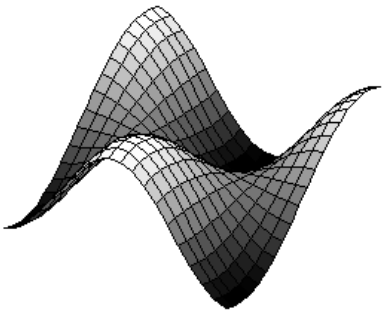


DOT

DESIGN OPTIMIZATION TOOLS

USERS MANUAL

Version 6

Vanderplaats Research & Development, Inc.

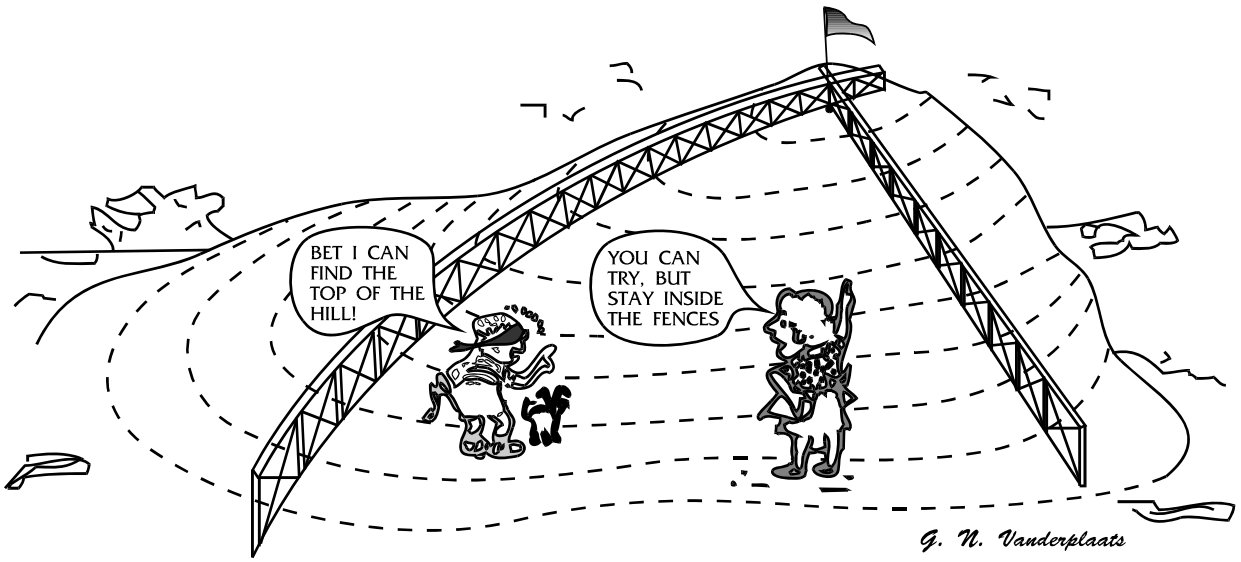
1767 S. 8th Street, Suite 200
Colorado Springs, CO 80905
http://www.vrand.com

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

OTHER VR&D SOFTWARE PRODUCTS

VisualDOC is a main ***D***esign ***O***ptimization ***C***ontrol program which replaces the DOT user's main program to greatly simplify 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), and much more.

GENESIS is a fully integrated finite element analysis and optimization program. Analysis capabilities include statics, normal modes, heat transfer, dynamic response and system buckling. Optimization capabilities include topology, member sizing and shape optimization. GENESIS uses the latest approximation techniques to gain exceptional efficiency in structural optimization. Typically, an optimum design is achieved using about ten detailed finite element analyses, even for design problems with thousands of variables and millions of constraints.



CONSTRAINED OPTIMIZATION

COPYRIGHT NOTICE

WARNING

DISCLAIMER

VR&D makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, VR&D reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of VR&D to notify any person or organization of such revision or change.

TRADEMARKS MENTIONED IN THIS MANUAL

VisualDOC, DOT and GENESIS are registered trademarks of Vanderplaats Research & Development, Inc. All other trademarks are the property of their respective corporations.

Contents

WHAT'S NEW

DOT Version 6

DOT Version 6 represents a major rewrite and enhancement of Version 5. Following is a partial list of enhancements contained in version 6:

- A few minor bugs have been corrected to make the basic optimization capability more robust. Most of these fixes are transparent to the average user, but are important to the overall reliability of the program. We have solved thousands of optimization problems in our quest to make DOT as robust as possible.
- The Golden Section Method is now available for use in the one-dimensional search. After the bounds have been reduced to the desired tolerance, polynomial interpolation is applied for a final refinement. If function evaluations are cheap, tightening the Golden Section tolerance will usually provide a better optimum.
- Major changes were made in the Modified Method of Feasible Directions (METHOD=1) to better follow curved constraints.
- Major changes were made in the Sequential Quadratic Programming Method, including addition of the Golden Section Method in the one-dimensional search.
- A SIMPLEX algorithm with built in upper bounds has been added to the Sequential Linear Programming Method (METHOD=2). In the past, the Modified Method of Feasible Directions was used to solve the linear sub-problem.
- Significant enhancements have been made to the Sequential Quadratic Programming algorithm (METHOD=3) to improve efficiency and robustness.
- Memory allocation has been greatly simplified.
- Subroutine DOT510, which estimates memory requirements, is still available to provide desired and maximum memory requirements. DOT510 no longer provides a minimum memory value since this unnecessarily restricted memory.
- Print levels have been modified to reduce the amount of output for lower IPRINT values.

CHAPTER 1

Introduction

- Introduction
- What You Will Find in this Manual
- DOT System Requirements
- Installing DOT on Your Computer
- Getting Started
- Getting Familiar with DOT
- What DOT Does
- The General Optimization Problem
- Equality Constraints
- Special Notes

1.1 Introduction

Welcome to VR&D's Design Optimization Tools, DOT. The DOT program is intended to help you solve a wide variety of nonlinear constrained or unconstrained optimization problems.

Optimization concepts and methods are not new. Indeed, optimization is fundamental to most of what we do. Whether we are engineers, athletes, or businessmen, our goal is to be best in some way. Numerical optimization, which is the basis for the DOT program, helps us for those cases where we are able to define the optimization problem in a consistent mathematical or numerical way.

The DOT manual is intended for the new user of optimization, as well as the experienced user. In this first chapter, we start by defining the computer requirements and identifying the files that you have received. In Chapter 2, we discuss the use of DOT with your own application programs. This is the real power of DOT, and you are shown how to couple it with your own “analysis” programs to solve sophisticated design tasks. Chapter 3 describes a variety of options available to help you “tune” DOT to work efficiently for your particular application. Finally, Chapter 4 offers examples to help you gain familiarity with DOT and to insure that it is working properly. Chapter 5 lists several references for further study.

Appendices are provided to assist you with specific questions and capabilities provided.

1.2 What You Will Find in this Manual

This manual has been written with the first time user in mind. As such, the first two chapters are intended to assist in using the program right away.

This chapter first defines system requirements and lists the distribution files that you have received. The remainder of this chapter is an introduction to optimization itself. Chapter 2 is all about interfacing DOT with user-supplied application programs for powerful optimization capabilities. Chapter 3 discusses advanced uses of DOT such as over-riding internal parameters, directly supplying gradients of the objective function and constraints, interrupting and restarting DOT, and writing output to a special file for later use. Chapter 4 presents examples of optimization problems taken from a variety of disciplines. 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 DOT. Appendix B defines storage requirements and Subroutine DOT510, which calculates the required working storage values of NRWK and NRIWK.

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 or access VR&D on the web at www.vrand.com.

1.3 DOT System Requirements

Versions of DOT are available for all levels of computers. DOT is provided as Object code.

1.4 Installing DOT on Your Computer

The CD-ROM or DVD you have received contains both DOT and VisualDOC. If you have received the software on a DVD, the GENESIS (and Design Studio) structural optimization software is also available. Alternatively, you may have downloaded the software from the VR&D web site. You are welcome to install VisualDOC and GENESIS also. This software will operate in a restricted mode immediately. On request, we can provide you with full capabilities of all VR&D software for a short term evaluation.

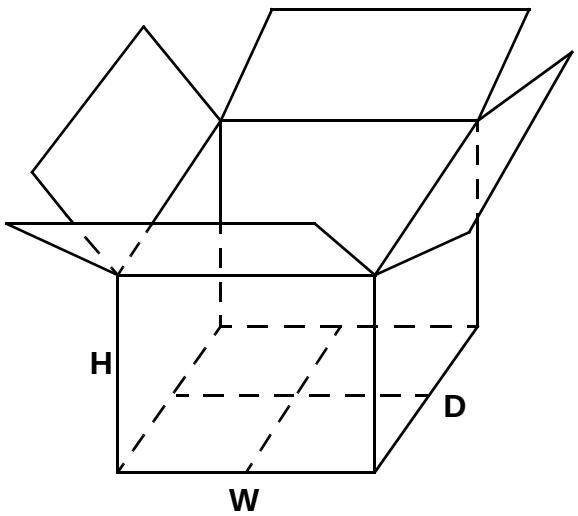
See the readme.txt file on the CD-ROM or DVD for the latest installation information.

1.5 Getting Started

DOT is a computer program for optimization. Specifically, it is used to automatically adjust parameters to maximize or minimize a calculated quantity while satisfying a multiple number of constraints. Functions considered by DOT may be linear or nonlinear and may be very complicated implicit functions of the design variables. That is, you do not need to provide explicit equations to define the responses in terms of the design variables.

DOT is coupled with your application program by writing a small interface. DOT is a tool to solve system optimization problems in engineering, business, social science and any other applications where system responses are analyzed and evaluated numerically.

For example, suppose you received a large quantity order to produce cardboard boxes. The order specifies that the volume of the boxes must be greater or equal to 2 cubic feet, and both top and bottom surfaces have double flaps, as shown in the figure below.



Since one of the large cost items is the amount of cardboard, we need to decide the box dimensions to minimize the amount of cardboard used to make each box. A minimum material box will also be the lightest weight design, thus helping to reduce shipping costs.

This problem can be expressed in the following form.

Find the dimensions W, D, and H which will

Minimize the surface area, S, where

$$S = 2(HW + HD + 2WD) \quad \text{Objective Function} \tag{1-1}$$

Subject to:

$$\text{Volume,} \quad WHD \geq 2.0 \quad \text{Inequality Constraint} \tag{1-2}$$

$$W, H, D \geq 0.0 \quad \text{Side Constraints} \tag{1-3}$$

In this problem, the surface area is the “objective function”. The condition that the volume must be greater than or equal to 2.0 is called a “constraint”. The requirements that W, H, and D be greater than 0.0 are called “side constraints”. This description of an objective function and constraints constitutes the formal optimization problem.

At this point you may realize that to find the optimum design you have to adjust three parameters simultaneously, which is rather difficult to visualize. With DOT, you can solve this problem almost instantly.

Now let's make the problem a bit more realistic. The boxes will be cut from a sheet of cardboard, then folded and glued along one vertical corner. To do this requires an extra 1.25 inch of material of height H. Now the problem becomes

Find the dimensions W, D and H to

Minimize

$$S = 2(HW + HD + 2WD) + \frac{1.25H}{12.0} \tag{1-4}$$

Subject to:

$$\text{Volume,} \quad HWD \geq 2.0 \tag{1-5}$$

$$W, H, D \geq 0.0 \tag{1-6}$$

Note that we have just added 1.25H/12.0 to the objective function where we divide by 12 to convert inches to feet. By doing so the problem has been made more difficult to solve by hand, although DOT solves it just as easily as before.

The method for solving general problems is to write a FORTRAN (or C/C++) program that calls DOT. The form of this program is defined in this manual and any experienced programmer will be able to quickly write such a program for most problems. Appendix A provides a simple program that can be used as a ‘template’ for using DOT.

The methods implemented in DOT are numerical search techniques known as mathematical programming. DOT represents the culmination of many years of research in system optimization by theoretical and applied mathematicians. Basic theories of mathematical programming are well known and many textbooks have been published. However, implementation of theoretical methods into reliable software has required a great deal of research and development. The author of DOT has been actively involved in software development as well as applications of system optimization methods since 1969.

Numerical optimization offers a number of improvements over the traditional approach to decision process and engineering design. Among the advantages of numerical optimization methods are:

- Perform system parameter adjustment far beyond human perception.
- Reduce the time required to make decisions.
- Provide a logical, systematic decision-making procedure.
- Virtually always improve system response, even if not arriving at the absolute optimal state.
- Not biased by intuition or experience. Hence it may produce new, non-traditional results.

The following is a brief list of engineering design projects to which serious application efforts have been made.

- Structural design for minimum weight. The GENESIS program from VR&D is a fully integrated finite element analysis and optimization program. GENESIS uses advanced approximation techniques to solve the optimization task (performed by DOT or BIGDOT) using only about ten detailed finite element analyses, even for very large optimization tasks.
- Aerodynamic design for maximum performance.
- Injection molding for uniform fill times.
- Maximum combustion efficiency of an internal combustion engine.
- Mechanical parts design for minimum material or for maximum performance.
- Conceptual aircraft, spacecraft and ship design.

Of course, applications are not limited to engineering design. Any system design or management problem involving numerical decisions may be cast in a form where numerical optimization is useful. Applications are limited only by the creativity of the user.

CAUTION

DOT solves the nonlinear optimization problem iteratively. It is designed to get a “near optimum” solution quickly, since in most practical problems a precise optimum (which takes much more computational effort) is not that meaningful. Therefore, you should not expect precise mathematical solutions. Chapter 4 gives examples of the differences between the theoretical optimum and that calculated by DOT for several cases. Usually, the difference is minor.

1.6 Getting Familiar with DOT

Now let's solve the simple box design of Section 1.5 using DOT. For convenience, the basic problem is restated here;

Minimize the surface area, S, where

$$S = 2(HW + HD + 2WD) \qquad \text{Objective Function} \qquad (1\text{-}7)$$

Subject to:

$$\text{Volume,} \quad WHD \geq 2.0 \qquad \text{Inequality Constraint} \qquad (1\text{-}8)$$

$$W, H, D \geq 0.0 \qquad \text{Side Constraints} \qquad (1\text{-}9)$$

The design variables, H, W and D are stored in the "Vector of Design Variables," **X**, so;

$$\mathbf{X} = \begin{Bmatrix} H \\ W \\ D \end{Bmatrix} \qquad (1\text{-}10)$$

We have three design variables, so NDV=3.

The objective function is, OBJ=S, so;

OBJ = 2(HW + HD + 2WD)

The inequality constraint is stored in the "Constraint Vector" **G** so;

G(1) = 1.0 - 0.5*X(1)*X(2)*X(3)

Note that the constraint G(1) is normalized by dividing by the bound (2.0) and converting to a non-positive inequality.

For this example, there is only one constraint, so NCON=1.

The side constraints are stored in the "Lower Bound Vector," **XL**, so;

$$\mathbf{XL} = \begin{Bmatrix} 0.0 \\ 0.0 \\ 0.0 \end{Bmatrix}$$

Because no upper bounds are prescribed, we will simply set the upper bounds on the design variables as 100.0 so the "Upper Bound Vector," **XU**, becomes;

$$\mathbf{XU} = \begin{Bmatrix} 100.0 \\ 100.0 \\ 100.0 \end{Bmatrix}$$

It is necessary to provide DOT with an initial design, so we will initially set the entries of vector **X** to unity for this example;

$$\mathbf{X} = \begin{Bmatrix} 1.0 \\ 1.0 \\ 1.0 \end{Bmatrix}$$

Note that this gives an initial volume of 1.0 ft.^{3}, which violates the minimum volume constraint of 2.0. This is acceptable in DOT and DOT will proceed to find a "feasible" optimum.

The vectors **X**, **G**, **XL** and **XU** are the basic arrays used by DOT to contain the user design information.

In this example, we could dimension **X**, **XL** and **XU** to 3 and **G** to 1, but will dimension each of them to 5. This is just to demonstrate that the required dimensions given in this manual are minimum dimensions. You may dimension arrays larger than the required value to allow for increasing the problem size in the future.

Additionally, the arrays **RPRM**(20) and **IPRM**(20) contain various control parameters. By initializing the contents to these arrays to zero, DOT will use all default parameters (defined in Chapter 3).

Also, arrays **WK** and **IWK** are used to store internal real and integer tables. These arrays are normally dimensioned rather large to insure sufficient memory is allocated. The user can call SUBROUTINE DOT510 to calculate the needed dimensions of the arrays **WK** and **IWK**.

Finally, we must tell DOT the number of design variables (NDV), the number of constraints (NCON), a print control parameter (IPRINT), whether to minimize or maximize (MINMAX=0 or -1 to minimize; +1 to maximize), the dimension of array **WK** (NRWK) and **IWK** (NRIWK) and an information parameter INFO. Initially, INFO=0. On return from DOT, if INFO=0, optimization is complete. **X** and **G** contain the optimum values of the design variables and constraints and OBJ is the optimum value of the objective function. If INFO=1, we calculate the value of the objective, OBJ, and constraints, **G**(I), I=1,NCON and call DOT again. In special cases, if we are providing gradient information to DOT, INFO=2 will be returned (if we use all default parameters, INFO will never equal 2, and DOT will calculate the gradients by finite difference methods). The parameter, METHOD, defines the optimization method to be used in DOT. Presently three methods are available for constrained optimization; METHOD=0 or 1 says use the Modified Method of Feasible Directions (MMFD), METHOD=2 says use the Sequential Linear Programming (SLP) method and METHOD=3 says use the Sequential Quadratic Programming (SQP) method. Two methods are available for unconstrained optimization (NCON=0). METHOD=0 or 1 says use the BFGS (Broydon-Fletcher-Goldfarb-Shanno) method and METHOD=2 says use the Fletcher-Reeves method.

It is required that we provide a main program to define the various information and call DOT, as well as a subroutine to calculate the objective and constraint functions in terms of **X**. The overall process is defined by the following 7 steps.

1. Dimension the required arrays and define the dimensions of **WK** and **IWK**.
2. Initialize the control parameters contained in **RPRM** and **IPRM** (normally set these to zero to use the default values; see Chapter 3).
3. Define the number of design variables and constraints, print control, method to be used, and whether to minimize or maximize.
4. Set the initial values of the design variables in **X**, the lower bounds, **XL**, and the upper bounds, **XU**.
5. Initialize the information parameter, INFO=0.
6. Call DOT to proceed with optimization.
7. On return from DOT, if INFO=0, terminate; the optimization process is complete. Otherwise, evaluate the objective and constraint functions and call DOT again. Eventually, DOT will return a value of INFO=0 to indicate that optimization is complete.

The FORTRAN listing for the box design problem is given below, along with a subroutine called EVAL which evaluates the functions (the function values could be calculated in the main program if you prefer). Note that arrays **X**, **XL**, **XU** and **G** are dimensioned larger than needed. It is only necessary to dimension these arrays large enough for the problem, but we can make them larger to allow for future expansion.

In this example, we used METHOD=1 (the MMFD method), and print control, IPRINT=1. The resulting output from DOT is also presented below.

From this simple example, we see that it is extremely easy to use DOT for optimization. Of course, the number of design variables and constraints, as well as the routine for evaluating the functions, may be quite large. If you simply couple DOT with your analysis, we suggest limiting the number of design variables to about 1000, although much larger problems have been solved. The number of constraints may be quite large (especially with METHOD=1 for constrained problems). Problems with over a million constraints have been solved with DOT. For optimization tasks in excess of about 2000 design variables, it is recommended that the BIGDOT optimizer be used [18].

BOX DESIGN FORTRAN PROGRAM

```
C
C     SAMPLE PROGRAM.  BOX DESIGN.
C
      DOUBLE PRECISION X(5),XL(5),XU(5),G(5),WK(800),RPRM(20),OBJ
      INTEGER IWK(200),IPRM(20),I,METHOD,NDV,NCON,IPRINT,MINMAX,INFO,
     *NRWK,NRIWK
C     DEFINE NRWK, NRIWK.
      NRWK=800
      NRIWK=200
C     ZERO RPRM AND IPRM.
      DO 10 I=1,20
         RPRM(I)=0.0
10       IPRM(I)=0
C     DEFINE METHOD,NDV,NCON.
      METHOD=1
C     THREE DESIGN VARIABLES.
      NDV=3
C     ONE CONSTRAINT
      NCON=1
C     DEFINE BOUNDS AND INITIAL DESIGN.
      DO 20 I=1,NDV
C     INITIAL VALUES.
         X(I)=1.0
C     LOWER BOUNDS.
         XL(I)=0.01
C     UPPER BOUNDS
20       XU(I)=100.
C     DEFINE IPRINT, MINMAX, INFO.
C     PRINT CONTROL.
      IPRINT=1
C     MINIMIZE
      MINMAX=-1
C     INITIALIZE INFO TO ZERO.
      INFO=0
C     OPTIMIZE.
100   CALL DOT (INFO,METHOD,IPRINT,NDV,NCON,X,XL,XU,
     *OBJ,MINMAX,G,RPRM,IPRM,WK,NRWK,IWK,NRIWK)
C     FINISHED?
      IF(INFO.EQ.0) STOP
C     EVALUATE OBJECTIVE AND CONSTRAINT.
      CALL EVAL(OBJ,X,G)
C     GO CONTINUE WITH OPTIMIZATION.
      GO TO 100
      END
      SUBROUTINE EVAL (OBJ,X,G)
C     SUBROUTINE TO EVALUATE THE OBJECTIVE FUNCTION AND CONSTRAINTS
C     FOR THE BOX DESIGN PROBLEM.
      DOUBLE PRECISION X(*),G(*),OBJ
      OBJ=2.0*X(2)*X(1)+2.0*X(3)*X(1)+4.0*X(2)*X(3)
      G(1)=1.0-0.5*X(1)*X(2)*X(3)
      RETURN
      END
```

DOT OUTPUT

```
DDDDD        OOOOO       TTTTTTT
D    D      O     O         T
D    D  ==  O  *  O  ==     T
D    D      O     O         T
DDDDD        OOOOO          T

   DESIGN OPTIMIZATION TOOLS


     (C) COPYRIGHT, 2010

       VANDERPLAATS R&D

ALL RIGHTS RESERVED, WORLDWIDE

         VERSION 6.0


CONTROL PARAMETERS

OPTIMIZATION METHOD,               METHOD =       1
NUMBER OF DECISION VARIABLES,         NDV =       3
NUMBER OF CONSTRAINTS,               NCON =       1
PRINT CONTROL PARAMETER,           IPRINT =       1
GRADIENT PARAMETER,                 IGRAD =       0
  FORWARD DIFFERENCE GRADIENTS ARE CALCULATED BY DOT
THE OBJECTIVE FUNCTION WILL BE MINIMIZED


-- SCALAR PROGRAM PARAMETERS

REAL PARAMETERS
  1) CT     = -3.00000E-02           9) FDCH   =  1.00000E-03
  2) CTMIN  =  3.00000E-03          10) FDCHM  =  1.00000E-04
  3) DABOBJ =  8.00000E-04          11) RMVLMZ =  5.00000E-01
  4) DELOBJ =  1.00000E-03          12) DABSTR =  8.00000E-04
  5) DOBJ1  =  1.00000E-01          13) DELSTR =  1.00000E-03
  6) DOBJ2  =  1.60000E+00          14) GSTOL  =  2.50000E-01
  7) DX1    =  1.00000E-02          15) GSTOLM =  1.00000E-04
  8) DX2    =  2.00000E-01

INTEGER PARAMETERS
  1) IGRAD  =      0    6) NGMAX  =      1   11) NONE   =      0
  2) ISCAL  =      3    7) IGMAX  =      1   12) NONE   =      0
  3) ITMAX  =    250    8) JTMAX  =    100   13) JWRITE =      0
  4) ITRMOP =      2    9) ITRMST =      2   14) NONE   =      0
  5) IWRITE =      6   10) JPRINT =      0   15) NSTORE =     694
```

```
STORAGE REQUIREMENTS
  ARRAY  DIMENSION    MINIMUM    MAXIMUM      USED
    WK        800        144        144        800
   IWK        200        102


  -- INITIAL VARIABLES AND BOUNDS

  LOWER BOUNDS ON THE DECISION VARIABLES (XL-VECTOR)
       1)   1.00000E-02   1.00000E-02   1.00000E-02

  DECISION VARIABLES (X-VECTOR)
       1)   1.00000E+00   1.00000E+00   1.00000E+00

  UPPER BOUNDS ON THE DECISION VARIABLES (XU-VECTOR)
       1)   1.00000E+02   1.00000E+02   1.00000E+02


  -- INITIAL FUNCTION VALUES

  OBJ =   8.0000

  CONSTRAINT VALUES (G-VECTOR)
       1)   5.00000E-01

  GMAX =  5.0000E-01  JGMAX =         1


  -- OPTIMIZATION IS COMPLETE

  NUMBER OF ITERATIONS =    5

  CONSTRAINT TOLERANCE, CT = -3.00000E-03

  THERE ARE       1 ACTIVE CONSTRAINTS AND       0 VIOLATED CONSTRAINTS
  CONSTRAINT NUMBERS
           1

  THERE ARE       0 ACTIVE SIDE CONSTRAINTS

  TERMINATION CRITERIA

  RELATIVE CONVERGENCE CRITERION WAS MET FOR  2 CONSECUTIVE ITERATIONS
```

```
-- OPTIMIZATION RESULTS


OBJECTIVE, F(X) =    1.19930E+01


DECISION VARIABLES, X

   ID         XL             X            XU
    1     1.00000E-02   2.01661E+00   1.00000E+02
    2     1.00000E-02   9.95420E-01   1.00000E+02
    3     1.00000E-02   9.95420E-01   1.00000E+02


CONSTRAINTS, G(X)

     1)   9.08397E-04

GMAX =  9.0840E-04  JGMAX =         1

FUNCTION CALLS =      63
```

1.7 What DOT Does

The DOT program uses numerical search methods to seek a minimum or maximum value of one function subject to limits on others. Numerical optimization methods such as this are formally known as Mathematical Programming Techniques. The functions involved must be calculated as functions of the “design variables.” If you wish to maximize a function, DOT internally does this by minimizing the negative of that function, but this will not be apparent from the printed output.

The user must specify an initial set of design variables (also called decision variables), and must provide the necessary code(s) to evaluate the objective and constraint functions each time the design variables are changed by DOT. Section 1.8 gives the general form of the optimization problem and Chapters 2 and 3 provide the detailed information needed to use DOT to solve a particular optimization problem.

The search algorithms used by DOT are described in detail in Reference 1 in Chapter 5. The basic concept is to solve the problem in two steps. The first is to determine a “Search Direction” which defines how the design variables will be changed. The key idea is that all variables will be changed simultaneously in a fashion that will improve the design. The second part is to determine how far to move in this direction, and this is called a “One-Dimensional Search.” This process of finding a search direction and then searching is called an “iteration” and is repeated until it converges to the optimum. This is the basic approach used by the Modified Method of Feasible Directions contained in DOT for constrained minimization (see reference 2 of Chapter 5). If no constraints are imposed, the minimization is called Unconstrained and DOT uses the Broydon-Fletcher-Goldfarb-Shanno (BFGS) algorithm if METHOD=0 or 1 and the Fletcher-Reeves (F.R.) algorithm if METHOD=2. These algorithms also proceed in the two step process of finding a search direction and performing a one-dimensional search.

When using the Sequential Linear Programming (SLP) or the Sequential Quadratic Programming (SQP) method contained in DOT, the process is modified somewhat, but the basic concept of moving from one design to an improved design is the same.

Chapter 5 provides a list of references for further study.

1.8 The General Optimization Problem

The optimization problem is formally stated as follows [1]:

Minimize or Maximize

$$F(\mathbf{X}) \qquad \text{Objective Function} \qquad (1\text{-}11)$$

Subject to;

$$g_j(\mathbf{X}) \le 0 \qquad j = 1,\text{NCON} \qquad \text{Inequality Constraints} \qquad (1\text{-}12)$$

$$X_i^L \le X_i \le X_i^U \qquad i = 1,\text{NDV} \qquad \text{Side Constraints} \qquad (1\text{-}13)$$

NDV is the number of design (decision) variables. DOT is designed to be a robust numerical optimizer for problems of (typically) up to 2000 design variables. However, there is nothing magic about this number. Problems in excess of two thousand design variables have been solved. Just be aware that larger optimization problems may be more difficult to solve, not to mention more time consuming. The BIGDOT optimization program is capable of solving problems with hundreds of thousands of design variables [18].

It should be noted that structural optimization applications, which use DOT as a subroutine in conjunction with a structural analysis code, often involve hundreds of design variables and thousands of nonlinear inequality constraints. Special methods exist for solving large structural optimization problems [3]. The GENESIS program combines the most advanced methods for structural optimization for design of structures [17].

NCON is the number of constraints in a particular problem. The number of constraints tends to get high for many problems. For example, consider a truss design problem (such as an electrical transmission tower) where each member has stress and buckling constraints, and displacement constraints are imposed at each joint. Also, the truss must support many independent load cases. For a large truss, there would be a great many constraints. Nevertheless, there is no maximum number of constraints to keep in mind.

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 mechanical part that has a negative thickness. Side constraints are allowed when solving unconstrained problems (NCON=0).

It is necessary to formulate optimization problems in this standard form. In Chapter 4, several examples from engineering and management are presented. These examples may be used as a guide to properly formulating optimization problems.

1.9 Equality Constraints

The formal problem statement of Section 1.8 includes only inequality constraints, which require a set of functions $g_j(\mathbf{X})$ to be less than or equal to zero. Suppose you want to specify that a function (or functions) must be equal to zero at the optimum. This is done by defining two separate inequality constraints for each function. One constraint requires the function to be less than or equal to zero and the other constraint requires the function to be greater than or equal to zero (in standard form, the negative of the function be less than or equal to zero). The only function value that satisfies both constraints is zero, which is just what an equality constraint requires. A simple example will demonstrate this.

Suppose you want the following relation to be true

$$-X_1^3 - X_2^2 + X_3 = 2.0$$

or, in the form that an optimizer understands,

$$-X_1^3 - X_2^2 + X_3 - 2.0 = 0.0$$

This requirement can be satisfied by imposing two equal and opposite inequality constraints as

$$g_1 = -X_1^3 - X_2^2 + X_3 - 2.0 \le 0.0$$

$$g_2 = -(-X_1^3 - X_2^2 + X_3 - 2.0) \le 0.0$$

That is;

$$g_2 = -g_1 \le 0.0$$

At the optimum, g_1 and g_2 should both equal zero within a small tolerance.

NOTE: In this case, an alternative would be to treat X_3 as a dependent variable and reduce the total number of independent design variables by one. Thus

$$X_3 = 2.0 + X_1^3 + X_2^2$$

Now, only X_1 and X_2 are the design variables. Whenever the optimizer requests function values, you would first calculate X_3 and then evaluate the functions in the usual way. This would reduce the number of design variables by one as well as eliminating both of the constraints. However, this is only possible when we have an explicit relationship such as this. In the general case, we just provide two equal and opposite constraints as we did above.

1.10 Special Notes

- Remember that optimization is iterative and usually nonlinear. You should try to formulate the problem so that the variables and functions are of the same general order of magnitude.
- Always normalize the constraints. For example,

  $Q \leq 20000.0$

  should be normalized as

  $Q/20000.0 - 1.0 \leq 0.0$
- DOT scales the design variables, **X** in an effort to improve the numerical conditioning of the optimization task. While there is no good theory for scaling, it usually works well. If DOT seems not to be working, try turning the scaling off by setting ISCAL = -1. [ISCAL is the internal parameter IPRM(2); see Chapter 3].

CHAPTER 2

DOT with Application Programs

- Introduction
- Methods Used by DOT
- Calling Statement
- Parameters in the Calling Statement
- Compiling and Linking
- A Simple Example

2.1 Introduction

Interfacing DOT 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 DOT in the manner described below.

A simple main program that calls DOT is provided in Appendix A. All that needs to be provided are the parameters and functions. The parameters that must be provided are defined in the following two sections. An example is presented in section 2.6.

2.2 Methods Used by DOT

The following table identifies the methods available. The choice of method is made with the METHOD parameter in the DOT calling statement.

Unconstrained Minimization (NCON = 0)	
METHOD	**DESCRIPTION**
0,1*	Broydon-Fletcher-Goldfarb-Shanno (BFGS) variable metric method. This method is considered best on theoretical grounds.
2	Fletcher-Reeves (F.R.) conjugate gradient method. This method uses very little computer memory and has been found to be reliable.
Constrained Minimization (NCON > 0)	
METHOD	**DESCRIPTION**
0,1*	Modified Method of Feasible Directions (MMFD). This method is reliable and uses the least computer memory.
2	Sequential Linear Programming (SLP). This method is often most efficient for general applications in terms of the number of function evaluations required, especially if there are as many active constraints as there are design variables at the optimum.
3	Sequential Quadratic Programming (SQP). This method is considered theoretically best if the optimization problem is “well conditioned.” This method should always be tried for your problem. If it works successfully, it is almost always the most efficient.

* Default Method.

It is generally agreed that there is no “Best” method for all applications. The user is encouraged to try all available methods on several applications. The method that performs best will usually be good for many optimization problems in this class.

2.3 Calling Statement

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

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

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

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

If you wish to call DOT from a C/C++ program, see Appendix A for a sample program.

2.4 Parameters in the Calling Statement

Table 2-1 lists the parameters in the calling statement to DOT. 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 DOT Argument List

PARAMETER	DEFINITION
INFO	Information parameter. Before calling DOT the first time, set INFO=0. When control returns from DOT to the calling program, INFO will normally have a value of 0 or 1. 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 DOT again. A third possibility, INFO= 2, exists also. In this case, the user must provide gradient information. This is an advanced feature and is described in Chapter 3. NOTE: If IPRM(18)>0 on return from DOT, a Fatal Error has occurred (See Chapter 3).

METHOD	Optimization method to be used. METHOD = 0 or 1 means use the modified method of feasible directions. METHOD = 2 means use the sequential linear programming method. METHOD = 3 means use the sequential quadratic programming method. If the problem in unconstrained (NCON=0), the BFGS algorithm will be used if METHOD=0 or 1 and the Fletcher-Reeves algorithm will be used if METHOD=2.
IPRINT	Print control parameter. IPRINT = 0 no output. IPRINT = 1 internal parameters, initial information and results. IPRINT = 2 same plus objective function and maximum constraint value (and constraint number) at each iteration as well as the number of active (NAC) and violated (NVC) constraints and the number of active side constraints (NACS). IPRINT = 3 same plus X-vector, G-vector and critical constraint numbers. IPRINT = 4 same plus gradients. IPRINT = 5 same plus search direction. IPRINT = 6 same plus set basic one-dimensional search information. IPRINT = 7 same plus detailed one-dimensional search information. NOTE: The IPRM Array contains additional print options. See Chapter 3.
NDV	Number of design (decision) variables contained in vector **X**.
NCON	Number of constraint values contained in array **G**. NCON=0 is allowed.
X(NDV)	Vector containing the design variables. On the first call to DOT, this is the user's best guess for the design. On the final return from DOT (INFO=0 is returned), the vector **X** contains the optimum design and vector **G** contains the corresponding constraint values. OBJ is the optimum objective function value.

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's -1.0E**+**15 and not -1.0E**-**15 (+15, not -15 exponent). Lower bounds may be used even for unconstrained (NCON=0) problems.
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. Upper bounds may be used even for unconstrained (NCON=0) problems.
OBJ	Value of the objective function corresponding to the current values of the design variables contained in **X**. On the first call to DOT, OBJ need not be defined. DOT will return a value of INFO=1 to indicate that the user must evaluate OBJ and call DOT again. Subsequently, any time a value of INFO=1 is returned from DOT, the objective, OBJ, must be evaluated for the current design and DOT 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 DOT, the constraint values need not be defined. On return from DOT, if INFO=1, the constraints must be evaluated for the current **X** and DOT must be called again. If NCON=0, array **G** must be dimensioned to 1 or larger, but no constraint values need to be provided.
RPRM(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.
IPRM(20)	Array containing the integer control parameters. As with the **RPRM** 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) variables. Array **WK** is used to store internal scalar variables and arrays used by DOT. If the user has not provided enough storage, DOT 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 1000 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) variables. Array **IWK** is used to store internal scalar variables and arrays used by DOT. If the user has not provided enough storage, DOT will print the appropriate message and terminate the optimization.
NRIWK	Dimensioned size of work array **IWK**. A good estimate is 300 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.

Note: The minimum required values of NRWK and NRIWK are defined in Appendix B. Those values are only minimums. The actual dimensions may be larger than this. DOT 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.

SUBROUTINE DOT510 can be called to provide desired and maximum required values of NRWK and NRIWK. See Appendix B for more information on this option.

2.5 Compiling and Linking

DOT is supplied as object code. When DOT is to be called by an application program, this DOT object code must be linked to a main program like the ones in Chapter 1, Chapter 4, or Appendix A.

You may use DOT in single precision (dot.a or dot*.lib files), or double precision (dot2.a or dot2*.lib files). The only difference between the two versions is that the double precision version contains DOUBLE PRECISION statements at the beginning of each routine.

Directions given here assume that you are using the double precision version. If you use the single precision version, the approach is the same.

Specific instructions for how to link DOT object code to the main program vary from system to system. Consult the manual that accompanies your compiler for detailed assistance.

An example of how some systems would compile and link a calling program MAIN.FOR to the DOT object code is as follows:

UNIX Systems:

```
f77 -o MAIN MAIN.FOR dot2.a
```

An executable file called MAIN will be created. To run DOT, simply execute file MAIN. This assumes that "f77" is the command to invoke FORTRAN 77 compiler on your system

PC (windows 95/98/NT/VISTA,Windows 7) with Intel FORTRAN:

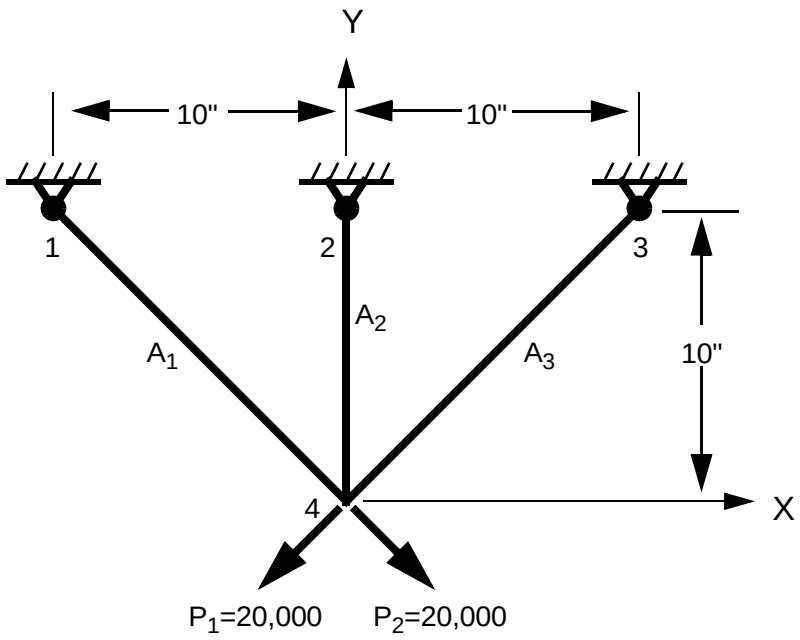
```
f77 /exe:MAIN.EXE MAIN.FOR dot_pc5.lib
```

An executable file called MAIN.EXE will be created. To run DOT, simply execute the file MAIN.EXE.

Sections 1.6 and 2.6, as well as Chapter 4 present example MAIN programs that the user can try immediately after receiving DOT. Appendix A provides a "generic" MAIN program which you may use as a "template" for your own applications.

2.6 A Simple Example

This is the optimization of the 3-bar truss shown below, which is the 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:

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

$$g_2 = \frac{1.0}{X_1 + \sqrt{2}X_2} - 1.0 \le 0.0 \tag{2-3}$$

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

The program used to solve this problem and the results are presented below. Please note that, depending on your computer precision, the results may differ slightly from those given here. This is to be expected since optimization is a nonlinear iterative process. However, your results should be close to those given here. The following pages provide results for METHOD = 1, METHOD = 2 and METHOD = 3. Note that the final values of the design variables are significantly different, although the value of the objective function is very nearly the same for each case. This is because the design space is quite "flat" so that many nearby designs are equally acceptable.

LISTING 2-1: PROGRAM ILLUSTRATING HOW TO USE DOT
WITH A SIMPLE CALLING PROGRAM

```
C
C     SAMPLE PROGRAM. THREE BAR TRUSS.
C
      DOUBLE PRECISION X(2),XL(2),XU(2),G(2),WK(800),RPRM(20),OBJ
      INTEGER IWK(200),IPRM(20),NRWK,NRIWK,I,METHOD,NDV,NCON,IPRINT,
     *MINMAX,INFO
C     DEFINE NRWK, NRIWK.
      NRWK=800
      NRIWK=200
C     ZERO RPRM AND IPRM.
      DO 10 I=1,20
         RPRM(I)=0.0
10       IPRM(I)=0
C     DEFINE METHOD,NDV,NCON.
      METHOD=1
      NDV=2
      NCON=2
C     DEFINE BOUNDS AND INITIAL DESIGN.
      DO 20 I=1,NDV
         X(I)=1.0
         XL(I)=0.1
20       XU(I)=100.
C     DEFINE IPRINT, MINMAX, INFO.
      IPRINT=3
      MINMAX=-1
      INFO=0
100   CALL DOT (INFO,METHOD,IPRINT,NDV,NCON,X,XL,XU,
     *OBJ,MINMAX,G,RPRM,IPRM,WK,NRWK,IWK,NRIWK)
      IF(INFO.EQ.0)STOP
      CALL EVAL(OBJ,X,G)
      GO TO 100
      END
      SUBROUTINE EVAL (OBJ,X,G)
C     SUBROUTINE TO EVALUATE THE OBJECTIVE FUNCTION AND CONSTRAINTS
C     FOR THE SPRING EQUILIBRIUM PROBLEM.
      DOUBLE PRECISION X(*),G(*),OBJ
      OBJ=2.0*SQRT(2.)*X(1)+X(2)
      G(1)=(2.*X(1)+SQRT(2.)*X(2))/(2.*X(1)*(X(1)+
     *SQRT(2.0)*X(2)))-1.
      G(2)=1./(2.*(X(1)+SQRT(2.)*X(2)))-1.
      RETURN
      END
```

DOT OUTPUT - METHOD=1

```
DDDDD        OOOOO       TTTTTTT
D    D      O     O         T
D    D  ==  O  *  O  ==     T
D    D      O     O         T
DDDDD        OOOOO          T

   DESIGN OPTIMIZATION TOOLS


     (C) COPYRIGHT, 2010

       VANDERPLAATS R&D

ALL RIGHTS RESERVED, WORLDWIDE

         VERSION 6.0
```

```
CONTROL PARAMETERS

OPTIMIZATION METHOD,                METHOD =       1
NUMBER OF DECISION VARIABLES,          NDV =       2
NUMBER OF CONSTRAINTS,                NCON =       2
PRINT CONTROL PARAMETER,             IPRINT =       3
GRADIENT PARAMETER,                   IGRAD =       0
  FORWARD DIFFERENCE GRADIENTS ARE CALCULATED BY DOT
THE OBJECTIVE FUNCTION WILL BE MINIMIZED


-- SCALAR PROGRAM PARAMETERS

REAL PARAMETERS
  1) CT     = -3.00000E-02           9) FDCH   =  1.00000E-03
  2) CTMIN  =  3.00000E-03          10) FDCHM  =  1.00000E-04
  3) DABOBJ =  3.82843E-04          11) RMVLMZ =  5.00000E-01
  4) DELOBJ =  1.00000E-03          12) DABSTR =  3.82843E-04
  5) DOBJ1  =  1.00000E-01          13) DELSTR =  1.00000E-03
  6) DOBJ2  =  7.65685E-01          14) GSTOL  =  2.50000E-01
  7) DX1    =  1.00000E-02          15) GSTOLM =  1.00000E-04
  8) DX2    =  2.00000E-01
```

```
INTEGER PARAMETERS
  1) IGRAD  =      0    6) NGMAX  =      2   11) NONE   =      0
  2) ISCAL  =      2    7) IGMAX  =      1   12) NONE   =      0
  3) ITMAX  =    250    8) JTMAX  =    100   13) JWRITE =      0
  4) ITRMOP =      2    9) ITRMST =      2   14) NONE   =      0
  5) IWRITE =      6   10) JPRINT =      0   15) NSTORE =      702


   STORAGE REQUIREMENTS
ARRAY  DIMENSION    MINIMUM    MAXIMUM      USED
  WK        800        137        137        800
 IWK        200        101


-- INITIAL VARIABLES AND BOUNDS

LOWER BOUNDS ON THE DECISION VARIABLES (XL-VECTOR)
     1)   1.00000E-01   1.00000E-01

DECISION VARIABLES (X-VECTOR)
     1)   1.00000E+00   1.00000E+00

UPPER BOUNDS ON THE DECISION VARIABLES (XU-VECTOR)
     1)   1.00000E+02   1.00000E+02


-- INITIAL FUNCTION VALUES

OBJ =   3.8284

CONSTRAINT VALUES (G-VECTOR)
     1)  -2.92893E-01  -7.92893E-01

GMAX = -2.9289E-01  JGMAX =         1


-- BEGIN CONSTRAINED OPTIMIZATION: MODIFIED METHOD OF FEASIBLE DIRECTIONS
```

```
 -- ITERATION NUMBER    1

 THERE ARE       0 ACTIVE CONSTRAINTS AND       0 VIOLATED CONSTRAINTS
 THERE ARE       2 RETAINED CONSTRAINTS
 CONSTRAINT NUMBERS
          1         2

 THERE ARE       0 ACTIVE SIDE CONSTRAINTS

 OBJECTIVE = 2.79490E+00

 DECISION VARIABLES (X-VECTOR)
      1)   6.75194E-01   8.85164E-01

 CONSTRAINT VALUES (G-VECTOR)
      1)  -2.47075E-06  -7.40530E-01

 NAC  =           0  NVC   =         0  NACS  =         0

 GMAX = -2.4708E-06  JGMAX =         1


 -- ITERATION NUMBER    2

 THERE ARE       1 ACTIVE CONSTRAINTS AND       0 VIOLATED CONSTRAINTS
 THERE ARE       2 RETAINED CONSTRAINTS
 CONSTRAINT NUMBERS
          1         2

 THERE ARE       0 ACTIVE SIDE CONSTRAINTS

 OBJECTIVE = 2.64114E+00

 DECISION VARIABLES (X-VECTOR)
      1)   7.91044E-01   4.03728E-01

 CONSTRAINT VALUES (G-VECTOR)
      1)  -8.17489E-04  -6.32893E-01

 NAC  =           1  NVC   =         0  NACS  =         0

 GMAX = -8.1749E-04  JGMAX =         1


 -- ITERATION NUMBER    3

 THERE ARE       1 ACTIVE CONSTRAINTS AND       0 VIOLATED CONSTRAINTS
 THERE ARE       2 RETAINED CONSTRAINTS
 CONSTRAINT NUMBERS
          1         2

 THERE ARE       0 ACTIVE SIDE CONSTRAINTS
```

```
-- OPTIMIZATION IS COMPLETE

NUMBER OF ITERATIONS =    3

CONSTRAINT TOLERANCE, CT = -3.00000E-02

THERE ARE       1 ACTIVE CONSTRAINTS AND       0 VIOLATED CONSTRAINTS
CONSTRAINT NUMBERS
         1

THERE ARE       0 ACTIVE SIDE CONSTRAINTS

TERMINATION CRITERIA

MAXIMUM K-T RESIDUAL =  0.00000E+00 IS LESS THAN  1.00000E-04

MAXIMUM POSSIBLE MOVE =  9.85798E-06 IS LESS THAN  1.00000E-05


-- OPTIMIZATION RESULTS


OBJECTIVE, F(X) =    2.64114E+00


DECISION VARIABLES, X

   ID         XL             X            XU
    1     1.00000E-01   7.91044E-01   1.00000E+02
    2     1.00000E-01   4.03728E-01   1.00000E+02


CONSTRAINTS, G(X)

     1)  -8.17489E-04  -6.32893E-01

GMAX = -8.1749E-04  JGMAX =         1


LAGRANGE MULTIPLIERS

     1)   2.64646E+00   0.00000E+00

FUNCTION CALLS =      32
```

DOT OUTPUT - METHOD=2

```
DDDDD        OOOOO       TTTTTTT
D    D      O     O         T
D    D  ==  O  *  O  ==     T
D    D      O     O         T
DDDDD        OOOOO          T

   DESIGN OPTIMIZATION TOOLS


     (C) COPYRIGHT, 2010

       VANDERPLAATS R&D

ALL RIGHTS RESERVED, WORLDWIDE

         VERSION 6.0


 CONTROL PARAMETERS

 OPTIMIZATION METHOD,                METHOD =       2
 NUMBER OF DECISION VARIABLES,          NDV =       2
 NUMBER OF CONSTRAINTS,                NCON =       2
 PRINT CONTROL PARAMETER,             IPRINT =       3
 GRADIENT PARAMETER,                   IGRAD =       0
   FORWARD DIFFERENCE GRADIENTS ARE CALCULATED BY DOT
 THE OBJECTIVE FUNCTION WILL BE MINIMIZED


 -- SCALAR PROGRAM PARAMETERS

 REAL PARAMETERS
   1) CT     = -3.00000E-02           9) FDCH   =  1.00000E-03
   2) CTMIN  =  3.00000E-03          10) FDCHM  =  1.00000E-04
   3) DABOBJ =  3.82843E-04          11) RMVLMZ =  5.00000E-01
   4) DELOBJ =  1.00000E-03          12) DABSTR =  3.82843E-04
   5) DOBJ1  =  1.00000E-01          13) DELSTR =  1.00000E-03
   6) DOBJ2  =  7.65685E-01          14) GSTOL  =  1.00000E-02
   7) DX1    =  1.00000E-02          15) GSTOLM =  1.00000E-04
   8) DX2    =  2.00000E-01
```

```
INTEGER PARAMETERS
  1) IGRAD  =      0    6) NGMAX  =      2   11) NONE   =      0
  2) ISCAL  =     -1    7) IGMAX  =      1   12) NONE   =      0
  3) ITMAX  =    250    8) JTMAX  =    100   13) JWRITE =      0
  4) ITRMOP =      2    9) ITRMST =      2   14) NONE   =      0
  5) IWRITE =      6   10) JPRINT =      0   15) NSTORE =      670


   STORAGE REQUIREMENTS
ARRAY  DIMENSION    MINIMUM    MAXIMUM      USED
  WK        800        169        169        800
 IWK        200        129


-- INITIAL VARIABLES AND BOUNDS

LOWER BOUNDS ON THE DECISION VARIABLES (XL-VECTOR)
     1)   1.00000E-01   1.00000E-01

DECISION VARIABLES (X-VECTOR)
     1)   1.00000E+00   1.00000E+00

UPPER BOUNDS ON THE DECISION VARIABLES (XU-VECTOR)
     1)   1.00000E+02   1.00000E+02


-- INITIAL FUNCTION VALUES

OBJ =   3.8284

CONSTRAINT VALUES (G-VECTOR)
     1)  -2.92893E-01  -7.92893E-01

GMAX = -2.9289E-01  JGMAX =         1
```

```
 -- BEGIN CONSTRAINED OPTIMIZATION: SEQUENTIAL LINEAR PROGRAMMING METHOD


 -- BEGIN SLP CYCLE    1     RELATIVE MOVE LIMIT = 0.50000

 APPROXIMATE FUNCTIONS

 OBJ = 2.20591E+00

 CONSTRAINT VALUES (G-VECTOR)
      1)   6.93889E-18  -6.98237E-01

 GMAX =  6.9389E-18  JGMAX =         1

 CALCULATED FUNCTIONS

 OBJ  = 2.20591E+00

 DECISION VARIABLES (X-VECTOR)
      1)   6.03130E-01   5.00000E-01

 CONSTRAINT VALUES (G-VECTOR)
      1)   2.10619E-01  -6.18390E-01

 NAC  =         0    NVC   =         1  NACS  =         0

 GMAX =  2.1062E-01  JGMAX =         1


 -- BEGIN SLP CYCLE    2     RELATIVE MOVE LIMIT = 0.50000

 APPROXIMATE FUNCTIONS

 OBJ = 2.45123E+00

 CONSTRAINT VALUES (G-VECTOR)
      1)   1.38778E-16  -5.27859E-01

 GMAX =  1.3878E-16  JGMAX =         1

 CALCULATED FUNCTIONS

 OBJ  = 2.45123E+00

 DECISION VARIABLES (X-VECTOR)
      1)   8.22445E-01   1.25000E-01

 CONSTRAINT VALUES (G-VECTOR)
      1)   1.08333E-01  -4.99610E-01

 NAC  =         0    NVC   =         1  NACS  =         0

 GMAX =  1.0833E-01  JGMAX =         1
```

```
-- BEGIN SLP CYCLE    3     RELATIVE MOVE LIMIT = 0.50000

APPROXIMATE FUNCTIONS

OBJ = 2.66003E+00

CONSTRAINT VALUES (G-VECTOR)
     1)  -9.71445E-17  -5.69747E-01

GMAX = -9.7145E-17  JGMAX =         1

CALCULATED FUNCTIONS

OBJ  = 2.66003E+00

DECISION VARIABLES (X-VECTOR)
     1)   8.74170E-01   1.87500E-01

CONSTRAINT VALUES (G-VECTOR)
     1)   1.08239E-02  -5.61147E-01

NAC  =         0    NVC   =         1  NACS  =         0

GMAX =  1.0824E-02  JGMAX =         1


-- BEGIN SLP CYCLE    4     RELATIVE MOVE LIMIT = 0.50000

APPROXIMATE FUNCTIONS

OBJ = 2.57625E+00

CONSTRAINT VALUES (G-VECTOR)
     1)   8.32667E-17  -6.45496E-01

GMAX =  8.3267E-17  JGMAX =         1

CALCULATED FUNCTIONS

OBJ  = 2.57625E+00

DECISION VARIABLES (X-VECTOR)
     1)   7.61687E-01   4.21875E-01

CONSTRAINT VALUES (G-VECTOR)
     1)   2.45430E-02  -6.31895E-01

NAC  =         0    NVC   =         1  NACS  =         0

GMAX =  2.4543E-02  JGMAX =         1
```

```
 -- BEGIN SLP CYCLE    5     RELATIVE MOVE LIMIT = 0.50000

 APPROXIMATE FUNCTIONS

 OBJ = 2.62859E+00

 CONSTRAINT VALUES (G-VECTOR)
      1)   2.63678E-16  -5.76299E-01

 GMAX =  2.6368E-16  JGMAX =         1

 CALCULATED FUNCTIONS

 OBJ  = 2.62859E+00

 DECISION VARIABLES (X-VECTOR)
      1)   8.54768E-01   2.10938E-01

 CONSTRAINT VALUES (G-VECTOR)
      1)   1.85759E-02  -5.66378E-01

 NAC  =         0    NVC   =         1  NACS  =         0

 GMAX =  1.8576E-02  JGMAX =         1


 -- BEGIN SLP CYCLE    6     RELATIVE MOVE LIMIT = 0.50000

 APPROXIMATE FUNCTIONS

 OBJ = 2.63394E+00

 CONSTRAINT VALUES (G-VECTOR)
      1)   4.85723E-17  -6.09153E-01

 GMAX =  4.8572E-17  JGMAX =         1

 CALCULATED FUNCTIONS

 OBJ  = 2.63394E+00

 DECISION VARIABLES (X-VECTOR)
      1)   8.19371E-01   3.16406E-01

 CONSTRAINT VALUES (G-VECTOR)
      1)   4.90750E-03  -6.05316E-01

 NAC  =         0    NVC   =         1  NACS  =         0

 GMAX =  4.9075E-03  JGMAX =         1
```

```
-- BEGIN SLP CYCLE    7     RELATIVE MOVE LIMIT = 0.50000

APPROXIMATE FUNCTIONS

OBJ = 2.61855E+00

CONSTRAINT VALUES (G-VECTOR)
     1)  -2.77556E-17  -6.55887E-01

GMAX = -2.7756E-17  JGMAX =         1

CALCULATED FUNCTIONS

OBJ  = 2.61855E+00

DECISION VARIABLES (X-VECTOR)
     1)   7.57997E-01   4.74609E-01

CONSTRAINT VALUES (G-VECTOR)
     1)   9.48069E-03  -6.50153E-01

NAC  =         0    NVC   =         1  NACS  =         0

GMAX =  9.4807E-03  JGMAX =         1

CONSTRAINT VIOLATION IS INCREASED.  MOVE HALF WAY

CALCULATED FUNCTIONS

OBJ  = 2.62624E+00

DECISION VARIABLES (X-VECTOR)
     1)   7.88684E-01   3.95508E-01

CONSTRAINT VALUES (G-VECTOR)
     1)   4.88278E-03  -6.29085E-01

NAC  =         0    NVC   =         1  NACS  =         0

GMAX =  4.8828E-03  JGMAX =         1
```

```
 -- BEGIN SLP CYCLE    8     RELATIVE MOVE LIMIT = 0.25000

 APPROXIMATE FUNCTIONS

 OBJ = 2.63601E+00

 CONSTRAINT VALUES (G-VECTOR)
      1)   9.02056E-17  -6.73304E-01

 GMAX =  9.0206E-17  JGMAX =         1

 CALCULATED FUNCTIONS

 OBJ  = 2.63601E+00

 DECISION VARIABLES (X-VECTOR)
      1)   7.39699E-01   5.43823E-01

 CONSTRAINT VALUES (G-VECTOR)
      1)   7.34427E-03  -6.68607E-01

 NAC  =         0    NVC   =         1  NACS  =         0

 GMAX =  7.3443E-03  JGMAX =         1

 CONSTRAINT VIOLATION IS INCREASED.  MOVE HALF WAY

 CALCULATED FUNCTIONS

 OBJ  = 2.63113E+00

 DECISION VARIABLES (X-VECTOR)
      1)   7.64191E-01   4.69666E-01

 CONSTRAINT VALUES (G-VECTOR)
      1)   4.32869E-03  -6.49958E-01

 NAC  =         0    NVC   =         1  NACS  =         0

 GMAX =  4.3287E-03  JGMAX =         1
```

```
-- BEGIN SLP CYCLE    9     RELATIVE MOVE LIMIT = 0.12500

APPROXIMATE FUNCTIONS

OBJ = 2.63582E+00

CONSTRAINT VALUES (G-VECTOR)
     1)  -8.67362E-17  -6.35112E-01

GMAX = -8.6736E-17  JGMAX =         1

CALCULATED FUNCTIONS

OBJ  = 2.63582E+00

DECISION VARIABLES (X-VECTOR)
     1)   7.86609E-01   4.10957E-01

CONSTRAINT VALUES (G-VECTOR)
     1)   1.19249E-03  -6.34447E-01

NAC  =         1    NVC   =         0  NACS  =         0

GMAX =  1.1925E-03  JGMAX =         1


-- BEGIN SLP CYCLE   10     RELATIVE MOVE LIMIT = 0.12500

APPROXIMATE FUNCTIONS

OBJ = 2.63889E+00

CONSTRAINT VALUES (G-VECTOR)
     1)  -1.71738E-16  -6.31098E-01

GMAX = -1.7174E-16  JGMAX =         1

CALCULATED FUNCTIONS

OBJ  = 2.63889E+00

DECISION VARIABLES (X-VECTOR)
     1)   7.92235E-01   3.98115E-01

CONSTRAINT VALUES (G-VECTOR)
     1)   5.99301E-05  -6.31066E-01

NAC  =         1    NVC   =         0  NACS  =         0

GMAX =  5.9930E-05  JGMAX =         1
```

```
 -- BEGIN SLP CYCLE   11     RELATIVE MOVE LIMIT = 0.12500

 APPROXIMATE FUNCTIONS

 OBJ = 2.63810E+00

 CONSTRAINT VALUES (G-VECTOR)
      1)  -7.97973E-17  -6.45353E-01

 GMAX = -7.9797E-17  JGMAX =         1

 CALCULATED FUNCTIONS

 OBJ  = 2.63810E+00

 DECISION VARIABLES (X-VECTOR)
      1)   7.74359E-01   4.47879E-01

 CONSTRAINT VALUES (G-VECTOR)
      1)   8.70890E-04  -6.44825E-01

 NAC  =         1    NVC   =         0  NACS  =         0

 GMAX =  8.7089E-04  JGMAX =         1


 -- BEGIN SLP CYCLE   12     RELATIVE MOVE LIMIT = 0.06250

 APPROXIMATE FUNCTIONS

 OBJ = 2.63839E+00

 CONSTRAINT VALUES (G-VECTOR)
      1)  -5.89806E-17  -6.37364E-01

 GMAX = -5.8981E-17  JGMAX =         1

 CALCULATED FUNCTIONS

 OBJ  = 2.63839E+00

 DECISION VARIABLES (X-VECTOR)
      1)   7.84361E-01   4.19887E-01

 CONSTRAINT VALUES (G-VECTOR)
      1)   2.61365E-04  -6.37200E-01

 NAC  =         1    NVC   =         0  NACS  =         0

 GMAX =  2.6136E-04  JGMAX =         1
```

```
-- OPTIMIZATION IS COMPLETE

NUMBER OF CONSTRAINED MINIMIZATIONS =   12

CONSTRAINT TOLERANCE, CT =-3.00000E-02

THERE ARE       1 ACTIVE CONSTRAINTS AND       0 VIOLATED CONSTRAINTS
CONSTRAINT NUMBERS
         1

THERE ARE       0 ACTIVE SIDE CONSTRAINTS

TERMINATION CRITERIA

RELATIVE CONVERGENCE CRITERION WAS MET FOR  2 CONSECUTIVE CYCLES


-- OPTIMIZATION RESULTS


OBJECTIVE, F(X) =    2.63839E+00


DECISION VARIABLES, X

   ID         XL             X            XU
    1     1.00000E-01   7.84361E-01   1.00000E+02
    2     1.00000E-01   4.19887E-01   1.00000E+02


CONSTRAINTS, G(X)

     1)   2.61365E-04  -6.37200E-01

GMAX =  2.6136E-04  JGMAX =         1


LAGRANGE MULTIPLIERS

     1)   2.62421E+00   0.00000E+00

FUNCTION CALLS =      39
```

DOT OUTPUT - METHOD=3

```
DDDDD        OOOOO       TTTTTTT
D    D      O     O         T
D    D  ==  O  *  O  ==     T
D    D      O     O         T
DDDDD        OOOOO          T

   DESIGN OPTIMIZATION TOOLS


     (C) COPYRIGHT, 2010

       VANDERPLAATS R&D

ALL RIGHTS RESERVED, WORLDWIDE

         VERSION 6.0


 CONTROL PARAMETERS

 OPTIMIZATION METHOD,                METHOD =       3
 NUMBER OF DECISION VARIABLES,          NDV =       2
 NUMBER OF CONSTRAINTS,                NCON =       2
 PRINT CONTROL PARAMETER,             IPRINT =       3
 GRADIENT PARAMETER,                   IGRAD =       0
   FORWARD DIFFERENCE GRADIENTS ARE CALCULATED BY DOT
 THE OBJECTIVE FUNCTION WILL BE MINIMIZED


 -- SCALAR PROGRAM PARAMETERS

 REAL PARAMETERS
   1) CT     = -3.00000E-02           9) FDCH   =  1.00000E-03
   2) CTMIN  =  3.00000E-03          10) FDCHM  =  1.00000E-04
   3) DABOBJ =  3.82843E-04          11) RMVLMZ =  5.00000E-01
   4) DELOBJ =  1.00000E-03          12) DABSTR =  3.82843E-04
   5) DOBJ1  =  1.00000E-01          13) DELSTR =  1.00000E-03
   6) DOBJ2  =  7.65685E-01          14) GSTOL  =  1.00000E-02
   7) DX1    =  1.00000E-02          15) GSTOLM =  1.00000E-04
   8) DX2    =  2.00000E-01
```

```
INTEGER PARAMETERS
  1) IGRAD  =      0    6) NGMAX  =      2   11) NONE   =      0
  2) ISCAL  =   1000    7) IGMAX  =      1   12) NONE   =      0
  3) ITMAX  =    250    8) JTMAX  =    100   13) JWRITE =      0
  4) ITRMOP =      2    9) ITRMST =      2   14) NONE   =      0
  5) IWRITE =      6   10) JPRINT =      0   15) NSTORE =      687


   STORAGE REQUIREMENTS
ARRAY  DIMENSION    MINIMUM    MAXIMUM      USED
  WK        800        152        152        800
 IWK        200        103


-- INITIAL VARIABLES AND BOUNDS

LOWER BOUNDS ON THE DECISION VARIABLES (XL-VECTOR)
     1)   1.00000E-01   1.00000E-01

DECISION VARIABLES (X-VECTOR)
     1)   1.00000E+00   1.00000E+00

UPPER BOUNDS ON THE DECISION VARIABLES (XU-VECTOR)
     1)   1.00000E+02   1.00000E+02


-- INITIAL FUNCTION VALUES

OBJ =   3.8284

CONSTRAINT VALUES (G-VECTOR)
     1)  -2.92893E-01  -7.92893E-01

GMAX = -2.9289E-01  JGMAX =         1


-- BEGIN CONSTRAINED OPTIMIZATION: SEQUENTIAL QUADRATIC PROGRAMMING METHOD


-- BEGIN SQP CYCLE    1

NAC =       0 NVC =       0 NGT =       2

OBJ  = 2.20808E+00

DECISION VARIABLES (X-VECTOR)
     1)   6.02328E-01   5.04442E-01

CONSTRAINT VALUES (G-VECTOR)
     1)   2.10133E-01  -6.19979E-01

NAC  =           0  NVC   =         1  NACS  =         0

GMAX =  2.1013E-01  JGMAX =         1
```

```
-- BEGIN SQP CYCLE    2

  NAC =       0 NVC =       1 NGT =       2

  OBJ  = 2.45630E+00

  DECISION VARIABLES (X-VECTOR)
       1)   8.21018E-01   1.34107E-01

  CONSTRAINT VALUES (G-VECTOR)
       1)   1.03719E-01  -5.05281E-01

  NAC  =           0  NVC   =         1  NACS  =         0

  GMAX =  1.0372E-01  JGMAX =         1


  -- BEGIN SQP CYCLE    3

  NAC =       0 NVC =       1 NGT =       2

  OBJ  = 2.55979E+00

  DECISION VARIABLES (X-VECTOR)
       1)   7.73237E-01   3.72742E-01

  CONSTRAINT VALUES (G-VECTOR)
       1)   3.11368E-02  -6.15495E-01

  NAC  =           0  NVC   =         1  NACS  =         0

  GMAX =  3.1137E-02  JGMAX =         1


  -- BEGIN SQP CYCLE    4

  NAC =       0 NVC =       1 NGT =       2

  OBJ  = 2.63978E+00

  DECISION VARIABLES (X-VECTOR)
       1)   7.94259E-01   3.93277E-01

  CONSTRAINT VALUES (G-VECTOR)
       1)  -2.31917E-04  -6.29749E-01

  NAC  =           1  NVC   =         0  NACS  =         0

  GMAX = -2.3192E-04  JGMAX =         1
```

```
-- BEGIN SQP CYCLE    5

NAC =       1 NVC =       0 NGT =       2

OBJ  = 2.63932E+00

DECISION VARIABLES (X-VECTOR)
     1)   7.88444E-01   4.09267E-01

CONSTRAINT VALUES (G-VECTOR)
     1)  -1.38342E-04  -6.34299E-01

NAC  =           1  NVC   =         0  NACS  =         0

GMAX = -1.3834E-04  JGMAX =         1


-- BEGIN SQP CYCLE    6

NAC =       1 NVC =       0 NGT =       2

Q.P. SUB-PROBLEM GAVE NULL SEARCH DIRECTION.  CONVERGENCE ASSUMED.


-- OPTIMIZATION IS COMPLETE

NUMBER OF CONSTRAINED MINIMIZATIONS =    6

CONSTRAINT TOLERANCE, CT =-3.00000E-02

THERE ARE       1 ACTIVE CONSTRAINTS AND       0 VIOLATED CONSTRAINTS
CONSTRAINT NUMBERS
         1

THERE ARE       0 ACTIVE SIDE CONSTRAINTS

TERMINATION CRITERIA

MAXIMUM S-VECTOR COMPONENT = 0.00000E+00 IS LESS THAN 1.00000E-04
```

```
-- OPTIMIZATION RESULTS


OBJECTIVE, F(X) =    2.63932E+00


DECISION VARIABLES, X

   ID         XL             X            XU
    1     1.00000E-01   7.88444E-01   1.00000E+02
    2     1.00000E-01   4.09267E-01   1.00000E+02


CONSTRAINTS, G(X)

     1)  -1.38342E-04  -6.34299E-01

GMAX = -1.3834E-04  JGMAX =         1


LAGRANGE MULTIPLIERS

     1)   2.64171E+00   0.00000E+00

FUNCTION CALLS =      25
```

CHAPTER 3

Advanced Use of DOT

- Introduction
- Over-Riding DOT Default Parameters
- Directly Supplying Gradients
- Interrupting and Restarting DOT
- Output to a Postprocessing Data File

3.1 Introduction

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

3.2 Over-Riding DOT Default Parameters

DOT 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 DOT. The default parameters include constraint tolerance, and the maximum finite difference step for gradient calculations, among others. The default parameters can be changed by simply setting the proper element of the RPRM or IPRM array to the desired value before calling DOT the first time. A sample program is included in this section. The figure below is a block diagram of the program to over-ride DOT default parameters. Tables 3-1 through 3-4 identify and define the internal parameters. Additional explanation of the parameters is given in Appendix D. Listing 3-1 shows an example FORTRAN file where the constraint tolerances CT and CTMIN, as well as the scaling parameter, ISCAL, are modified.

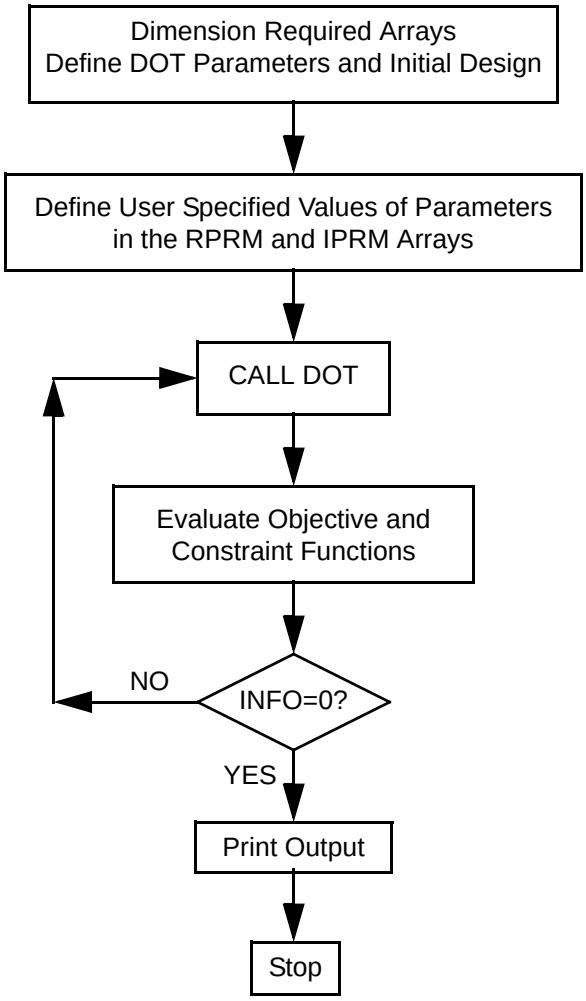

Dimension Required Arrays
Define DOT Parameters and Initial Design
Define User Specified Values of Parameters
in the RPRM and IPRM Arrays
CALL DOT
Evaluate Objective and
Constraint Functions
NO
INFO=0?
YES
Print Output
Stop

Table 3-1: Scalar Parameters Stored in the RPRM Array

LOCATION	NAME	DEFAULT VALUE
RPRM(1)	CT	-0.03
RPRM(2)	CTMIN	0.003
RPRM(3)	DABOBJ	MAX[0.0001*ABS(F0),1.0E-20]
RPRM(4)	DELOBJ	0.001
RPRM(5)	DOBJ1	0.1
RPRM(6)	DOBJ2	0.2*ABS(F0)
RPRM(7)	DX1	0.01
RPRM(8)	DX2	MAX{0.2*ABS[X(I)]}, I=1,NDV
RPRM(9)	FDCH	0.001
RPRM(10)	FDCHM	0.0001
RPRM(11)	RMVLMZ	0.5
RPRM(12)	DABSTR	MAX[0.0001*ABS(F0),1.0E-20]
RPRM(13)	DELSTR	0.001
RPRM(14)	GSTOL	0.25 For METHOD=1, Not used for METHOD=2, 0.01 for the sub-optimization problem when METHOD=3.
RPRM(15)	GSTOLM	0.0001
RPRM(16)	GSTOLS	0.25
RPRM(17)-RPRM(20)	RESERVED FOR INTERNAL USE	

NOTE: F0 = The value of the objective function at the start of optimization (for the initial values of **X**).

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

LOCATION	PARAMETER	DEFINITION
1	CT	A constraint is active if its numerical value is more positive than CT. CT is a small negative number.
2	CTMIN	A constraint is violated if its numerical value is more positive than CTMIN.
3	DABOBJ	Maximum absolute change in the objective for ITRMOP consecutive iterations to indicate convergence in optimization.
4	DELOBJ	Maximum relative change in the objective for ITRMOP consecutive iterations to indicate convergence in optimization.
5	DOBJ1	Relative change in the objective function attempted on the first optimization iteration. Used to estimate initial move in the one-dimensional search. Updated as the optimization progresses.
6	DOBJ2	Absolute change in the objective function attempted on the first optimization iteration. Used to estimate initial move in the one-dimensional search. Updated as the optimization progresses.
7	DX1	Maximum relative change in a design variable attempted on the first optimization iteration. Used to estimate the initial move in the one-dimensional search. Updated as the optimization progresses.
8	DX2	Maximum absolute change in a design variable attempted on the first optimization iteration. Used to estimate the initial move in the one-dimensional search. Updated as the optimization progresses.
9	FDCH	Relative finite difference step when calculating gradients.
10	FDCHM	Minimum absolute value of the finite difference step when calculating gradients. This prevents too small a step when X(I) is near zero.
11	RMVLMZ	Maximum relative change in design variables during the first approximate subproblem in the Sequential Linear Programming Method. That is, each design variable is initially allowed to change by $\pm$50%. This move limit is reduced as the optimization progresses.
12	DABSTR	Maximum absolute change in the objective for ITRMST consecutive iterations of the Sequential Linear Programming and Sequential Quadratic Programming methods to indicate convergence to the optimum.
13	DELSTR	Maximum relative change in the objective for ITRMST consecutive iterations of the Sequential Linear Programming method to indicate convergence to the optimum.

14	GSTOL	Golden Section tolerance as a fraction of the initial bounds in the one-dimensional search. The bounds will be reduced to this fraction. If GSTOL > 1.0, DOT will not use the Golden Section Method.
15	GSTOLM	The minimum Golden Section tolerance to be used.
16	GSTOLS	The value of GSTOL used in the one-dimensional search when METHOD=3.The bounds will be reduced to this fraction. If GSTOLS > 1.0, DOT will not use the Golden Section Method.

Table 3-3: Parameters Stored in the IPRM Array

LOCATION	NAME	DEFAULT VALUE
IPRM(1)	IGRAD	0
IPRM(2)	ISCAL	NDV
IPRM(3)	ITMAX	250
IPRM(4)	ITRMOP	2
IPRM(5)	IWRITE	6
IPRM(6)	NGMAX	NCON, but not more than 2*NDV
IPRM(7)	IGMAX	0 if IGRAD > 0. 1 if IGRAD $\leq$ 0. 1 for sub-optimization problem when METHOD = 3.
IPRM(8)	JTMAX	100
IPRM(9)	ITRMST	2
IPRM(10)	JPRINT	0
IPRM(11)		Not used.
IPRM(12)		Not used.
IPRM(13)	JWRITE	0
IPRM(14)		
IPRM(15)	NSTORE	Internally defined.
IPRM(16)-IPRM(17)	Internally used by DOT.	
IPRM(18)	IERROR	INTERNALLY DEFINED FATAL ERROR FLAG
IPRM(19)	NEWITR	INTERNALLY DEFINED ITERATION COUNTER
IPRM(20)	NGT	INTERNALLY DEFINED NUMBER OF NEEDED CONSTRAINT GRADIENTS

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

LOCATION	PARAMETER	DEFINITION
1	IGRAD	Specifies whether the gradients are calculated by DOT (IGRAD=-1 or 0) or by the user (IGRAD=1). If IGRAD=0, gradients are calculated by first forward finite difference. If IGRAD=-1, gradients are calculated by central finite difference. Note: If IGRAD=-1, the gradients are more accurate, but the number of function evaluations is almost doubled.
2	ISCAL	Design variables are rescaled every ISCAL iterations. Set ISCAL = -1 to turn off scaling. ISCAL is not used if METHOD=2.
3	ITMAX	Maximum number of iterations allowed at the optimization level.
4	ITRMOP	The number of consecutive iterations for which the absolute or relative convergence criteria must be met to indicate convergence at the optimizer level.
5	IWRITE	File number for printed output.
6	NGMAX	Number of retained constraints used for METHOD=2 or 3. Also, the maximum number of constraints retained for gradient calculations when METHOD=1.
7	IGMAX	If IGMAX=0, only gradients of active and violated constraints are calculated for METHOD = 1. If IGMAX = 1, up to NGMAX gradients are calculated, including active, violated, and near active constraints. IGMAX = 2 will use IGMAX = 0 in sub-optimization problem when METHOD = 3.
8	JTMAX	Maximum number of cycles allowed for the Sequential Linear Programming and Sequential Quadratic Programming methods. This is the number of linearized subproblems solved.
9	ITRMST	The number of consecutive cycles for which the absolute or relative convergence criteria must be met to indicate convergence in the Sequential Linear Programming and Sequential Quadratic Programming methods.
10	JPRINT	Sequential Linear Programming and Sequential Quadratic Programming subproblem print. If JPRINT>0, IPRINT is turned on during approximate subproblem. This is for debugging only. The values assigned JPRINT have the same meaning as IPRINT.
11		Not used.
12		Not used.
13	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.

14		Not used.
15	NSTORE	Storage allocated for constraint gradients and solution of the direction finding sub-problem. NSTORE is internally calculated by DOT.
16-17	Used internally by DOT.	
18	IERROR	Fatal error parameter returned by DOT. 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 or 4 indicates that storage for constraint gradients is too small. IERROR = 5 indicates that a violated constraint has a zero gradient. Therefore, no feasible design can be found.
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. If METHOD=0,1, for constrained problems, or if METHOD=0,1,2 for unconstrained problems, this is after each one-dimensional search. If METHOD=2,3, for constrained problems, this is after each approximate optimization. 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 DOT.
20	NGT	The number of constraint gradients needed. If the user supplies gradients to DOT, 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 DOT.

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

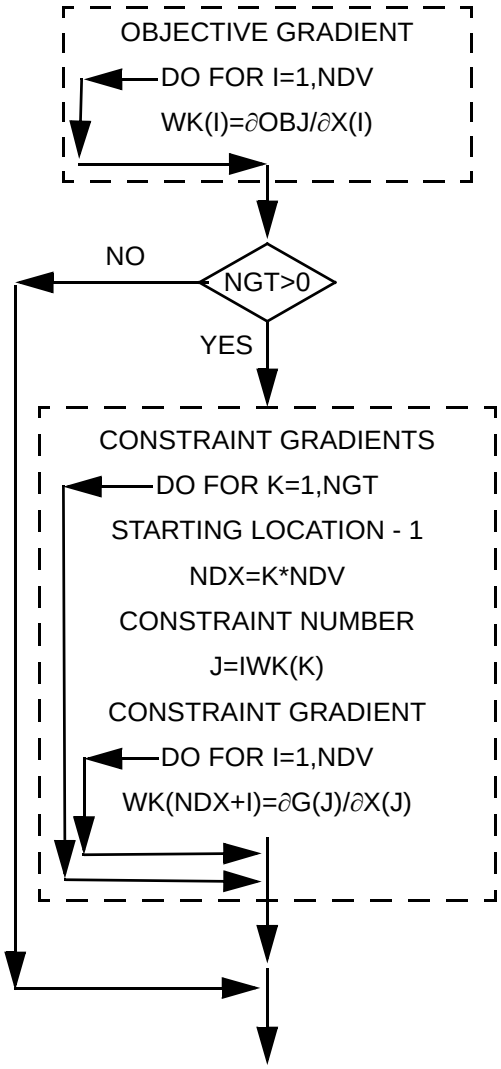
```
C
C     SAMPLE PROGRAM. THREE BAR TRUSS.
C
      DOUBLE PRECISION X(2),XL(2),XU(2),G(2),WK(800),RPRM(20),OBJ
      INTEGER IWK(200),IPRM(20),NRWK,NRIWK,I,METHOD,NDV,NCON,IPRINT,
     *MINMAX,INFO
C     DEFINE NRWK, NRIWK.
      NRWK=800
      NRIWK=200
C     ZERO RPRM AND IPRM.
      DO 10 I=1,20
         RPRM(I)=0.0
10       IPRM(I)=0
C * * OVER-RIDE THE DEFAULT FOR CT AND CTMIN.
C * * CT VALUE.
      RPRM(1)=-0.1
C * * CTMIN VALUE.
      RPRM(2)=0.002
C * * TURN OFF SCALING (ISCAL = -1).
      IPRM(2)=-1
C     DEFINE METHOD,NDV,NCON.
      METHOD=3
      NDV=2
      NCON=2
C     DEFINE BOUNDS AND INITIAL DESIGN.
      DO 20 I=1,NDV
         X(I)=1.0
         XL(I)=0.1
20       XU(I)=100.
C     DEFINE IPRINT, MINMAX, INFO.
      IPRINT=3
      MINMAX=-1
      INFO=0
100   CALL DOT (INFO,METHOD,IPRINT,NDV,NCON,X,XL,XU,
     *OBJ,MINMAX,G,RPRM,IPRM,WK,NRWK,IWK,NRIWK)
      IF(INFO.EQ.0)STOP
      CALL EVAL(OBJ,X,G)
      GO TO 100
      END
      SUBROUTINE EVAL (OBJ,X,G)
C     SUBROUTINE TO EVALUATE THE OBJECTIVE FUNCTION AND CONSTRAINTS
C     FOR THE SPRING EQUILIBRIUM PROBLEM.
      DOUBLE PRECISION X(*),G(*),OBJ
      OBJ=2.0*SQRT(2.)*X(1)+X(2)
      G(1)=(2.*X(1)+SQRT(2.)*X(2))/(2.*X(1)*(X(1)+
     *SQRT(2.0)*X(2)))-1.
      G(2)=1./(2.*(X(1)+SQRT(2.)*X(2)))-1.
      RETURN
      END
```

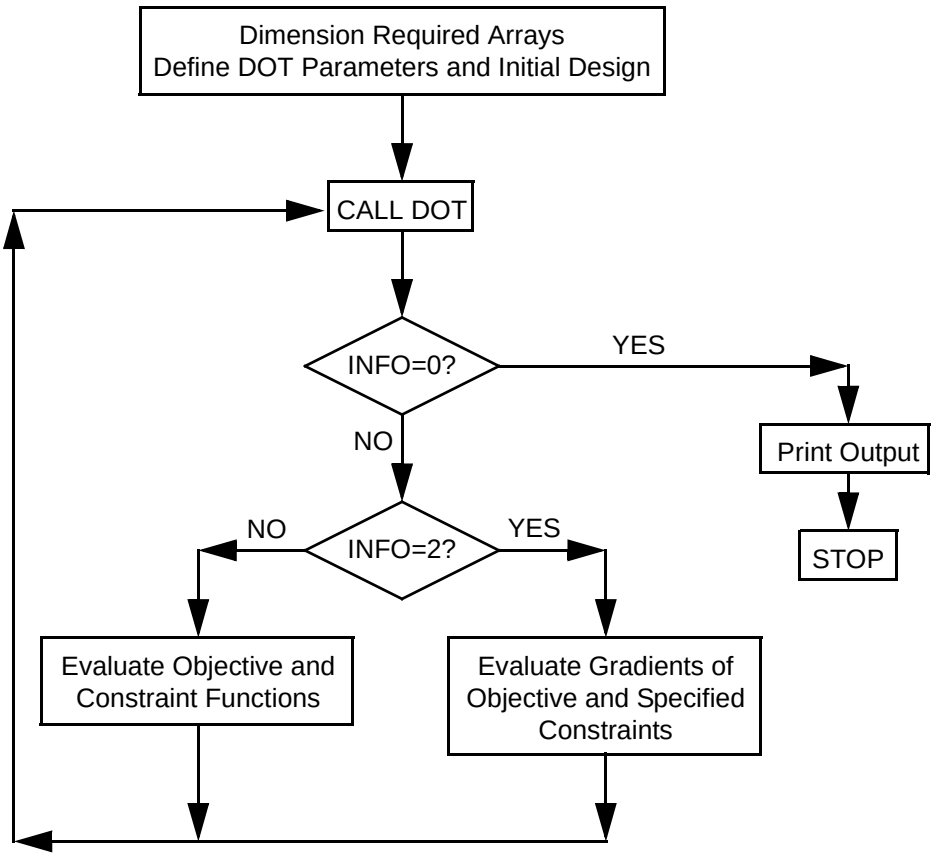
3.3 Directly Supplying Gradients

The default option of DOT is to use first forward finite difference methods to calculate the gradients of the objective function and constraints. This approach can become time consuming for large problems. The user may wish to directly evaluate the gradients in the FORTRAN or C/C++ program and send them to DOT. This is done by setting IPRM(1) = 1 before calling DOT.

The gradients of the objective function are stored in the first NDV locations of the **WK** array. DOT requires gradients of the active/violated and near active constraints only. The active/violated and near active constraints are identified in the first NGT elements of the **IWK** array, where NGT is given in **IPRM**(20) and is the number of active/violated and near active constraints. 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 following figures. The second figure is a block diagram of the standard calling program for DOT 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 2.6.


OBJECTIVE GRADIENT
DO FOR I=1,NDV
WK(I)=∂OBJ/∂X(I)
NO
NGT>0
YES
CONSTRAINT GRADIENTS
DO FOR K=1,NGT
STARTING LOCATION - 1
NDX=K*NDV
CONSTRAINT NUMBER
J=IWK(K)
CONSTRAINT GRADIENT
DO FOR I=1,NDV
WK(NDX+I)=∂G(J)/∂X(J)


Dimension Required Arrays
Define DOT Parameters and Initial Design
CALL DOT
INFO=0?
YES
NO
Print Output
STOP
NO
INFO=2?
YES
Evaluate Objective and Constraint Functions
Evaluate Gradients of Objective and Specified Constraints

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

```
C     USER-SUPPLIED GRADIENTS: THE THREE-BAR TRUSS.
C     REQUIRED ARRAYS.
C
      DOUBLE PRECISION X(2),XL(2),XU(2),G(2),WK(1000),RPRM(20),AA(2,2),
     *BB(2,2),OBJ,D1
      INTEGER IWK(500),IPRM(20),NRWK,NRIWK,I,METHOD,NDV,NCON,IPRINT,
     *MINMAX,INFO,NGT,K,J,N1
C
C     DIMENSIONS OF WK AND IWK
      NRWK=1000
      NRIWK=500
C     ZERO RPRM AND IPRM
      DO 10 I=1,20
        RPRM(I)=0.0
10    IPRM(I)=0
C     SPECIFY THAT GRADIENTS ARE TO BE PROVIDED
      IPRM(1)=1
C     DEFINE METHOD, NDV, NCON, IPRINT, MINMAX
      METHOD=3
      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 DOT (INFO,METHOD,IPRINT,NDV,NCON,X,XL,XU,OBJ,
     *MINMAX,G,RPRM,IPRM,WK,NRWK,IWK,NRIWK)
C     PROVIDE GRADIENTS IF DOT IS REQUESTING THEM
      IF(INFO.EQ.2)GO TO 30
C     EXIT IF CONVERGENCE IS OBTAINED
      IF(INFO.EQ.0)GO TO 70
      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=IPRM(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)
      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
```

3.4 Interrupting and Restarting DOT

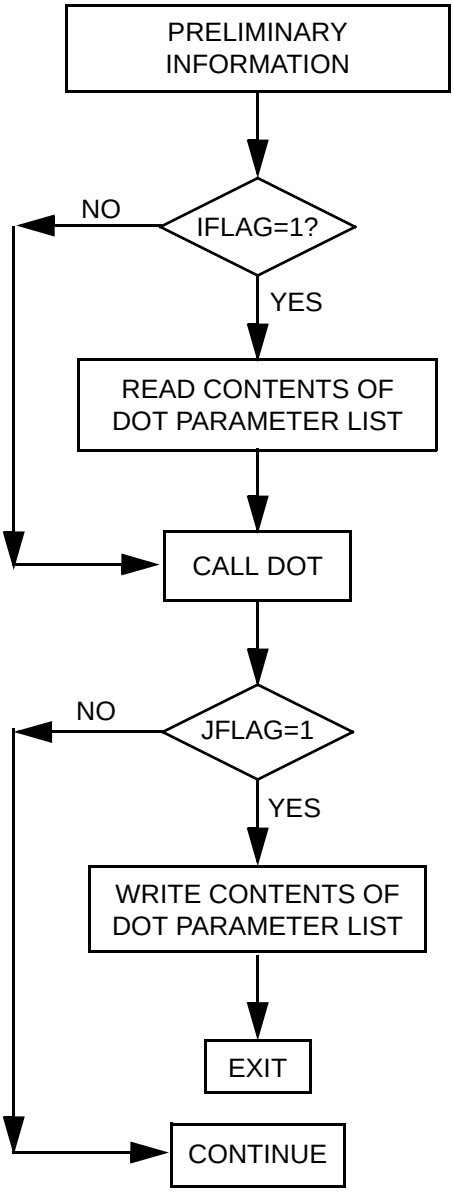
It is a simple matter to stop the optimization when you wish and then restart from that point at a later time. All that is required is that, on return from DOT, you write the entire contents of the parameter list to a file and then exit. Upon restarting, you simply read this information again and continue from the point where you exited. All internal parameters of DOT 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 below. 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 **IPRM**(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 13 of **IPRM**), 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.


PRELIMINARY INFORMATION
IFLAG=1?
NO
YES
READ CONTENTS OF DOT PARAMETER LIST
CALL DOT
JFLAG=1
NO
YES
WRITE CONTENTS OF DOT PARAMETER LIST
EXIT
CONTINUE

3.5 Output to a Postprocessing Data File

By opening a file and setting JWRITE [**IPRM**(13)] to the value of that file number, the user can output useful design iteration history information. 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

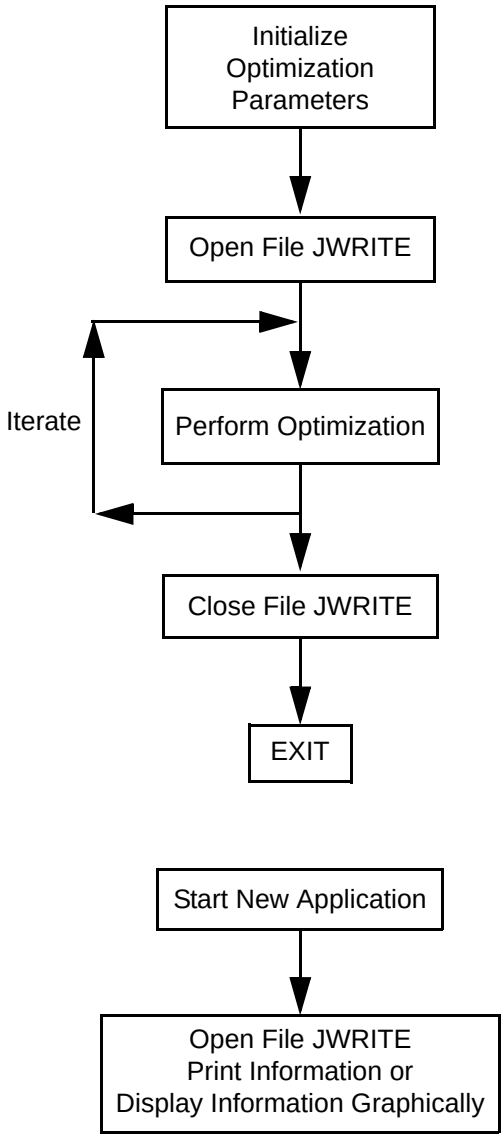
Note that the X-Vector and G-Vector are written in groups of 5 entries.

If METHOD = 0 or 1, or if METHOD = 2 and NCON = 0, this information is written after each one-dimensional search. If NCON > 0 and METHOD = 2 or 3, this information is written after each approximation to the problem is solved.

In addition to the information written after each iteration, the initial optimization information (ITER = 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.


Initialize Optimization Parameters
Open File JWRITE
Iterate
Perform Optimization
Close File JWRITE
EXIT
Start New Application
Open File JWRITE
Print Information or
Display Information Graphically

CHAPTER 4

Examples

- Introduction
- Box Design
- Three-Bar Truss
- Cantilevered Beam
- Equilibrium of a Spring System
- Construction Management
- Piston Design
- Portfolio Selection
- Equality Constraints

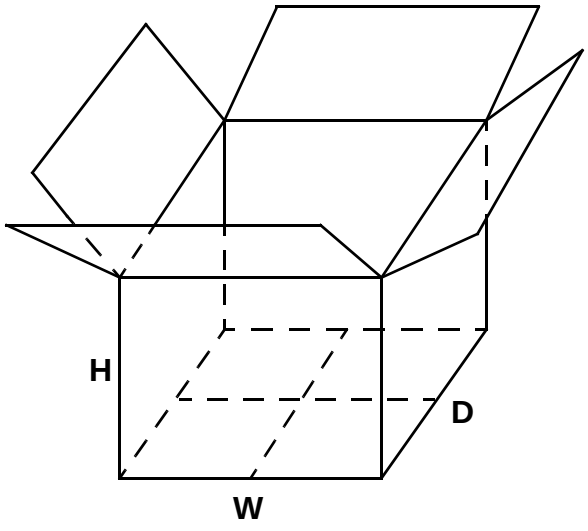
4.1 Introduction

This section presents examples of optimization problems taken from a variety of disciplines. There is a FORTRAN calling program included at the end of each example. You can begin using DOT immediately by solving these example problems. Refer to Section 2.5 for instructions on compiling the programs and linking them with DOT. The examples are as follows.

- Box Design (Material Minimization)
- Three-Bar Truss (Volume Minimization)
- Cantilevered Beam (Volume Minimization)
- Spring System Equilibrium (Nonlinear Structural Analysis Problem)
- Construction Management (Cost Minimization)
- Piston Oil Minimization (Volume Minimization)
- Portfolio Selection (Yield Maximization)
- Equality Constraint Example

4.2 Box Design

This is the very simple box design problem that described in Chapter 1. The goal is to determine dimensions H, W, and D to minimize the carton surface area required to enclose a volume of at least 2 cubic feet. The box is drawn as follows



The problem is presented in standard form as

Minimize Surface Area, S, where

$$S = 2.0*(W*H + D*H + 2.0*W*D) \quad (4\text{-}1)$$

Subject to;

$$2.0 - H*D*W \leq 0.0 \quad (4\text{-}2)$$

$$H, W, D \geq 0.0 \quad (4\text{-}3)$$

The theoretical optimum design for this problem is

$S = 12.00\ ft^2$, $V = 2.00\ ft^3$, $H = 2.00$ ft, $W = 1.00$ ft, $D = 1.00$ ft

The DOT solutions are

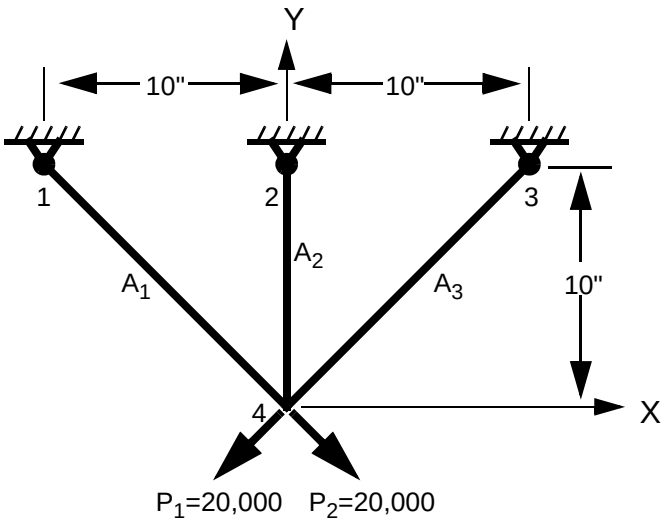
PARAMETER	INITIAL VALUE	OPTIMUM Method=1	OPTIMUM Method=2	OPTIMUM Method=3
H	1.0	2.017	1.985	2.007
W	1.0	0.995	1.004	0.999
D	1.0	0.995	1.003	0.999
OBJECTIVE	8.0	11.993	11.999	12.016
MAX G	1.0	9.1e-4	1.2E-4	-2.1E-3
FUNCTIONS		63	63	39

LISTING 4-1: BOX DESIGN FORTRAN PROGRAM.

```
C
C     SAMPLE PROGRAM.  BOX DESIGN.
C
      DOUBLE PRECISION X(5),XL(5),XU(5),G(5),WK(800),RPRM(20),OBJ
      INTEGER IWK(200),IPRM(20),I,METHOD,NDV,NCON,IPRINT,MINMAX,INFO
C     DEFINE NRWK, NRIWK.
      NRWK=800
      NRIWK=200
C     ZERO RPRM AND IPRM.
      DO 10 I=1,20
         RPRM(I)=0.0
10       IPRM(I)=0
C     DEFINE METHOD,NDV,NCON.
      METHOD=1
C     THREE DESIGN VARIABLES.
      NDV=3
C     ONE CONSTRAINT
      NCON=1
C     DEFINE BOUNDS AND INITIAL DESIGN.
      DO 20 I=1,NDV
C     INITIAL VALUES.
         X(I)=1.0
C     LOWER BOUNDS.
         XL(I)=0.01
C     UPPER BOUNDS
20       XU(I)=100.
C     DEFINE IPRINT, MINMAX, INFO.
C     PRINT CONTROL.
      IPRINT=1
C     MINIMIZE
      MINMAX=-1
C     INITIALIZE INFO TO ZERO.
      INFO=0
C     OPTIMIZE.
100   CALL DOT (INFO,METHOD,IPRINT,NDV,NCON,X,XL,XU,
     *OBJ,MINMAX,G,RPRM,IPRM,WK,NRWK,IWK,NRIWK)
C     FINISHED?
      IF(INFO.EQ.0) STOP
C     EVALUATE OBJECTIVE AND CONSTRAINT.
      CALL EVAL(OBJ,X,G)
C     GO CONTINUE WITH OPTIMIZATION.
      GO TO 100
      END
      SUBROUTINE EVAL (OBJ,X,G)
C     SUBROUTINE TO EVALUATE THE OBJECTIVE FUNCTION AND CONSTRAINTS
C     FOR THE BOX DESIGN PROBLEM.
      DOUBLE PRECISION X(*),G(*),OBJ
      OBJ=2.0*X(2)*X(1)+2.0*X(3)*X(1)+4.0*X(2)*X(3)
      G(1)=1.0-0.5*X(1)*X(2)*X(3)
      RETURN
      END
```

4.3 Three-Bar Truss

This is the optimization of the 3-bar truss solved in Chapter 2 and shown in the figure below. This is a classical example in structural design. The objective function is the total volume of the material of the members. The decision variables X_1 and X_2 correspond to the cross-sectional 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 P1. The loads, P_1 and P_2, are applied separately and the material specific weight is 0.1 lb. per cubic inch. The allowable stress in the members is 20,000 psi. This problem is formulated into the form given below. The original problem actually consists of 12 constraints, being the stress limit in each member under each of the 2 loading conditions. Here, the problem has been abbreviated for clarity.



The problem is formally stated as follows

$$\text{Minimize} \quad OBJ = 2\sqrt{2}X_1 + X_2 \tag{4-4}$$

Subject to:

$$g_1 = \frac{2X_1 + \sqrt{2}X_2}{2X_1(X_1 + \sqrt{2}X_2)} - 1.0 \le 0.0 \tag{4-5}$$

$$g_2 = \frac{1.0}{X_1 + \sqrt{2}X_2} - 1.0 \le 0.0 \tag{4-6}$$

$$0.01 \le X_i \le 100.0 \quad i = 1,2 \tag{4-7}$$

The DOT solutions are

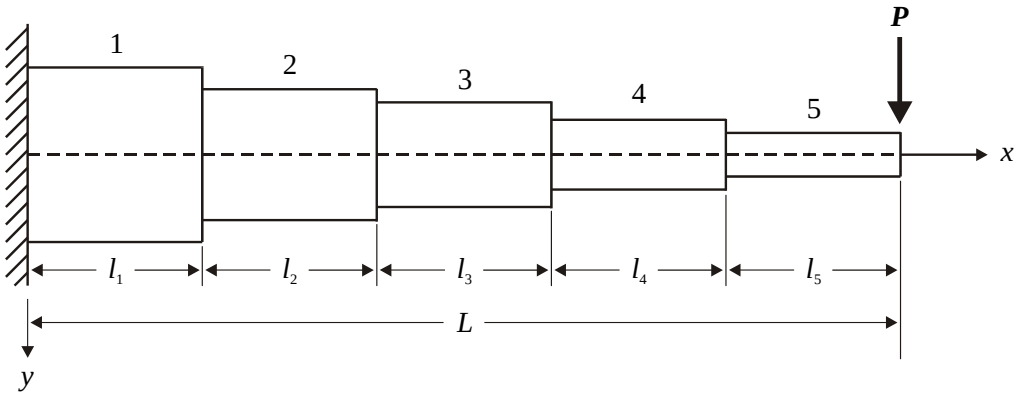
PARAMETER	INITIAL VALUE	OPTIMUM Method=1	OPTIMUM Method=2	OPTIMUM Method=3
A1	1.00	0.791	0.784	0.771
A2	1.00	0.404	0.420	0.442
A3	1.00	0.791	0.784	0.771
OBJECTIVE	3.828	2.641	2.638	2.640
MAX G	-0.293	-8.17E-4	2.6E-4	5.5E-5
FUNCTIONS		32	39	48

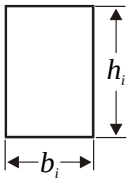
LISTING 4-2: THREE BAR TRUSS FORTRAN PROGRAM.

```
C
C     SAMPLE PROGRAM. THREE BAR TRUSS.
C
      DOUBLE PRECISION X(2),XL(2),XU(2),G(2),WK(800),RPRM(20),OBJ
      INTEGER IWK(200),IPRM(20),NRWK,NRIWK,I,METHOD,NDV,NCON,IPRINT,
     *MINMAX,INFO
C     DEFINE NRWK, NRIWK.
      NRWK=800
      NRIWK=200
C     ZERO RPRM AND IPRM.
      DO 10 I=1,20
         RPRM(I)=0.0
10       IPRM(I)=0
C     DEFINE METHOD,NDV,NCON.
      METHOD=1
      NDV=2
      NCON=2
C     DEFINE BOUNDS AND INITIAL DESIGN.
      DO 20 I=1,NDV
         X(I)=1.0
         XL(I)=0.1
20       XU(I)=100.
C     DEFINE IPRINT, MINMAX, INFO.
      IPRINT=3
      MINMAX=-1
      INFO=0
100   CALL DOT (INFO,METHOD,IPRINT,NDV,NCON,X,XL,XU,
     *OBJ,MINMAX,G,RPRM,IPRM,WK,NRWK,IWK,NRIWK)
      IF(INFO.EQ.0)STOP
      CALL EVAL(OBJ,X,G)
      GO TO 100
      END
      SUBROUTINE EVAL (OBJ,X,G)
C     SUBROUTINE TO EVALUATE THE OBJECTIVE FUNCTION AND CONSTRAINTS
C     FOR THE SPRING EQUILIBRIUM PROBLEM.
      DOUBLE PRECISION X(*),G(*),OBJ
      OBJ=2.0*SQRT(2.)*X(1)+X(2)
      G(1)=(2.*X(1)+SQRT(2.)*X(2))/(2.*X(1)*(X(1)+
     *SQRT(2.0)*X(2)))-1.
      G(2)=1./(2.*(X(1)+SQRT(2.)*X(2)))-1.
      RETURN
      END
```

4.4 Cantilevered Beam

The cantilevered beam shown below 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), deflection under the load, and the geometric requirement that the height of any segment does not exceed twenty times the width.





Cross section

P = 50,000 N
E = 2.0x10^7 N/cm^2
L = 500 cm
$\overline{\sigma}$ = 14,000 N/cm^2
$\overline{y}$ = 2.5 cm

The deflection y_i at the right end of segment i is calculated by the following recursion formulas:

$$y_0 = y'_0 = 0 \tag{4-8}$$

$$y'_i = \frac{P l_i}{E I_i}\left[L + \frac{l_i}{2} - \sum_{j=1}^{i} l_j\right] + y'_{i-1} \tag{4-9}$$

$$y_i = \frac{P l_i^2}{2 E I_i}\left[L - \sum_{j=1}^{i} l_j + \frac{2 l_i}{3}\right] + y'_{i-1} l_i + y_{i-1} \tag{4-10}$$

where the deflection y is defined as positive downward, y'_i is the derivative of y with respect to x (the slope), and l_i is the length of segment i. Young's modulus E is the same for all segments, and the moment of inertia for segment i is

$$I_i = \frac{b_i h_i^3}{12} \tag{4-11}$$

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

$$M_i = P\left[L + l_i - \sum_{j=1}^{i} l_j\right] \tag{4-12}$$

and the corresponding maximum bending stress is

$$\sigma_i = \frac{M_i h_i}{2I_i} \tag{4-13}$$

The design task is now defined as

Minimize:
$$V = \sum_{i=1}^{N} b_i h_i l_i \tag{4-14}$$

Subject to:

$$\frac{\sigma_i}{\bar{\sigma}} - 1 \le 0 \qquad i = 1, N \tag{4-15}$$

$$h_i - 20 b_i \le 0 \qquad i = 1, N \tag{4-16}$$

$$\frac{y_N}{\bar{y}} - 1 \le 0 \tag{4-17}$$

$$b_i \ge 1.0 \qquad i = 1, N \tag{4-18}$$

$$h_i \ge 5.0 \qquad i = 1, N \tag{4-19}$$

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. (4-15) and (4-17), N linear constraints defined by Eq. (4-16), and $2N$ side constraints on the design variables defined by Eqs. (4-18) and (4-19). The side constraints are treated here as general inequality constraints. Additionally, lower bounds of 0.1 are imposed explicitly on b_i and h_i, $i = 1, N$ within the optimization program to ensure that the design remains physically meaningful.

This problem was solved using five segments (10 design variables).

The DOT solutions are

PARAMETER	INITIAL VALUE	OPTIMUM Method=1	OPTIMUM Method=2	OPTIMUM Method=3
b_1	5.00	3.12	3.10	3.12
b_2	5.00	2.87	2.88	2.87
b_3	5.00	2.56	2.58	2.57
b_4	5.00	2.20	2.20	2.20
b_5	5.00	1.75	1.75	1.75
h_1	40.00	62.50	61.96	62.36
h_2	40.00	57.37	57.54	57.38
h_3	40.00	51.12	51.64	51.34
h_4	40.00	44.10	44.09	44.11
h_5	40.00	35.00	35.00	35.00
OBJECTIVE	100,000	64,908	64,927	64,934
MAX G	0.538	6.9E-4	3.7E-4	3.2E-4
FUNCTIONS		219	182	163

Note that at the beginning of the program listing a parameter called NSEG is defined. In the above example, NSEG=5. You may change NSEG to increase or decrease the problem size. The program is dimensioned to allow a value of NSEG up to 50, to yield 100 design variables. If you wish to solve even larger problems, be sure to increase the array sizes.

This problem was solved again using fifty segments (100 design variables).

The DOT solutions are

PARAMETER	INITIAL VALUE	OPTIMUM Method=1	OPTIMUM Method=2	OPTIMUM Method=3
OBJECTIVE	100,000	63,202	63,207	63,196
MAX G	0.5625	7.2E-6	2.8E-4	3.3E-4
FUNCTIONS		2,367	2,027	1,501

Finally, this problem was solved without the displacement constraint. Note that, now the optimum should be fully constrained so METHOD=2 works very well.

The DOT solutions are

PARAMETER	INITIAL VALUE	OPTIMUM Method=1	OPTIMUM Method=2	OPTIMUM Method=3
OBJECTIVE	100,000	54,923	54,605	54,843
MAX G	0.5625	2.9E-3	1.0E-7	4.8E-4
FUNCTIONS		6,207	809	864

LISTING 4-3: CANTILEVER BEAM ANALYSIS FORTRAN PROGRAM.

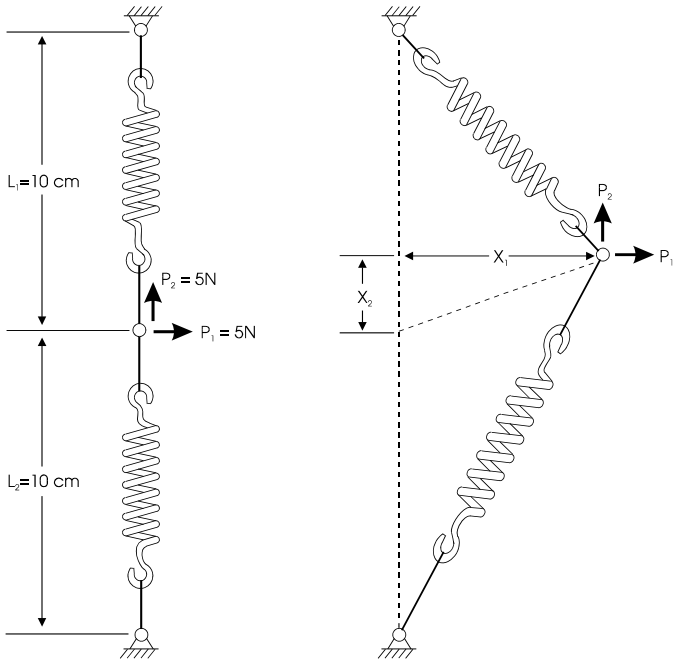
```
C
C     SAMPLE PROGRAM.  NSEG ELEMENT BEAM DESIGN.
C
      DOUBLE PRECISION X(1000),XL(1000),XU(1000),G(1001),WK(10000000),
     *RPRM(20),OBJ
      INTEGER IWK(10000),IPRM(20),NRWK,NRIWK,NSEG,I,METHOD,NDV,NCON,
     *IPRINT,MINMAX,INFO
C     DEFINE NRWK, NRIWK.
      NRWK=10000000
      NRIWK=10000
C
C     NUMBER OF BEAM SEGMENTS
C
      NSEG=50
C     ZERO RPRM AND IPRM.
      DO 10 I=1,20
         RPRM(I)=0.0
10       IPRM(I)=0
C     DEFINE METHOD,NDV,NCON.
      METHOD=1
C     NDV = TWO TIMES NSEG DESIGN VARIABLES.
      NDV=2*NSEG+1
C     TWO TIMES NSEG + 1 CONSTRAINTS.
      NCON=2*NSEG
C     DEFINE BOUNDS AND INITIAL DESIGN.
      DO 20 I=1,NSEG
C     INITIAL VALUES.
         X(I)=5.0
         X(I+NSEG)=40.0
C     LOWER BOUNDS.
         XL(I)=1.0
         XL(I+NSEG)=5.0
C     UPPER BOUNDS
         XU(I)=100.
         XU(I+NSEG)=100.
20    CONTINUE
C     DEFINE IPRINT, MINMAX, INFO.
C     PRINT CONTROL.
      IPRINT=2
C     MINIMIZE
      MINMAX=-1
C     INITIALIZE INFO TO ZERO.
      INFO=0
C     OPTIMIZE.
100   CALL DOT (INFO,METHOD,IPRINT,NDV,NCON,X,XL,XU,
     *OBJ,MINMAX,G,RPRM,IPRM,WK,NRWK,IWK,NRIWK)
C     FINISHED?
       if(info.eq.0) write(*,*)' final obj =',obj
      IF(INFO.EQ.0) STOP
C     EVALUATE OBJECTIVE AND CONSTRAINT.
      CALL EVAL(OBJ,X,G,NSEG)
C     GO CONTINUE WITH OPTIMIZATION.
      GO TO 100
      END
```

Examples

```
      SUBROUTINE EVAL (OBJ,X,G,NSEG)
      DOUBLE PRECISION X(*),G(*),P,E,AL,ALI,SIG,YMX,VOL,ALA,Y,YP,AI,AM,
     *SIGI,OBJ
      INTEGER NSEG,I
C     NSEG-ELEMENT BEAM.
C     NDV=10, NCON=11.
      P=50000.
      E=2.0E+7
      AL=500.
      ALI=AL/FLOAT(NSEG)
      SIG=14000.
      YMX=2.54
C     VOLUME, STRESS CONSTRAINTS, H/B CONSTRAINTS, DISPL. CONSTRAINT.
      VOL=0.
      ALA=0.
      Y=0.
      YP=0.
      DO 22 I=1,NSEG
      VOL=VOL+ALI*X(I)*X(I+NSEG)
      AI=X(I)*(X(I+NSEG)**3)/12.
      ALA=ALA+ALI
      AM=P*(AL+ALI-ALA)
      SIGI=.5*AM*X(I+NSEG)/AI
C     STRESS CONSTRAINTS.
      G(I)=SIGI/SIG-1.
C     H/B CONSTRAINTS.
      G(I+NSEG)=X(I+NSEG)-20.*X(I)
      G(I+NSEG)=.1*G(I+NSEG)
      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)
22    CONTINUE
      G(2*NSEG+1)=Y/YMX-1.
      OBJ=VOL
      RETURN
      END
```

4.5 Equilibrium of a Spring System

The figure below shows a simple spring for which we wish to find the equilibrium position under the applied loads [1]. This is a nonlinear analysis problem. Loads P_1 and P_2 are applied to a two-spring system as shown. The equilibrium position is calculated by minimizing the potential energy (PE).



(a) Undeformed (b) Deformed

The total potential energy of the system is defined as;

$$PE = \frac{1}{2}K_1\delta L_1^2 + \frac{1}{2}K_2\delta L_2^2 - P_1X_1 - P_2X_2$$

$$= \frac{1}{2}K_1[\sqrt{X_1^2 + (L_1 - X_2)^2} - L_1]^2 + \frac{1}{2}K_2[\sqrt{X_1^2 + (L_2 + X_2)^2} - L_2]^2 \quad (4\text{-}20)$$

$$- P_1X_1 - P_2X_2$$

This equation is solved with unconstrained minimization (NCON = 0).

The DOT solutions are

PARAMETER	INITIAL VALUE	OPTIMUM Method=1	OPTIMUM Method=2
X1	0.00	8.581	8.508
X2	0.00	4.442	4.290
OBJECTIVE	0.00	-41.80	-41.76
FUNCTIONS		134	134

LISTING 4-4: SPRING EQUILIBRIUM ANALYSIS FORTRAN PROGRAM.

```
C
C     SAMPLE PROGRAM.  SPRING EQUILIBRIUM ANALYSIS.
C
      DOUBLE PRECISION X(2),XL(2),XU(2),G(1),WK(800),RPRM(20),OBJ
      INTEGER IWK(200),IPRM(20),NRWK,NRIWK,I,METHOD,NDV,NCON,IPRINT,
     *MINMAX,INFO
C     DEFINE NRWK, NRIWK.
      NRWK=800
      NRIWK=200
C     ZERO RPRM AND IPRM.
      DO 10 I=1,20
         RPRM(I)=0.0
10       IPRM(I)=0
C     DEFINE METHOD,NDV,NCON.
      METHOD=1
      NDV=2
      NCON=0
C     DEFINE BOUNDS AND INITIAL DESIGN.
      DO 30 I=1,NDV
         X(I)=0.0
         XL(I)=-100.0
30       XU(I)=100.0
C     DEFINE IPRINT, MINMAX, INFO
      IPRINT=1
      MINMAX=-1
      INFO=0
100   CALL DOT (INFO,METHOD,IPRINT,NDV,NCON,X,XL,XU,OBJ,MINMAX,G,RPRM,
     *IPRM,WK,NRWK,IWK,NRIWK)
      IF(INFO.EQ.0)STOP
      CALL EVAL(OBJ,X,G)
      GO TO 100
      END
      SUBROUTINE EVAL (OBJ,X,G)
C
C     SUBROUTINE TO EVALUATE THE OBJECTIVE FUNCTION AND CONSTRAINTS
C     FOR THE SPRING EQUILIBRIUM PROBLEM.
C
      DOUBLE PRECISION X(*),G(*),OBJ
      OBJ=4.*(SQRT(ABS(X(1))**2 + ABS(10. - X(2))**2 ) - 10.)**2
     *+.5*(SQRT(ABS(X(1))**2 + ABS(10. + X(2))**2 ) - 10.)**2
     *- 5.*X(1)- 5.*X(2)
      RETURN
      END
```

4.6 Construction Management

A contractor is considering two gravel pits from which he may purchase material to supply a project [3]. The unit cost to load and deliver the material to the project site is \$5.00/yd.^{3} from pit 1 and \$7.00/yd.^{3} from pit 2. He must deliver a minimum of 10,000 yd.^{3} to the site.

The mix that he delivers must consist of at least 50 percent sand, no more than 60 percent gravel, nor more than 8 percent silt. (Note that there is some redundancy in the requirements.) The material at pit 1 consists of 30 percent sand and 70 percent gravel. The material at pit 2 consists of 60 percent sand, 30 percent gravel, and 10 percent silt. Determine how much material should be taken from each pit.

Since the gravel from pit 1 does not contain the minimum amount of sand to meet project requirements, the contractor may not utilize the cheaper material exclusively. He must mix the material from pits 1 and 2 to produce the required proportions.

We define the decision variables to be

X_1 = amount of material taken from pit 1 (in cubic yards)

X_2 = amount of material taken from pit 2 (in cubic yards)

The objective function is

$$\text{Minimize } C = 5X_1 + 7X_2$$

Let $X_1 + X_2$ equal the total amount of standard mix delivered to the project site. The contractor must deliver at least 10,000 cubic yards, thus the delivery constraint is

$$X_1 + X_2 \geq 10{,}000$$

The mixture must contain at least 50 percent sand. The contractor may obtain the desired amount of sand by combining the materials from each pit.

$$0.3X_1 + 0.6X_2 \geq 0.5(X_1 + X_2)$$

The products $0.3X_1$ and $0.6X_2$ are the amounts of sand taken from pits 1 and 2, respectively. The term $0.5(X_1 + X_2)$ is the amount of sand in the mix. Similarly, the constraint on the amount of gravel to be delivered is

$$0.7X_1 + 0.3X_2 \leq 0.6(X_1 + X_2)$$

Finally, the constraint equation for silt is

$$0.1X_2 \leq 0.08(X_1 + X_2)$$

Examples

The minimum cost model may be written as

Minimize $C = 5X_1 + 7X_2$

Subject to:

$X_1 + X_2 \geq 10{,}000$ (delivery)

$0.3X_1 + 0.6X_2 \geq 0.5(X_1 + X_2)$ (sand)

$0.7X_1 + 0.3X_2 \leq 0.6(X_1 + X_2)$ (gravel)

$0.1X_2 \leq 0.08(X_1 + X_2)$ (silt)

$X_1, X_2 \geq 0$

or in standard form,

Minimize $F(X_1, X_2) = 5X_1 + 7X_2$ (4-21)

Subject to:

$10{,}000 - X_1 - X_2 \leq 0$ (4-22)

$2X_1 - X_2 < 0$ (4-23)

$X_1 - 3X_2 < 0$ (4-24)

$X_2 - 4X_1 < 0$ (4-25)

$X_1, X_2 \geq 0$ (4-26)

The DOT solutions are

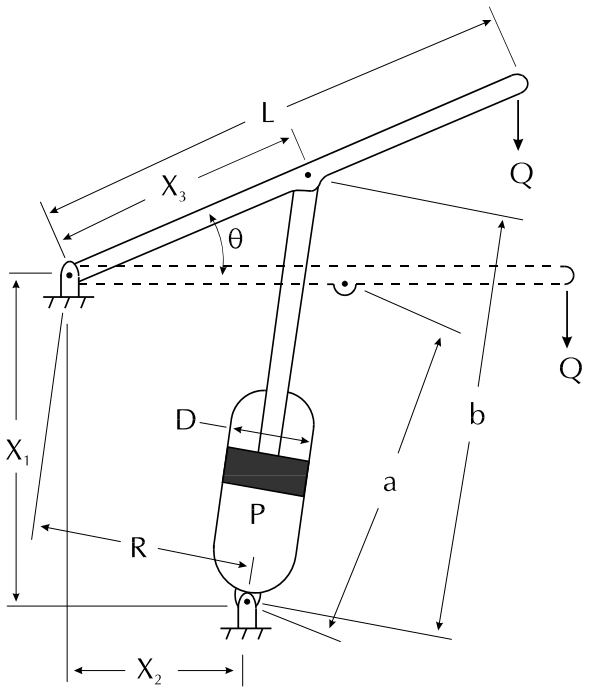
PARAMETER	INITIAL VALUE	OPTIMUM Method=1	OPTIMUM Method=2	OPTIMUM Method=3
X_1	3,000.0	3,333	3,333	3,332
X_2	3,000.0	6,667	6,667	6,664
OBJECTIVE	36,000.0	6,333	6,333	6,330
MAX G	0.40	0.0	0.000	4.7E-4
FUNCTIONS		32	9	13

LISTING 4-5: CONSTRUCTION MANAGEMENT FORTRAN PROGRAM.

```
C
C     SAMPLE PROGRAM.  CONSTRUCTION MANAGEMENT.
C
      DOUBLE PRECISION X(2),XL(2),XU(3),G(4),WK(800),RPRM(20),OBJ
      INTEGER IWK(200),IPRM(20),NRWK,NRIWK,I,METHOD,NDV,NCON,IPRINT,
     *MINMAX,INFO
C     DEFINE NRWK, NRIWK.
      NRWK=800
      NRIWK=200
C     ZERO RPRM AND IPRM.
      DO 10 I=1,20
         RPRM(I)=0.0
10       IPRM(I)=0
C     DEFINE METHOD,NDV,NCON.
      METHOD=1
      NDV=2
      NCON=4
C     DEFINE BOUNDS AND INITIAL DESIGN.
      DO 20 I=1,NDV
         X(I)=3000.
         XL(I)=0.0
20       XU(I)=10000.
C     DEFINE IPRINT, MINMAX, INFO.
      IPRINT=1
      MINMAX=-1
      INFO=0
100   CALL DOT (INFO,METHOD,IPRINT,NDV,NCON,X,XL,XU,
     *OBJ,MINMAX,G,RPRM,IPRM,WK,NRWK,IWK,NRIWK)
      IF(INFO.EQ.0)STOP
      CALL EVAL(OBJ,X,G)
      GO TO 100
      END
      SUBROUTINE EVAL (OBJ,X,G)
C
C     SUBROUTINE TO EVALUATE THE OBJECTIVE FUNCTION AND CONSTRAINTS
C     FOR THE CONSTRUCTION MANAGEMENT PROBLEM.
C
      DOUBLE PRECISION X(*),G(*),OBJ
      OBJ=5.*X(1)+7.*X(2)
      G(1)=(-X(1)-X(2)+10000.)/10000.
      G(2)=(2.*X(1)-X(2))/10000.
      G(3)=(X(1)-3.*X(2))/10000.
      G(4)=(-4.*X(1)+X(2))/10000.
      RETURN
      END
```

4.7 Piston Design

A piston lifts a load Q, as depicted below.



The specifications for the problem are

Payload (Q)= 10,000 lbs.

Beam Length (L)= 120 in.

Oil Pressure (P)= 1,500 psi

Maximum Allowable Bending Moment of the Beam $= 1.8 \times 10^6$ lbs-in.

The objective is to minimize the volume of oil required to lift the load from 0 to 45 degrees. There are constraints on force equilibrium, maximum bending moment of the beam, and minimum piston stroke. The pivot position, X_3, must not exceed L/2 and the support position must be at least half the piston diameter. The design variables are the distances X_1, X_2, X_3, and D. The problem is formulated as

$$\text{Minimize Volume of Oil, } V = \frac{\pi}{4}D^2(b-a) \tag{4-27}$$

Subject to;

$$\text{Equilibrium at } \theta = 45^o \quad QL\cos\theta - R(\theta)F \le 0 \tag{4-28}$$

$$\text{Bending Moment} \quad Q(L-X_3) - 1.8\times10^6 \le 0 \tag{4-29}$$

$$\text{Piston Stroke} \quad 1.2(b-a) - a \le 0 \tag{4-30}$$

Support Location $\frac{D}{2} - X_2 \le 0$ (4-31)

where

$$R = \frac{\left| -X_2(X_3 \sin\theta + X_1) + X_1(X_2 - X_3\cos\theta) \right|}{\sqrt{(X_3\sin\theta + X_1)^2 + (X_2 - X_3\cos\theta)^2}}$$

$$F = \frac{\pi}{4} P D^2$$

$$a = \sqrt{(X_3 - X_2)^2 + X_1^2}$$

$$b = \sqrt{(X_3 \sin 45^o + X_1)^2 + (X_1^2 - X_3 \cos 45^o)^2}$$

$$\text{Volume of Oil, } V = \frac{\pi}{4} D^2 (b - a)$$

The DOT solutions are

PARAMETER	INITIAL VALUE	OPTIMUM Method=1	OPTIMUM Method=2	OPTIMUM Method=3
X1	84.0	50.78	50.86	69.27
X2	60.0	3.26	3.26	2.51
X3	84.0	120.00	120.0	111.1
D	6.0	6.52	6.52	5.41
OBJECTIVE	1584.4	1034.8	1035.3	1073.7
MAX G	-0.133	-3.0e-5	0.0	-1.2E-6
FUNCTIONS		94	41	79

LISTING 4-6: PISTON DESIGN FORTRAN PROGRAM.

```
C     SAMPLE PROGRAM.  PISTON DESIGN.
C
      DOUBLE PRECISION X(4),XL(4),XU(4),G(5),WK(800),RPRM(20),OBJ
      INTEGER IWK(200),IPRM(20),NRWK,NRIWK,I,METHOD,NDV,NCON,IPRINT,
     *MINMAX,INFO
C     DEFINE NRWK, NRIWK.
      NRWK=800
      NRIWK=200
C     ZERO RPRM AND IPRM.
      DO 10 I=1,20
         RPRM(I)=0.0
10       IPRM(I)=0
C     DEFINE METHOD,NDV,NCON.
      METHOD=1
      NDV=4
      NCON=5
C     DEFINE BOUNDS AND INTIAL DESIGN.
      DO 20 I=1,NDV
         XL(I)=.05
20       XU(I)=1000.
      XU(3)=120.
      X(1)=84.0
      X(2)=60.0
      X(3)=84.0
      X(4)=6.0
C     DEFINE IPRINT, MINMAX, INFO.
      IPRINT=1
      MINMAX=-1
      INFO=0
100   CALL DOT (INFO,METHOD,IPRINT,NDV,NCON,X,XL,XU,
     *OBJ,MINMAX,G,RPRM,IPRM,WK,NRWK,IWK,
     *NRIWK)
      IF(INFO.EQ.0)STOP
      CALL EVAL(OBJ,X,G)
      GO TO 100
      END
```

```
      SUBROUTINE EVAL (OBJ,X,G)
C     SUBROUTINE TO EVALUATE THE OBJECTIVE FUNCTION AND CONSTRAINTS
C     FOR THE PISTON DESIGN PROBLEM.
      DOUBLE PRECISION X(*),G(*),Q,AL,P,BNDMAX,PIO4,BO,B45,RNO,RDO,RO,
     *RN45,RD45,R45,OBJ,F
C     DEFINE LOAD, LENGTH, PRESSURE, AND ALLOWABLE BENDING MOMENT.
      Q=10000.
      AL=240.
      P=1500.
      BNDMAX=1.8E+6
C     CALCULATE THE OBJECTIVE FUNCTION AND CONSTRAINTS.
      PIO4=3.1415927/4.0
      BO=SQRT(ABS(X(3)-X(2))**2+X(1)*X(1))
      B45=SQRT(ABS(X(3)*SIN(PIO4)+X(1))**2+ABS(X(3)*COS(PIO4)-X(2))**2)
      RNO=-X(2)*X(1)+X(1)*(X(2)-X(3))
      RDO=SQRT(X(1)**2+(X(2)-X(3))**2)
      RO=ABS(RNO/RDO)
      RN45=-X(2)*(X(3)*SIN(PIO4)+X(1))+X(1)*(X(2)-X(3)*COS(PIO4))
      RD45=SQRT((X(3)*SIN(PIO4)+X(1))**2+(X(2)-X(3)*COS(PIO4)**2))
      R45=ABS(RN45/RD45)
      OBJ=((PIO4)*X(4)*X(4))*(B45-BO)
      F=PIO4*P*X(4)*X(4)
      G(1)=(Q*AL/(RO*F)-1.0)
      G(2)=(Q*AL*COS(PIO4)/(R45*F)-1.0)
      G(3)=Q*(AL-X(3))/BNDMAX-1.
      G(4)=1.2*(B45-BO)/BO-1.
      G(5)=.5*X(4)-X(2)
      RETURN
      END
```

4.8 Portfolio Selection

A bank has $10 million to invest. Securities available for purchase are [4].

Bond Name	Bond Type	Quality	Years To Mature	Yield	After Tax Yield
A	Municipal	2	9	4.3%	4.3%
B	Agency	2	15	5.4%	2.7%
C	Government	1	4	5.0%	2.5%
D	Government	1	3	4.4%	2.2%
E	Municipal	5	2	4.5%	4.5%

Restrictions on the investments are:

1. Government and agency bonds must total at least $4 million.
2. Average quality must be 1.4 or less.
3. Average years to maturity must be 5 years or less.

The bank wants to maximize after tax earnings.

Decision variables can be defined as follows:

X_a = Amount invested in A (in $millions)

X_b = Amount invested in B (in $millions)

X_c = Amount invested in C (in $millions)

X_d = Amount invested in D (in $millions)

X_e = Amount invested in E (in $millions)

The after tax earnings can be expressed as follows:

Objective = $0.043X_a + 0.027X_b + 0.025X_c + 0.022X_d + 0.045X_e$

The constraints are expressed as follows:

Total investments should not exceed $10 million

$$X_a + X_b + X_c + X_d + X_e \leq 10.0$$

At least $4 million must be invested in government and agency bonds

$$X_b + X_c + X_d \geq 4.0$$

Average quality (total quality / total volume) must not exceed 1.4

$$\frac{2X_a + 2X_b + X_c + X_d + 5X_e}{X_a + X_b + X_c + X_d + X_e} \leq 1.4$$

$$0.6X_a + 0.6X_b - 0.4X_c - 0.9C_d + 3.6X_e \leq 0$$

Average maturity must not exceed 5 yrs.

$$\frac{9X_a + 15X_b + 4X_c + 3X_d + 2X_e}{X_a + X_b + X_c + X_d + X_e} \leq 5.0$$

$$4X_a + 10X_b - X_c - 2X_d - 3X_e \leq 0.0$$

The problem can now be formally stated as

$$\text{Maximize } 0.043X_a + 0.027X_b + 0.025X_c + 0.022X_d + 0.045X_e \quad (4\text{-}32)$$

Subject to

$$X_a + X_b + X_c + X_d + X_e - 10.0 \leq 0.0 \quad (4\text{-}33)$$

$$4.0 - (X_b + X_c + X_d) \leq 0.0 \quad (4\text{-}34)$$

$$0.6X_a + 0.6X_b - 0.4X_c - 0.4X_d - 3.6X_e \leq 0.0 \quad (4\text{-}35)$$

$$4X_a + 10X_b - X_c - 2X_d - 3X_e \leq 0.0 \quad (4\text{-}36)$$

The DOT solutions are

PARAMETER	INITIAL VALUE	OPTIMUM Method=1	OPTIMUM Method=2	OPTIMUM Method=3
Xa	5.0	2.29	2.18	2.28
Xb	5.0	0.00	0.00	0.27
Xc	5.0	7.43	7.36	3.52
Xd	5.0	0.01	0.00	3.57
Xe	5.0	0.51	0.45	0.36
OBJECTIVE	0.810	0.308	0.298	0.288
MAX G	0.40	2.6E-3	0.0	1.5E-6
FUNCTIONS		183	30	38

LISTING 4-7: PORTFOLIO SELECTION FORTRAN PROGRAM.

```
C
C     SAMPLE PROGRAM.  PORTFOLIO SELECTION.
C
      DOUBLE PRECISION X(5),XL(5),XU(5),G(4),WK(800),RPRM(20),OBJ
      INTEGER IWK(200),IPRM(20),NRWK,NRIWK,I,NDV,NCON,METHOD,IPRINT,
     *MINMAX,INFO
C     DEFINE NRWK, NRIWK.
      NRWK=800
      NRIWK=200
C     ZERO RPRM AND IPRM.
      DO 10 I=1,20
         RPRM(I)=0.0
10       IPRM(I)=0
C     DEFINE NDV, NCON, METHOD.
      NDV=5
      NCON=4
      METHOD=1
C     DEFINE BOUNDS AND INITIAL DESIGN.
      DO 20 I=1,NDV
         X(I)=5.0
         XL(I)=0.0
20       XU(I)=10.
C     DEFINE IPRINT, MINMAX, INFO.
      IPRINT=1
      MINMAX=1
      INFO=0
100   CALL DOT (INFO,METHOD,IPRINT,NDV,NCON,X,XL,XU,
     *OBJ,MINMAX,G,RPRM,IPRM,WK,NRWK,IWK,
     *NRIWK)
      IF(INFO.EQ.0)STOP
      CALL EVAL(OBJ,X,G)
      GO TO 100
      END
      SUBROUTINE EVAL (OBJ,X,G)
C     SUBROUTINE TO EVALUATE THE OBJECTIVE FUNCTION AND CONSTRAINTS
C     FOR THE PORTFOLIO SELECTION PROBLEM.
      DOUBLE PRECISION X(*),G(*),OBJ
      OBJ=0.043*X(1)+0.027*X(2)+0.025*X(3)+0.022*X(4)+0.045*X(5)
      G(1)=(X(1)+X(2)+X(3)+X(4)+X(5)-10.)/100.
      G(2)=(4.-X(2)-X(3)-X(4))/100.
      G(3)=(0.6*X(1)+0.6*X(2)-0.4*X(3)-0.4*X(4)+3.6*X(5))/100.
      G(4)=(4.*X(1)+10.*X(2)-X(3)-2.*X(4)-3.*X(5))/100.
      RETURN
      END
```

4.9 Equality Constraints

This is an example of how equality constraints are formulated. Consider the following mathematical programming problem

$$\text{Minimize OBJ} = (X_1 + X_2)^2 + (X_2 + X_3)^2$$

$$\text{Subject to: } h_1 = X_1 + 2X_2 + 3X_3 - 1.0 = 0.0$$

Note that the constraint in this case is an equality constraint. The constraint function must equal zero. This problem is set up for DOT as follows

$$\text{Minimize OBJ} = (X_1 + X_2)^2 + (X_2 + X_3)^2 \tag{4-37}$$

Subject to;

$$g_1 = X_1 + 2X_2 + 3X_3 - 1.0 \leq 0.0 \tag{4-38}$$

$$g_2 = -(X_1 + 2X_2 + 3X_3 - 1.0) \leq 0.0 \tag{4-39}$$

or

$$g_1 = -g_2 \leq 0$$

The DOT solutions are

PARAMETER	INITIAL VALUE	OPTIMUM Method=1	OPTIMUM Method=2	OPTIMUM Method=3
X1	-4.00	0.488	0.332	0.500
X2	1.00	-0.488	-0.305	-0.501
X3	2.00	0.496	0.426	0.500
OBJECTIVE	18.00	6.7E-5	0.015	1.1E-7
MAX G	3.00	0.0	0.0	4.8E-4
FUNCTIONS		97	445	68

LISTING 4-8: EQUALITY CONSTRAINTS FORTRAN PROGRAM.

```
C
C     SAMPLE PROGRAM.  EQUALITY CONSTRAINTS.
C
      DOUBLE PRECISION X(3),XL(3),XU(3),G(2),WK(800),RPRM(20),OBJ
      INTEGER IWK(200),IPRM(20),NRWK,NRIWK,I,METHOD,NDV,NCON,IPRINT,
     *MINMAX,INFO
C     DEFINE NRWK, NRIWK.
      NRWK=800
      NRIWK=200
C     ZERO RPRM AND IPRM.
      DO 10 I=1,20
         RPRM(I)=0.0
         IPRM(I)=0
10    CONTINUE
C     DEFINE METHOD,NDV,NCON.
      METHOD=1
      NDV=3
      NCON=2
C     DEFINE BOUNDS AND INTIAL DESIGN.
      DO 20 I=1,NDV
         XL(I)=-100.
         XU(I)=100.
20    CONTINUE
       X(1)=-4.
       X(2)=1.
       X(3)=2.
C     DEFINE IPRINT, MINMAX, INFO.
      IPRINT=2
      MINMAX=-1
      INFO=0
100   CALL DOT (INFO,METHOD,IPRINT,NDV,NCON,X,XL,XU,
     *OBJ,MINMAX,G,RPRM,IPRM,WK,NRWK,IWK,NRIWK)
      IF(INFO.EQ.0)STOP
      CALL EVAL(OBJ,X,G)
      GO TO 100
      END
      SUBROUTINE EVAL (OBJ,X,G)
C
C     SUBROUTINE TO EVALUATE THE OBJECTIVE FUNCTION AND CONSTRAINTS
C     FOR THE EQUALITY CONSTRAINT PROBLEM.
C
      DOUBLE PRECISION X(*),G(*),OBJ
      OBJ=(X(1)+X(2))**2 + (X(2)+X(3))**2
      G(1)=X(1)+2.*X(2)+3.*X(3)-1.
      G(2)=-G(1)
      RETURN
      END
```

CHAPTER 5

References

- Introduction
- References

5.1 Introduction

There is much to be gained from a review of some of the basic optimization literature. While DOT 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., **Mulfidiscipline Design Optimization**, Vanderplaats Research & Development, Inc., Colorado Springs, CO, 2007.
2. Vanderplaats, G. N., "An Efficient Feasible Direction Algorithm for Design Synthesis," AIAA Journal, Vol. 22, No. 11, Nov. 1984.
3. Haftka, R. T., and Kamat, M. P., **Elements of Structural Optimization**, Nijhoff Publishers, Netherlands, 1985.
4. Ossenbruggen, P. J., **Systems Analysis for Civil Engineers**, John Wiley and Sons, N. Y., 1984.
5. Bradley, S. P., Hax, A. C. and Magnanti, T. L., **Applied Mathematical Programming**, Addison-Wesley, Mass. 1977.
6. Zangwill, W. I., **Nonlinear Programming: A Unified Approach**, Prentice-Hall, Englewood Cliffs, N.J., 1969.
7. Fletcher, R. and Reeves, C. M., "Function Minimization by Conjugate Gradients," Br. Computer J., Vol. 7, No. 2, pp. 149-154, 1964.
8. Broyden, C. G., "The Convergence of a Class of Double Rank Minimization Algorithms, Parts I and II," J. Inst. Math. Appl., Vol. 6, pp. 76-90, 222-231, 1970.
9. Fletcher, R., "A New Approach to Variable Metric Algorithms," Computer J., Vol. 13, pp. 317-322, 1970.
10. Goldfarb, D., "A Family of Variable Metric Methods Derived by Variational Means," Math. Comput., Vol. 24, pp. 23-26, 1970.
11. Shanno, D. F., "Conditioning of Quasi-Newton Methods for Function Minimization," Math. Comput., Vol. 24, pp. 647-656, 1970.
12. Kelly, J. E., "The Cutting Plane Method for Solving Convex Programs," J. SIAM, Vol. 8, pp. 702-712, 1960.
13. Dantzig, G. B., **Linear Programming and Extensions**, Princeton University Press, Princeton, N.J., 1963.
14. Dantzig, G. B. and Thapa, M. N., **Linear Programming 1: Introduction**, Springer, 1997.
15. Powell, M. J. D., "Algorithms for Nonlinear Constraints that use Lagrangian Functions," Math. Prog., Vol. 14, No. 2, pp. 224-248, 1978.
16. Vanderplaats, G. N. and Sugimoto, H., "Application of Variable Metric Methods to Structural Synthesis," Engineering Computations, Vol. 2, No. 2, June, 1985.

17. GENESIS User's Manuals, VMA Engineering, Colorado Springs, CO, 1999.

18. BIGDOT User’s Manual, Vanderplaats Research & Development, Inc., 2002.

19. Thomas, H., Vanderplaats, G. and Shyy, Y-K, “A Study of Move Limit Adjustment Strategies in the Approximation Concepts Approach to Structural Synthesis,” Proc. 4th AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, Holiday Inn, Cleveland, OH, Sept. 21-23, 1992, AIAA-92-4839.

20. Zoutendijk, K. G., **Methods of Feasible Directions**, Elsevier Publishing Co., Amsterdam, Netherlands, 1960.

APPENDIX **A**

Structure of Program Calling DOT

- **Introduction**
- **Basic Program Organization**
- **Structure of Program Interfacing with DOT**
- **Box Design Program in C Language Interfacing DOT Object Code Compiled in FORTRAN 77**

A.1 Introduction

The program given here may be used as a prototype as a main calling program for using DOT. All default parameters are defined prior to calling DOT. The defaults are contained in the **RPRM** and **IPRM** arrays. These are normally initialized to zero. Then, any over-ride values are set in the proper locations. For detailed information about using DOT 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.

A.3 Structure of Program Interfacing with DOT

FORTRAN Example.

```
      DOUBLE PRECISION X(NDV),XL(NDV),XU(NDV),G(NCON),WK(BIG),RPRM(30)
      INTEGER IWK(BIG),IPRM(20),NRWK,NRIWK,I,NDV,NCON,IPRINT,MINMAX,
     *METHOD,INFO
C     DIMENSIONS OF WK AND IWK.
      NRWK=?
      NRIWK=?
C     ZERO RPRM AND IPRM
      DO 10 I=1,20
        RPRM(I)=0.0
10    IPRM(I)=0
C     AT THIS POINT SET ANY ENTRIES OF RPRM AND IPRM
C     TO THEIR DESIRED VALUES IF THE DEFAULTS ARE
C     TO BE OVER-RIDDEN.
C     E.G.
C     RPRM(1)=-0.01
C     DEFINE NDV, NCON, IPRINT, MINMAX, METHOD
      NDV=?
      NCON=?
      IPRINT=?
      MINMAX=?
      METHOD=?
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
20    CALL DOT (INFO,METHOD,IPRINT,NDV,NCON,X,XL,XU,
     *OBJ,MINMAX,G,RPRM,IPRM,WK,NRWK,IWK,NRIWK)
C     EVALUATE OBJECTIVE AND CONSTRAINTS.  YOU MAY CALL ONE
C     OR MORE SUBROUTINES TO DO THIS.
      OBJ=?
      G(I)=?, I=1,NCON
      IF(INFO.GT.0) GO TO 20
C     OPTIMIZATION IS COMPLETE. OUTPUT RESULTS.
      STOP
      END
```

A.4 Box Design Program in C Language Interfacing DOT Object Code Compiled in FORTRAN 77

```
/* BOX DESIGN C MAIN PROGRAM */
/* MACHINE DEPENDENT DECLARATIONS */
/* PC */
#if defined (PC)
#define DOT DOT
extern void __stdcall DOT(int *INFO, int *METHOD, int *IPRINT,
            int *NDV, int *NCON, double *X, double *XL, double *XU,
            double *OBJ, int *MINMAX, double *G, double *RPRM, int *IPRM,
            double *WK, int *NRWK, int *IWK, int *NRIWK);
#endif
/* SUN and SGI Unix machines  */
#if defined (SUN) || defined (SGI)
#define DOT dot_
extern void DOT(int *INFO, int *METHOD, int *IPRINT,
            int *NDV, int *NCON, double *X, double *XL, double *XU,
            double *OBJ, int *MINMAX, double *G, double *RPRM, int *IPRM,
            double *WK, int *NRWK, int *IWK, int *NRIWK);
#endif
/*  IBM and HP Unix machines  */
#if defined (IBM) || defined (HP)
#define DOT dot
extern void DOT(int *INFO, int *METHOD, int *IPRINT,
            int *NDV, int *NCON, double *X, double *XL, double *XU,
            double *OBJ, int *MINMAX, double *G, double *RPRM, int *IPRM,
            double *WK, int *NRWK, int *IWK, int *NRIWK);
#endif
int main()
{
     int info, method, iprint, ndv, ncon, minmax, iprm[20], nrwk, iwk[200], nriwk;
        float x[5], xl[5], xu[5], obj, g[5],rprm[20],wk[800];
      int i;
/* define nrwk, nriwk */
            nrwk  = 800;
            nriwk = 200;
/* zero rprm and iprm arrays */
            for (i = 1; i <= 20; ++i) {
                  rprm[i - 1] = 0.0;
                  iprm[i - 1] = 0;
               }
/* modified method of feasible directions */
            method = 1;
/* three design variables */
            ndv = 3;
/* one constraint */
```

```
                ncon = 1;
/* define bounds and initial design */
                for (i = 1; i <= ndv; ++i) {
                        x [i-1] = 1.0;
                        xl[i-1] = 0.01;
                        xu[i-1] = 100.0;
                  }
/* print control */
                iprint = 1;
/* minimize */
                minmax  =-1;
/* initialize infor to zero */
                info = 0;
/* call DOT optimizer */
do
{
   dot_(&info, &method, &iprint, &ndv, &ncon, x, xl, xu, &obj, &minmax, g,
   rprm, iprm, wk, &nrwk, iwk, &nriwk);
/* evaluate objective and constraint */
                eval(&obj, x, g);
 }while ( info != 0);
return 0;
}
/* function to evaluate the objective function and constraints */

int eval(obj, x, g)
float *obj, *x, *g;
{
   --g;
   --x;
                *obj = x[2] * 2. * x[1] + x[3] * 2. * x[1] + x[2] * 4. * x[3];
                g[1] = 1. - x[1] * .5 * x[2] * x[3];
                return 0;
} /* end of program */
```

Compiling and Linking:

The C main program must be compiled using your C/C++ compiler. You should preferably link the object code of the C main program to DOT object code using a FORTRAN 77 linker. If you want to link using C linker, you must also link the FORTRAN 77 libraries, which means you must have FORTRAN 77 installed on your computer.

A Example of how to compile and link the box design program (boxdot.c) on a SGI workstation:

```
> cc -DSGI -c boxdot.c
> f77 -o boxdot boxdot.o dot.a
```

This will create the executable file “boxdot” which you may execute to perform optimization.

APPENDIX **B**

Calculating DOT Array Sizes

- **Introduction**
- **Unconstrained Problems (NCON=0)**
- **Constrained Problems (NCON > 0)**
- **DOT510 Storage Calculations**

B.1 Introduction

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

B.2 Unconstrained Problems (NCON=0)

METHOD = 1

NRWK = 16*NDV+NDV*(NDV+1)/2+50

NRIWK = 3*NDV+91

METHOD = 2

NRWK = 16*NDV+50

NRIWK = 3*NDV+91

B.3 Constrained Problems (NCON > 0)

The parameters NCOLA and NRB are used to determine internal storage needs for storing constraint gradients and solving the direction finding problem. In DOT, a single block of storage called **WK(**NRWK**)** is used to provide the needed space for real arrays and another single block of storage called **IWK(**NRIWK**)** is used to store integer arrays. NRWK is the storage needed for **WK** and **NRIWK** is the storage needed for IWK. Two estimates are made, desired and maximum.

The needed information is calculated using the following formulas.

Basic Real Storage:

NRWKMN = 16*NDV+8*NCON+50

NSIDE = NDV

For I = 1,NDV

DXI = (XU(I)-XL(I))/MAX(ABS(XL(I)),ABS(XU(I)),0.001)

If DXI < 0.01, NSIDE is increased by 1.

Calculating DOT Array Sizes

NCOLA and NRB desired values.

NCOLAD=MIN(NCON,2*NDV)

If NGMAX > 0.0, NCOLAD = MIN(NGMAX,NCON)

If NGMAX < 0, NGMAX = NCOLAD

NRBD = NCOLAD+NSIDE+1

NCOLA and NRB maximum values.

NCOLMX = NCON

NRBMX = NCOLMX + NSIDE + 1

Desired Basic Storage:

NSTRD=NRBD*(NRBD+2)+NDV*NGMAX

Maximum Basic Storage:

NCOLMX=NCON

NRBMX=NCOLMX+NSIDE+1

NSTRMX=NRBMX*(NRBMX+2)+NDV*NCOLMX

If METHOD=2

NRWKMN=NRWKMN+(NDV+2*NGMAX+2)*(NGMAX+2)

If METHOD=3

NRWKMN=NRWKMN+NGMAX*NDV+3*NCON+MAX(NDV,NCON)
+NDV*(NDV+1)/2

Desired Real Storage:

NRWKD = NRWKMN + NSTRD

Maximum Real Storage:

NRWKMX=NRWKMN+NSTRMX

Basic Integer Storage:

NRIWK=3*NDV+2*NCON+91

If METHOD=2

NRIWK=NRIWK+4*NDV+7*NCON+6

If METHOD=3

NRIWK=NRIWK+NCON

If the value of NRWK provided to DOT is less than NSTRD, or if NRIWK is less than NRIWK calculated here, the optimization will be terminated with an error message. If sufficient storage is available, NRWK should be dimensioned to equal (or greater than) NSTRMX.

B.4 DOT510 Storage Calculations

Alternatively, NRWK and NRIWK can be calculated by calling Subroutine DOT510.

The parameter list of SUBROUTINE DOT510 is as follows

SUBROUTINE DOT510 (NDV,NCON,METHOD,NRWK,NRWKMN,NRWKD,

*NRWKMX,NRIWK,NSTORE,NGMAX,XL,XU,NTUSD1,NTUSD2)

The input parameters to DOT510 are;

NDV - The number of design variables.

NCON - The number of constraints.

METHOD -The optimization method to be used.

XL and **XU** - Arrays containing the lower and upper bounds on the design variables, respectively.

NRWK -The dimensioned size of the **WK** array. NRWK may be input as zero. If NRWK is non-zero, DOT510 will attempt to adjust memory requirements to accommodate the dimensioned size of **WK**.

The output from DOT510 is;

NRWKMN - Same as NRWKD.

NRWKD - Desired dimension of the **WK** array. Normally, this is sufficient to achieve an optimum.

NRWKMX - Maximum dimension that can be used for the **WK** array. This amount of storage should be provided if storage is not an issue.

NRIWK- the required dimension of the **IWK** array.

NSTORE- The desired storage for gradients and arrays used in the direction finding problem. Used internally by DOT.

NGMAX - The maximum number of gradients that DOT will calculate or request.

NTUSD1, NTUSD2 are place holders so users of earlier DOT versions do not have to change the calling statement for DOT510.

Note that, if you are doing dynamic memory allocation in your program, you may call DOT510 directly to calculate the needed memory for the **WK** and **IWK** arrays.

APPENDIX C

In Case of Difficulty

- **Introduction**
- **Debugging Procedure**

C.1 Introduction

DOT is a robust numerical optimizer, and there should seldom be a case where no progress is made during the optimization. Also, numerous internal checks are made to avoid exponent overflows, divide by zero, and similar run-time errors.

Usually, when something seems wrong, it can be traced to the basic setup of the optimization problem or (more often) simple programming errors. Thus, while it is difficult to project all possible errors of this sort, some are common enough to be able to offer a short list of items to check.

C.2 Debugging Procedure

It is suggested that the following steps be followed in order to try to isolate problems:

1. Check all array dimension statements. Be sure the values of NRWK and NRIWK are correct. If your code is written in double precision, be sure you are using the double precision version of DOT.
2. Check the parameter list for calling DOT. Be sure that all parameters are present and in the proper order. A common error is to create a program with an editor that allows 80 column lines, while using a compiler that ignores all characters after column 72.
3. Turn off the automatic scaling and try again. Sometimes the scaling actually makes the conditioning of the problem worse. If the difficulties still exist, leave the scaling turned off during further testing.
4. Set the print control, IPRINT, to 5. This will cause the gradient information to be printed during the optimization. If the gradient of the objective or any constraint function has all zeroes, this parameter is not a function of the design variables. While it is theoretically possible to have a zero gradient, it is extremely rare on a digital computer. Check the problem formulation.
5. Check the order of magnitude of the components of the gradients. A well conditioned problem will have roughly the same order of magnitude values (within a factor of 100). If one term is several orders of magnitude greater that the others, it may help to scale this design variable by dividing by a number of that order of magnitude. If the components of the gradient of a constraint are all very large it will probably help to divide that constraint by a large number. A common error in problem formulation is to have a function, say Q, that must be less than QQ, where QQ is on the order of 10,000. In creating the constraint (which is required to be less than or equal to zero) we may write **G**(I) = Q - QQ. This will make the constraint very difficult to deal with by DOT, because Q must equal about 9,999.97 before the constraint is considered active. Therefore, it is important to normalize the constraint as **G**(I) = Q/QQ - 1.0. Now a constraint value of -0.01 will identify the constraint as being within one percent of being critical.
6. As a last resort, set IPRINT equal to 7 or IPRNT2 equal to 2. This will turn on print in the one-dimensional search. Plot the objective and constraint functions versus the move parameter, ALPHA. If one or more are extremely nonlinear, reformulation of

the problem by dividing that function by a large number is indicated. Another possibility here is that the finite difference gradient parameters, FDCH and FDCHM are either too large or too small. If the analysis is iterative, it often helps to try FDCH = 0.02 or larger and FDCHM = 0.01 or larger. This will mask the inaccuracies in the analysis. On the other hand, if the analysis is calculated very precisely as functions of the design variables, an order of magnitude smaller than the default value is indicated.

7. If the last resort fails, please contact VR&D for assistance. If possible, provide some printed output. Set IPRINT = 7 so all of the information will be there. We will do our best to help.

APPENDIX **D**

Internal Parameters in DOT

- **Introduction**
- **Parameters Contained in RPRM**
- **Parameters Contained in IPRM**

D.1 Introduction

In this Appendix, a description of the DOT internal parameters is given. While this is necessarily brief, it is somewhat more detailed than the cryptic information contained in Chapter 3.

The parameters are listed in the order they appear in the **RPRM** and **IPRM** arrays. If it is unlikely that the parameter should be changed from its default value, this is stated. Reference 1 describes the algorithms contained in DOT, and may be referred to for a more detailed description of how a parameter is used in a given algorithm.

D.2 Parameters Contained in RPRM

1. CT - This is the constraint tolerance. This parameter defines when a constraint is considered active.

   One of the key issues in constrained optimization is determining when a constraint is numerically "critical". If a constraint, **G**(I) is numerically greater than CT (CT is a negative number), it is considered critical for purposes of finding a new search direction or deciding if the optimum has been found. This is also why the constraint should be normalized to order of magnitude unity. Thus if **G**(I) is numerically greater than CT (say -0.03) then it is assumed to be within 3 percent of being critical. Numerically, this is considered to be an "active" constraint.

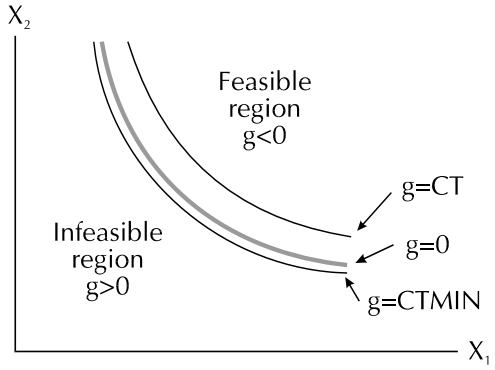
   For highly nonlinear problems, it is often helpful to make the value of CT more negative, say -0.05 or -0.10. By this method, the constraint is "trapped" sooner and the optimization process will direct the design away from this constraint. On the other hand, if the constraints are nearly linear, it may help to make CT closer to zero, say -0.01. Then, when interpolating for **G**(I) = 0.0, a more precise value of **G**(I) is obtained. In either case, the value of CT is progressively reduced during optimization to a value of -CTMIN, which is the value at which a constraint becomes strongly critical. In fact, if **G**(I) exceeds CTMIN (a positive number), the constraint is considered to be violated. (See the definition of CTMIN)

   If one or more constraints repeatedly become active on one iteration and inactive on the next, CT should be increased in magnitude (say try CT = -0.05 or -0.10), or the offending constraint should be divided by a factor of ten to reduce its sensitivity.

2. CTMIN -This is a constraint tolerance for defining when inequality constraints are violated. CTMIN is a small positive number. A constraint is considered inactive if its value is more negative than CT. If the constraint value is more positive than CTMIN, it is considered violated.

   Since, mathematically a constraint is violated any time its value is greater than zero, there may be a temptation to set CTMIN=0. However, this should not be done because the optimization algorithms interpolate on zero and some numerical bandwidth should be provided to allow for inaccuracies. The default value allows for about a half of a percent constraint violation for normalized constraints.

The geometric relationship between a constraint **G** and the parameters CT and CTMIN is shown in the following figure



3. DABOBJ - This is the absolute convergence criterion for optimization. If the objective function is changed by less than this value for ITRMOP consecutive iterations, the optimization will terminate. If the objective function changes by more than one order of magnitude during optimization, the default value for DABOBJ will probably cause premature convergence. In this case, it is usually desirable to set DABOBJ to a small number, say 0.001, and let the optimization process converge based on the relative change criterion defined by DELOBJ.

4. DELOBJ -This parameter is used in conjunction with DABOBJ. Here the convergence is tested on the relative change in the objective function. The combination of DABOBJ and DELOBJ work together to form the diminishing returns convergence criteria in DOT. Here by relative change we mean the fractional change in the value of the objective function between successive iterations.

   If the objective function is quite small in magnitude, a relative change of, say, one percent may not be meaningful. The absolute convergence criterion are relied on to detect convergence. On the other hand, for large values of the objective function, the absolute change is considered of lesser importance and the relative criterion tends to control the optimization convergence.

5. DOBJ1 -This is used in the one-dimensional search. On the first search, it is difficult to estimate a desirable move parameter, ALPHA, because the optimization process has no history. DOBJ1 is used to estimate the ALPHA which will reduce the objective function by this fraction, based on a linear approximation to the problem. Thus, for DOBJ1 = 0.1, the first step in the one-dimensional search will attempt to reduce the objective by ten percent.

   If the problem is highly nonlinear, so that the calculated ALPHA is consistently less than the proposed ALPHA, efficiency will be improved by reducing DOBJ1. Alternatively, if the calculated ALPHA is consistently greater than the proposed ALPHA, it is desirable to increase DOBJ1.

6. DOBJ2 -If the objective function is quite large in magnitude, a move to reduce the objective by the fraction DOBJ1 may be too large. In this case, DOBJ2 is used to limit the change in the objective function to the magnitude of DOBJ2. In other words, DOBJ1 is a fractional change and DOBJ2 is an absolute change. As with DOBJ1, if the proposed moves are too large, DOBJ2 may be reduced. If the proposed moves are too small, DOBJ2 may be increased.

   Both DOBJ1 and DOBJ2 are updated during the optimization process by keeping track of progress. Therefore, their initial values are usually not too critical except for highly nonlinear problems where no progress can be made due to very large estimates for ALPHA.

7. DX1, DX2 -These are used in the one-dimensional search. These parameters have an equivalent meaning to DOBJ1 and DOBJ2, but here are applied to each component of the **X** vector. The same general rules apply. The purpose of DX1 and DX2 is to prevent very large initial changes in the components of the X vector. DX1 and DX2 are also updated during the optimization process.

8. FDCH -Used if **IPRM**(1) = 0 for internal gradient calculations by DOT. Gradients are calculated by first forward finite difference unless a variable is at its upper bound. In this case, a first backwards finite difference step is taken and no check is made to insure that the resulting design variable is above its lower bound. FDCH is the finite difference step size as a fraction of the design variable being perturbed.

   If high precision is available and required in evaluating the objective and constraint functions, FDCH should be reduced. If the analysis is iterative, with its own internal convergence parameters, FDCH may have to be increased. For iterative analysis, a value of FDCH up to 0.05 may be appropriate for constrained problems, but FDCH = 0.02 is a more reasonable limit for unconstrained problems. The reason for this is that DOT seeks the point where the gradient is zero for unconstrained problems, and if FDCH is large, this is numerically difficult and will lead to false gradient information. On the other hand, for constrained problems, the gradients of the objective and critical constraints are usually non-zero at the optimum. Hence, precision is less important for constrained problems.

9. FDCHM -This is used if **IPRM**(1) = 0 for internal gradient calculations by DOT. This is the minimum absolute step size for gradient calculations. This is used if the component of **X** is near zero since a fractional change may not be meaningful. The same general rules apply as with FDCH.

10. RMVLMZ - This is the relative change allowed in each design variable during the first iteration of the Sequential Linear Programming and Sequential Quadratic Programming Methods. Because the objective and constraint functions are approximated by a first order Taylor Series expansion, the information is only valid in the region of the approximation. Thus, we do not allow the design variables to change too much for each approximation. Depending on the progress of the optimization, this parameter will be updated. If the approximate optimization does not lead to significant constraint violations, this parameter is unchanged. However, if after an approximate optimization, one or more constraints are found to be violated more than they were at the beginning of the approximate optimization, RMVLMZ is reduced by 50% (this is only done after the second and subsequent iterations of the SLP method since on the first iteration it is common to go into the infeasible region).

    If the problem is known to be linear, RMVLMZ should be increased to a large value, say 100. If the problem is found to be quite nonlinear (the more common case), RMVLMZ should be reduced to 0.2 or even 0.1.

    Move limits on the individual design variables are internally adjusted, based on whether the design variable is consistently changing in one direction or if its value is oscillating [19.].

11. DABSTR - This parameter has the same meaning as DABOBJ, but here it is applied to convergence of the Sequential Linear Programming and Sequential Quadratic Programming Methods. That is, if the absolute value of the objective does not change by more than this amount between two consecutive solutions of the approximate optimization problem, this criterion is considered satisfied. Normally this parameter does not need to be changed.

12. DELSTR - This parameter has the same meaning as DELOBJ, but here it is applied to convergence of the Sequential Linear Programming and Sequential Quadratic Programming Methods. That is, if the relative value of the objective does not change by more than this amount between two consecutive solutions of the approximate optimization problem, this criterion is considered satisfied. Normally this parameter does not need to be changed.

13. GSTOL - This is the Golden Section tolerance as a fraction of the initial bounds determined in the one-dimensional search. This is used by the Modified Method of Feasible directions. If METHOD=1, the default is to not use the Golden Section method. If METHOD=3, the Golden Section Method will be used in the Sub-Problem because the function values are cheap. During the one-dimensional search, DOT first finds bounds on the minimum. Then the bounds are refined if the Golden Section Method is turned on (GSTOL<1.0). After the bounds are reduced to the GSTOL fraction of the initial bounds, polynomial interpolation is used to obtain the final value of ALPHA.

14. GSTOLM - This is the minimum tolerance that will be allowed in the Golden Section Method. As the optimization process proceeds, the proposed value of ALPHA can become quite small. At this point, further refinement by the Golden Section Method is not meaningful. Thus the Golden Section Method will be bypassed if ALPHA is very small and polynomial interpolation will be used.

D.3 Parameters Contained in IPRM

1. IGRAD -Specifies whether the gradients of the objective function and the constraints are calculated by DOT using finite difference methods (IGRAD = 0) or are supplied by the user (IGRAD = 1). If the gradients are readily available then directly providing gradients can save much computer time.

2. ISCAL -Specifies whether the design variables, objective and constraint functions, and gradients are to be scaled by DOT (ISCAL = 0 or greater) or if the problem is to remain unscaled (ISCAL = -1). The problem is actually re-scaled every ISCAL iterations where the default (if the user sets ISCAL = 0) is NDV. While there has not been much research into the theory of scaling, practical experience shows that scaling often serves to improve the conditioning of many problems. It should be noted, however, that sometimes scaling actually makes a problem worse. In this case the scaling should be turned off by setting ISCAL = -1 (ISCAL = 0 is the default).

3. ITMAX -Maximum number of iterations in the optimizer. If function evaluations are extremely expensive, reduce ITMAX. In the extreme case ITMAX = 1 or 2 is justified because the first few iterations are where most progress is made. If function evaluations are not expensive and the optimization terminates by reaching ITMAX, it should be increased.

4. ITRMOP -The number of consecutive iterations that must satisfy the absolute or relative convergence criteria before optimization is terminated in the Modified Method of Feasible Directions or the BFGS or Fletcher-Reeves Methods. Usually ITRMOP should be at least 2 because it is common to make little progress on one iteration, only to make major progress on the next. Therefore, the default of ITRMOP = 2 will allow a second try before terminating. If progress toward the optimum seems slow, but consistent, and function evaluations are not too expensive, it may improve the solution to increase ITRMOP to a value of 3 to 5.

5. IWRITE - File number to which DOT output will be sent. The default value is 6.

6. NGMAX - This is the maximum number of columns in the A-Matrix. This matrix is used to store the gradients of all active, violated, and near active constraints. The A-Matrix is stored in the **WK** array and so storage must be allocated for it. Ideally, NGMAX = NCON since it is conceivable that all constraints are active or violated. On the other hand, theoretically there should never be more than NDV active or violated constraints since this would define a “fully constrained” problem. However, this rule is often violated at the start because we do not have a good feasible starting point.

   The reason that the value of NGMAX is a concern is that we may have only a few design variables but thousands of constraints. Therefore, if we set NGMAX = NCON, this would require a large amount of storage. Normally, it is best do dimension **WK** and **IWK** as large as possible and then let DOT use the default value for NGMAX. If a storage error occurs, NGMAX (and if necessary, the dimensions of **WK** and **IWK**) should be increased if possible. Otherwise, choose a different starting design in an attempt to reduce the number of critical constraints.

7. IGMAX -

8. JTMAX - This is the maximum number of iterations allowed in the Sequential Linear Programming and Sequential Quadratic Programming Methods. It is equivalent to ITMAX for the Modified Method of Feasible Directions. This is only used if METHOD=2 or 3 and NCON>0. When using these methods, ITMAX should be set to it's default value since solution of the approximate sub-problem is not usually expensive. If function values are very expensive, it is suggested to set JTMAX to a value of 1 or 2 to judge the progress of the optimization before continuing.

9. ITRMST - The number of consecutive iterations that must satisfy the absolute or relative convergence criteria before optimization is terminated in the Sequential Linear Programming and Sequential Quadratic Programming Methods. This is equivalent to ITRMOP, but now defines the number of approximate problems that will be solved before exiting. Usually ITRMST should be at least 2 because it is common to make little progress on one iteration, only to make major progress on the next. Therefore, the default of ITRMOP = 2 will allow a second try before terminating. If progress toward the optimum seems slow, but consistent, and function evaluations are not too expensive, it may improve the solution to set ITRMOP to a value of 3 or 4.

10. JPRINT - This is a debugging print control with the Sequential Linear Programming and Sequential Quadratic Programming Methods. Normally it should remain at its default value of 0. If JPRINT>0, IPRINT will be turned on during the sub-optimization problem. This will generate a considerable amount of output and should be used only for debugging purposes. Normally, this sub-problem is solved reliably and so it is not desirable to monitor its progress.

11. JWRITE - This is a file number for outputting information about the optimization history. If JWRITE=0, no output will be provided. IF JWRITE>0, it is assumed that the user has opened the file before calling DOT. DOT will then output the iteration history to this file. This is useful if you wish to create a graphics file showing the optimization progress or if you wish to output a summary of the optimization process in tabular form. In version 1.xx of DOT, this information was difficult to get because when DOT returned to the calling program, at the beginning of a new iteration, the values of **X**, **G** and OBJ were often different than they were on the last return. This is because, during the one-dimensional search, the last design investigated may be rejected in favor of an earlier design. The information on file JWRITE is consistent and includes the initial design information as well as the results at the end of each design iteration.

    If the Modified Method of Feasible Directions, BFGS or Fletcher-Reeves method is used, this information is written at the end of each one-dimensional search. If the Sequential Linear Programming or Sequential Quadratic Programming Method is used, this information is written at the end of each approximate optimization.

12. NSTORE - Storage available for solving the direction finding problem in the Modified Method of Feasible Directions or for Gradient storage in the sub-problem in the Sequential Linear Programming or Sequential Quadratic Programming methods.

13. IERROR - Error flag. On return from DOT, if IERROR > 0, an error has occurred and the optimization should be terminated.

14. NEWITR - This parameter is not input to DOT, but is calculated internally. NEWITR will have a value of -1 or n. If it's value is n, optimization iteration number

'n' has just ended and a new iteration has begun (or if INFO=0, the optimization is complete). The value of n = 0 is returned to indicate that the initial design has just been evaluated (this is the result of iteration 0).

If finite difference methods are used to calculate gradients, the first term in the **X** vector will already be changed when NEWITR = -1 (NEWITR = -1 will be returned NDV times during gradient calculations). Therefore, it is difficult for the user to interact to create graphics files or to monitor the optimization progress. NEWITR provides a means of determining exactly what is happening and acting accordingly.

If JWRITE>0, the current optimization information has just been written to that file. Therefore, this is a good place to either interrogate that file or to interrupt and later restart the program.

15. NGT - This parameter is not input to DOT, but is calculated internally. NGT is the number of constraints for which gradients are needed and it is only meaningful if the user provides gradients. NGT will usually be much less than NCON since gradients of all constraints are usually not needed. The first NGT locations of the **IWK** array identify the constraint numbers for which gradients are required. The user must, in turn, store these gradients in the **WK** array, following the gradient of the objective function.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V