the past been available on proprietary computer architectures such as CDC/CYBER and VAX/VMS. As Unix systems are becoming more available, NASTRAN is migrating to these new machines in order to take advantage of open systems. This environment is especially amenable to programmers who wish to integrate stand-alone programs into a package but either cannot or choose not to rewrite stand-alone programs in subroutine format and link operation by a main driver program. Since we have programs such as NASTRAN to perform structural analyses , programs such as CONMIN to perform optimization studies, and many miscellaneous programs to formulate input values required for optimization from output values from NASTRAN, Unix provides us with the necessary capability to synthesize these programs into one system capable of performing structural optimization tasks.

The OPTNAST computer program was created to demonstrate the feasibility of integrating NASTRAN with optimization methods in the context of structural design. OPTNAST, which capitalizes on previously written optimization code and the Unix operating system, consists of several fortran programs and a Unix shell script program. The Unix c-shell script was written to perform a loop operation between the analysis program (NASTRAN) and the optimizer (CONMIN). In order to use the script the user must obey some basic rules regarding his design problem. These rules are imposed on the user in order to simplify the code development process. The restrictions are as follows:

- No free format
- Only one material card
- All elements will be designed
- Constraints will be applied to all elements/nodes
- All load cases will be designed; limit of 5 load cases

With more extensive code development, any of these restrictions can be removed. However, our intent is to develop a reasonably practical methodology to conduct optimization with NASTRAN and thus some restrictions are acceptable.

There are two input files required by the OPTNAST program. They are a standard NASTRAN input file (e. g. tenbar.nid) and a file of optimization parameters (e. g. tenbar.opt). The input file must obey the previously discussed restrictions and must also include the following statements:

- Request for OUTPUT2 file with KELM matrix (for use in gradient computations)
- Request for punch file with displacement and/or stress data (for use in constraint calculations)

The optimization parameter file must contain the following:

- New CONMIN parameters to override defaults (if any are desired)
- Number of and values for displacement and stress constraints

Examples of each are contained in Appendices 1 and 2.

Once the user has properly prepared the NASTRAN input file and the optimization parameter file, the user is ready to run the OPTNAST program (Figure 1). The OPTNAST program consists of a Unix script (Appendix 3) file that calls the executable programs and processes the data shared by the executables. There are three executable files called by OPTNAST. The first is PREPARE, which preprocesses the bulk data file. The second is NASTRAN, and the third is COSOPT, which performs all of the optimization computations (Appendix 4). The OPTNAST script performs the following operations:

- Reads the name of the input file
- Processes the input file to include load cases to calculate virtual load vector response (for gradient calculations)
- Submits the problem to NASTRAN to calculate initial structural design response to the applied loads
- Sends the data to the COSOPT program to:

(1) calculate constraint and objective function and gradient values
(2) submit to CONMIN for optimization
(3) return a new NASTRAN input file if design has not converged or a converge flag if it has

- Loop back and submit new input file to NASTRAN to continue optimization task
- Continue looping until optimum is reached or maximum number of 16 iterations is reached

While the OPTNAST program is not an integrated package, but rather a collection of executables driven by a script file, it is fully capable of performing all tasks necessary to solve the optimization problem.

Results and Discussion

The OPTNAST program was used to perform design optimization studies on two structural models, each with varying constraint values and load cases. The first model, the Ten Bar Truss (Figure 1) was modeled with the properties as illustrated in the NASTRAN input file example (Appendix 1). This problem was solved with six different conditions, with minimization of structure weight being the objective in each case. The first case featured 2.0" displacement constraints applied to all grid points. The second case featured 25000 psi stress constraints (both tensile and compressive) on each element. The third case synthesized both the first two cases. The fourth case featured stress constraints with two separate load cases applied. The fifth case was identical to the second case except that no linear approximations were made during the redesign phase (NASTRAN was called to recalculate structural response after each iteration). The sixth case was again identical to the second, except that the initial design variable values are set to minimum gauge. This is what is described as an infeasible design because all constraints are violated.

The second structural model designed was a Two Hundred Bar Truss (Figure 3). The objective of this model is to provide an example of a large structure in order to indicate feasibility of designing a large model. This structure was solved with stress constraints applied to each element and with two separate load cases. Since there are two hundred elements and two load cases, this design model includes two hundred design variables and four hundred constraints.

Each of the previously described models was run with the OPTNAST program, and results are provided to compare with those provided by the ASTROS program. Since ASTROS input is generally compatible with NASTRAN and since ASTROS uses a similar optimization algorithm, approximation concepts and gradient calculations, results gained from each code should be comparable. This comparison is bourne out when viewing the final results tabulated in Table 1. The results show that for any design the final design's optimal weight for each method agree to within one percent. One obvious penalty is that the amount of time required is much less with an integrated package like ASTROS. Improvements to the

OPTNAST program can be made to improve efficiency, but an integrated package with a centralized database like ASTROS benefits from inherently more efficient methods of processing, storing and sharing data between modules. It should also be noted that the timing summary for the OPTNAST program is only an approximation since the code was not included to keep track of the actual time spent.

Concluding Remarks

This study has proved the feasibility of conducting optimization studies with NASTRAN. The OPTNAST program generated for this study can be used for designing truss structures with displacement and stress constraints. As many as five different load cases can be considered with the program. The program can achieve optimum designs very similar to integrated design optimization packages such as ASTROS, but a computational performance penalty is inherent and unavoidable. Still, this method is very attractive when integrated packages do not offer the necessary capabilities, such as element types or constraints that the user needs to design for. As a result, this is a viable alternative when the user has highly specialized design needs.

References

1. Tischler, V. A.; Venkayya, V. B.: Sensitivity Analysis and Optimization Issues in NASTRAN, 1991 NASTRAN Colloquium

2. Vanderplaats, Garret N.: CONMIN - A Fortran Program for Constrained Function Minimization User's Manual, NASA TMX-62282, Ames Research Center, August 1973

3. COSMIC NASTRAN User's Manual, NASA SP-222(08), June 1986

Tables

Table 1: Results

Case		OPTNAST			ASTROS		
Model Name	Constra-ints	Weight (lbs)	Iteration cycles	Clock (min)	Weight (lbs)	Iteration cycles	Clock (min)
10 Bar Truss	Disp	5024	12	12:00	5102	12	1:15
10 Bar Truss	Disp & Stress	5066	10	10:00	5104	12	1:41
10 Bar Truss	Stress Const	1594	14	14:00	1594	18	2:31
10 Bar Truss	2xLoads, Stress	1741	9	9:00	1738	15	1:26
10 Bar Truss	Stress, No Approx	1609	144	Hrs	-	-	-
10 Bar Truss	Stress, Infeas.	1594	13	13:00	1593	16	1:30
200 Bar Truss	2xLoads, Stress	98.62	12	5 Hrs	98.75	6	8:47

Figures

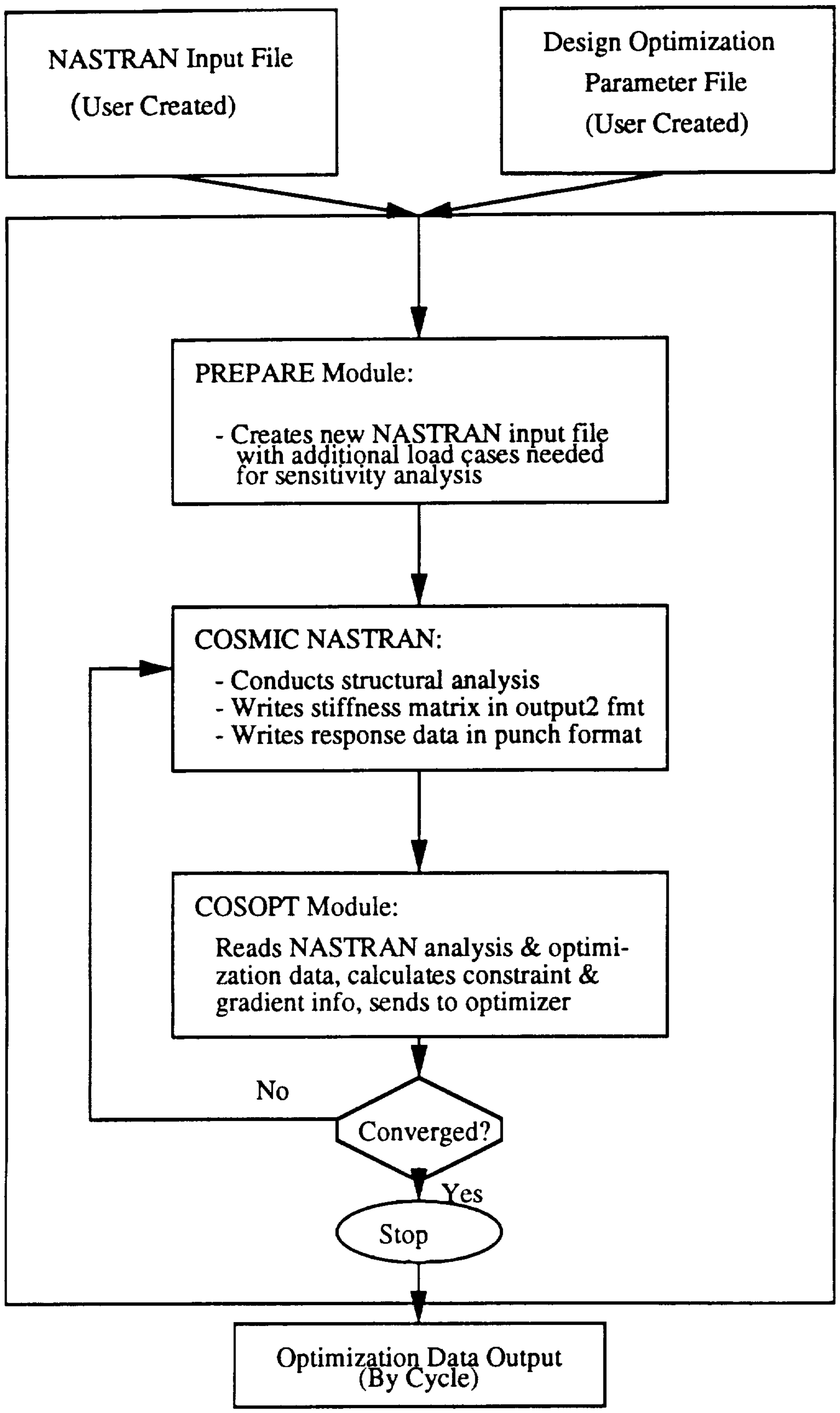
NASTRAN Input File
(User Created)
Design Optimization
Parameter File
(User Created)
PREPARE Module:
- Creates new NASTRAN input file with additional load cases needed for sensitivity analysis
COSMIC NASTRAN:
- Conducts structural analysis
- Writes stiffness matrix in output2 fmt
- Writes response data in punch format
COSOPT Module:
Reads NASTRAN analysis & optimization data, calculates constraint & gradient info, sends to optimizer
No
Converged?
Yes
Stop
Optimization Data Output
(By Cycle)

Figure 1: OPTNAST Program

apshot

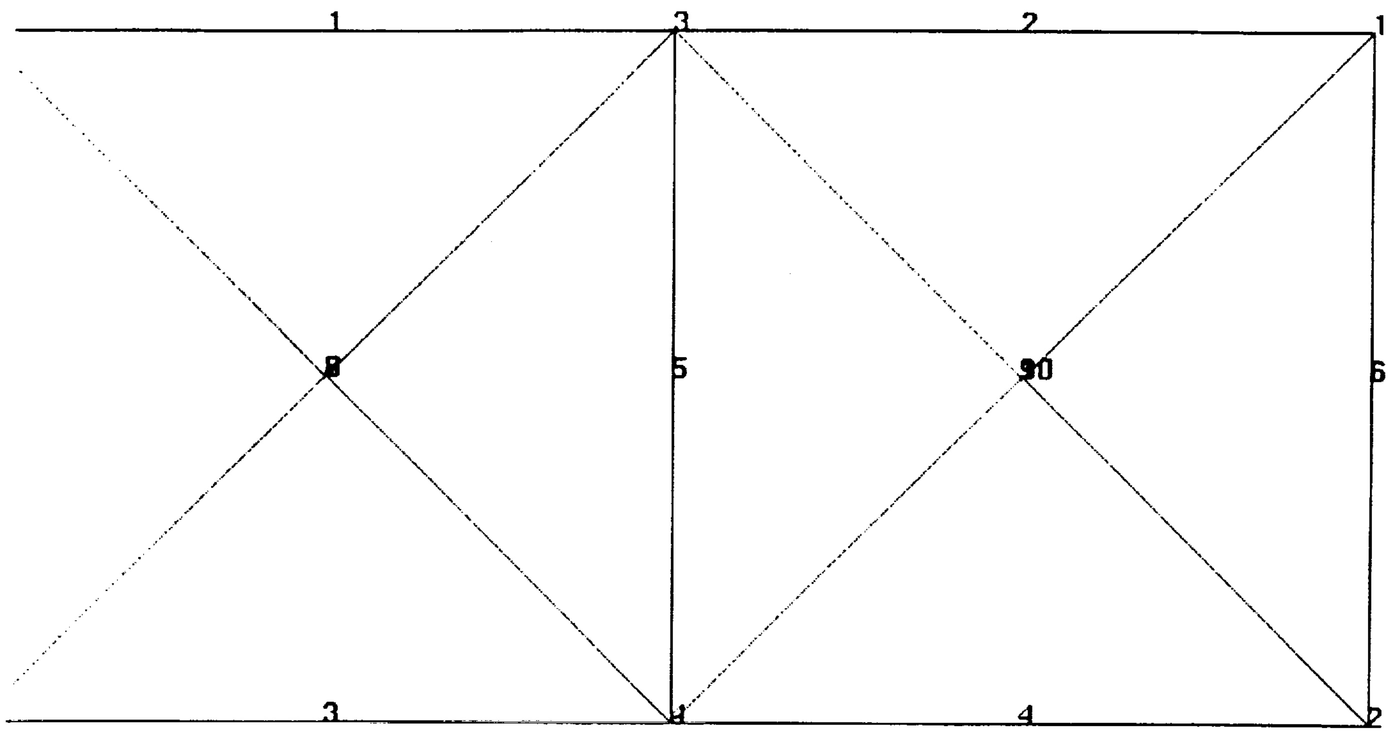

Figure 2: Ten Bar Truss

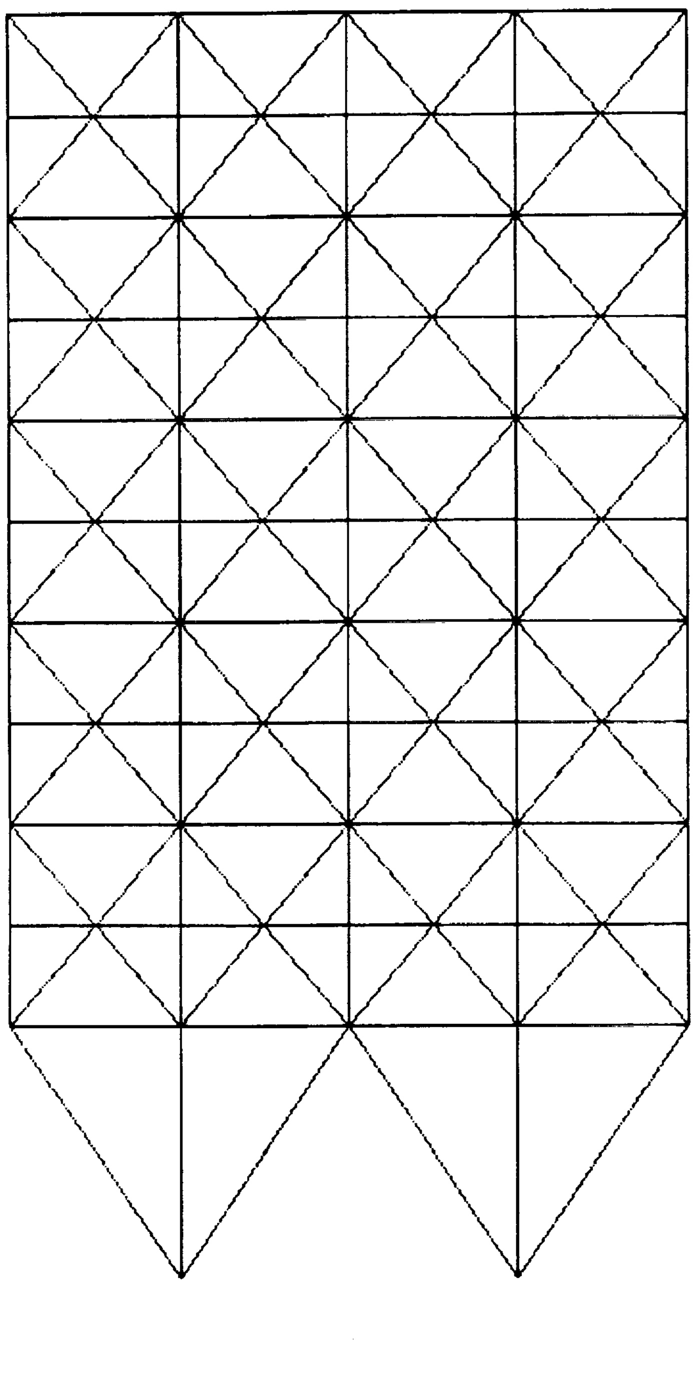

Figure 3: Two Hundred Bar Truss

```
ID TENB,TENB
SOL 1,0
TIME 50
ALTER 37 $
OUTPUT2 KELM//-1/15/ V,N,Z $
OUTPUT2,,,,//-9/15 $
ENDALTER $
CEND
TITLE = TEN BAR TRUSS
DISP(PRINT,PUNCH)=ALL
STRESS(PRINT,PUNCH)=ALL
SPC = 1
SUBCASE 1
LOAD = 1
BEGIN BULK
$
$             TEN BAR TRUSS MODEL
$             FROM SCHMIT, L.A., JR. AND MIURA, H., " APPROXIMATION
$                     CONCEPTS FOR EFFICIENT STRUCTURAL SYNTHESIS ",
$                     NASA CR-2552, MARCH 1976.
$
GRID      1               720.0   360.0    0.0
GRID      2               720.0    0.0     0.0
GRID      3               360.0   360.0    0.0
GRID      4               360.0    0.0     0.0
GRID      5                 0.0   360.0    0.0
GRID      6                 0.0    0.0     0.0
CROD      1       1       3       5
CROD      2       2_      1       3
CROD      3       3       4       6
CROD      4       4       2       4
CROD      5       5       3       4
CROD      6       6       1       2
CROD      7       7       4       5
CROD      8       8       3       6
CROD      9       9       2       3
CROD     10      10       1       4
PROD     1        2      30.0
PROD     2        2      30.0
PROD     3        2      30.0
PROD     4        2      30.0
PROD     5        2      30.0
PROD     6        2      30.0
PROD     7        2      30.0
PROD     8        2      30.0
PROD     9        2      30.0
PROD     10       2      30.0
$
MAT1      2       1.E+7           0.3     0.1       25000.0
$
SPC1,     1,    123456,   5,    6
SPC1,     1,    3456,     1,    THRU,   4
$
FORCE,    1,    2,   ,   -1.E5,   0.0,    1.0,    0.0
FORCE,    1,    4,   ,   -1.E5,   0.0,    1.0,    0.0
$
ENDDATA
```

Appendix 1: NASTRAN Input File

```
$ AN EXAMPLE PROBLEM
INDMIN=0
0.10
$ PRINT CONTROL
 IPRCTL=3
$ DISPLACEMENT CONSTRAINT
LMTDSP=2
6,-2.0  2  1
  -2.0  2  2
  -2.0  2  3
  -2.0  2  4
  -2.0  2  5
  -2.0  2  6
 NZLMIT=4
 IPRINT=1
 FXMIN=1.0E+10
 ITERT=40
 XMIN=.1
 XMAX=1000.0
```

Appendix 2: Optimization Parameter Input File

```
# Unix c-shell script to optimize rod structure for displacement
# and stress constraints using NASTRAN to derive structural response
# quantities (displacements, stresses, K matrix), CONMIN optimization
# algorithm for optimization and assorted routines to calculate
# objective function, constraint values and sensitivities (sensitivity
# analysis uses virtual load vector method
#
# Inputs to program are NASTRAN input deck (no free format) <filename.nid> and
# optimization parameter file <filename.opt>
#
# get model name if not provided
if ($1 == "") then
echo 'model name?'
set a = $<
else
set a = $1
endif
# check to see if optimization parameter file exists
if (! -e $a.opt) then
echo "RUN REQUIRES OPTIMIZATION PARAMETERS ($a.opt)
exit
endif
echo "1.0" >fort.85              #initialize last obj fn value to 1.0
cp $a.nid $a.nid.old             #save old input
cp $a.opt fort.4                 #get optimization date
cp $a.nid fort.55                #copy input to unit 55
prepare <$a.nid >$a.out          #add virtual load vectors to NASTRAN input
rm $a.out $a.nid
mv fort.65 $a.nid
# build script to execute cosmic
echo "c" >cosfeed
echo $a >>cosfeed
echo "o" >>cosfeed
echo "i" >>cosfeed
echo "y" >>cosfeed
set it = 0
#begin loop
while ($it < 16)                 #maximum 16 iterations
cp $a.nid fort.55
@ it = $it + 1                   #counter
cosmic <cosfeed >$a.out          #execute cosmic interactively
#prepare for optimization segment
#cp $a.nid fort.55               #copy input file to unit 55
cp $a/PCH fort.25                #punch file to unit 25
cp $a/INP1 fort.15               #output2 file to unit 15
rm -rf $a
cosopt <fort.55 >$a.opt.it$it   #submit to optimization program
if ( -e fort.65 ) mv fort.65 $a.nid
set loop = `cat fort.75`
if ( $loop == "0" ) set it="16" #if optimization converged end loop
end
rm cosfeed fort.15 fort.25 fort.55 fort.75 fort.4 fort.85
```

Appendix 3: OPTNAST Unix Shell Script

```
      PROGRAM COSOPT
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C-----------------------------------------------------------------------
C     Program to submit NASTRAN output to CONMIN optimization algorithn
C     for rod structures with displacement and stress constraints
C-----------------------------------------------------------------------
      EXTERNAL SETFUN
C-----------------------------------------------------------------------
      INCLUDE 'cosopt.inc'
C-----------------------------------------------------------------------
      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
C-----------------------------------------------------------------------
      SAVE/FUNPAR/
      COMMON/FUNPAR/FXMIN,XL,XU,NZLMIT,ITERT,IPRINT1
C-----------------------------------------------------------------------
C  Thickness (of membrane elements or area of bars--input)
C
      DIMENSION TH( MAXELM )
      SAVE   /ANLYZ1/
      COMMON /ANLYZ1/ TH, MEMBS, JOINTS, MM, NFI
C-----------------------------------------------------------------------
C
C  Index to elements' material properties
C
      INTEGER MYOUNG( MAXELM )
C
C  Material properties
C
      DIMENSION YOUNGM( MAXMTL ), POISON( MAXMTL ), RHO1( MAXMTL )
C
C  Allowable Stresses
C
      DIMENSION ALSTRS( 3, MAXMTL )
C
      SAVE   /ANLYZ2/
      COMMON /ANLYZ2/ EEE, PMU, RHO, YOUNGM, POISON, RHO1, MYOUNG,
     +                ALSTRS, NMAT, MSSTRS
C-----------------------------------------------------------------------
      INTEGER NNODES( MAXELM )
C
C  Node number connectivities for each element
C
      INTEGER MA( MAXELM ), MB( MAXELM ), MC( MAXELM ), MD( MAXELM )
C
C  Nodal coordinates for each joint
C
      DIMENSION X( MAXJNT ), Y( MAXJNT ), Z( MAXJNT )
C
      SAVE   /ANLYZ3/
      COMMON /ANLYZ3/ NNODES, MA, MB, MC, MD, X, Y, Z, INCHES
C-----------------------------------------------------------------------
C
C  Degree of freedom numbers for restrained nodes (boundary conditions)
C
      DIMENSION IBND( MAXBND )
C
C  Number of load components for each loading condition
C
```

```
      DIMENSION NJLODS( MAXLOD )
C
C  Displacement and force resultants for each degree of freedom and
C   and loading condition
C
      DIMENSION DR( NNMAX,MAXLOD ), FR( NNMAX,MAXLOD )
C
C  Stiffness and mass matrices
C
      DIMENSION SK( MAXSK ), GM( MAXSK )
C
C  Pointers to diagonal elements in stiffness matrix, SK;
C  Row number for first nonzero element in each Column of SK
C
      DIMENSION IDIAG( NNMAX ), ICOL( NNMAX )
C
      SAVE   /ANLYZ4/
      COMMON /ANLYZ4/ IBND, NJLODS, FR, DR, SK, IDIAG, ICOL, GM,
     +                NBNDRY, NN, KIPS, NR, NONZRO
C-----------------------------------------------------------------------
      SAVE   /ANLYZ5/
      COMMON /ANLYZ5/ LOADS
C-----------------------------------------------------------------------
C
C  Element Area (size) Minimum and Maximums,
C  Variable Bounds factor limits
C
      DIMENSION AEMIN( MAXMEM ),  AEMAX( MAXMEM )
      DOUBLE PRECISION VBMIN, VBMAX
      LOGICAL INDMIN, INDMAX
C
C  Key to Limited Displacements;
C  Number of Displacement Constraints;
C  Deflection constraints:  maximum deflection for all nodes or
C   magnitude, direction, and node number for each node's constraint
C
      INTEGER LMTDSP, NDSPCN
      DIMENSION DEFMAX( 3 ),
     +          DEFMAG( MAXDEF ), IDRDEF( MAXDEF ), NNDDEF( MAXDEF )
C
C  Frequency limits (negative for lower bound);
C  Number of Frequencies Constrained, Mode number of Constrained freq.
C
      DIMENSION FRQLMT( MAXFQL )
      INTEGER NFRQCN, MODECN( MAXFQL )
C
C  Flag for Rayleight Quotient Frequency Constraint Approximation;
C  Flag for inverting form of Frequency constraint.
C
      LOGICAL FRQAPX, FRQINV
C
C  Structural to total mass modal energy ratios
C
      DIMENSION GAMMAJ( MAXFQL )
      SAVE /OPTIM2/
      COMMON /OPTIM2/ FRQLMT, GAMMAJ, DEFMAX, DEFMAG, IDRDEF, NNDDEF,
     +                AEMIN, AEMAX, VBMIN, VBMAX, INDMIN, INDMAX,
     +                LMTDSP, NDSPCN, NFRQCN, FRQAPX, FRQINV, MODECN,
     +                LMTSTR, NSTRCN, NDUMMY
C-----------------------------------------------------------------------
```

```
C
C  Von Mises Effective Stress Ratio for each element
C
      DIMENSION VMEFSR( MAXCON, MAXLC )
C
C  Strain energies for each element & axial stress values
C
      DIMENSION ENRG( MAXCON+1, MAXLC ), SX(MAXCON)
C
      COMMON /OPTIM3/ VMEFSR, ENRG, SX
      SAVE   /OPTIM3/
C-----------------------------------------------------------------------
C
C Allowable stress values
      DIMENSION ALS(3)
      SAVE   /OPTIM12/
      COMMON /OPTIM12/ ALS
C-----------------------------------------------------------------------
      DIMENSION A(N1,N3),A1(N6,N7),AS(N1,N3),AD(N1,N3),XOBJ(N1),VLB(N1),
     +VUB(N1),G(N2),SCAL(N1),S(N1),G1(N2),G2(N2),B(N3,N3),C(N4), DF(N1),
     +ISC(N2),IC(N3),MS1(N5), ITYPG(N2),IHAC(MAXCON+3),KMAT(K1,K1)
      REAL OBJOLD
C Override selected CONMIN default parameters
      DELFUN = 0.0001
      DABFUN = 0.01
      CTMIN = .0005
      CTLMIN = .001
      CT = -.003
      CTL = -.01
      ITRM = 3
      NFDG = 1
      NSCAL = 0
      LINOBJ = 1
      ITMAX = 75
      NSIDE = 20
      IGOTO = 0
      INDEX=6
      MM=3
C Read NASTRAN data deck to get structural data
      CALL INPUT(SETFUN, NDV, NCON)
C Calculate initial design variable and objective function values
      CALL INIDV(XOBJ, DF)
      IF (NDSPCN .GT. 0 ) NFI=NFI+JOINTS
      IF (NSTRCN .GT. 0 ) NFI=NFI+MEMBS
      DO I = 1,NCON
        ISC(I) = 0
      ENDDO
      DO I = 1,NDV
        VLB(I)=XL
        VUB(I)=XU
      ENDDO
      WRITE(75,*)1
C Calculate objective function value
      CALL CALOBJ(OBJ,DF,XOBJ,NDV,.FALSE.)
C Calculate constraint values
      CALL CALCON(XOBJ,G,ITYPG)
      NAC = 0
      SF=1.0
      PRINT*,'Constraint values'
      DO J=1,NCON
```

```
        PRINT*,'g(j)=',G(J)
        ic(j)=0
        IF (G(J) .GE. CT) THEN
          NAC = NAC + 1
          ic(nac)=j
        ENDIF
      ENDDO
      PRINT*,'Number of active constraints:',NAC
C CALCULATE CONSTRAINT GRADIENTS
      CALL VICKY1(KMAT)
      IF (NDSPCN .GT. 0) THEN
        CALL VICKY2(KMAT,INDEX,AD)
        DO I=1,N7
          DO J=1,N6
            A(J,I)=AD(J,I)
          ENDDO
        ENDDO
        IF (NSTRCN .GT. 0) THEN
          CALL VICKY4(KMAT,INDEX,AS)
          DO K=1,NSTRCN
            DO J=1,N6
              A(J,K+NDSPCN)=AS(J,K)
              ENDDO
          ENDDO
        ENDIF
      ELSE IF (NSTRCN .GT. 0) THEN
      CALL VICKY4(KMAT,INDEX,AS)
      DO I=1,N7
        DO J=1,N6
          A(J,I)=AS(J,I)
        ENDDO
      ENDDO
      ELSE
      PRINT*,'ERROR - NO CONSTRAINTS IDENTIFIED'
      STOP
      ENDIF
      PRINT*,'Constraint Gradients'
      do i=1,n7
        do j=1,n6
        WRITE(6,70)(a(j,i))
        enddo
      enddo
70    FORMAT(6E15.6)
      CALL APXCMN(XOBJ, VLB, VUB, G, A, NDV, NCON, OBJ, DF, IHAC,
     +            RTCNV, INVFLG, MAXCON, MAXNDV, IACT, IVIOL, ITYPG,NVC)
      IF (NVC .EQ. 0) THEN
        READ(85,*)OBJOLD
        IF (ABS((OBJOLD-OBJ)/OBJOLD) .LE. 0.001) THEN
          REWIND(75)
          WRITE(75,*)0
          PRINT*,'COSOPT HAS CONVERGED'
          ENDIF
      ENDIF
      REWIND(85)
      WRITE(85,*)OBJ
      PRINT*,'XOBJ=',(XOBJ(I),I=1,NDV)
      CALL UPDATE(XOBJ,MAXNDV)
      WRITE(6,187)OBJ
  187 FORMAT(5X,21HOBJECTIVE FUNCTION = ,E15.8)
      STOP
      END
```