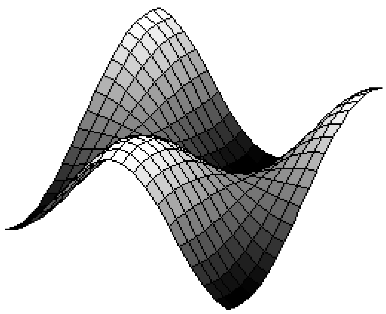
BIGDOT

VERY LARGE SCALE DESIGN OPTIMIZATION TOOL



BIGDOT

VERY LARGE SCALE DESIGN OPTIMIZATION TOOL

USERS MANUAL

Version 4.X

Vanderplaats Research & Development, Inc.

1767 S. 8th Street, Suite 200
Colorado Springs, CO 80905
http://www.vrand.com

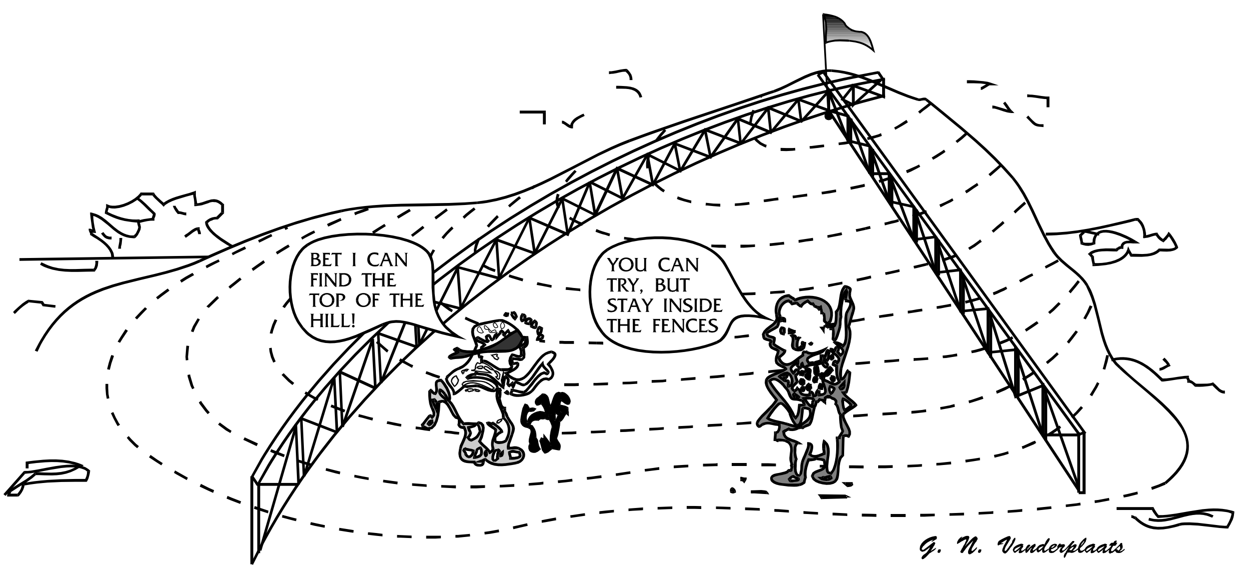
Phone (719) 473-4611   FAX (719) 473-4638

OTHER VR&D SOFTWARE PRODUCTS

DOT (***D***esign ***O***ptimization ***T***ools) is our basic optimization program for general purpose applications. DOT is the standard optimizer for numerous commercial products in addition to our own.

VisualDOC is a main ***D***esign ***O***ptimization ***C***ontrol program which greatly simplifies coupling your analysis with optimization. The design problem is defined in the windows environment for convenient pre- and post-processing. VisualDOC provides additional features such as multi-objective and discrete variable optimization, approximations based on curve fits (response surface approximations), design of experiments and much more.

GENESIS is a fully integrated finite element analysis and optimization program for member sizing, shape and topology optimization. GENESIS provides state of the art linear elastic analysis for statics, normal modes, heat transfer and dynamic response. It uses the latest approximation techniques to produce an optimum design quickly and easily. GENESIS was written from the beginning to perform optimization. It is not just another analysis program with optimization added.



CONSTRAINED OPTIMIZATION

Contents

CHAPTER 1

Overview

- Introduction
- The General Optimization Problem
- What You Will Find in this Manual
- BIGDOT System Requirements
- Installing BIGDOT on Your Computer
- Documentation and the Adobe ACROBAT Reader

1.1 Introduction

Welcome to VR&D's Design Optimization Tool, BIGDOT. The BIGDOT program is intended to solve very large, nonlinear, constrained problems where gradient information is available and function and gradient evaluation is efficient. The specific goal in writing BIGDOT is to solve very large optimization problems using limited central memory.

This manual is intended for developers who are familiar with optimization. Normally, the user is expected to be proficient using the DOT [2] optimizer from VR&D. While there are differences in the parameters and organization of the control arrays RPRM and IPRM, those which are the same as used in the DOT program have the same meaning in almost all cases (IPRINT is different because we do not want to print thousands of variable and constraint values except for debugging purposes). In BIGDOT, we use parameter arrays names RPRMBD and IPRMBD to distinguish them from DOT.

The program ALLDOT may be used to combine DOT and BIGDOT into a single optimization system. In this case modified RPRM and IPRM arrays are used and data is internally transferred to RPRMBD and IPRMBD when BIGDOT is called.

BIGDOT is capable of solving continuous, discrete, integer or mixed continuous/discrete/integer variable problems. For discrete variable problems, there is no guarantee of obtaining the true minimum. The techniques used in BIGDOT are intended to efficiently obtain a near optimum discrete solution very efficiently.

1.1.1 BIGDOT Compared to DOT

The so-called “modern” optimization methods used by the DOT optimizer are quite efficient, but have two key drawbacks as the size of the optimization problem grows.

1. A sub-optimization task is performed to calculate the search direction at each iteration. This sub-task can become quite time consuming.
2. It is necessary to store all active and violated constraint gradients in the optimizer. This requires extensive memory. While it may be possible to use out of core operations to deal with this, such techniques are complicated and expensive.

BIGDOT overcomes both of these limitations by using a modern exterior point method. Depending on the options used, the optimizer itself requires almost no computational time and it is not necessary to store a large number of gradients in memory. BIGDOT allows the user to provide gradients via a hard drive file in compacted form to even further reduce memory requirements.

Version 4 of BIGDOT provides two new methods for calculating the individual penalty parameters on the constraints. These methods usually produce a marginally better optimum but at some computational cost. If there are fewer than about 1,000 active/violated constraints, the new methods are efficient. However, when the number of active/violated constraints grows beyond 1000 (10,000, depending on overall run times), the computational cost of BIGDOT itself may grow unacceptably. In this case, the original calculation method may be preferred. The parameter PENLTD contained in the IPRMBD array controls both the method of calculating the penalty parameters and the desired precision of the optimum.

Note that the reduction in efficiency is based on the number of active/violated constraints and not on the number of design variables. For problems of 1,000,000 variables where there are only a few hundred active/violated constraints, there should be little loss in efficiency but an increase in precision of the optimum.

1.2 The General Optimization Problem

BIGDOT solves the optimization problem:

Minimize or Maximize

$$F(\mathbf{X}) \qquad \text{Objective Function} \qquad (1\text{-}1)$$

Subject to;

$$g_j(\mathbf{X}) \le 0 \qquad j = 1,\text{NCON} \qquad \text{Inequality Constraints} \qquad (1\text{-}2)$$

$$X_i^L \le X_i \le X_i^U \qquad i = 1,\text{NDV} \qquad \text{Side Constraints} \qquad (1\text{-}3)$$

$$X_i \in S_k \qquad (1\text{-}4)$$

NDV is the number of design (decision) variables. BIGDOT is designed to be a robust numerical optimizer for problems in excess of 1,000,000 design variables. However, there is nothing magic about this number and much larger problems are have been solved. As of the writing of this manual, a structural optimization problem to minimize mass subject to frequency constraints has been solved using 190,000 design variables and a topology optimization problem has been solved in excess of 2,500,000 variables.

NCON is the number of constraints in a particular problem. The number of constraints tends to get high for many problems. For example, consider an aircraft wing design problem where each finite element has stress limits, and displacement constraints are imposed at each joint. Also, the wing must support many independent load cases. For a large finite element model, there would be a great many constraints. Nevertheless, there is no maximum number of constraints to keep in mind. Also, there is no minimum number of constraints. Thus, NCON=0 is allowed.

X_i^L and X_i^U are called side constraints. These are lower and upper bounds on the design variables. A common use of lower bounds is to prevent the design variables from going below zero. For example, it would make no sense to design a wing panel that has a negative thickness.

X_i may be continuous, integer or discrete, where discrete values are contained in a user supplied set. If X_i is to be chosen from a discrete set, this set of values is provided as given in Eq. (1-4).

If equality constraints are to be included, this may be done by simply creating two equal and opposite inequality constraints. BIGDOT solves equality constrained problems with no loss in efficiency.

It is necessary to formulate optimization problems in this standard form.

For detailed study of optimization methods and applications, the textbook, Multidiscipline Design Optimization, by Dr. Vanderplaats is available directly from VR&D. Please contact VR&D for details and pricing (web page www.vrand.com).

1.3 What You Will Find in this Manual

This chapter first defines system requirements. Chapter 2 describes interfacing BIGDOT with user-supplied application programs. Chapter 3 discusses advanced uses of BIGDOT such as over-riding internal parameters, supplying gradients of the objective function and constraints, interrupting and restarting BIGDOT, and writing output to a special file for later use. Chapter 4 presents examples to demonstrate the efficiency to be expected from BIGDOT. Chapter 5 is a list of references which may be useful to those seeking a better understanding of numerical optimization.

Appendix A gives a main calling program that may be used as a prototype for using BIGDOT. Appendix B defines the BIGDOT storage requirements and discusses Subroutine BDT507, which calculates the minimum required working storage values of NRWK and NRIWK.

1.4 BIGDOT System Requirements

Versions of BIGDOT are available for all levels of computers. The actual storage needed for BIGDOT is quite small, less than one megabyte. Execution requirements will depend on the program that BIGDOT is linked with and the number of variables and constraints.

BIGDOT is provided in double precision format. If a single precision version is needed, please contact VR&D.

1.5 Installing BIGDOT on Your Computer

BIGDOT is normally provided on a CD-ROM which also contains our VisualDOC and DOT software. Specific installation instructions are provided on an installation sheet provided with the CD-ROM. Alternatively, BIGDOT may be downloaded directly from the VF&&D web site.

Installation will include procedures for obtaining a licensing file, normally via email.

BIGDOT is provided in object code form for a variety of machines and compilers. If the specific machine or compiler you are using is not included, please contact VR&D for assistance. Also, if DLL or API formats are desired, contact VR&D.

1.6 Documentation and the Adobe ACROBAT Reader

The CD-ROM includes this manual on-line in the Adobe PDF format. This may be read using the Adobe Acrobat Reader, which is freely available from Adobe.

CHAPTER 2

BIGDOT with Application Programs

2.1 Introduction

Interfacing BIGDOT with your program is simple, as explained in this chapter. A part of the application program where all of the parameters and responses are available (usually in the main program) is modified to call BIGDOT in the manner described below.

A simple main program that calls BIGDOT is provided in Appendix A. All that needs to be provided are the parameters and function values. The parameters that must be provided are defined in the following two sections. An example is presented in section 2.7.

2.2 Methods Used by BIGDOT

BIGDOT solves unconstrained and constrained optimization problems. Also, for unconstrained problems, side constraints (lower and upper bounds on the design variables) are allowed.

BIGDOT uses a Sequential Unconstrained Minimization Technique (SUMT) to solve the constrained optimization problem as a sequence of unconstrained optimization sub-problems. To minimize storage requirements, the Fletcher-Reeves algorithm is used for the unconstrained sub-problem.

If discrete variable optimization is to be performed, BIGDOT first solves the continuous variable problem. Then, beginning with this solution, optimization is continued with the following set of additional constraints;

$$P(X) = R\sum_{i=1}^{ND} 0.5\left\{1 - \sin 2\pi\left[\frac{X_i - 0.25(X_i^- + 3X_i^+)}{X_i^- - X_i^+}\right]\right\} \qquad (2\text{-}1)$$

where X_i^- is the next lower discrete value and X_i^+ is the next higher discrete value and ND is the number of discrete and/or integer variables. The addition of these constraints to the original set attempts to drive the design to a discrete solution with minimum increase in the objective, while still satisfying the original constraints.

The proprietary Sequential Unconstrained Minimization (SUMT) algorithm used here requires gradients of only a critical subset of the constraints. Furthermore storage requirements are quite low, due of the internal algorithms used.

The first parameter in the IPRMBD array is NGMAX. BIGDOT requires the gradient of the objective and active/violated constraints. The number of constraint gradients that BIGDOT needs at any point is given by NGT, contained in IPRMBD(20) when BIGDOT returns to the user with INFO=2. Two options are available for providing gradients.

The first option is to store gradients directly in the **WK** array. The constraint gradients may be stored in an array of dimension NDV X NGMAX. Since NGT is not known in advance, NGMAX must be large enough to store whatever number of constraints become active/violated. For very large problems, this creates a massive storage requirement. Therefore, BIGDOT has a feature to request only subsets of constraint gradients. Indeed, NGMAX=1 is allowed, as the minimum possible required storage. However, while this is possible, it is not desirable because BIGDOT will return the user NGT times with INFO=2, [and with IPRMBD(20)=1] to get the needed gradients. Depending on how gradients are calculated, this may be quite inefficient. In general, if the working array storage parameter, NRWK should be as large as possible, within your computer memory limits, to allow BIGDOT to utilize the available storage.

In addition to the original method, BIGDOT Version 4 uses two new methods for calculating the penalty multipliers. If JPENLT=21, 22 or 23, (see JPENLT in Chapter 3), the multipliers will be calculated by solving a set of simultaneous equations until NGT exceeds NGMAX. Beyond that, the method will switch to solving these equations as an optimization sub-problem. This is because method 2X solves the set of simultaneous equations in central memory. If there is not enough memory, the equations will be solved as a sub-optimization problem requiring very little memory.

The second option is to store gradients in an unformatted file in compact form. In this case the gradient of the objective and the NGT active/violated constraints are stored in the file.

2.3 Calling Statement

BIGDOT is invoked by the following FORTRAN calling statement in the user's program:

CALL BIGDOT (INFO, IPRINT, NDV, NCON, **X**, **XL**, **XU**, OBJ, MINMAX,

* **G**, RPRMBD, IPRMBD, **WK**, NRWK, **IWK**, NRIWK,**IDISCR**,**DISCRT**)

All information needed by BIGDOT is passed via the parameter list, except gradients that may be pass in a file. Also, when BIGDOT requires the values of the objective function and constraints, or their gradients, it returns to the calling program instead of calling a user-supplied subroutine. This gives the user considerable flexibility in using BIGDOT, allowing for restarting the optimization process or for calling BIGDOT from the user’s analysis subroutine(s) to perform sub-optimization tasks.

If you wish to call BIGDOT from a C/C++ or other program, please contact VR&D for assistance.

2.4 Parameters in the Calling Statement

Table 2-1 lists the parameters in the calling statement to BIGDOT. Where arrays are defined, the required dimension size is given as the array argument. These are minimum dimensions. The arrays can be dimensioned larger than this to allow for program expansion.

Table 2-1 Parameters in the BIGDOT Argument List

PARAMETER	DEFINITION
INFO	Information parameter. Before calling BIGDOT the first time, set INFO=0. When control returns from BIGDOT to the calling program, INFO will normally have a value of 0, 1or 2. If INFO= 0, the optimization is complete (or terminated with an error message). If INFO= 1, the user must evaluate the objective, OBJ, and constraint functions, G(j), j=1,NCON, and call BIGDOT again. If INFO= 2, the user must provide gradient information. This is described in Chapter 3. **NOTE:** If IPRMBD(18)>0 on return from BIGDOT, a Fatal Error has occurred (See Chapter 3).
IPRINT	Print control parameter. IPRINT = 0 no output. IPRINT = 1 internal parameters, initial information and results. IPRINT = 2 same plus objective function and miscellaneous information at each optimization cycle. IPRINT = 3 same plus initial and final X-vector and G-vector. IPRINT = 4 same plus X-vector and G-vector at each cycle. IPRINT = 5 same plus iteration information during unconstrained minimization sub-problem. IPRINT = 6 same plus gradients and individual penalty parameters. IPRINT = 7 same plus one-dimensional search information during unconstrained minimization sub-problem.
NDV	Number of design (decision) variables contained in vector **X**.

NCON	Number of constraint values contained in array **G**. NCON=0 is allowed. If unconstrained problems are to be solved, lower and upper bounds on the design variables are still imposed. If NCON=0 dimension **G** to unity.
X(NDV)	Vector containing the design variables. On the first call to BIGDOT, this is the user's best guess of the design. On the final return from BIGDOT (INFO=0 is returned), the vector **X** contains the optimum design and vector **G** contains the corresponding constraint values.
XL(NDV)	Array containing lower bounds on the design variables, **X**. If no lower bounds are imposed on one or more of the design variables, the corresponding component(s) of **XL** must be set to a large negative number, say -1.0E+15. Be sure it is-1.0E**+**15 and not -1.0E**-**15 (+15, not -15 exponent).
XU(NDV)	Array containing upper bounds on the design variables, **X**. If no upper bounds are imposed on one or more of the design variables, the corresponding component(s) of **XU** must be set to a large positive number, say 1.0 E+15.
OBJ	Value of the objective function corresponding to the current values of the design variables contained in **X**. On the first call to BIGDOT, OBJ need not be defined. BIGDOT will return a value of INFO=1 to indicate that the user must evaluate OBJ and call BIGDOT again. Subsequently, any time a value of INFO=1 is returned from BIGDOT, the objective, OBJ, must be evaluated for the current design and BIGDOT must be called again. OBJ has the same meaning as F(**X**) in the mathematical problem statement given in Chapter 1.
MINMAX	Integer parameter specifying whether the minimum (MINMAX=0,-1) or maximum (MINMAX=1) of the objective function is to be found.
G(NCON)	Array containing the NCON inequality constraint values corresponding to the current design contained in **X**. On the first call to BIGDOT, the constraint values need not be defined. On return from BIGDOT, if INFO=1, the constraints must be evaluated for the current **X** and BIGDOT must be called again. If NCON=0, array **G** must be dimensioned to 1 or larger, but no constraint values need to be provided.

RPRMBD(20)	Array containing the real (floating point numbers) control parameters. Initialize the entire array to 0.0 to use all default values. If you use other values than the defaults, set the corresponding entries to the desired values. Chapter 3 describes how to change the value of these parameters.
IPRMBD(20)	Array containing the integer control parameters. As with the **RPRMBD** array, set the array to zero to use the default values, or set the proper entries to the desired values. Chapter 3 describes how to change the value of these parameters.
WK(NRWK)	User provided work array for real (floating point) numbers. Array **WK** is used to store internal scalar variables and arrays used by BIGDOT. If the user has not provided enough storage, BIGDOT will print the appropriate message and terminate the optimization.
NRWK	Dimensioned size of work array **WK**. NRWK should be set quite large, starting at about 100000 for a small problem. If NRWK has been given too small a value, an error message will be printed and the optimization will be terminated.
IWK(NRIWK)	User provided work array for integer (fixed point) numbers. Array **IWK** is used to store internal scalar variables and arrays used by BIGDOT. If the user has not provided enough storage, BIGDOT will print the appropriate message and terminate the optimization.
NRIWK	Dimensioned size of work array **IWK**. A good estimate is 3000 for a small problem. Increase the size of NRIWK as the problem grows larger. If NRIWK is too small, an error message will be printed and the optimization will be terminated.

IDISCR(2*NDV)	Discrete variable identifiers. Location I gives the identifier for design variable X_i. If **IDISCR**(I)=0, X_i is a continuous variable. If **DISCR**(I)=-1, X_i is an integer variable between **XL**(I) and **XU**(I). If **IDISCR**(I)>0, X_i is a discrete variable between **XL**(I) and **XU**(I). Let J=**IDISCR**(I). Then J is the first location in **DISCRT** containing the available discrete values. Let K=**IDISCR**(I+NDV). Then K is the last location in **DISCRT** containing the available discrete values. Note that **IDISCR** must always be dimensioned 2*NDV, even if no discrete variable optimization is to be performed. **Alternate Form**: Discrete values can be provided as first, last and increment. Let J=**IDISCR**(I). Then J is the first location in **DISCRT** containing the available discrete values. Let K=**IDISCR**(I+NDV). If K $\leq$ 0; **DISCRT**(J) = Lower bound, DISCRT(J+1) = Upper bound and DISCRT(J+2) = Increment. For example, a variable may be between 1.0 and 10.0 in increments of 0.3.
DISCRT(NN)	Array containing sets of discrete values. If there are no discrete variable sets, **DISCRT** must be dimensioned to one (NN=1). **DISCRT** contains each set of discrete variables referenced by **IDISCR**. The discrete values must be provided in ascending order. Only those values between **XL**(I) and **XU**(I) will be considered in the optimization process. If all discrete variables are taken from the same set, NN equals the number of discrete values provided. If several sets of discrete values are used, NN equals the sum of the discrete values for all sets. **Alternate Form**: If **IDSCRT**(I+NDV) $\leq$ 0, **DISCRT** will contain the lower bound, upper bound and increment for variable I. **NOTE**: The lower and upper bounds contained in **DISCRT** will over-ride the values provided by **XL** and **XU** if they are more restrictive.

Note: The minimum required values of NRWK and NRIWK are defined in Appendix B. Those values are only requirements. The actual dimensions may be larger than this. BIGDOT uses a large number of internal arrays. The arrays **WK** and **IWK** are used to store these and the internal data management allocates the appropriate locations to store the internal arrays.

A subroutine called BDT507 is provided as part of BIGDOT. You can call this subroutine from your main program to determine the required dimensions of the **WK** and **IWK** arrays. See Appendix B for more information on this.

2.5 Compiling and Linking

BIGDOT is supplied as object code. When BIGDOT is to be called by an application program, this BIGDOT object code must be linked to the calling program. If you wish to call BIGDOT from a C or C++ program or require a dynamic link library (DLL) or Application Program Interface (API), please contact VR&D.

BIGDOT is provided in double precision. If you require a single precision version, please contact VR&D.

2.6 DOT Users

If you are currently using the DOT program and wish to add BIGDOT to your library of optimizers, this is easily achieved using SUBROUTINE ALLDOT, which is provided with BIGDOT. ALLDOT allows you to call either DOT or BIGDOT, where BIGDOT is called if METHOD=4.

2.6.1 SUBROUTINE ALLDOT

You will notice that the only difference in the calling statement between DOT and BIGDOT is that the METHOD parameter is not included in the BIGDOT statement and arrays **IDISCR** and **DISCRT** have been added. Also, the **RPRM** and **IPRM** arrays used in DOT are similar to the **RPRMBD** and **IPRMBD** arrays used in BIGDOT. However, the actual entries are different or in a different order.

With ALLDOT, we use the **RPRM** and **IPRM** arrays, just as with DOT. BIGDOT uses four new real parameters called PENALT, PMULT, PENLTD and PMULTD, contained in locations 14 through 17, respectively, of the **RPRM** array. BIGDOT uses one new integer parameter called NDSCRD and this is stored in **IPRM**(16) when using ALLDOT.

The calling statement to ALLDOT is that same as the calling statement for DOT with the addition of the discrete variable information contained in arrays **DISCRT** and **IDISCR**.

To use ALLDOT, define everything just as if you are calling DOT. If you wish to override the default parameters in **RPRM** or **IPRM**, you just define these as you would for calling DOT.

To change from calling DOT to calling ALLDOT, simply change the calling statement of DOT to;

CALL ALLDOT(INFO, METHOD ,IPRINT, NDV, NCON, **X**, **XL**, **XU** ,OBJ,

*MINMAX, **G**, **RPRM**, **IPRM**, **WK**, NRWK, **IWK**, NRIWK,

***DISCRT**, **IDISCR**)

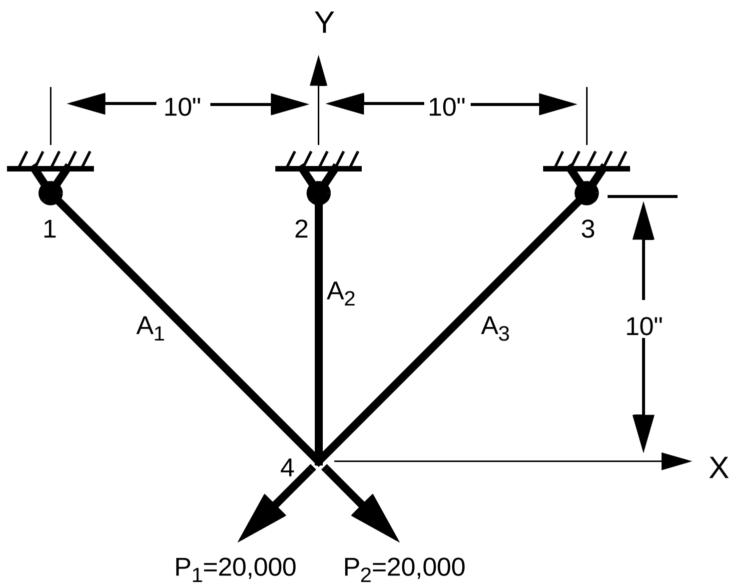
You can now call ALLDOT with METHOD = 1, 2, 3 or 4. METHODS 1, 2 and 3 have the same meaning as for DOT. METHOD = 4 will invoke BIGDOT.

Internally, ALLDOT simply puts the parameters in the RPRM and IPRM arrays into the BIGDOT arrays **RPRMBD** and **IPRMBD**.

Remember that using BIGDOT, you cannot specify that finite difference gradients will be used. You must provide the needed gradients. Of course this may be by finite difference and the DOT003 routine may be used for this.

2.7 A Simple Example

This is the optimization of the 3-bar truss shown below, which is a classical example in structural synthesis.



The objective is to minimize the total volume of the material of the members. The decision variables X_1 and X_2 correspond to the areas of member 1 (and 3) and member 2, respectively. The area of member 3 is “linked” to be the same as member 1 for symmetry. The constraints are tensile stress constraints in members 1 and 2 under load P_1. The loads, P_1 and P_2, are applied separately. This problem, in standard form for optimization, is given below. The original problem actually consists of 12 constraints, being the stress limit in each of the three members under each of the 2 loading conditions. The problem has been abbreviated here for clarity.

$$\text{Minimize} \quad OBJ = 2\sqrt{2}X_1 + X_2 \tag{2-1}$$

Subject to:

$$g_1 = \frac{2X_1 + \sqrt{2}X_2}{2X_1(X_1 + \sqrt{2}X_2)} - 1.0 \le 0.0 \tag{2-2}$$

$$g_2 = \frac{1.0}{X_1 + \sqrt{2}X_2} - 1.0 \le 0.0 \qquad (2\text{-}3)$$

$$0.01 \le X_i \le 100.0 \qquad i = 1,2 \qquad (2\text{-}4)$$

This problem was solved using both DOT and BIGDOT by calling ALLDOT. The results for each method are given in

Table 2-2 Three-Bar Truss Results

METHOD	X_1	X_2	Objective	Function Calls	Gradient Calls
Initial	1.000	1.000	3.828	- -	- -
1	0.7777	0.4356	2.635	52	7
2	0.7875	0.4125	2.640	13	12
3	0.7890	0.4076	2.639	16	6
4	0.7826	0.4255	2.639	135	23

Note that BIGDOT (METHOD = 4) is much less efficient than the DOT methods. Thus, for problems that have under about 1000 active or violated constraints, DOT is usually preferred unless discrete variable optimization is needed.

CHAPTER 3

Advanced Use of BIGDOT

- **Introduction**
- **Over-Riding BIGDOT Default Parameters**
- **Supplying Gradients**
- **Interrupting and Restarting BIGDOT**
- **Output to a Postprocessing Data File**
- **Using BIGDOT with DOT**

3.1 Introduction

This chapter discusses advanced uses of BIGDOT. These include over-riding the internal default parameters, supplying the function gradients, interrupting and restarting the optimization, and output to a post-processing file. These features can be invoked by simple modifications to the user-supplied main program.

BIGDOT solves the constrained optimization problem as a sequence of unconstrained minimization tasks. The pseudo-objective function is defined as

$$\Phi(X) = F(X) + r_p \sum_{j=1}^{M} r_p^j \mathrm{Max}[0, g_j(X)]^2$$

Here r_p is the overall penalty parameter, called PENALT here. The individual constraints $g_j(X)$ are also multiplied by individual penalty parameters, r_p^j, that generally approximate the Lagrange Multipliers λ_j. The individual parameters, r_p^j, are calculated internally by one of three proprietary algorithms. 1. Ad-hoc calculation. 2. Solve a set of equations. 3. Solve a sub-optimization problem.

The parameter, JPENLT is a two digit parameter. The first digit defines the algorithm, 1 or 2. The second digit defines the desired precision of the optimum, 1, 2 or 3, where 1 is the most efficient but least precise and 3 is the more time consuming but most accurate.

3.2 Over-Riding BIGDOT Default Parameters

BIGDOT contains a variety of internal parameters that effect the efficiency and reliability of the optimization process. Each of these is assigned a “default” value to be used unless the user explicitly changes it. Occasionally, the user may wish to over-ride some of the internal parameters of BIGDOT. The default parameters include constraint tolerance, and penalty parameters, among others. The default parameters can be changed by simply setting the proper element of the **RPRMBD** or **IPRMBD** array to the desired value before calling BIGDOT the first time. A sample program is included in this section. Figure Figure 3-1 is a block diagram of the program to over-ride BIGDOT default parameters. Tables 3-1 through 3-4 identify and define the internal parameters. Listing 3-1 shows an example FORTRAN file where the constraint tolerance CTMIN, as well as the constraint gradient storage parameter, NGMAX, are modified.

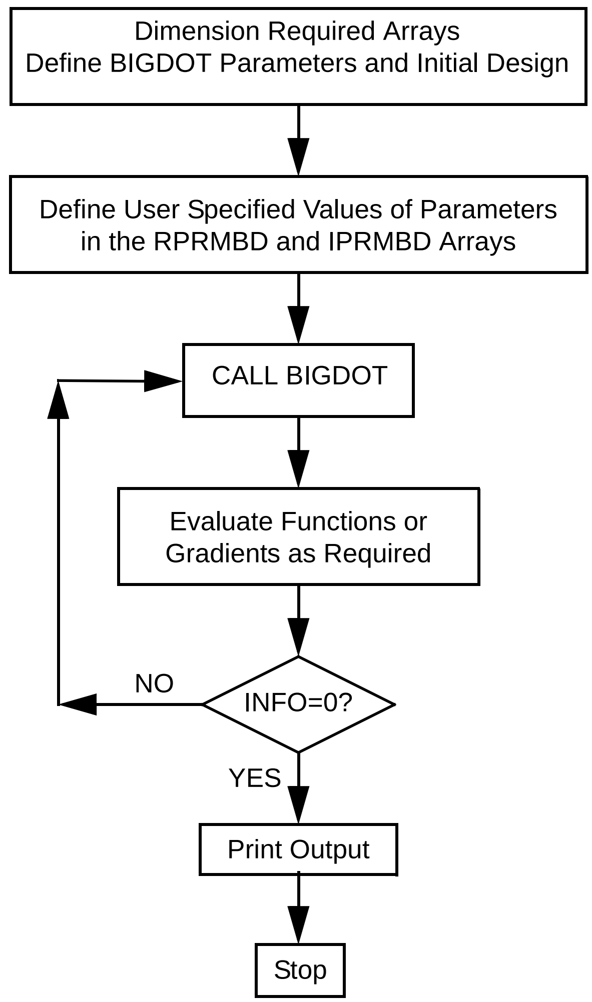

Dimension Required Arrays
Define BIGDOT Parameters and Initial Design
Define User Specified Values of Parameters
in the RPRMBD and IPRMBD Arrays
CALL BIGDOT
Evaluate Functions or
Gradients as Required
NO
INFO=0?
YES
Print Output
Stop

Figure 3-1

BIGDOT uses three separate methods to calculate the individual penalty parameters and allows 3 levels of desired precision for finding the optimum. These are defined by the second digit in JPENLT, that is JPENLT(2) in the tables

Table 3-1: Scalar Parameters Stored in the **RPRMBD** Array

LOCATION	NAME	DEFAULT VALUE		
		Less Precise	**More Precise**	**Most Precise**
		JPENLT(2)=1	**JPENLT(2)=2**	**JPENLT(2)=3**
RPRMBD(1)	PENALT	1.0		
RPRMBD(2)	PMULT	2.5	1.33	
RPRMBD(3)	CTMIN	0.003		
RPRMBD(4)	DABSTR	0.01	0.001	0.0001
RPRMBD(5)	DELSTR	0.01	0.0001	
RPRMBD(6)	DABOBJ	0.01	0.001	0.0001
RPRMBD(7)	DELOBJ	0.01	0.0001	
RPRMBD(8)	PENLTD	0.01	0.001	
RPRMBD(9)	PMULTD	1.25		
RPRMBD(10)	GSTOL	0.25	0.1	0.05
RPRMBD(11)	GSTOLM	0.001	0.00001	0.00001
RPRMBD(12)- **RPRMBD**(20)	NOT USED			

NOTE: F0 = The value of the objective function at the start of optimization (for the initial values of **X**). DABSTR=DABOBJ=MAX[MULT*ABS(F0),1.0E-20] where MULT is the number given in the table.

Table 3-2: Definitions of Parameters Contained in the **RPRMBD** Array

LOCATION	PARAMETER	DEFINITION
1	PENALT	Initial penalty multiplier. Updated by multiplying by PMULT after each optimization cycle.
2	PMULT	Penalty parameter multiplier. $PENALT^{NEW}=PMULT*PENALT^{OLD}$
3	CTMIN	A constraint is violated if its numerical value is more positive than CTMIN.
4	DABSTR	Maximum absolute change in the objective between ITRMST consecutive iterations of the Sequential Unconstrained Minimization process to indicate convergence to the optimum.
5	DELSTR	Maximum relative change in the objective between ITRMST consecutive iterations of the Sequential Unconstrained Minimization Process to indicate convergence to the optimum.
6	DABOBJ	Maximum absolute change in the objective between ITRMOP consecutive iterations to indicate convergence in optimization.
7	DELOBJ	Maximum relative change in the objective between ITRMOP consecutive iterations to indicate convergence in optimization.
8	PENLTD	Initial penalty multiplier for discrete variable constraints. Updated by multiplying by PMULTD after each optimization cycle.
9	PMULTD	Penalty parameter multiplier for discrete variable constraints. $PENLTD^{NEW}=PMULTD*PENLTD^{OLD}$
10	GSTOL	Golden Section tolerance for one-dimensional search.
11	GSTOLM	Minimum Golden Section tolerance for one-dimensional search.

Table 3-3: Parameters Stored in the IPRMBD Array

LOCATION	NAME	DEFAULT VALUE		
		Less Precise	**More Precise**	**Most Precise**
		JPENLT(2)=1	**JPENLT(2)=2**	**JPENLT(3)=3**
IPRMBD(1)	NGMAX	NCON		
IPRMBD(2)	ISCAL	1		
IPRMBD(3)	JTMAX	50		
IPRMBD(4)	ITRMST	2	3	4
IPRMBD(5)	ITMAX	10000 2*NDV If NCON=0		
IPRMBD(6)	ITRMOP	2	3	4
IPRMBD(7)	IWRITE	6		
IPRMBD(8)	JWRITE	0		
IPRMBD(9)	MAXINT	2000000000		
IPRMBD(10)	NDSCRT	0		
IPRMBD(11)	IGRAD	0		
IPRMBD(12)	JGRAD	10		
IPRMBD(13)	JPENLT	21	22	23
IPRMBD(14)- **IPRMBD**(16)	Not Used			
IPRMBD(17)	IOPT	INTERNALLY DEFINED. NORMALLY 0. SET TO 1 TO INDICATE OPTIMUM CONTINUOUS SOLUTION		
IPRMBD(18)	IERROR	INTERNALLY DEFINED FATAL ERROR FLAG		
IPRMBD(19)	NEWITR	INTERNALLY DEFINED ITERATION COUNTER		
IPRMBD(20)	NGT	INTERNALLY DEFINED NUMBER OF NEEDED CONSTRAINT GRADIENTS		

JPENLT(2) = The second digit of JPENLT.

Note; If the value of JPENLT is input, the default value for that precision level is given here. If JPENLT = 0 is input, the Default becomes JPENLT=22.

Table 3-4: Definitions of Parameters Contained in the **IPRMBD** Array

LOCATION	PARAMETER	DEFINITION
1	NGMAX	Number of constraint gradients that can be stored at one time. If less than NCON gradients can be stored, NGMAX will be adjusted to use the greatest available storage.
2	ISCAL	Design variables are scaled by dividing each variable by the magnitude of its initial value. Set ISCAL = -1 to turn off scaling.
3	JTMAX	Maximum number of unconstrained minimizations (cycles) allowed.
4	ITRMST	The number of consecutive iterations for which the absolute or relative convergence criteria must be met to indicate convergence in the Sequential Unconstrained Minimization process.
5	ITMAX	Maximum number of iterations allowed at the unconstrained sub-optimization level.
6	ITRMOP	The number of consecutive iterations for which the absolute or relative convergence criteria must be met to indicate convergence at the unconstrained sub-optimization level.
7	IWRITE	File number for printed output.
8	JWRITE	File number to write iteration history information to. This is useful for using postprocessing programs to plot the iteration process. This is only used if JWRITE>0.
9	MAXINT	Maximum integer number that can be defined.
10	NDSCRT	Number of applications of discrete variable optimization after continuous variable optimization is complete. NDSCRT=0 means do continuous variable optimization only. NDSCRT>0 means do discrete variable optimization NDSCRT times. Normally NDSCRT should not exceed 1. However, NDSCRT>1 may allow the design to move more than one discrete location from the continuous solution.
11	IGRAD	Unit number from which gradients are stored when INFO=2. This file is used if gradients are provided by the calling routine in the file.
12	JGRAD	Unit number from which gradients are stored when INFO=2. This file is opened internally by BIGDOT and is only used if the number of constraint gradients exceeds available storage and gradients are not stored on unit IGRAD.

13	JPENLT	Defines the method used to estimate the individual penalty parameters and the precision desired for the optimization process. JPENLT is a two digit number, KL, where K=1 Use the original method for calculating multipliers. K=2 Calculate multipliers by solving equations. If K=2 generates one or more negative multipliers, a sub-optimization problem will be solved to guarantee non-negative multiplier values. L=1 Least precise but most efficient. Provides a feasible design and will provide a good optimum if the problem is well conditioned. L=2 Default precision. Usually provides a good optimum. L=3 Most precise but uses the most function and gradient evaluations.
14-16	Not presently used	
17	IOPT	Continuous optimum indicator. Normally 0. When IOPT is set to 1, the optimum continuous solution is contained in OBJ, X and G. When discrete variable optimization begins, IOPT is set back to 0.
18	IERROR	Fatal error parameter returned by BIGDOT. Normally = 0. IERROR = 1 indicates WK or IWK dimension is too small. IERROR = 2 indicates some XL(I) is greater than XU(I). IERROR = 3 indicates that a needed gradient is not found or is out of order in the IGRADU file. IERROR = 4 indicates that and end of file was encountered reading gradients from the IGRADU file.
19	NEWITR	Normally = -1. Set = n at the start of a new iteration, where n is the number of the iteration just completed. At the beginning of optimization, n=0 will be returned to indicate that the initial analysis is being done. At the beginning of discrete variable optimization NEWITR is set to -2 to indicate that the current design is the optimum continuous solution. If JWRITE>0, the optimization information will have just been written to that file. If you wish to stop after each iteration (or after a particular iteration) and then re-start later, NEWITR is a flag to do this. NEWITR is defined internally by BIGDOT.
20	NGT	The number of constraint gradients needed. If the user supplies gradients to BIGDOT, this will be needed. The constraint numbers for which gradients are needed are contained in position 1 through NGT of the IWK array. NGT is defined internally by BIGDOT.

LISTING 3-1: OVER-RIDING DEFAULT PARAMETERS: THE 3-BAR TRUSS.

```
C     REQUIRED ARRAYS. WK and IWK are dimensioned very large.
      DIMENSION X(2),XL(2),XU(2),G(2),WK(1000),
     *IWK(500),RPRMBD(20),IPRMBD(20)
C     DIMENSIONS OF WK AND IWK
      NRWK=1000
      NRIWK=500
C     ZERO RPRMBD AND IPRMBD
      DO 10 I=1,20
        RPRMBD(I)=0.0
10    IPRMBD(I)=0
C * * OVER-RIDE THE DEFAULT FOR CTMIN
      RPRMBD(3)=-0.05
C * * SET NGMAX TO 1
      IPRMBD(1)=1
C     DEFINE METHOD, NDV, NCON, IPRINT, MINMAX, X, XL, XU
      METHOD=1
      NDV=2
      NCON=2
      IPRINT=1
      MINMAX=-1
      X(1)=1.0
      X(2)=1.0
      XL(1)=0.1
      XL(2)=0.1
      XU(1)=1.0E+20
      XU(2)=1.0E+20
C     READY TO OPTIMIZE
      INFO=0
20    CALL BIGDOT (INFO,IPRINT,NDV,NCON,X,XL,XU,OBJ,
     *MINMAX,G,RPRMBD,IPRMBD,WK,NRWK,IWK,NRIWK)
C     EVALUATE OBJECTIVE AND CONSTRAINTS
      OBJ=2.*SQRT(2.)*X(1)+X(2)
      G(1)=(2.*X(1)+SQRT(2.)*X(2))/(2.*X(1)*(X(1)+
     *SQRT(2.)*X(2)))-1.
      G(2)=1./(X(1)+SQRT(2.)*X(2))-1.
      IF(INFO.GT.0)GO TO 20
C     INFO=0. OPTIMIZATION IS COMPLETE. PRINT RESULTS.
      WRITE(6,40)OBJ,X(1),X(2),G(1),G(2)
      STOP
40    FORMAT(//5X,'OPTIMUM',5X,'OBJ =',E12.5//5X,'
     *X(1) =',E12.5,5X,'X(2) =',E12.5/5X,
     *'G(1) =',E12.5,5X,'G(2) =',E12.5)
      END
```

3.3 Supplying Gradients

The only option in BIGDOT is to use analytic or user supplied gradients of the objective function and constraints. These gradients are stored in the work array if there is sufficient space. It there is not sufficient space, multiple sets of gradients will be requested. In this case, if multipliers are calculated by solving a sub-optimization task, they will be stored internally in BIGDOT by writing them to the scratch file, IGRAD.

3.3.1 Storing Gradients Directly in the Work Array

The gradients of the objective function are stored in the first NDV locations of the **WK** array. BIGDOT requires gradients of the active and violated constraints only. The active and violated constraints are identified in the first NGT elements of the **IWK** array, where NGT is given in **IPRMBD**(20) and is the number of required constraint gradients. The gradients of these constraints are stored in the next NDV*NGT elements of the **WK** array (following the gradient of the objective function).

The general outline of the code to provide gradients is shown in Figure 3-2 and Figure 3-3. The second figure is a block diagram of the standard calling program for BIGDOT with user-supplied gradients. The logic described there is used in the block where gradients are calculated in the first figure. Listing 3-2 gives an example FORTRAN program for supplying gradients while optimizing the three-bar truss problem of Section 2.7.

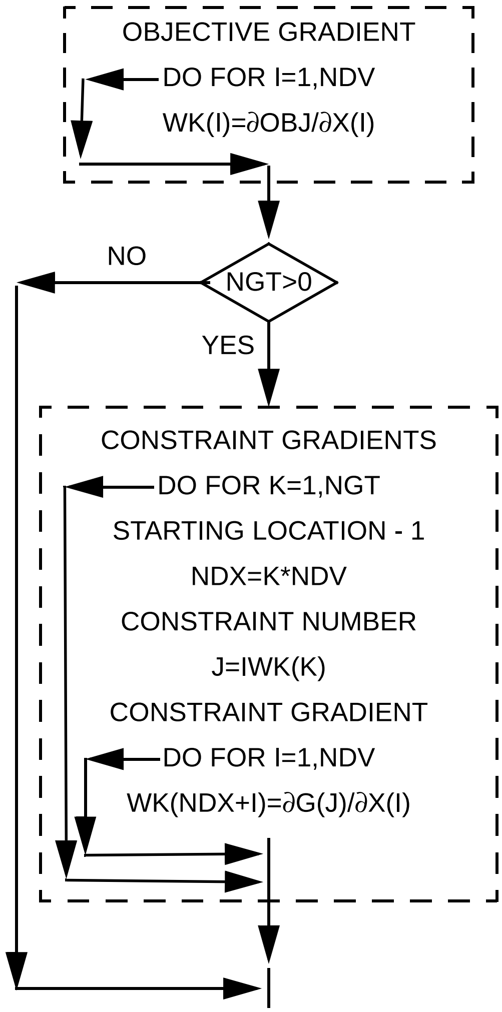


Figure 3-2Supplying Gradients

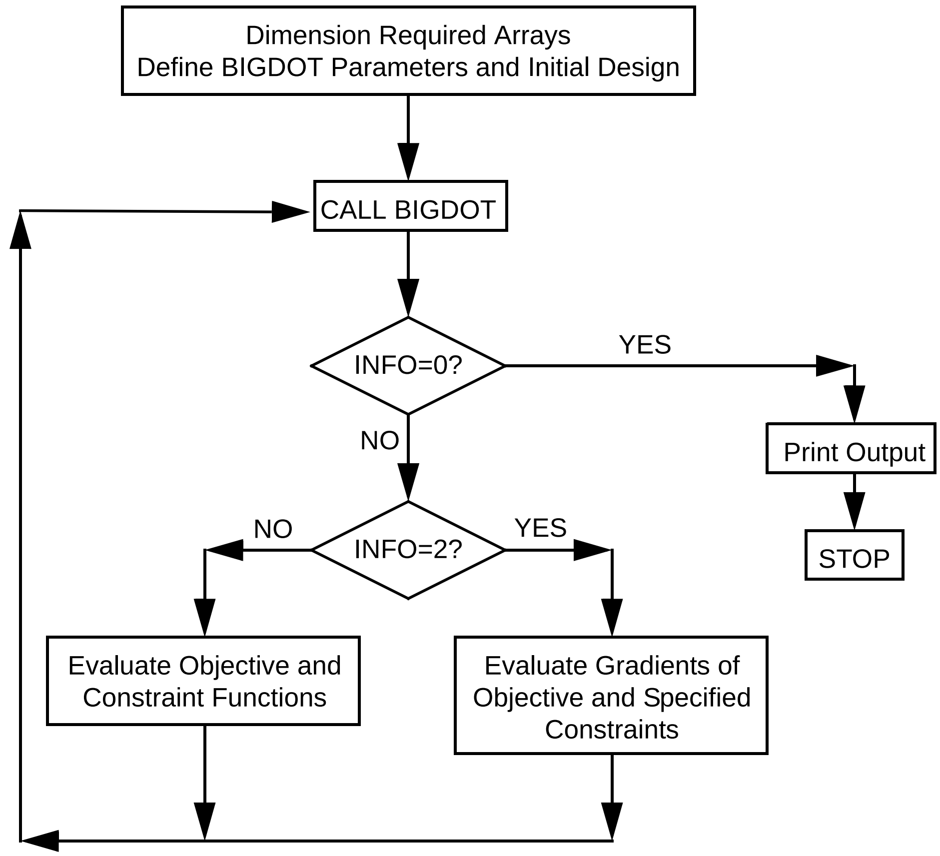


Figure 3-3Program Flow

LISTING 3-2: THREE-BAR TRUSS. USER-SUPPLIED GRADIENTS.

```
C     USER-SUPPLIED GRADIENTS: THE THREE-BAR TRUSS.
C     REQUIRED ARRAYS.
C
      DIMENSION X(2),XL(2),XU(2),G(2),
     *WK(1000),IWK(500),RPRMBD(20),IPRMBD(20),AA(2,2),BB(2,2)
C
C     DIMENSIONS OF WK AND IWK
      NRWK=1000
      NRIWK=500
C     ZERO RPRMBD AND IPRMBD
      DO 10 I=1,20
        RPRMBD(I)=0.0
10    IPRMBD(I)=0
C     SPECIFY THAT GRADIENTS ARE TO BE PROVIDED
      IPRMBD(1)=1
C     DEFINE METHOD, NDV, NCON, IPRINT, MINMAX
      METHOD=1
      NDV=2
      NCON=2
      IPRINT=1
      MINMAX=-1
C     DEFINE X,XL,XU
      X(1)=1.0
      X(2)=1.0
      XL(1)=0.1
      XL(2)=0.1
      XU(1)=1.0E+20
      XU(2)=1.0E+20
C     READY TO OPTIMIZE
      INFO=0
20    CALL BIGDOT (INFO,IPRINT,NDV,NCON,X,XL,XU,OBJ,
     *MINMAX,G,RPRMBD,IPRMBD,WK,NRWK,IWK,NRIWK)
C     EXIT IF CONVERGENCE IS OBTAINED
      IF(INFO.EQ.0)GO TO 70
C     PROVIDE GRADIENTS IF DOT IS REQUESTING THEM
      IF(INFO.EQ.2)GO TO 30
      OBJ=2.*SQRT(2.)*X(1)+X(2)
      G(1)=(2.*X(1)+SQRT(2.)*X(2))/(2.*X(1)*(X(1)+
     *SQRT(2.)*X(2)))-1.
      G(2)=1./(X(1)+SQRT(2.)*X(2))-1.
      GO TO 20
C     -----------------------------------------------
30    CONTINUE
C     GRADIENT OF OBJECTIVE
      WK(1)=2.*SQRT(2.)
      WK(2)=1.0
      NGT=IPRMBD(20)
      IF(NGT.EQ.0)GO TO 20
```

```
C     CONSTRAINT GRADIENTS. USE ARRAY BB FOR TEMPORY
C     STORAGE
      D1=(X(1)+SQRT(2.)*X(2))**2
C     GRADIENT OF G(1)
      BB(1,1)=-(2.*X(1)*X(1)+2.*SQRT(2.)*X(1)*X(2)+
     *SQRT(2.)*X(2)*X(2))/(2.*X(1)*X(1)*D1)
      BB(2,1)=-1./(SQRT(2.)*D1)
C     GRADIENT OF G(2)
      BB(1,2)=-0.5/D1
      BB(2,2)=SQRT(2.)*BB(1,2)
C     STORE APPROPRIATE GRADIENTS IN ARRAY AA
      DO 40 K=1,NGT
      J=IWK(K)
      AA(1,K)=BB(1,J)
40    AA(2,K)=BB(2,J)
C     PUT THE GRADIENTS IN THE WK ARRAY
      N1=NDV
      DO 60 K=1,NGT
      DO 50 I=1,NDV
50       WK(I+N1)=AA(I,K)
      N1=N1+NDV
60    CONTINUE
      GO TO 20
C     ------------------------------------------------
C     INFO = 0. OPTIMIZATION IS COMPLETE. TERMINATE.
C     PRINT RESULTS
70    WRITE(6,80)OBJ,X(1),X(2),G(1),G(2)
      STOP
80    FORMAT(//5X,'OPTIMUM',5X,'OBJ =',E12.5//5X,'X(1) =',
     1E12.5,5X,'X(2) =',E12.5/5X,'G(1) =',E12.5,5X,'G(2) =',
     2E12.5)
      END
```

3.3.2 Storing Gradients In an Unformatted File

This option is no longer available in BOGDOT. When BIGDOT does not have sufficient storage in the **WK** array, it will return repeatedly for a subset of gradients and, if necessary will write them internally in the IGRAD scratch file.

Strictly Linear Problems

BIGDOT provides an alternative to the traditional SIMPLEX or similar linear programming problems. If the problem is strictly linear, it is only required to store all gradients in the unformatted file (beginning with the objective followed by constraints in ascending order) before ever calling BIGDOT. Then with BIGDOT returns a value of INFO=2, simply read this file for the requested gradients and return to BIGDOT.

3.4 Interrupting and Restarting BIGDOT

It is a simple matter to stop the optimization when you wish and then restart from that point at a later time. All that is required is that, on return from BIGDOT, you write the entire contents of the parameter list to a file and then exit. Upon restarting, you simply read this information again and continue from the point where you exited. All internal parameters of BIGDOT will have the values that they had at the time you exited. This provides you with the flexibility to review the optimization progress before continuing, and is particularly useful if computer resources are limited or if you just want to insure that the optimization is performing as expected.

The basic program flow is shown in the Figure 3-4. It is assumed here that you have defined parameters called IFLAG and JFLAG to indicate what you wish to do. The values of IFLAG and JFLAG are assumed to have the following meanings;

IFLAG = 1 - This is a restart. Read saved parameters and continue.

JFLAG = 1 - Save information and exit.

It is perhaps more common to interrupt the optimization process at the end of an iteration. This is easily accomplished by checking the value of NEWITR which is contained in **IPRMBD**(19). If NEWITR=n, iteration n+1 has begun. If you interrupt at this point, you can review the progress of the optimization before proceeding. If JWRITE>0, (JWRITE is stored in location 8 of **IPRMBD**), the current design will have been written to file JWRITE just before this return to the calling program. The modification to interrupt after each iteration is simply to check if JFLAG=1 and NEWITR=n, where n is the iteration number after you wish to exit. If both are true, write the information to a file and exit. Note that NEWITR will be set to 0 at the beginning of the first iteration, after the analysis has been performed for the initial design. Therefore, when NEWITR=0, you have only evaluated the functions for the initial design, but have not changed the design.

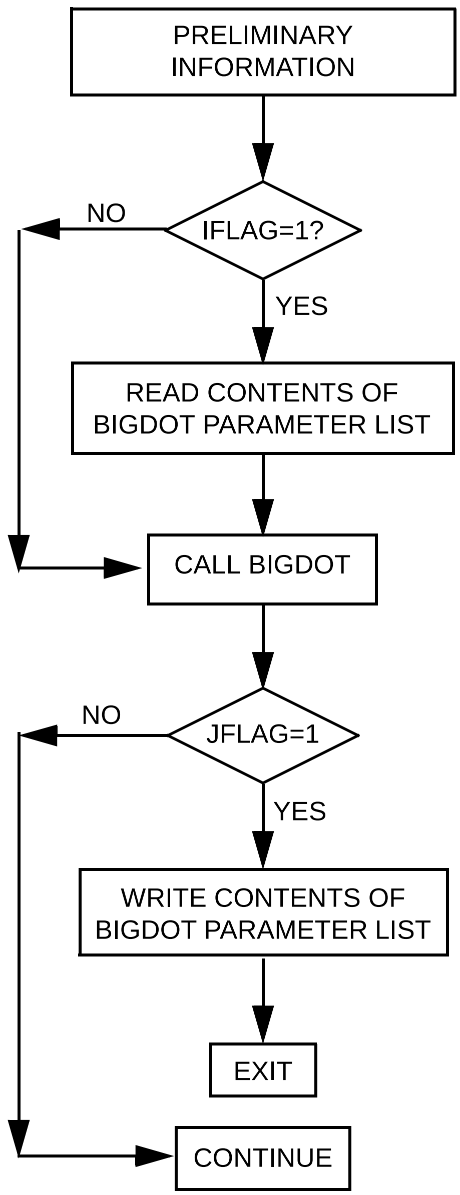


Figure 3-4Restarting BIGDOT

3.5 Output to a Postprocessing Data File

By opening a file and setting JWRITE [**IPRMBD**(8)] to the value of that file number, the user can output useful design iteration history information. The program flow for this is shown in Figure 3-5. This can be used to make decisions based on the progress of the optimization, as well as for plotting the iteration history during or after the optimization process is complete.

If JWRITE is greater than zero, the following information is written to this file during the optimization process. This information is written as an ASCII file using the FORTRAN FORMATS shown;

INFORMATION	**FORMAT**
ITER, NDV, NCON	1X,3I10
OBJ	1X,E15.8
X-VECTOR	1X,5E15.8
G-VECTOR (if NCON > 0)	1X,5E15.8

Note that the X-Vector and G-Vector are written in groups of 5 entries.

This information is written after each unconstrained minimization sub-problem is solved.

In addition to the information written after each iteration, the initial optimization information (JTER = 0) is also output.

It is the user's responsibility to position the file according to his/her needs. The usual application here is to open the file before optimization begins and then access it after the optimization process ends. This is shown in the figure below.

During the iteration loop of optimization, file JWRITE may be accessed. However, it is important to keep track of the file position so that all desired information is saved.

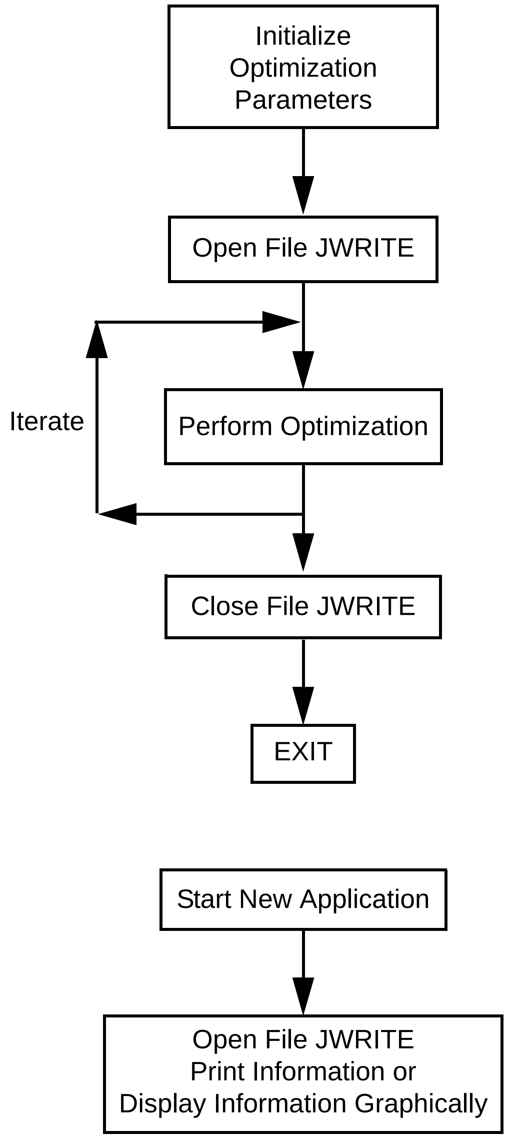


Figure 3-5Output to a Post-Processing File

3.6 Using BIGDOT with DOT

If you are a current DOT user and wish to add BIGDOT to your library of optimizers, this is very easy. You can set up your program just as if you are using DOT, with the **RPRM** and **IPRM** arrays defined as in DOT. In addition, set locations 17 through 20 to PENALT, PMULT, PENLTD, and PMULTD if you wish to over-ride the default values. Also, set IPRM(16) to the value of NDSCRT. Then change the calling statement from calling DOT to calling ALLDOT (provided with BIGDOT). You must link DOT as well as BIGDOT with your program. Now if you call ALLDOT with METHOD = 1, 2 or 3, DOT will be called. If you call ALLDOT with METHOD = 4, BIGDOT will be called. Internally, **RPRMBD** and **IPRMBD** will be used to transfer the information from **RPRM** and **IPRM**, respectively, so the ordering is as needed for BIGDOT.

The calling statement to ALLDOT is as follows;

CALL ALLDOT(INFO, METHOD ,IPRINT, NDV, NCON, **X**, **XL**, **XU** ,OBJ,

*MINMAX, **G**, **RPRM**, **IPRM**, **WK**, NRWK, **IWK**, NRIWK,

***DISCRT**, **IDISCR**)

Note that this is the same as the calling statement for DOT except that the discrete variable tables, **DISCRT** and **IDISCR** have been added.

See Section 2.6 for additional details.

Tables Table 3-1 andTable 3-2 provide the relationship between DOT and BIGDOT parameter locations. When using ALLDOT, the BIGDOT parameters are stored in the specified locations of RPRM and IPRM. Thus, when using ALLDOT, you need to add parameters to locations 14-17 of RPRM and location 16 of IPRM.

NOTE: When calling BIGDOT, either directly or via ALLDOT, you MUST provide gradients. BIGDOT does not calculate gradients by finite difference.

Table 3-1 Real Parameter Cross-reference

NAME	RPRM LOCATION	USED BY		RPRMBD LOCATION
		DOT	BIGDOT	
CT	1	X		
CTMIN	2	X	X	3
DABOBJ	3	X	X	6
DELOBJ	4	X	X	7
DOBJ1	5	X		
DOBJ2	6	X		
DX1	7	X		
DX2	8	X		
FDCH	9	X		
FDCHM	10	X		
RMVLMZ	11	X		
DABSTR	12	X	X	4
DELSTR	13	X	X	5
GSTOL	14	X	X	10
GSTOLM	15	X	X	11
GSTOLS	16	X		
PENALT	17		X	1
PMULT	18		X	2
PENLTD	19		X	8
PMULTD	20		X	9

Bold indicates new parameters used by BIGDOT only.

Table 3-2 Integer Parameter Cross-reference

NAME	IPRM LOCATION	USED BY		IPRMBD LOCATION
		DOT	BIGDOT	
IGRAD	1	X		11
ISCAL	2	X	X	2
ITMAX	3	X	X	5
ITRMOP	4	X	X	6
IWRITE	5	X	X	7
NGMAX	6	X	X	1
IGMAX	7	X		
JTMAX	8	X	X	3
ITRMST	9	X	X	4
JPRINT	10	X		
JGRAD	11		X	12
JPENLT	12		X	13
JWRITE	13	X	X	**8**
MAXINT	14	X	X	9
NSTORE	15	X	X	
NDSCRT	16		X	10
IERROR	18	X	X	18
NEWITR	19	X	X	19
NGT	20	X	X	20

Bold indicates new parameters used by BIGDOT. Note: BOGDOT does not use NSTORE so location 15 of the IPRM array is used to transfer JPENLT.

CHAPTER 4

Examples

- Introduction
- Equilibrium of Spring-Mass System
- Cantilevered Beam
- Topology Optimization

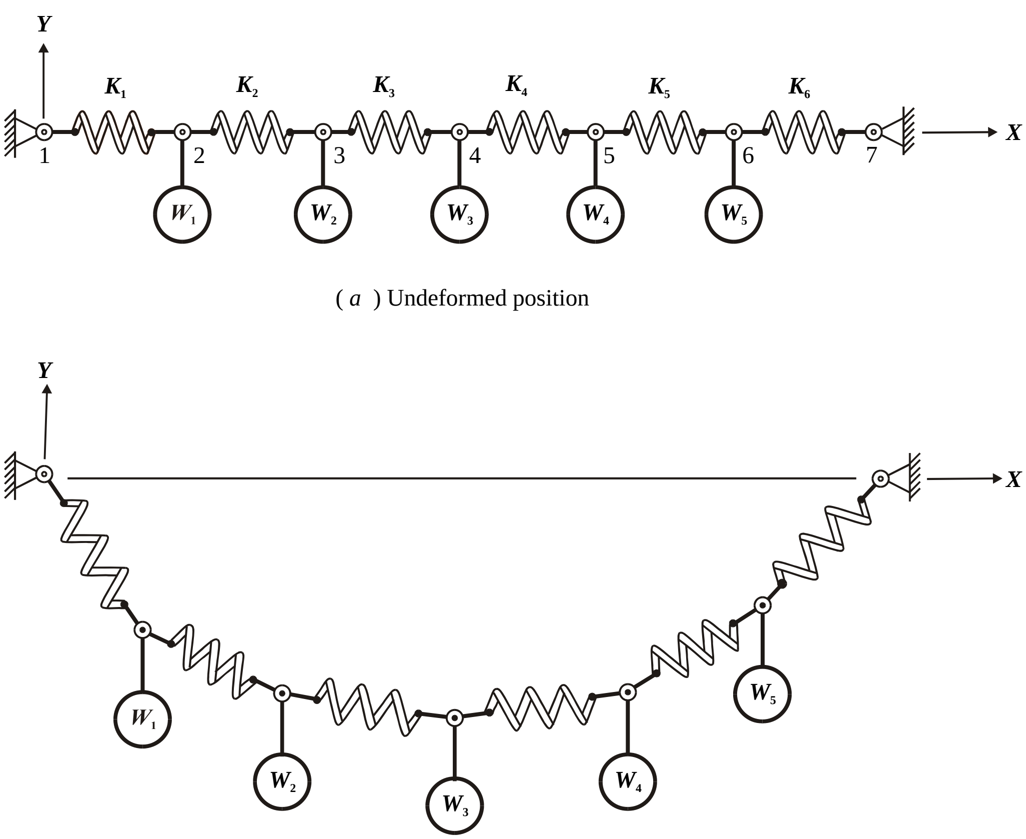
4.1 Introduction

This section presents three examples to demonstrate the use of BIGDOT. There are FORTRAN calling programs included with the program distribution CD for the first two examples. You can begin using BIGDOT immediately by solving these example problems. Refer to Section 2.5 for instructions on compiling the programs and linking them with BIGDOT. The examples are as follows.

- Equilibrium of Spring-Mass System (Potential Energy Minimization)
- Cantilevered Beam (Volume Minimization)
- Topology Optimization in the GENESIS program (Material Distribution)

4.2 Equilibrium of Spring-Mass System

First consider an unconstrained minimization problem, where NCON=0. Figure 4-1 shows a simple spring system supporting weights at the connections between the springs. This system can be analyzed to determine the equilibrium position by minimizing the total potential energy of the system. We can create a problem of whatever size we choose using simple formulas.



(b) Deformed position

Figure 4-1Spring-Mass System

Here we will assume the system is comprised of N masses and N+1 springs, shown in the undeformed position at the top of the figure and the deformed position at the bottom. While the coordinate system shown is for the total system, we can threat the displacements of the masses as design variables, so the initial design is all zeros.

The deformation of spring *i* is

$$\Delta L_i = [(X_{i+1} - X_i)^2 + (Y_{i+1} - Y_i)^2]^{1/2} - L_i^0 \tag{4-1}$$

where the length L^0 of each spring is taken to be the total length divided by N+1.

The stiffness of spring i is taken to be

$$K_i = 500 + 200\left(\frac{N}{3} - i\right)^2 \quad \text{N/m} \tag{4-2}$$

Mass W_j is defined to be

$$W_j = 50j \quad \text{N} \tag{4-3}$$

where *j* corresponds to the joint where W_j is applied.

Note that if K_i = Constant and W_j = Constant, the deformed shape should be a symmetric parabola.

The total potential energy, PE, is now

$$PE = \sum_{i=1}^{N+1} \frac{1}{2} K_i \Delta L_i^2 + \sum_{j=1}^{N} W_j Y_{j=1} \tag{4-4}$$

where PE has units of Newton-meters and the coordinates are positive as shown in the figure.

The objective is now to minimize PE and the are 2N design variables, being the X and Y displacements.

Here, the gradient of the objective function is calculated analytically.

For discrete variable optimization, the design variables are chosen in increments of 0.1.

In each case, the initial objective function value is 0.0. The optimum value will be different depending on the number of masses and springs.

The BIGDOT solutions are

PARAMETER	OPTIMUM N=5,000 (NDV=10,000)	OPTIMUM N=25,000 (NDV=50,000)	OPTIMUM N=50,000 (NDV=100,000)
Continuous Optimization			
OBJECTIVE	$-1.41X10^{10}$	$-5.74X10^{11}$	$-2.48X10^{12}$
NUMBER OF ITERATIONS	5,113	6,089	4,690
FUNCTION EVALUATIONS	40,923	48,678	37,501
GRADIENT EVALUATIONS	5,113	6,089	4,690
Additional Computations for Discrete Optimization			
OBJECTIVE	$-1.36X10^{10}$	$-5.47X10^{11}$	$-2.39X10^{11}$
NUMBER OF CYCLES	7	7	8
FUNCTION EVALUATIONS	547	573	1,066
GRADIENT EVALUATIONS	64	68	192

The program provided here allows the user to input the number of masses, N=NMASS, as well as the print control value. Here, N = 5,000, 12,500 and 25,000 was used, to create problems of 10,000, 25,000 and 50,000 design variables, respectively. You may change NMASS to increase or decrease the problem size. The program is dimensioned to allow a value of NMASS (N) up to 25,000, to yield 50,000 design variables. If you wish to solve even larger problems, be sure to increase the array sizes.

The computer program for this example is given in Listing 4-1. Output for a 1,000 variable example is given in Listing 4-2.

LISTING 4-1: SPRING-MASS SYSTEM FORTRAN PROGRAM

```
C
C     SAMPLE PROGRAM. NMASS HANGING MASS ANALYSIS.
C
      DOUBLE PRECISION X(1000000),XL(1000000),XU(1000000),G(1),
     *WK(40000000),RPRM(20),DISCRT(10000),OBJ
      INTEGER IDISCR(2001000),IPRM(20),IWK(4000000),NGT,NDV,INFO,
     *I,NMASS,METHOD,IPRINT,NRWK,NRIWK,NDSCRT,NCON,NA,MINMAX,JPENLT
C     DEFINE NRWK, NRIWK.
      NRWK=40000000
      NRIWK=4000000
C
C     NUMBER OF MASSES IS NMASS
C
10    WRITE(*,*)' INPUT NMASS, IPRINT, METHOD, NDSCRT, JPENLT'
      READ(*,*)NMASS,IPRINT,METHOD,NDSCRT,JPENLT
      IF(NMASS.EQ.0) STOP
C
      INFO=0
C
C     DEFINE NDV,NCON.
C     TWO TIMES NMASS DESIGN VARIABLES.
      NDV=2*NMASS
C     NO CONSTRAINTS.
      NCON=0
C
C --- TEMP FOR TESTING DISCRETE VARIABLES
C
      IF(METHOD.EQ.4.AND.INFO.EQ.0) THEN
C
C     DISCRETE VARIABLE INFORMATION
C
         DO 20 I=1,NDV
            IDISCR(I)=1
20          IDISCR(NDV+I)=6000
CC20          IDISCR(NDV+I)=-1
         DO 30 I=1,6000
            DISCRT(I)=.1*FLOAT(I)-300.
30       CONTINUE
        IDISCR(2)=6001
        IDISCR(4)=6001
        IDISCR(5)=6001
        IDISCR(NDV+2)=-1
        IDISCR(NDV+4)=-1
        IDISCR(NDV+5)=-1
        DISCRT(6001)=-300
        DISCRT(6002)=600.
        DISCRT(6003)=.1
      ENDIF
C
C --- END TEMP
C
C     DEFINE INITIAL DESIGN.
```

```
      DO 40 I=1,NMASS
         X(I)=0.0
         X(I+NMASS)=0.0
40    CONTINUE
      NA=NDV+1
C     MINIMIZE
      MINMAX=-1
C     OPTIMIZE.
      CALL EVAL(OBJ,X,NMASS,WK,NDV,INFO)
C     DEFINE BOUNDS.
      DO 50 I=1,NMASS
C     LOWER BOUNDS.
         XL(I)=-5000.
         XL(I+NMASS)=-5000.
C     UPPER BOUNDS
         XU(I)=5000.
         XU(I+NMASS)=5000.
50    CONTINUE
C     INITIALIZE INFO TO ZERO.
      INFO=0
C     ZERO RPRM AND IPRM.
      DO 60 I=1,20
         RPRM(I)=0.0
60       IPRM(I)=0
      IPRM(16)=NDSCRT
      IPRM(2)=-1
      IPRM(12)=JPENLT
70    CONTINUE
      CALL ALLDOT(INFO,METHOD,IPRINT,NDV,NCON,X,XL,XU,OBJ,MINMAX,
     1G,RPRM,IPRM,WK,NRWK,IWK,NRIWK,DISCRT,IDISCR)
C     FINISHED?
      NGT=0
      CALL EVAL(OBJ,X,NMASS,WK,NDV,INFO)
      IF(INFO.EQ.0) WRITE(*,*)' FINAL OBJ =',OBJ
      IF(INFO.EQ.0) GO TO 10
      GO TO 70
      END
      SUBROUTINE EVAL (OBJ,X,NMASS,DF,NDV,INFO)
      INTEGER NDV,INFO,I,J,NMASS
      DOUBLE PRECISION X(*),DF(*),AL,ALI,PE1,PE2,XI,YI,WI,
     *XIP1,YIP1,AKI,DLI1,DLI,OBJ
C
      IF(INFO.GT.1) THEN
         DO 10 I=1,NDV
            DF(I)=0.
10       CONTINUE
      ENDIF
C
C     NMASS MASSES.
C
      AL=60.
      ALI=AL/(FLOAT(NMASS)+1.)
      PE1=0.
      PE2=0.
      XI=0.
```

```
      YI=0.
      DO 20 I=1,NMASS
         J=I-1
         WI=50.*FLOAT(I)
         PE1=PE1+WI*X(I+NMASS)
         IF(INFO.EQ.2) DF(I+NMASS)=DF(I+NMASS)+WI
         XIP1=X(I)
         YIP1=X(I+NMASS)
         AKI=500.+200.*((FLOAT(NMASS)/3.-FLOAT(I))**2)
         DLI1=SQRT((ALI+XIP1-XI)**2+(YIP1-YI)**2)
         DLI=DLI1-ALI
         IF(INFO.EQ.2) THEN
           DF(I)=DF(I)+AKI*DLI*(ALI+XIP1-XI)/DLI1
           DF(I+NMASS)=DF(I+NMASS)+AKI*DLI*(YIP1-YI)/DLI1
           IF(J.GT.0) THEN
             DF(J)=DF(J)-AKI*DLI*(ALI+XIP1-XI)/DLI1
             DF(J+NMASS)=DF(J+NMASS)-AKI*DLI*(YIP1-YI)/DLI1
           ENDIF
         ENDIF
         PE2=PE2+0.5*AKI*(DLI**2)
         XI=XIP1
         YI=YIP1
20    CONTINUE
C
C     LAST SPRING
C
      AKI=500.+200.*((FLOAT(NMASS)/3.-FLOAT(NMASS+1))**2)
      DLI1=SQRT((ALI-XI)**2+YI**2)
      DLI=DLI1-ALI
      PE2=PE2+0.5*AKI*(DLI**2)
      IF(INFO.EQ.2) THEN
        DF(NMASS)=DF(NMASS)-AKI*DLI*(ALI-XI)/DLI1
        DF(NDV)=DF(NDV)+AKI*DLI*YI/DLI1
      ENDIF
      OBJ=PE1+PE2
      RETURN
      END
```

LISTING 4-2: SPRING-MASS SYSTEM OUTPUT: NMASS = 500

```
BBBBB       III      GGGGG      DDDDD       OOOOO      TTTTTTT
B    B       I      G           D    D     O     O        T
BBBBB   ==   I   ==  G    GG  == D    D  ==  O  *  O  ==    T
B    B       I      G     G     D    D     O     O        T
BBBBB       III      GGGGG      DDDDD       OOOOO         T

                 DESIGN OPTIMIZATION TOOLS


                    (C) COPYRIGHT, 2012

                     VANDERPLAATS R&D

               ALL RIGHTS RESERVED, WORLDWIDE

                       VERSION 4.0




 CONTROL PARAMETERS

 NUMBER OF DECISION VARIABLES,          NDV =      10000
 NUMBER OF CONSTRAINTS,                NCON =          0
 PRINT CONTROL PARAMETER,            IPRINT =          1
 THE OBJECTIVE FUNCTION WILL BE MINIMIZED


 -- SCALAR PROGRAM PARAMETERS

 REAL PARAMETERS
   1) PENALT =  1.00000E+00           7) DELOBJ =  1.00000E-04
   2) PMULT  =  1.33000E+00           8) PENLTD =  1.00000E+02
   3) CTMIN  =  3.00000E-03           9) PMULTD =  1.25000E+00
   4) DABSTR =  1.00000E-10          10) GSTOL  =  1.00000E-01
   5) DELSTR =  1.00000E-04          11) GSTOLM =  1.00000E-05
   6) DABOBJ =  1.00000E-10

 INTEGER PARAMETERS
   1) NGMAX  =          0             8) JWRITE =       0
   2) ISCAL  =         -1             9) MAXINT =  2000000000
   3) JTMAX  =         50            10) NDSCRT =       1
   4) ITRMST =          4            11) IGRAD  =       0
   5) ITMAX  =      20000            12) JGRAD  =      10
   6) ITRMOP =          5            13) JPENLT =      22
   7) IWRITE =          6
```

```
    STORAGE REQUIREMENTS
ARRAY  DIMENSION        USED
  WK    40000000      120041
 IWK     4000000       30081


-- INITIAL FUNCTION VALUES

OBJ =   0.0000


-- OPTIMIZATION IS COMPLETE

NUMBER OF ITERATIONS = 6878

THERE ARE         0 ACTIVE SIDE CONSTRAINTS

TERMINATION CRITERIA

RELATIVE CONVERGENCE CRITERION WAS MET FOR  5 CONSECUTIVE ITERATIONS


-- OPTIMIZATION RESULTS


OBJECTIVE, F(X) =   -7.79443E+10

FUNCTION CALLS =   55004

GRADIENT CALLS =    6878



-- BEGIN DISCRETE VARIABLE OPTIMIZATION


-- INITIAL FUNCTION VALUES

OBJ = -7.79443E+10


-- OPTIMIZATION IS COMPLETE

NUMBER OF UNCONSTRAINED MINIMIZATIONS =    7

THERE ARE         0 ACTIVE SIDE CONSTRAINTS

THERE ARE     10000 DISCRETE VALUES

TERMINATION CRITERIA

RELATIVE CONVERGENCE CRITERION WAS MET FOR  4 CONSECUTIVE ITERATIONS
```

```
-- OPTIMIZATION RESULTS


OBJECTIVE, F(X) =   -7.45619E+10

FUNCTION CALLS =   55584

GRADIENT CALLS =    6946
```

4.3 Cantilevered Beam

The cantilevered beam shown in Figure 4-2 is to be designed for minimum material volume. The design variables are the width *b* and height *h* at each of *N* segments. We wish to design the beam subject to limits on stress (calculated at the left end of each segment) and the geometric requirement that the height of any segment does not exceed twenty times the width. For this case, if we allowed the height and width to be continuously variable, we can calculate the optimum to be 53,714. This is a theoretical lower bound on the discrete problem solved here.

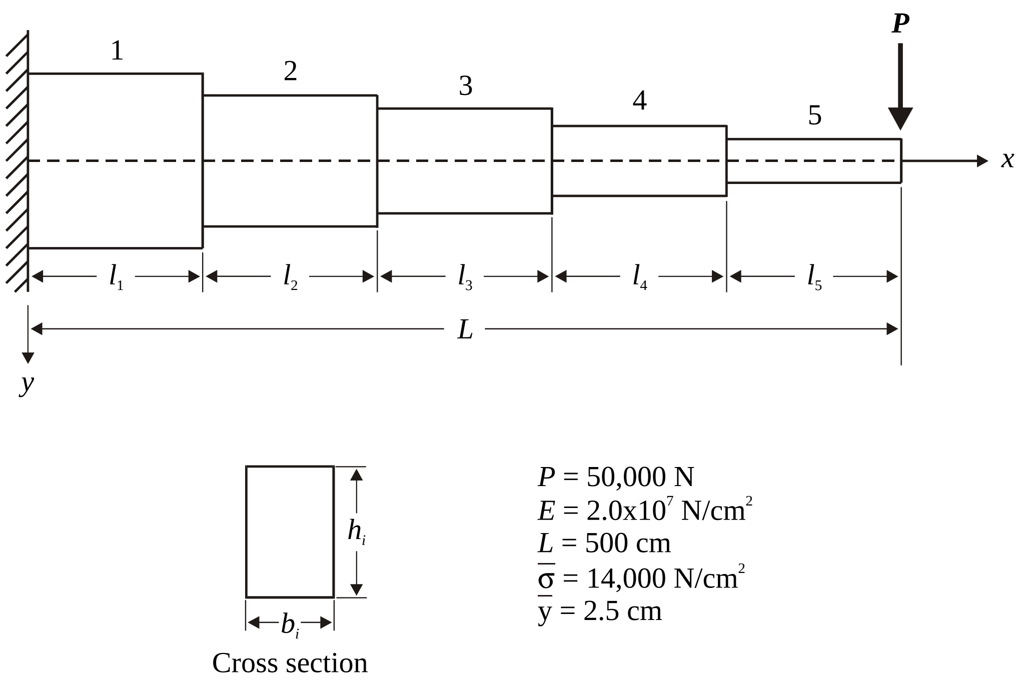


Figure 4-2Cantilevered Beam

The bending moment at the left end of segment *i* is calculated as

$$M_i = P\left[L + l_i - \sum_{j=1}^{i} l_j\right] \tag{4-5}$$

and the corresponding maximum bending stress is

$$\sigma_i = \frac{M_i h_i}{2I_i} \tag{4-6}$$

where

$$I_i = \frac{b_i h_i^3}{12} \tag{4-7}$$

The design task is now defined as

Minimize: $$V = \sum_{i=1}^{N} b_i h_i l_i \tag{4-8}$$

Subject to:

$$\frac{\sigma_i}{\bar{\sigma}} - 1 \leq 0 \qquad i = 1, N \tag{4-9}$$

$$h_i - 20b_i \leq 0 \qquad i = 1, N \tag{4-10}$$

$$b_i \geq 1.0 \qquad i = 1, N \tag{4-11}$$

$$h_i \geq 5.0 \qquad i = 1, N \tag{4-12}$$

Here $\bar{\sigma}$ is the allowable bending stress and $\bar{y}$ is the allowable displacement. This is a design problem in $n = 2N$ variables. There are $N + 1$ nonlinear constraints defined by Eqs. (4-9), N linear constraints defined by Eq. (4-10), and $2N$ side constraints on the design variables defined by Eqs. (4-11) and (4-12). Additionally, lower and upper bounds are imposed explicitly on b_i and h_i, $i = 1, N$ within the optimization program to ensure that the design remains physically meaningful.

The program provided here allows the user to input the number of beam segments (elements), NSEG, as well as the print control value.Here, NSEG=500 was used, to create problems of 1000 design variables. You may change NSEG to increase or decrease the problem size. The program is dimensioned to allow a value of NSEG up to 25,000, to yield 50,000 design variables. If you wish to solve even larger problems, be sure to increase the array sizes.

Here the gradients are calculated analytically. In the GENESIS program used in the second example, gradients are also calculated analytically, which quite efficient. Three tables are presented below. JPENLT=1x is similar to previous versions of BIGDOT but includes three levels of accuracy. JPENLT=2x solves for internal penalty parameters by solving a set of simultaneous equations.

The BIGDOT solutions for JPENLT=1x are

PARAMETER	JPENLT=11	JPENLT=12	JPENLT=13
Continuous Optimization			
OBJECTIVE	53,787	53,758	53,723
# ACTIVE CONSTRAINTS	999	999/2	999/2
NUMBER OF CYCLES	8	22	20
FUNCTION EVALUATIONS	252	1101	761
GRADIENT EVALUATIONS	30	123	83
Additional Computations for Discrete Optimization			
OBJECTIVE	54,846	54,921	54,890
# ACTIVE CONSTRAINTS	862/1	837/2	843/2
NUMBER OF CYCLES	4	6	6
FUNCTION EVALUATIONS	77	231	151
GRADIENT EVALUATIONS	6	14	13
Run Time (Minutes)	0.1	0.2	0.3

The BIGDOT solutions for JPENLT=2x are

PARAMETER	**JPENLT=21**	**JPENLT=22**	**JPENLT=23**
Continuous Optimization			
OBJECTIVE	53,829	53,765	53,775
# ACTIVE CONSTRAINTS	999	999/2	999/2
NUMBER OF CYCLES	11	25	26
FUNCTION EVALUATIONS	304	1435	1169
GRADIENT EVALUATIONS	36	161	127
Additional Computations for Discrete Optimization			
OBJECTIVE	54,945	54,933	54,928
# ACTIVE CONSTRAINTS	826	832/2	837/2
NUMBER OF CYCLES	5	6	3
FUNCTION EVALUATIONS	121	233	79
GRADIENT EVALUATIONS	10	12	5
Run Time (Minutes)	6	26	22

The FORTRAN program used here is shown in Listing 4-3. The BIGDOT output for the JPENLT=22 case is shown in Listing 4-4. Note that sufficient storage was not provided to store the desired number of constraint gradients, so NGMAX was set to 3484. Because BIGDOT required more constraint gradients than this during optimization, reduced storage logic was used. This increased the run time, but otherwise had no effect on the optimum.

LISTING 4-3: CANTILEVER BEAM ANALYSIS FORTRAN PROGRAM

```
C
C     SAMPLE PROGRAM. NSEG ELEMENT BEAM DESIGN.
C
      INTEGER IPRM(20),IDISCR(1000000),NRWK,NRIWK,NSEG,IPRINT,METHOD,
     *NDSCRT,NDV,NCON,INFO,I,NA,MINMAX,IGRAD,NEWITR,NGT,IWK(2100000),
     *VALUES(8),JPENLT
      DOUBLE PRECISION X(500000),XL(500000),XU(500000),G(500001),
     *WK(40000000),RPRM(20),DISCRT(10000),OBJ,T1,T2,T3
      CHARACTER*10 DATE,TIME,ZONE
C
C     DEFINE NRWK, NRIWK.
C
      NRWK=40000000
      NRIWK=2100000
C
C     NUMBER OF BEAM SEGMENTS IS NSEG
C
      OBJ=0.
10    CONTINUE
      WRITE(*,*)' INPUT NSEG, IPRINT, METHOD, NDSCRT, JPENLT'
      READ(*,*)NSEG,IPRINT,METHOD,NDSCRT,JPENLT
      IF(NSEG.EQ.0) STOP
      INFO=0
      CALL DATE_AND_TIME(DATE, TIME, ZONE, VALUES)
      T1=3600.*VALUES(5)+60.*VALUES(6)+VALUES(7)+VALUES(8)/1000.
      DO 20 I=1,NRWK
         WK(I)=OBJ
20    CONTINUE
      DO 30 I=1,NRIWK
         IWK(I)=-1
30    CONTINUE
C
C     DEFINE NDV,NCON.
C     TWO TIMES NSEG DESIGN VARIABLES.
C
      NDV=2*NSEG
C
C     TWO TIMES NSEG CONSTRAINTS.
C
      NCON=2*NSEG
```

```
C
C --- DISCRETE VALUES
C
      IF(METHOD.EQ.4.AND.INFO.EQ.0) THEN
C
C     DISCRETE VARIABLE INFORMATION
C
         DO 40 I=1,NDV
            IDISCR(I)=1
40          IDISCR(NDV+I)=6000
         DO 50 I=1,6000
            DISCRT(I)=.1*FLOAT(I)
50       CONTINUE
      ENDIF
C
C     DEFINE INITIAL DESIGN.
C
      DO 60 I=1,NSEG
C
C     INITIAL VALUES.
C
         X(I)=5.0
         X(I+NSEG)=40.0
60    CONTINUE
C
C     MINIMIZE
C
      MINMAX=-1
C
C     DEFINE BOUNDS.
C
      DO 70 I=1,NSEG
C
C     LOWER BOUNDS.
C
         XL(I)=.5
         XL(I+NSEG)=5.
C
C     UPPER BOUNDS
C
         XU(I)=100.
         XU(I+NSEG)=100.
70    CONTINUE
C
C     INITIALIZE INFO TO ZERO.
C
      INFO=0
```

```
C
C     ZERO RPRM AND IPRM.
C
      DO 80 I=1,20
         RPRM(I)=0.0
         IPRM(I)=0
80    CONTINUE
      IPRM(12)=JPENLT
      IPRM(16)=NDSCRT
      IGRAD=IPRM(1)
      IF(IGRAD.GT.0) OPEN(UNIT=10,FILE='JUNK',ACCESS='SEQUENTIAL',
     *STATUS='UNKNOWN',FORM='UNFORMATTED')
C
90    CONTINUE
      CALL ALLDOT(INFO,METHOD,IPRINT,NDV,NCON,X,XL,XU,OBJ,MINMAX,
     1G,RPRM,IPRM,WK,NRWK,IWK,NRIWK,DISCRT,IDISCR)
      IGRAD=IPRM(1)
      NEWITR=IPRM(19)
C
C     FINISHED?
      IF(INFO.EQ.0) THEN
         WRITE(*,*)' FINAL OBJ =',OBJ
         CALL DATE_AND_TIME(DATE, TIME, ZONE, VALUES)
         T2=3600.*VALUES(5)+60.*VALUES(6)+VALUES(7)+VALUES(8)/1000.
         T3=(T2-T1)/60.
         WRITE(*,*)' ELAPSED TIME',T3,' MINUTES'
      ENDIF
      NGT=IPRM(20)
      NA=NDV+1
      CALL EVAL(OBJ,X,G,NSEG,WK,WK(NA),NDV,INFO,NGT,IWK,IGRAD)
      IF(INFO.EQ.0) GO TO 10
      GO TO 90
      END
      SUBROUTINE EVAL (OBJ,X,G,NSEG,DF,A,NDV,INFO,NGT,IC,IGRAD)
      INTEGER IC(*),NDV,IGRAD,NSEG,INFO,I,IREC,N1,N2,J,NGT,MGRAD
      DOUBLE PRECISION X(*),G(*),DF(*),DG(500000),A(NDV,*),P,E,AL,ALI,
     *SIG,YMX,VOL,ALA,Y,YP,BI,HI,AI,AM,SIGI,OBJ
C
      IF(IGRAD.GT.0) REWIND IGRAD
C
C     NSEG-ELEMENT BEAM.
C
      P=50000.
      E=2.0E+7
      AL=500.
      ALI=AL/FLOAT(NSEG)
      SIG=14000.
      YMX=2.54
```

```
C
C     GRADIENT OF OBJECTIVE FUNCTION
C
      IF(INFO.GT.1) THEN
         DO 10 I=1,NSEG
            DF(I)=ALI*X(I+NSEG)
            DF(I+NSEG)=ALI*X(I)
10       CONTINUE
         IREC=0
         IF(IGRAD.GT.0) CALL BDT009(DF,NDV,IREC,IGRAD)
      ENDIF
C
C     VOLUME, STRESS CONSTRAINTS, H/B CONSTRAINTS.
C
      VOL=0.
      ALA=0.
      Y=0.
      YP=0.
      N1=1
      N2=IC(N1)
      DO 40 I=1,NSEG
         BI=X(I)
         HI=X(I+NSEG)
         VOL=VOL+BI*HI
         AI=BI*(HI**3)/12.
         ALA=ALA+ALI
         AM=P*(AL+ALI-ALA)
         SIGI=.5*AM*X(I+NSEG)/AI
C
C     STRESS CONSTRAINTS.
C
         G(I)=1.*(SIGI/SIG-1.)
C
C     GRADIENTS
C
         IF(INFO.GT.1) THEN
            IF(N2.EQ.I) THEN
C
C     STRESS GRADIENT
C
               IF(IGRAD.GT.0) THEN
                  DO 20 J=1,NDV
                     DG(J)=0.
20                CONTINUE
                  DG(I)=-6.*AM/(BI**2*HI**2*SIG)
                  DG(I+NSEG)=-12.*AM/(BI*HI**3*SIG)
                  CALL BDT009(DG,NDV,N2,IGRAD)
               ELSE
                  DO 30 J=1,NDV
                     A(J,N1)=0.
30                CONTINUE
                  A(I,N1)=-6.*AM/(BI**2*HI**2*SIG)
                  A(I+NSEG,N1)=-12.*AM/(BI*HI**3*SIG)
               ENDIF
```

```
               N1=N1+1
               IF(N1.LE.NGT) N2=IC(N1)
            ENDIF
         ENDIF
C
C     H/B CONSTRAINTS.
C
         G(I+NSEG)=0.01*(X(I+NSEG)-20.*X(I))
         Y=Y+.5*P*ALI*ALI*(AL-ALA+2.*ALI/3.)/(E*AI)+YP*ALI
         YP=YP+P*ALI*(AL+.5*ALI-ALA)/(E*AI)
         OBJ=VOL*AL/FLOAT(NSEG)
40    CONTINUE
C
C     H/B GRADIENT
C
      IF(INFO.GT.1) THEN
         DO 70 I=1,NSEG
            MGRAD=I+NSEG
            IF(N2.EQ.MGRAD) THEN
               IF (IGRAD.GT.0) THEN
                  DO 50 J=1,NDV
                     DG(J)=0.
50                CONTINUE
                  DG(I)=-.2
                  DG(I+NSEG)=0.01
                  CALL BDT009(DG,NDV,N2,IGRAD)
               ELSE
                  DO 60 J=1,NDV
                     A(J,N1)=0.
60                CONTINUE
                  A(I,N1)=-.2
                  A(I+NSEG,N1)=0.01
               ENDIF
                  N1=N1+1
                  IF (N1.LE.NGT) N2=IC(N1)
            ENDIF
70       CONTINUE
      ENDIF
      RETURN
      END
```

LISTING 4-4: CANTILEVER BEAM OUTPUT: NSEG = 500

```
BBBBB       III       GGGGG       DDDDD        OOOOO       TTTTTTT
B    B       I       G            D    D      O     O         T
BBBBB   ==   I   ==  G    GG  ==  D    D  ==  O  *  O  ==     T
B    B       I       G     G      D    D      O     O         T
BBBBB       III       GGGGG       DDDDD        OOOOO          T

                   DESIGN OPTIMIZATION TOOLS


                      (C) COPYRIGHT, 2012

                        VANDERPLAATS R&D

                 ALL RIGHTS RESERVED, WORLDWIDE

                           VERSION 4.0




 CONTROL PARAMETERS

 NUMBER OF DECISION VARIABLES,          NDV =      1000
 NUMBER OF CONSTRAINTS,                NCON =      1000
 PRINT CONTROL PARAMETER,            IPRINT =         2
 THE OBJECTIVE FUNCTION WILL BE MINIMIZED


 -- SCALAR PROGRAM PARAMETERS

 REAL PARAMETERS
   1) PENALT =  1.00000E+00          7) DELOBJ =  1.00000E-04
   2) PMULT  =  1.33000E+00          8) PENLTD =  1.00000E+02
   3) CTMIN  =  3.00000E-03          9) PMULTD =  1.25000E+00
   4) DABSTR =  1.00000E+02         10) GSTOL  =  1.00000E-01
   5) DELSTR =  1.00000E-04         11) GSTOLM =  1.00000E-05
   6) DABOBJ =  1.00000E+02

 INTEGER PARAMETERS
   1) NGMAX  =      1000             8) JWRITE =      0
   2) ISCAL  =         1             9) MAXINT =  2000000000
   3) JTMAX  =        50            10) NDSCRT =      1
   4) ITRMST =         4            11) IGRAD  =      0
   5) ITMAX  =     10000            12) JGRAD  =     10
   6) ITRMOP =         5            13) JPENLT =     22
   7) IWRITE =         6
```

```
    STORAGE REQUIREMENTS
ARRAY  DIMENSION        USED
  WK    40000000     2017040
 IWK     2100000        4081


-- INITIAL FUNCTION VALUES

OBJ =  1.00000E+05

MAXIMUM CONSTRAINT VALUE =  0.33929     IS CONSTRAINT NUMBER         1


-- BEGIN OPTIMIZATION


-- BEGIN CONTINUOUS CYCLE    1


MAXIMUM CONSTRAINT VALUE = 3.39286E-01 IS CONSTRAINT         1

THERE ARE        12 ACTIVE CONSTRAINTS AND       126  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  1.00005E+05 OBJECTIVE =  1.00000E+05

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  3.66186E+04   OBJECTIVE =  2.72268E+04


MAXIMUM CONSTRAINT VALUE = 1.07986E+00 IS CONSTRAINT       496


-- BEGIN CONTINUOUS CYCLE    2


MAXIMUM CONSTRAINT VALUE = 1.07986E+00 IS CONSTRAINT       496

THERE ARE         4 ACTIVE CONSTRAINTS AND       986  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  3.97179E+04 OBJECTIVE =  2.72268E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  4.03724E+04   OBJECTIVE =  3.06222E+04


MAXIMUM CONSTRAINT VALUE = 9.68551E-01 IS CONSTRAINT       498
```

```
-- BEGIN CONTINUOUS CYCLE    3


MAXIMUM CONSTRAINT VALUE = 9.68551E-01 IS CONSTRAINT       498

THERE ARE         0 ACTIVE CONSTRAINTS AND       995  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  4.35899E+04 OBJECTIVE =  3.06222E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  4.36095E+04   OBJECTIVE =  3.57509E+04


MAXIMUM CONSTRAINT VALUE = 6.51573E-01 IS CONSTRAINT       498


-- BEGIN CONTINUOUS CYCLE    4


MAXIMUM CONSTRAINT VALUE = 6.51573E-01 IS CONSTRAINT       498

THERE ARE         4 ACTIVE CONSTRAINTS AND       995  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  4.62028E+04 OBJECTIVE =  3.57509E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  4.60994E+04   OBJECTIVE =  3.97951E+04


MAXIMUM CONSTRAINT VALUE = 5.17951E-01 IS CONSTRAINT       498


-- BEGIN CONTINUOUS CYCLE    5


MAXIMUM CONSTRAINT VALUE = 5.17951E-01 IS CONSTRAINT       498

THERE ARE         2 ACTIVE CONSTRAINTS AND       996  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  4.81798E+04 OBJECTIVE =  3.97951E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  4.79930E+04   OBJECTIVE =  4.30110E+04


MAXIMUM CONSTRAINT VALUE = 2.47807E-01 IS CONSTRAINT       498
```

```
-- BEGIN CONTINUOUS CYCLE    6


MAXIMUM CONSTRAINT VALUE = 2.47807E-01 IS CONSTRAINT       498

THERE ARE         3 ACTIVE CONSTRAINTS AND       994  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  4.96370E+04 OBJECTIVE =  4.30110E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  4.94268E+04   OBJECTIVE =  4.55556E+04


MAXIMUM CONSTRAINT VALUE = 1.52154E-01 IS CONSTRAINT       975


-- BEGIN CONTINUOUS CYCLE    7


MAXIMUM CONSTRAINT VALUE = 1.52154E-01 IS CONSTRAINT       975

THERE ARE         1 ACTIVE CONSTRAINTS AND       998  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.07042E+04 OBJECTIVE =  4.55556E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.05058E+04   OBJECTIVE =  4.75667E+04


MAXIMUM CONSTRAINT VALUE = 1.20018E-01 IS CONSTRAINT       976


-- BEGIN CONTINUOUS CYCLE    8


MAXIMUM CONSTRAINT VALUE = 1.20018E-01 IS CONSTRAINT       976

THERE ARE         1 ACTIVE CONSTRAINTS AND       998  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.14757E+04 OBJECTIVE =  4.75667E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.13213E+04   OBJECTIVE =  4.90743E+04


MAXIMUM CONSTRAINT VALUE = 9.03155E-02 IS CONSTRAINT       984
```

```
-- BEGIN CONTINUOUS CYCLE    9


MAXIMUM CONSTRAINT VALUE = 9.03155E-02 IS CONSTRAINT       984

THERE ARE         1 ACTIVE CONSTRAINTS AND       998  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.20629E+04 OBJECTIVE =  4.90743E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.19355E+04   OBJECTIVE =  5.02166E+04


MAXIMUM CONSTRAINT VALUE = 6.57379E-02 IS CONSTRAINT       987


-- BEGIN CONTINUOUS CYCLE   10


MAXIMUM CONSTRAINT VALUE = 6.57379E-02 IS CONSTRAINT       987

THERE ARE         1 ACTIVE CONSTRAINTS AND       998  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.25028E+04 OBJECTIVE =  5.02166E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.23975E+04   OBJECTIVE =  5.10486E+04


MAXIMUM CONSTRAINT VALUE = 5.10538E-02 IS CONSTRAINT       991


-- BEGIN CONTINUOUS CYCLE   11


MAXIMUM CONSTRAINT VALUE = 5.10538E-02 IS CONSTRAINT       991

THERE ARE         1 ACTIVE CONSTRAINTS AND       998  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.28426E+04 OBJECTIVE =  5.10486E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.27477E+04   OBJECTIVE =  5.17172E+04


MAXIMUM CONSTRAINT VALUE = 3.97129E-02 IS CONSTRAINT       992
```

```
-- BEGIN CONTINUOUS CYCLE   12


MAXIMUM CONSTRAINT VALUE = 3.97129E-02 IS CONSTRAINT       992

THERE ARE         1 ACTIVE CONSTRAINTS AND       998  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.30878E+04 OBJECTIVE =  5.17172E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.30089E+04   OBJECTIVE =  5.22254E+04


MAXIMUM CONSTRAINT VALUE = 5.98997E-02 IS CONSTRAINT       499


-- BEGIN CONTINUOUS CYCLE   13


MAXIMUM CONSTRAINT VALUE = 5.98997E-02 IS CONSTRAINT       499

THERE ARE         1 ACTIVE CONSTRAINTS AND       998  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.32675E+04 OBJECTIVE =  5.22254E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.32060E+04   OBJECTIVE =  5.26124E+04


MAXIMUM CONSTRAINT VALUE = 3.81077E-02 IS CONSTRAINT       499


-- BEGIN CONTINUOUS CYCLE   14


MAXIMUM CONSTRAINT VALUE = 3.81077E-02 IS CONSTRAINT       499

THERE ARE         1 ACTIVE CONSTRAINTS AND       998  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.34019E+04 OBJECTIVE =  5.26124E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.33545E+04   OBJECTIVE =  5.29186E+04


MAXIMUM CONSTRAINT VALUE = 1.81327E-02 IS CONSTRAINT       996
```

```
-- BEGIN CONTINUOUS CYCLE   15


MAXIMUM CONSTRAINT VALUE = 1.81327E-02 IS CONSTRAINT       996

THERE ARE         1 ACTIVE CONSTRAINTS AND       998  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.34984E+04 OBJECTIVE =  5.29186E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.34660E+04   OBJECTIVE =  5.31388E+04


MAXIMUM CONSTRAINT VALUE = 1.42524E-02 IS CONSTRAINT       996


-- BEGIN CONTINUOUS CYCLE   16


MAXIMUM CONSTRAINT VALUE = 1.42524E-02 IS CONSTRAINT       996

THERE ARE         1 ACTIVE CONSTRAINTS AND       998  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.35740E+04 OBJECTIVE =  5.31388E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.35498E+04   OBJECTIVE =  5.33025E+04


MAXIMUM CONSTRAINT VALUE = 1.12477E-02 IS CONSTRAINT       997


-- BEGIN CONTINUOUS CYCLE   17


MAXIMUM CONSTRAINT VALUE = 1.12477E-02 IS CONSTRAINT       997

THERE ARE         1 ACTIVE CONSTRAINTS AND       998  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.36314E+04 OBJECTIVE =  5.33025E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.36126E+04   OBJECTIVE =  5.34199E+04


MAXIMUM CONSTRAINT VALUE = 1.43450E-02 IS CONSTRAINT       499
```

```
-- BEGIN CONTINUOUS CYCLE   18


MAXIMUM CONSTRAINT VALUE = 1.43450E-02 IS CONSTRAINT       499

THERE ARE         1 ACTIVE CONSTRAINTS AND       998  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.36762E+04 OBJECTIVE =  5.34199E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.36599E+04   OBJECTIVE =  5.35162E+04


MAXIMUM CONSTRAINT VALUE = 6.02369E-03 IS CONSTRAINT       499


-- BEGIN CONTINUOUS CYCLE   19


MAXIMUM CONSTRAINT VALUE = 6.02369E-03 IS CONSTRAINT       499

THERE ARE         1 ACTIVE CONSTRAINTS AND       998  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37074E+04 OBJECTIVE =  5.35162E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.36956E+04   OBJECTIVE =  5.35862E+04


MAXIMUM CONSTRAINT VALUE = 7.69721E-03 IS CONSTRAINT       499


-- BEGIN CONTINUOUS CYCLE   20


MAXIMUM CONSTRAINT VALUE = 7.69721E-03 IS CONSTRAINT       499

THERE ARE       797 ACTIVE CONSTRAINTS AND       202  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37317E+04 OBJECTIVE =  5.35862E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37224E+04   OBJECTIVE =  5.36407E+04


MAXIMUM CONSTRAINT VALUE = 3.44231E-03 IS CONSTRAINT       499
```

```
-- BEGIN CONTINUOUS CYCLE   21


MAXIMUM CONSTRAINT VALUE = 3.44231E-03 IS CONSTRAINT       499

THERE ARE       996 ACTIVE CONSTRAINTS AND         3  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37493E+04 OBJECTIVE =  5.36407E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37425E+04   OBJECTIVE =  5.36816E+04


MAXIMUM CONSTRAINT VALUE = 4.22535E-03 IS CONSTRAINT       499


-- BEGIN CONTINUOUS CYCLE   22


MAXIMUM CONSTRAINT VALUE = 4.22535E-03 IS CONSTRAINT       499

THERE ARE       998 ACTIVE CONSTRAINTS AND         1  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37626E+04 OBJECTIVE =  5.36816E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37577E+04   OBJECTIVE =  5.37115E+04


MAXIMUM CONSTRAINT VALUE = 1.90318E-03 IS CONSTRAINT       998


-- BEGIN CONTINUOUS CYCLE   23


MAXIMUM CONSTRAINT VALUE = 1.90318E-03 IS CONSTRAINT       998

THERE ARE       999 ACTIVE CONSTRAINTS AND         0  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37729E+04 OBJECTIVE =  5.37115E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37691E+04   OBJECTIVE =  5.37344E+04


MAXIMUM CONSTRAINT VALUE = 2.13056E-03 IS CONSTRAINT       499
```

```
-- BEGIN CONTINUOUS CYCLE   24


MAXIMUM CONSTRAINT VALUE = 2.13056E-03 IS CONSTRAINT       499

THERE ARE       999 ACTIVE CONSTRAINTS AND         0  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37805E+04 OBJECTIVE =  5.37344E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37776E+04   OBJECTIVE =  5.37517E+04


MAXIMUM CONSTRAINT VALUE = 1.50885E-03 IS CONSTRAINT       499


-- BEGIN CONTINUOUS CYCLE   25


MAXIMUM CONSTRAINT VALUE = 1.50885E-03 IS CONSTRAINT       499

THERE ARE       999 ACTIVE CONSTRAINTS AND         0  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37862E+04 OBJECTIVE =  5.37517E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37840E+04   OBJECTIVE =  5.37645E+04


MAXIMUM CONSTRAINT VALUE = 1.11714E-03 IS CONSTRAINT       499


-- OPTIMIZATION IS COMPLETE

NUMBER OF UNCONSTRAINED MINIMIZATIONS =   25

CONSTRAINT TOLERANCE, CT =-3.00000E-02   CTMIN = 3.00000E-03

THERE ARE       999 ACTIVE CONSTRAINTS AND         0 VIOLATED CONSTRAINTS

THERE ARE         2 ACTIVE SIDE CONSTRAINTS

TERMINATION CRITERIA

ABSOLUTE CONVERGENCE CRITERION WAS MET FOR  4 CONSECUTIVE ITERATIONS
```

```
-- OPTIMIZATION RESULTS


OBJECTIVE, F(X) =    5.37645E+04

MAXIMUM CONSTRAINT VALUE =  1.11714E-03  IS CONSTRAINT NUMBER       499

FUNCTION CALLS =    1435

GRADIENT CALLS =     161


-- BEGIN DISCRETE VARIABLE OPTIMIZATION


-- INITIAL FUNCTION VALUES

OBJ =   53765.

MAXIMUM CONSTRAINT VALUE =  1.11714E-03 IS CONSTRAINT NUMBER       499


-- BEGIN DISCRETE CYCLE    1


MAXIMUM CONSTRAINT VALUE = 1.11714E-03 IS CONSTRAINT       499

THERE ARE       999 ACTIVE CONSTRAINTS AND         0  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  1.03144E+05 OBJECTIVE =  5.37645E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.49349E+04   OBJECTIVE =  5.49326E+04


MAXIMUM CONSTRAINT VALUE = 1.41174E-02 IS CONSTRAINT       500

THERE ARE       999 DISCRETE VALUES
```

```
-- BEGIN DISCRETE CYCLE    2


MAXIMUM CONSTRAINT VALUE = 1.41174E-02 IS CONSTRAINT       500

THERE ARE       831 ACTIVE CONSTRAINTS AND         1  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.49356E+04 OBJECTIVE =  5.49326E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.49356E+04   OBJECTIVE =  5.49326E+04


MAXIMUM CONSTRAINT VALUE = 1.41174E-02 IS CONSTRAINT       500

THERE ARE       999 DISCRETE VALUES


-- BEGIN DISCRETE CYCLE    3


MAXIMUM CONSTRAINT VALUE = 1.41174E-02 IS CONSTRAINT       500

THERE ARE       831 ACTIVE CONSTRAINTS AND         1  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.49366E+04 OBJECTIVE =  5.49326E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.49353E+04   OBJECTIVE =  5.49331E+04


MAXIMUM CONSTRAINT VALUE = 1.00003E-03 IS CONSTRAINT       591

THERE ARE      1000 DISCRETE VALUES


-- BEGIN DISCRETE CYCLE    4


MAXIMUM CONSTRAINT VALUE = 1.00003E-03 IS CONSTRAINT       591

THERE ARE       832 ACTIVE CONSTRAINTS AND         0  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.49361E+04 OBJECTIVE =  5.49331E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.49361E+04   OBJECTIVE =  5.49331E+04


MAXIMUM CONSTRAINT VALUE = 1.00003E-03 IS CONSTRAINT       591

THERE ARE      1000 DISCRETE VALUES
```

```
-- BEGIN DISCRETE CYCLE    5


MAXIMUM CONSTRAINT VALUE = 1.00003E-03 IS CONSTRAINT       591

THERE ARE       832 ACTIVE CONSTRAINTS AND         0  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.49371E+04 OBJECTIVE =  5.49331E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.49371E+04   OBJECTIVE =  5.49331E+04


MAXIMUM CONSTRAINT VALUE = 1.00003E-03 IS CONSTRAINT       591

THERE ARE      1000 DISCRETE VALUES


-- BEGIN DISCRETE CYCLE    6


MAXIMUM CONSTRAINT VALUE = 1.00003E-03 IS CONSTRAINT       591

THERE ARE       832 ACTIVE CONSTRAINTS AND         0  VIOLATED CONSTRAINTS

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.49384E+04 OBJECTIVE =  5.49331E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.49384E+04   OBJECTIVE =  5.49331E+04


MAXIMUM CONSTRAINT VALUE = 1.00003E-03 IS CONSTRAINT       591

THERE ARE      1000 DISCRETE VALUES
```

```
-- OPTIMIZATION IS COMPLETE

NUMBER OF UNCONSTRAINED MINIMIZATIONS =    6

CONSTRAINT TOLERANCE, CT =-3.00000E-02   CTMIN = 3.00000E-03

THERE ARE       832 ACTIVE CONSTRAINTS AND         0 VIOLATED CONSTRAINTS

THERE ARE         2 ACTIVE SIDE CONSTRAINTS

THERE ARE      1000 DISCRETE VALUES

TERMINATION CRITERIA

RELATIVE CONVERGENCE CRITERION WAS MET FOR  4 CONSECUTIVE ITERATIONS

ABSOLUTE CONVERGENCE CRITERION WAS MET FOR  4 CONSECUTIVE ITERATIONS


-- OPTIMIZATION RESULTS


OBJECTIVE, F(X) =    5.49331E+04

MAXIMUM CONSTRAINT VALUE =  1.00003E-03  IS CONSTRAINT NUMBER       591

FUNCTION CALLS =    1668

GRADIENT CALLS =     173
FINAL OBJ =   54933.0517222882
ELAPSED TIME   26.6282552083333       MINUTES
```

4.4 Topology Optimization

The GENESIS structural analysis and optimization program from VR&D will optimize very large structures using what is known as approximation concepts. When solving topology optimization problems, the number of design variables can become very large.

Here, a classical problem known as the Mitchell truss is designed using GENESIS.

The goal is to find the structure supported by the round bar and subject to a single point load on the right, as shown in Figure 4-3. The initial design is a planar structure made up of over 47,000 quadrilateral plate elements filling the design region. GENESIS treats the density of each element as an independent design variable, where the “density factor” can range from zero to one. In this case symmetry was imposed about the horizontal mid-plane for the left half of the structure, leaving a total of just over 35,000 independent design variables. Young’s modulus is linked to the density, so as the density approaches zero, so does the stiffness. We wish to find the optimum structure which is as stiff as possible while using only 10% of the original material.

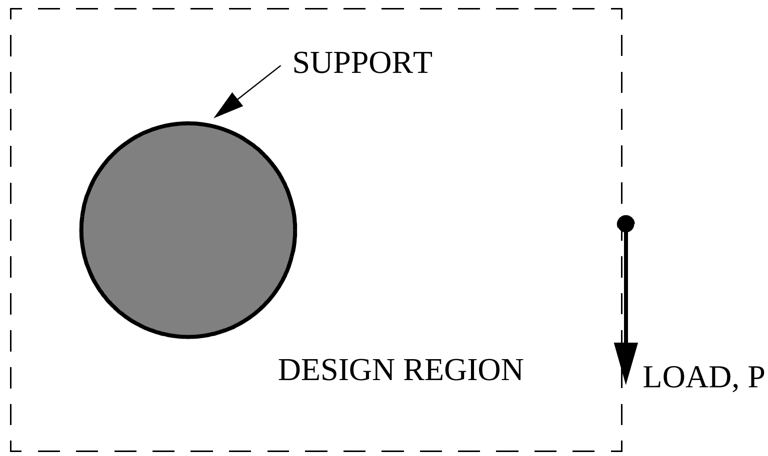


Figure 4-3Topology Design Space

Figure 4-4 shows the optimum topology obtained by GENESIS. The approximate optimization phase of GENESIS required approximately seven percent of the total run time.

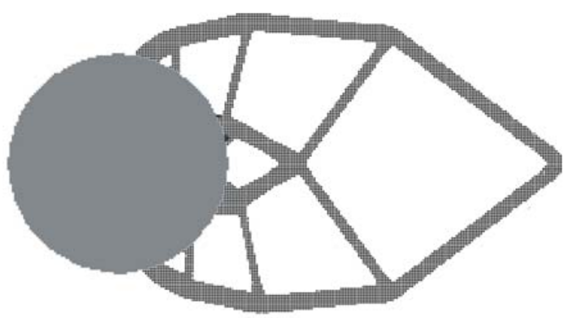


Figure 4-4Topology Results

CHAPTER 5

References

5.1 Introduction

There is much to be gained from a review of some of the basic optimization literature. While BIGDOT can often be satisfactorily used by the optimization novice, a better understanding of the theory of optimization can lead to more effective use of the program. The following is a list of publications which may be useful to those seeking a better understanding of numerical optimization.

5.2 References

1. Vanderplaats, G. N., Multidiscipline Design Optimization, Vanderplaats Research & Development, Inc., Colorado Springs, CO, 2007.
2. DOT User’s Manual, Vanderplaats Research & Development, Inc., Colorado Springs, CO.
3. VisualDOC User’s Manual, Vanderplaats Research & Development, Inc., Colorado Springs, CO.
4. GENESIS User's Manuals, Vanderplaats Research & Development, Inc., Colorado Springs, CO.
5. Vanderplaats, G. N., “Very Large Scale Optimization,” Proc. AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA, Sept. 6-1, 2000, AIAA Paper Number 2000-4809.

APPENDIX A

Structure of Program Calling BIGDOT

- **Introduction**
- **Basic Program Organization**
- **Structure of FORTRAN Program Interfacing with BIGDOT**

A.1 Introduction

The program given here may be used as a prototype as a main calling program for using BIGDOT. All default parameters are defined prior to calling BIGDOT. The defaults are contained in the **RPRMBD** and **IPRMBD** arrays. These are normally initialized to zero. Then, any over-ride values are set in the proper locations. For detailed information about using BIGDOT with application programs, see Chapters 2 and 3 of this manual.

A.2 Basic Program Organization

NOTE: In the following outline of the program, a question mark (**?**) means that a value or values must be provided. Also, the parameters, such as NDV, given in the dimension statement are required values and must be replaced by actual numbers. Each place where a parameter must be supplied by the user is highlighted in ***italic***. Remember that the arrays can be dimensioned larger than the required values to allow for future expansion. Thus, the parameter BIG for **WK** and **IWK**, means that these arrays must be dimensioned at least large enough to solve the problem, but may be dimensioned larger. It is good practice to dimension these arrays as large as possible to allow for future expansion of the number of design variables and constraints.

This sample program stores gradients in the **WK** array. If it is desired to store gradients in an unformatted file to be read by BIGDOT, please refer to Section 3.3.2 for directions.

A.3 Structure of FORTRAN Program Interfacing with BIGDOT

```
      DIMENSION X(NDV),XL(NDV),XU(NDV),G(NCON),WK(BIG),IWK(BIG),
     * RPRMBD(20),IPRMBD(20)
C     DIMENSIONS OF WK AND IWK.
      NRWK=?
      NRIWK=?
C     ZERO RPRMBD AND IPRMBD
      do 10 I=1,20
         RPRMBD(I)=0.0
         IPRMBD(I)=0
10    CONTINUE
C     AT THIS POINT SET ANY ENTRIES OF RPRMBD AND IPRMBD
C     TO THEIR DESIRED VALUES IF THE DEFAULTS ARE
C     TO BE OVER-RIDDEN.
C     E.G.
C     RPRMBD(1)=5.0
C     DEFINE NDV, NCON, IPRINT, MINMAX
      NDV=?
      NCON=?
      IPRINT=?
      MINMAX=?
C     DEFINE X, XL, XU
      X(I)=?, I=1,NDV
      XL(I)=?, I=1,NDV
      XU(I)=?, I=1,NDV
C     READY TO OPTIMIZE
      INFO=0
      END
20    CALL BIGDOT (INFO,IPRINT,NDV,NCON,X,XL,XU,
     * OBJ,MINMAX,G,RPRMBD,IPRMBD,WK,NRWK,IWK,NRIWK)
C     EVALUATE OBJECTIVE AND CONSTRAINTS OR GRADIENTS.
C     YOU MAY CALL ONE OR MORE SUBROUTINES TO DO THIS.
      IF(INFO.LE.1) THEN
C     EVALUATE OBJECTIVE AND CONSTRAINTS
         OBJ=?
         G(I)=?, I=1,NCON
         IF(INFO.EQ.1) GO TO 20
      ENDIF
      IF(INFO.EQ.2) THEN
C     EVALUATE GRADIENTS
         WL(I)=GRADIENT OF OBJ, I=1,NDV
         NGT=IPRMBD(20)
         NN=NDV
         DO 30 I=1,NGT
            IGRAD=IWK(I)
            WK(I)=GRADIENT OF G(IGRAD), I=NN+1,NN+NDV
            NN=NN+NDV
30       CONTINUE
         GO TO 20
      ENDIF
      IF(INFO.GT.0) GO TO 20
C     OPTIMIZATION IS COMPLETE. OUTPUT RESULTS.
      STOP
```

APPENDIX B

Calculating BIGDOT Array Sizes

- Storage Requirements
- Computational Storage Calculations

B.1 Storage Requirements

Arrays **WK** and **IWK** must be dimensioned in any program that calls BIGDOT. The minimum required dimensions, NRWK and NRIWK, can be calculated using the formulas given here:

NRWK = 12NDV + 3NCON + NDV*NGMAX + 40

If JPENLT = 2X,   NRWK = NRWK+2NCON + NGMAX**2

NRIWK = NDV + NCON + 2*MAX(NDV,NCON) + 81

where NCON = 0 is allowed.

If gradients are to be stored in the WK array, the default value for NGMAX is NCON. However, this can generate very large storage requirements.

BIGDOT will actually run with NGMAX = 1, but if NCON is large, and many constraint become active, this will generate many returns to the main program for gradients since they are calculated only one at a time.

If the available storage, NRWK is known, the largest possible NGMAX can be calculated as

If JPENLT = 1X, $$\mathrm{NGMAX} = \frac{\mathrm{NRWK} - 12\mathrm{NDV} - 3\mathrm{NCON} - 40}{\mathrm{NDV}}$$

If JPENLT = 2X, let

A0=FLOAT(NRWK-12*NDV-5*NCON-40)

A1=FLOAT(NDV)

B=(-A1+SQRT(A1**2-4.0*A0))/2.

Then,

NGMAX = MIN (NCON, INT(B))

In practice, if NRWK is input to BIGDOT, NGMAX will be calculated to be as large as possible, up to NCON.

B.2 Computational Storage Calculations

Alternatively, NRWK and NRIWK can be calculated by SUBROUTINE BDT507, which may be directly called from a user program. The parameter list of SUBROUTINE BDT507 is as follows

SUBROUTINE
BDT507(NDV,NCON,NGMAX,NRWK,NRIWK,MAXINT,JPENLT)

where NDV is the number of design variables and NCON is the number of constraints. NGMAX is the desired number of columns to store gradients of constraints (ideally, NGMAX = NCON). NRWK is the number of rows in the **WK** array, and NRIWK is the number of rows in the **IWK** array. MAXINT is the maximum integer value possible (if MAXINT is input as 0, the default value is 2000000000) and JPENLT is the penalty method parameter.

On input, BDT507 requires NDV, NCON, NGMAX, MAXINT and JPENLT. BDT507 outputs the desired values of NRWK, and NRIWK.

Note that if you call BDT507 with NGMAX = 0, NRWK will be returned with a value needed to store all but the constraint gradients.

If you call BDT507 with NGMAX = 1, NRWK will the returned with the minimum value.

If you call BDT507 with NGMAX = NCON, NRWK will the returned with the maximum value.

The value that BDT507 returns for NRIWK is always the required value, based on the value of NGMAX provided.

Index

A

B

C

D

E

G

I

J

L

M

N

O

P

R

S

T

V