NASA/CR-2002-211768

Very Large Scale Optimization

Garrett Vanderplaats
Vanderplaats Research and Development, Inc., Colorado Springs, Colorado

August 2002

NASA/CR-2002-211768

Very Large Scale Optimization

Garrett Vanderplaats
Vanderplaats Research and Development, Inc., Colorado Springs, Colorado

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NAS1-00102

August 2002

VERY LARGE SCALE OPTIMIZATION

1.0 Introduction

Schmit [1] introduced numerical optimization to the engineering community in 1960 by solving a two variable structural optimization problem. This was the classical 3-bar truss and it represented the first time finite element analysis and nonlinear optimization was combined into a single program. Since that time, thousands of research papers have been published on this subject and the optimization algorithms have been continually enhanced and refined, along with the methods for formulating and solving engineering problems. This is particularly true in the area of structural optimization with the use of formal approximations [2 -4]. Today, several commercial structural optimization programs are available to solve a wide range of design tasks.

Figure 1 shows the general trend in problem size in engineering since 1960 and projected to 2010. While there is considerable scatter in the actual problem size solved over the years (for example, results with hundreds of variables using optimality criteria were published in the 1970s), the general growth in broad based applications has been as shown.

With this increased problem size, we have exceeded the capabilities of most current optimization algorithms and this has motivated the effort reported here. An interesting outcome of this study has been that we have re-created a modern version of methods popular in the 1960s.

The principal difficulties with present methods are that, as problem size grows, they require considerable computer memory and they internally take a great deal of computational effort.

The BIGDOT software created here overcomes the key problems encountered with existing software and also offers some very attractive features.

1. It requires very little computer memory.
2. It does not solve a complex and time intensive direction finding problem.
3. It handles redundant constraints (constraints with the same values and gradients) easily.
4. It deals with equality constraints with no loss in efficiency or reliability.
5. It solves discrete variable problems to efficiently produce a "good" discrete solution.
6. It scales very well. That is, the efficiency is about the same regardless of problem size.

The BIGDOT program has been tested extensively, both on mathematical test cases and on real structural optimization problems. As shown in Figure 1, we have already solved an optimization problem in excess of 135,000 variables. This was a topology optimization problem where we maximized stiffness subject to mass limits using the GENESIS [5] software from Vanderplaats Research & Development, Inc. (VR&D). Additionally, as shown in the example section, we have solved 50,000 variable fully constrained problems.

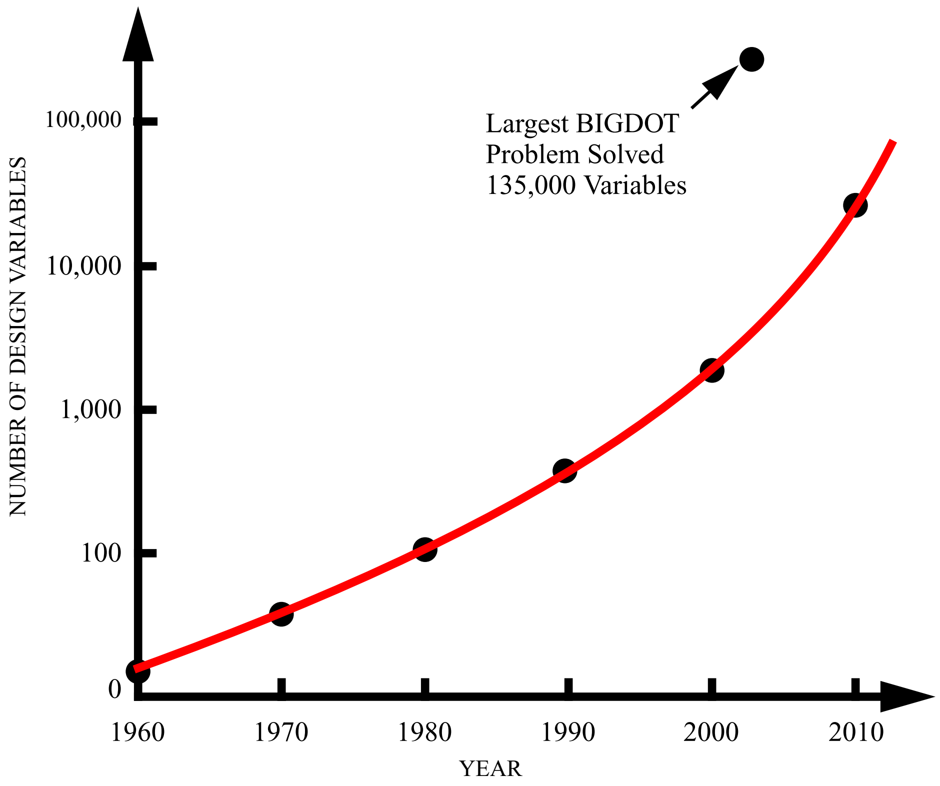


Figure 1 Trends in Engineering Optimization Problem Size

The original goal of this project was to develop methods for solving structural optimization problems with very large numbers of design variables and large numbers of critical constraints. As the project progressed, it became apparent that the methods being developed were applicable to general optimization tasks and not just structural optimization. Therefore, the resulting BIGDOT program is not restricted in any way to structural applications.

The basic requirements were to develop methods for solving nonlinear constrained optimization problems in excess of 10,000 variables. Key issues include efficiency, reliability, memory requirements, and gradient calculation requirements.

As part of this effort, an algorithm was developed to include discrete variables in the optimization task. Based on the success of this part of the effort, the resulting program was modified to consider strictly unconstrained problems in addition to the original target of dealing with constrained problems.

Based on the theoretical developments, a general purpose optimization program was created, called BIGDOT, which is already used by the GENESIS structural optimization software from VR&D.

Here, we will first define the general optimization problem and identify candidate algorithms. We will then focus on the final algorithms that were chosen for software development. Test cases are provided to demonstrate the efficiency and reliability of the methods and software created here. It is concluded that we now have commercial optimization software capable of solving nonlinear constrained optimization problems with tens of thousands of design variables and without any intrinsic limits on the number of constraints.

2.0 General Problem Statement

The general nonlinear optimization problem is [6];

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

Subject to;

$$g_j(\mathbf{X}) \le 0 \qquad j = 1, m \qquad \text{General Linear and Nonlinear Inequality Constraints} \tag{2}$$

$$h_k(\mathbf{X}) = 0 \qquad k = 1, l \qquad \text{General Linear and Nonlinear Equality Constraints} \tag{3}$$

$$X_i^L \le X_i \le X_i^U \qquad i = 1, n \qquad \text{Side Constraints} \tag{4}$$

$$X_i \in \mathbf{S}_i \tag{5}$$

The objective and constraint functions may be linear or nonlinear. The inequality constraints defined by Equation 2 and equality constraints defined by Equation 3 may or may not exist. That is, the problem may be unconstrained. The side constraints defined by Equation 4 also may or may not exist. If side constraints are not needed for one or more variables, their values are set to be a large negative number for the lower bound and large positive number for the upper bound. Side constraints may be included for unconstrained functions.

Equality constraints defined by Equation 3 are not considered explicitly here. Instead, equality constraints may be included as two equal and opposite inequality constraints.

The design variables can be contained in a set of allowed values given by Equation 5. The set of allowed values may be a general set of discrete values or may be integer values. If a set of allowed values is not provided for a given design variable, that variable is continuous. Thus, any design variable may be continuous, discrete or integer.

This definition of the design problem is extremely general. Prior to this project, we were able to routinely solve such problems with (typically) up to a few hundred variables with no specific limit on the number of constraints. The methods and software developed here extend the problem size to tens of thousands of design variables, also with no specific limit on the number of constraints. The price paid for this is that the optimization efficiency, as measured by the number of function and gradient evaluations, is about twenty to thirty percent of that of existing methods.

3.0 Present Methods

Presently, the DOT optimizer from VR&D contains three algorithms [7];

1. Modified Method of Feasible Directions (MMFD).
2. Sequential Linear Programming (SLP).
3. Sequential Quadratic Programming (SQP).

Each of these methods has its own attractive features. However, they all have two things in common;

1. They require storage of the gradients of many constraints.
2. They each require solution of a sub-problem, either to find a search direction or (in the case of SLP and SQP) to solve a sub-optimization problem.

This is not a significant issue if there are a large number of design variables, but only a few critical or near critical constraints, including side constraints. However, if there are a large number of critical or near critical constraints (including side constraints), both the storage requirements and the computational costs rise quite dramatically. Also, for such problems, experience shows that reliability diminishes as the number of active constraints increases.

As optimization becomes more accepted by the engineering community, the problem size has rapidly increased, quickly reaching the limits of current methods.

Therefore, it is reasonable to address methods that may not be as efficient, in terms of the number of function and gradient calculations, as the three methods listed above, but require relatively little computational time in the direction-finding sub-problem, and with limited storage. This was the motivation for the present study.

4.0 Candidate Methods

The Sequential Unconstrained Minimization Technique (SUMT) approach offers a way to solve very large problems with minimum memory requirements while avoiding solution of a large direction finding sub-problem. Here, we identify several SUMT approaches which were studied in order to find the most reliable method. Reliability is more important than efficiency. Thus, it is recognized that while the cost of solving the optimization problem will grow, we will be able to solve problems of a size in excess of anything possible today. Interest in the Operations Research community has recently focused on SUMT, with emphasis on interior point methods, for solving large problems [8]. For nonlinear engineering problems considered here, we have found that a revised exterior penalty function techniques meets our needs best.

Sequential Linear Programming (SLP) with adjustable move limits deserves note here since such an approach is reasonably reliable and we are able to solve very large LP problems [9, 10]. However, while reliable, very large LP problems can themselves become inefficient. More importantly, in structural optimization we have high quality <u>nonlinear</u> approximations available. Therefore, we would like to utilize the high quality of the approximations if possible, rather than losing the higher order information that is available. While we could sequentially linearize the nonlinear approximations, this would add another iterative loop with limited reliability, and so this is not considered to be a viable approach.

4.1 Sequential Unconstrained Minimization Techniques (SUMT)

SUMT was a very popular approach in the 1960's, where the original constrained problem was converted to a sequence of unconstrained problems [11]. This was a natural approach because unconstrained minimization techniques were reasonably well developed. As research in optimization algorithms progressed, other methods were shown to be more efficient and reliable for "Typical" optimization problems. However, as problem size has grown, it has been found that the more modern methods can become computationally inefficient. Thus, especially for structural optimization using approximation techniques, a new look at SUMT is appropriate.

The basic SUMT approach is to;

$$\text{Minimize } \Phi(\mathbf{X}) = F(\mathbf{X}) + P(\mathbf{X}) \tag{6}$$

where $\Phi(\mathbf{X})$ is called the *pseudo-objective function* and the penalty term, P(**X**), depends on the method being used. This general approach is described in some detail in Reference 6. The key distinction of each method is the form of P(**X**).

During this project, the following methods were considered;

1. Exterior Penalty Function
2. Interior Penalty Function, Reciprocal
3. Interior Penalty Function, Original Log Barrier
4. Interior Penalty Function, Polyak's Log Barrier
5. Interior Penalty Function, Polyak's Log-Sigmoid
6. Interior Penalty Function, Linear Extended Penalty
7. Interior Penalty Function, Quadratic Extended Penalty
8. Interior Penalty Function, Variable Extended Penalty
9. Augmented Lagrange Multiplier Method
10. Duality Theory

4.1.1 Exterior Penalty Function

The exterior penalty is the easiest to incorporate into the optimization process. Here the penalty function P(**X**) is typically given by [6]

$$P(\mathbf{X}) = r_p \sum_{j=1}^{m} \{\mathrm{MAX}[0, g_j(\mathbf{X})]\}^2 + r_p \sum_{k=1}^{l} [h_k(\mathbf{X})]^2 \tag{7}$$

The subscript p is the outer loop counter which we will call the cycle number.

We begin with a small value of the penalty parameter, r_p, and minimize the pseudo-objective function, $\Phi(\mathbf{X})$. We then increase r_p and repeat the process until convergence.

4.1.2 Interior Penalty Function

The basic concept of the interior penalty is that, as the constraints approach zero (from the negative side), the penalty grows to drive the design away from the constraint bounds [11].

Reciprocal

In the past, a common penalty function used for the interior method was

$$P(\mathbf{X}) = \sum_{j=1}^{m} \frac{-1}{g_j(\boldsymbol{X})} \tag{8}$$

Using Equation 6 and including equality constraints via the exterior penalty function of Equation 7,

$$\Phi(\mathbf{X}, r'_p, r_p) = F(\mathbf{X}) + r'_p \sum_{j=1}^{m} \frac{-1}{g_j(\mathbf{X})} + r_p \sum_{k=1}^{l} [h_k(\mathbf{X})]^2 \tag{9}$$

Here r'_p is initially a large number, and is decreased as the optimization progresses.

The last term on Equation 9 is the exterior penalty as before, because we wish to drive $h_k(\mathbf{X})$ to zero. Also, r_p has the same definition as before and $F(\mathbf{X})$ is the original objective function.

In our remaining discussion, we will omit the equality constraints, remembering that they are normally added as an exterior penalty function as in Equation 9.

Log Barrier Method

An alternative form of Equation 9 is

$$P(\mathbf{X}) = r'_p \sum_{j=1}^{m} -\log[-g_j(\mathbf{X})] \tag{10}$$

and this is often recommended as being slightly better numerically conditioned.

Polyak's Log Barrier Method

Polyak [8, 12] suggests a modified log-barrier function which has features of both the extended penalty function and the Augmented Lagrange multiplier method described in Section 4.1.4.

The modified logarithmic barrier penalty function is defined as:

$$M(X, r'_p, \lambda^p) = -r'_p \sum_{j=1}^{M} \lambda_j^p \log\left[1 - \frac{g_j(X)}{r'_p}\right] \tag{11}$$

where the nomenclature has been changed to be consistent with the present discussion.

Using this, we create the pseudo-objective function

$$\Phi(X, r'_p, \lambda^p) = F(X) - r'_p \sum_{j=1}^{m} \lambda_j^p \log\left[1 - \frac{g_j(X)}{r'_p}\right] \tag{12}$$

We only consider inequality constraints and ignore side constraints. Equality constraints can be treated using the exterior penalty function approach and side constraints can be considered directly in the optimization problem. Alternatively, equality constraints can be treated as two equal and opposite inequality constraints because this method acts much like an extended penalty function method, allowing for constraint violations.

The first order optimality conditions are:

$$\nabla_x \Phi(\mathbf{X}, r'_p, \lambda^p) = \nabla_x F(\mathbf{X}) + \sum_{j=1}^{m} \frac{\lambda_j^p}{\left[1 - \frac{g_j(\mathbf{X})}{r'_p}\right]} \nabla_x g_j(\mathbf{X}) \tag{13}$$

The update for the Lagrange multipliers is:

$$\lambda_j^{p+1} = \frac{\lambda_j^p}{\left[1 - \frac{g_j(\mathbf{X})}{r'_p}\right]} \quad \text{if} \quad g_j(\mathbf{X}) \le k r'_p \tag{14}$$

$$\lambda_j^{p+1} = \frac{\lambda_j^p}{2r'_p\,(1-k)} \quad \text{if} \quad g_j(\mathbf{X}) > k r'_p \tag{15}$$

where k <1.0. Typically, k = 0.8.

By using the $\log\left[1 - \frac{g_j(\mathbf{X})}{r'_p}\right]$ we allow $g_j(\mathbf{X})$ to be as positive as r'_p . This dictates that our initial choice of r'_p must be greater than the maximum violated constraint value. If no constraints are violated, we can pick an initial r'_p to make the value of the objective function and the penalty function equal.

When solving the unconstrained sub-problem, care must be taken with the one-dimensional search to insure that we are always inside the barrier, where the penalty function becomes undefined, just as in the case of the interior penalty function method.

This method has been tested by the author and others using quadratic functions and it has been shown to have strong theoretical characteristics for this class of problems.

Polyak's Log-Sigmoid Method

A more recent method by Polyak, et al [13] appears to have better properties than the log-barrier method by eliminating the barrier. Here, we create the penalty function as;

$$P(\mathbf{X}) = \frac{2}{r_p} \sum_{j=1}^{m} \lambda_j^p \{ \ln[1 + e^{r_p g_j(\mathbf{X})}] - \ln(2) \} \tag{16}$$

The update for the Lagrange multipliers is:

$$\lambda_j^{p+1} = \frac{2\lambda_j^p}{[1 + e^{-r_p g_j(\mathbf{X})}]} \tag{17}$$

4.1.3 Extended Interior Penalty Function

This approach attempts to incorporate the best features of the reciprocal interior and exterior methods for inequality constraints. For equality constraints, the exterior penalty is used as before and so is omitted here for brevity.

The Linear Extended Penalty Function

The first application of extended penalty functions in engineering design is attributable to Kavlie and Moe [14]. This concept was revised and improved upon by Cassis and Schmit [15]. Here the penalty function used in Equation 6 takes the form

$$P(\mathbf{X}) = \sum_{j=1}^{m} \tilde{g}_j(\mathbf{X}) \tag{18}$$

where

$$\tilde{g}_j(\mathbf{X}) = -\frac{1}{g_j(\mathbf{X})} \qquad \text{if } g_j(\mathbf{X}) \le \varepsilon \tag{19}$$

$$\tilde{g}_j(\mathbf{X}) = -\frac{2\varepsilon - g_j(\mathbf{X})}{\varepsilon^2} \qquad \text{if } g_j(\mathbf{X}) > \varepsilon \tag{20}$$

The parameter ε is a small negative number which marks the transition from the interior penalty given by Equation 8 to the extended penalty given by Equation 20.

If a sequence of improving feasible designs is to be produced, it is necessary to choose ε so that the pseudo-objective function $\Phi(\mathbf{X}, r'_p)$ has a positive slope at the constraint boundary. Haftka and Starnes [18] recommend that ε be defined by

$$\varepsilon = -C(r'_p)^a \qquad \frac{1}{3} \le a \le \frac{1}{2} \tag{21}$$

where *C* is a constant. At the beginning of the optimization, they choose ε in the range $-0.3 \le \varepsilon \le -0.1$. Also, they pick the initial r'_p so that the two terms on the right-hand side of Equation 6 are equal. This defines the value of *C* to be used in Equation 21. Now, at the beginning of each unconstrained minimization, ε is defined by Equation 21 and is kept constant throughout that unconstrained minimization. While this definition of ε was created in conjunction with the quadratic extended penalty function, it works here also. Equation 21 provides the desired feature of maintaining the minimum of the unconstrained function inside the feasible region. A similar approach which was developed for use in the linear extended penalty function method is given by Cassis [16, 17].

The Quadratic Extended Penalty Function

The extended penalty given by Equations 18 - 20 is defined as the linear extended penalty function. This function is continuous and has continuous first derivatives at $g_j(\mathbf{X}) = \varepsilon$. However, the second derivative is discontinuous, and so if a second-order method is used for unconstrained minimization, some numerical problems may result. Haftka and Starnes [18] overcome this by creating a quadratic extended penalty function as

$$\tilde{g}_j(\mathbf{X}) = -\frac{1}{g_j(\mathbf{X})} \qquad \text{if } (g_j(\mathbf{X}) \le \varepsilon) \tag{22}$$

$$\tilde{g}_j(\mathbf{X}) = \frac{-1}{\varepsilon}\left\{\left[\frac{g_j(\mathbf{X})}{\varepsilon}\right]^2 - 3\left[\frac{g_j(\mathbf{X})}{\varepsilon}\right] + 3\right\} \qquad \text{if } (g_j(\mathbf{X}) > \varepsilon) \tag{23}$$

Equations 22 and 23 may be most useful if a second-order method is used for unconstrained minimization. However, the price paid for this second-order continuity is that the degree of nonlinearity of the pseudo-objective function is increased.

The Variable Penalty Function

Both the exterior and interior penalty function methods have been used with success. The significant modifications to these traditional methods have been related to improving the numerical conditioning of the optimization problem, as exemplified by the extended interior methods. A formulation is presented by Prasad [19] which offers a general class of penalty functions and also avoids the occurrence of extremely large numerical values for the penalty associated with large constraint violations.

The variable penalty function approach creates a penalty which is dependent on three parameters: *s*, *A*, and ε as follows:

For $s \neq 1$,

$$\tilde{g}_j(\mathbf{X}) = \frac{[-g_j(\mathbf{X})]^{1-s}}{s-1} \qquad \text{if } (g_j(\mathbf{X}) \le \varepsilon) \tag{24}$$

$$\tilde{g}_j(\mathbf{X}) = \left(A\left[\frac{g_j(\mathbf{X})}{\varepsilon} - 1\right]^3 + \frac{s}{2}\left[\frac{g_j(\mathbf{X})}{\varepsilon} - 1\right]^2 - \left[\frac{g_j(\mathbf{X})}{\varepsilon} - 1\right] + \frac{1}{s-1}\right)(-\varepsilon)^{1-s}$$

$$\text{if } (g_j(\mathbf{X}) > \varepsilon) \tag{25}$$

and for $s = 1$,

$$\tilde{g}_j(\mathbf{X}) = -\log[-g_j(\mathbf{X})] \qquad \text{if } (g_j(\mathbf{X}) \le \varepsilon) \tag{26}$$

$$\tilde{g}_j(\mathbf{X}) = A\left[\frac{g_j(\mathbf{X})}{\varepsilon} - 1\right]^3 + \frac{1}{2}\left[\frac{g_j(\mathbf{X})}{\varepsilon} - 1\right]^2 - \left[\frac{g_j(\mathbf{X})}{\varepsilon} - 1\right] - \log(-\varepsilon) \qquad \text{if } (g_j(\mathbf{X}) > \varepsilon) \tag{27}$$

The parameters *s* and *A* are used to create a class of penalty functions. That is, Equations 24 and 25 includes each class of extended penalty functions we have discussed.

The small negative number ε is defined the same as for the extended penalty function method and is the transition to the extended penalty. Prasad recommends that ε be defined as follows:

$$\varepsilon = -\beta(r'_p)^q \tag{28}$$

$$\text{where} \quad \frac{1}{2+s} \le q \le \frac{1}{s} \qquad \text{for} \qquad s > 0 \tag{29}$$

and β is a positive constant chosen so ε is initially near zero, say ε = – 0.01. Note the similarity of Equations 21 and 28.

The final parameter *A* controls the shape of the extended penalty. On the first iteration of the unconstrained minimization subproblem

if all $g_j(\mathbf{X}) \le 0$

$$A = \frac{1+s}{3} \tag{30}$$

and if some $g_j(\mathbf{X}) > 0$

$$A = \frac{1 - s(c^*/\varepsilon - 1)}{3(c^*/\varepsilon - 1)^2} \tag{31}$$

where *c** is the value of the most-violated constraint. On subsequent unconstrained minimizations, Equations 30 - 31 are used if all constraints are satisfied. If all constraints are violated

$$A = \frac{s}{6(1 - c^*/\varepsilon)} \tag{32}$$

If only a portion of the constraints are violated, the maximum value of *A* given by Equations 30 and 32 are used.

Figure 2 qualitatively compares the various penalty function methods considered here. The variable penalty function, although more complicated to use, ensures second-order continuity at the transition point and, for badly violated constraints, the penalty does not increase at the same dramatic rate as the quadratic extended penalty. However, for badly violated constraints, the penalty may begin to decrease so we must take care to deal with this possibility.

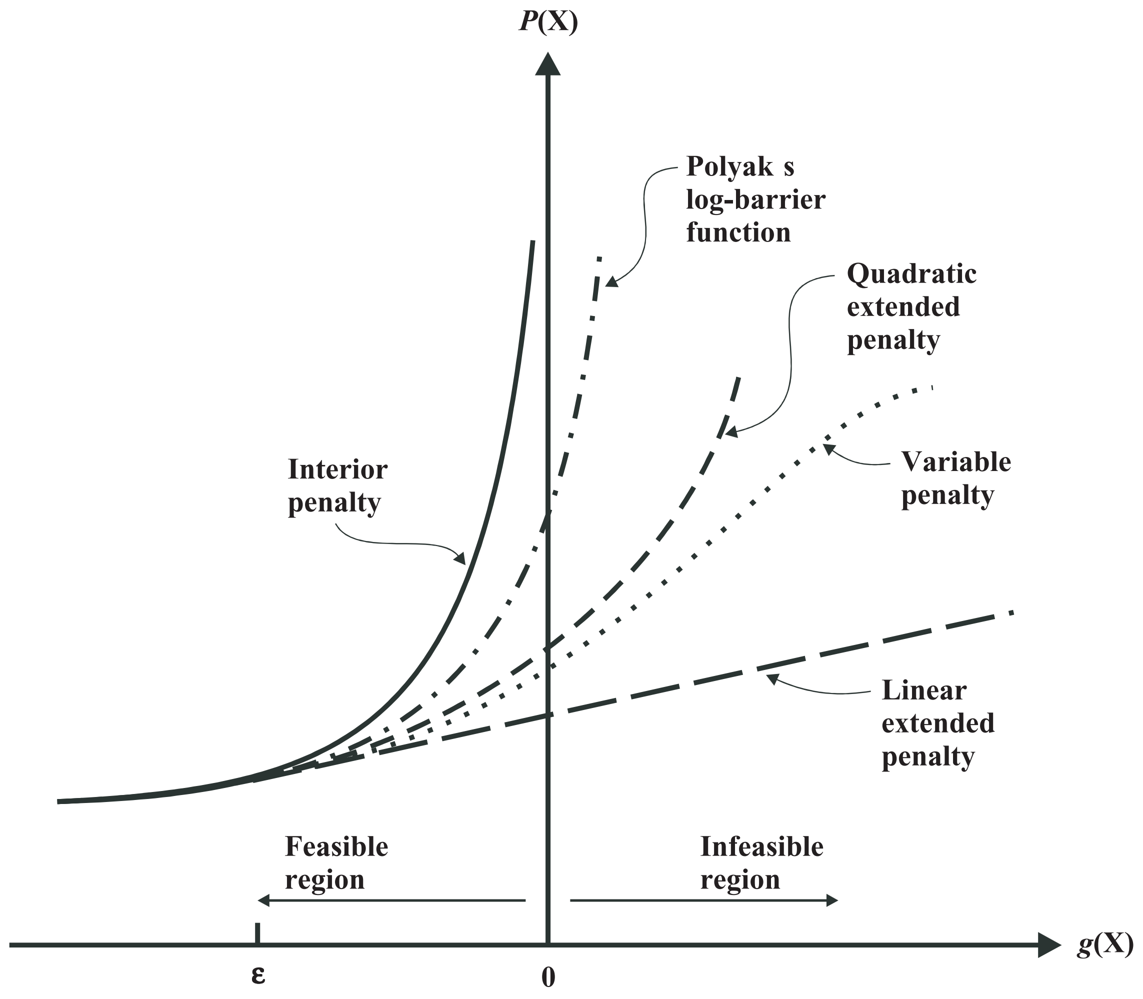


Figure 2 Qualitative comparison of penalty function methods.

4.1.4 Augmented Lagrange Multiplier Method

The Augmented Lagrange Multiplier (ALM) method uses the augmented Lagrangian [24, 25]

$$A(\mathbf{X}, \lambda, r_p) = F(\mathbf{X}) + \sum_{j=1}^{m} [\lambda_j \psi_j + r_p \psi_j^2] + \sum_{k=1}^{l} \left\{ \lambda_{k+m} h_k(\mathbf{X}) + r_p [h_k(\mathbf{X})]^2 \right\} \tag{33}$$

$$\text{where} \quad \psi_j = \max\left[g_j(\mathbf{X}), \frac{-\lambda_j}{2r_p} \right] \tag{34}$$

The update formulas for the Lagrange multipliers are now

$$\lambda_j^{p+1} = \lambda_j^p + 2r_p\left\{\max\left[g_j(\mathbf{X}), \frac{-\lambda_j^p}{2r_p}\right]\right\} \qquad j = 1, m \tag{35}$$

$$\lambda_{k+m}^{p+1} = \lambda_{k+m}^p + 2r_p h_k(\mathbf{X}) \qquad k = 1, l \tag{36}$$

The ALM method is generally considered to be more robust than the interior or exterior methods.

4.1.5 Duality Theory

Using the knowledge that ($\mathbf{X}^*$, λ^*) defines a saddle point of the Lagrangian, we know that this will correspond to a maximum with respect to λ and a minimum with respect to $\mathbf{X}$. Thus, we can define the Lagrangian in terms of λ alone as

$$L(\lambda) = \min_{\mathbf{X}} L(\mathbf{X}, \lambda) \tag{37}$$

Now since $L(\lambda)$ is a maximum with respect to λ at the optimum, we wish to maximize $L(\lambda)$, or

$$\max_{\lambda}\ L(\lambda) = \max_{\lambda}\ \min_{\mathbf{X}}\ L(\mathbf{X}, \lambda) \tag{38}$$

Alternatively, we could state this as an equivalent min-max problem as

$$\min_{\mathbf{X}}\ L(\mathbf{X}) = \min_{\mathbf{X}}\ \max_{\lambda}\ L(\mathbf{X}, \lambda) \tag{39}$$

We can now expressly define the primal and dual problems. The primal problem is the same as we have seen before

$$\text{Minimize:} \qquad F(\mathbf{X}) \tag{40}$$

Subject to:

$$g_j(\mathbf{X}) \le 0 \qquad j = 1, m \tag{41}$$

$$h_k(\mathbf{X}) = 0 \qquad k = 1, l \tag{42}$$

$$X_i^l \le X_i \le X_i^u \tag{43}$$

We now state the dual optimization problem as

$$\text{Maximize:} \quad L(\lambda) \tag{44}$$

Subject to:

$$\lambda_j \geq 0 \qquad j = 1, m \tag{45}$$

$$\lambda_{m+k} \text{ unrestricted in sign, } k = 1, l \tag{46}$$

Now we call the Lagrange multipliers *dual variables*. If we could solve the primal problem, then we could retrieve the dual variables λ^*. Similarly, it may be possible in some cases to solve the dual problem and then retrieve the optimum primal variables $\mathbf{X}^*$. Depending on the form of the approximations to structural responses, Schmit and Fleury have shown that duality theory may offer a good candidate for structural synthesis, including discrete variables[26, 27]

4.2 Choosing A Method

While each of these methods has its strong features, several were quickly eliminated based on computational considerations. These were the interior methods which are discontinuous at the constraint boundaries (methods 2, 3 and 4), as well as methods which, experience has shown, do not offer either computational or theoretical advantages (methods 6, 7, and 8).

The remaining methods (1, 5, 9 and 10) were chosen for further study.

Table 1 briefly compares the important features of each of the selected methods.

Table 1 Comparison of Methods

METHOD	FEATURES
Exterior Penalty Function Method	Easy to program. Reliable. Approaches the optimum from outside the feasible region. Easily deals with equality constraints and redundant constraints.
Polyak's Log Sigmoid Method	Many of the features of the original log penalty function and the extended penalty function method. Strong theoretical basis. Early studies are encouraging. Difficulty if constraint becomes non-active because the Lagrange multiplier goes to zero and cannot be recovered if the constraint later becomes active.
Augmented Lagrange Multiplier Method	Considered best of the "traditional" penalty function methods. Adjusts both the penalty parameter and the Lagrange Multipliers as optimization progresses. Can approach the optimum from the feasible or infeasible region.
Duality Theory	Good if the problem can be approximated well as a separable task. Good candidate for discrete variables, though theoretically weak.

4.2.1 Prototype Software

A prototype software code was created to test the algorithms listed above. Numerous tests were performed to demonstrate the feasibility of using these techniques for structural optimization. Comparisons are made with the methods contained in the existing DOT software.

Duality Theory was dropped from the study because it solves a convex approximation which is often not valid for general optimization tasks and because of difficulties with discrete variables which require solving a complex and time consuming sub-problem to deal with discontinuous derivatives.

Polyak's Log Sigmoid method was dropped because we were unable to create a reliable code in the presence on numerical inaccuracies. For example, when constraints become inactive during the optimization, their Lagrange multiplier goes to zero. There is no direct way in the algorithm to calculate a proper multiplier later if the constraint again becomes active. This causes numerical difficulties even for active constraints that become very slightly feasible. Thus, while we were able to make the algorithm find a near optimum quickly, a precise optimum could not be reliably achieved.

Finally, the Augmented Lagrange Multiplier method was dropped because, at least theoretically, it requires retaining all constraints. This generates large storage requirements which are undesirable in the present context.

This left the Exterior Penalty Function method as the method of choice. However, this method was modified to consider individual penalties for each constraint. While these penalty parameters are conceptually similar to the Lagrange multipliers in the Augmented Lagrange Multiplier method, a unique proprietary algorithm was created to estimate their values.

Here, the original constrained optimization problem is converted to a sequence of unconstrained problems of the form;

Minimize

$$\Phi(\mathbf{X}) = F(\mathbf{X}) + r_p \sum_{j=1}^{m} q_j^p \{MAX[0, g_j(\mathbf{X})]\}^2 \tag{47}$$

Subject to;

$$X_i^L \le X_i \le X_i^U \qquad i = 1, n \tag{48}$$

where **X** is the vector of design variables, F(**X**) is the objective function and g_j(**X**) are the constraints.

The subscript/superscript, p is the outer loop counter which we will call the cycle number. The penalty parameter, r_p, is initially set to a small value and then increased after each design cycle. The only difference between this formulation and the traditional exterior penalty function is the addition of individual penalty parameters, q_j^p, on each constraint. These multipliers are similar to the Lagrange multipliers used in the Augmented Lagrange Multiplier Method [24], but are calculated by a proprietary formula. Equation 48 imposes limits on the design variables (side constraints) which are handled directly.

If equality constraints are considered, they can just be converted to two equal and opposite inequality constraints.

The unconstrained penalized function defined by Equation 47 is solved by the Fletcher-Reeves conjugate direction method [28].

The gradient of $\Phi(\mathbf{X})$ is required during the optimization process.

$$\nabla\Phi(\mathbf{X}) = \nabla F(\mathbf{X}) + 2r_p \sum_{j=1}^{m} q_j^p \{\mathrm{MAX}[0, g_j(\mathbf{X})\nabla g_j(\mathbf{X})]\} \tag{49}$$

Here, the choice of the exterior penalty function becomes apparent because only gradients of violated constraints are required. Furthermore, it is not necessary to store all gradients at once. Noting that Equation 49 is a simple addition of gradient vectors, in the limit, we can calculate only one gradient (objective or constraint) at a time.

As an indication of computer memory required by various methods, the proposed method is compared with the three methods used by the DOT program. This is presented in Table 2, where MMFD is the Modified Method of Feasible Directions, SLP is the Sequential Linear Programming Method and SQP is the Sequential Quadratic Programming Method.

Table 2 Storage Requirements

	Number of Design Variables		
Method	100	1,000	10,000
MMFD	53,000	5,000,000	$5x10^8$
SLP	113,000	11,000,000	$11X10^8$
SQP	119,000	11,500,000	$12X10^8$
Proposed Method	1,400 To 11,000	14,000 To 1,000,000	140,000 To $10X10^7$

The memory requirements for the DOT methods are the number of words for storage of all internal arrays. For the proposed method, the two memory requirements are the minimum, where only 1 gradient vector is calculated at a time, and the maximum, were all possible gradients are stored in memory. In this calculation, the number of constraints equals the number of design variables. In practice, many more constraints may be considered, in which case DOT requirements will increase but BIGDOT requirements will not.

As can be seen, as the problem size grows, storage requirements for the present methods grow exponentially. However, for the proposed method, storage is much less and the requirement grows only linearly with problem size. If there are many more constraints than design variables, the requested storage for the present methods grows even more rapidly.

As noted above, the unconstrained minimization sub-problem is solved by the Fletcher-Reeves algorithm. Here, letting q be the iteration number in the unconstrained sub-problem, the search direction is found as;

If q = 1

$$S^q = -\nabla\Phi(X^0) \tag{50}$$

If q > 1

$$S^q = -\nabla\Phi(X^{q-1}) + \beta S^{q-1} \tag{51}$$

where

$$\beta = \frac{\left|\nabla\Phi(X^{q-1})\right|^2}{\left|\nabla\Phi(X^{q-2})\right|^2} \tag{52}$$

In the present study, once the search direction is calculated, the one-dimensional search is performed using polynomial interpolation.

It can be argued that more modern quasi-Newton methods such as the Broydon-Fletcher-Goldfarb-Shanno method [29 - 32] are a better choice for solving the sub-problem. However, these methods require much more storage. Also, computational experience has shown that the Fletcher-Reeves algorithm is comparable in efficiency and reliability if carefully programmed.

During the unconstrained minimization sub-problem, almost no effort is required to calculate the search direction, so the computational time spent in the optimizer itself is negligible. The cost of optimization is almost completely the cost of analysis and gradient computations. Thus, it is desirable that high quality approximations are available, as is the case in modern structural optimization.

4.3 Discrete Variable Optimization

As mentioned above, Duality theory was a candidate for solving discrete variable problems, but was rejected as inefficient and because it requires significant memory. Thus, we looked for another approach for solving discrete variable problem. It is known that the “theoretically correct” branch and bound method is hopelessly inefficient for problems in excess of only a few variables [33, 34].

Our goal is to create an algorithm that will find a “good” discrete optimum very efficiently with large numbers of variables and constraints. To achieve this, we created a very simple penalty function approach which is quite similar to the method of Shin, Gurdal and Griffen [35].

The approach used here is as follows;

1. Solve the continuous variable problem.
2. Beginning with the continuous solution, add penalty terms which will drive the discrete variables to the next lower or higher value.

Two methods were considered here. The first is an added penalty function of the form;

$$P(X) = R\sum_{i=1}^{NDV}(X_i - X_i^L)(X_i^U - X_i) \tag{53}$$

where X_i^L is the next lower discrete value and X_i^U is the next higher discrete value.

The second is a sin function form of this added penalty;

$$P(X) = R\sum_{i=1}^{NDV} 0.5\left\{1 - \sin 2\pi\left[\frac{X_i - 0.25(X_i^L + 3X_i^U)}{X_i^U - X_i^L}\right]\right\} \tag{54}$$

These are similar to the penalty functions of Reference 35. However, unlike Reference 35, we concluded that the sin form is most reliable and therefore, this is the method used in the BIGDOT program.

During this phase of the optimization, it is important to include the original penalty function as well to maintain feasibility with respect to the general constraints.

Equation 54 attempts to drive the variables to a discrete value. However, this penalty function creates numerous relative minima and has little assurance of insuring feasibility with respect to the original constraints or of actually driving all discrete variables to a discrete value. Therefore, after several cycles, progress is evaluated. If all discrete variables are not driven to an allowed value, we ask how to change each variable such that it will move to a discrete value with minimum effect on the objective function and at the same time maintaining feasibility with respect to the original constraints.

To achieve this, we first get the gradient of the objective function and the penalty term of the pseudo-objective function, with the penalty multipliers set to unity (get the sum of violated constraint gradients).

The general algorithm for this is;

1. Including only discrete variables, and bypassing variables that are already at a discrete value, search for MAX$\left|\dfrac{\frac{\partial P}{\partial X_i}}{\frac{\partial F}{\partial X_i}}\right|$
2. Calculate the changes in X_i that will move X_i to its next larger and smaller discrete value.
3. For each such δX_i estimate the maximum constraint value based on a linear approximation.
4. Move X_i to the discrete value that maintains feasibility.
5. Repeat from step 1 until all discrete variables are at a discrete value.

This algorithm drives the design variables to a discrete value while still including the original constraints via the original SUMT algorithm. This has the limitation that variables can move only one discrete value up or down from the continuous solution.

The final algorithm finds a feasible, discrete solution efficiently and seldom fails. However, it must be remembered that this is not guaranteed to be a theoretical optimum, only a good discrete solution.

This algorithm has the advantage that it is a straightforward addition to the continuous variable algorithm and that it requires very little memory and computational effort.

4.4 The BIGDOT Optimization Software

The algorithm described above has been coded for commercial application and distribution under the name “BIGDOT,” and the general, continuous variable algorithm has been presented at a technical conference [36] The discrete variable capability will be presented at the MDO conference in September, 2002 [37]. The usage of BIGDOT is very similar to that of VR&D’s current optimization software, DOT. In fact, current users of DOT can include BIGDOT by calling an intermediate program called ALLDOT. When calling ALLDOT with METHOD = 1, 2 or 3, the DOT algorithms will be used. If ALLDOT is called with METHOD = 4, the SUMT algorithm outlined here is used.

A complete users manual has been created for BIGDOT [38] and the program has been ported to numerous computers. BIGDOT has been added to the GENESIS structural optimization program [5] from VR&D and has successfully solved structural optimization problems in excess of 135,000 design variables.

In the following section, examples are offered to demonstrate the BIGDOT.

4.5 Test Case - Unconstrained Minimization of a Spring - Mass System

First consider an unconstrained minimization problem. Figure 3 shows a simple spring system supporting weights at the connections between the springs. This system can be analyzed to determine the equilibrium position by minimizing the total potential energy of the system. We can create a problem of whatever size we choose using simple formulas.

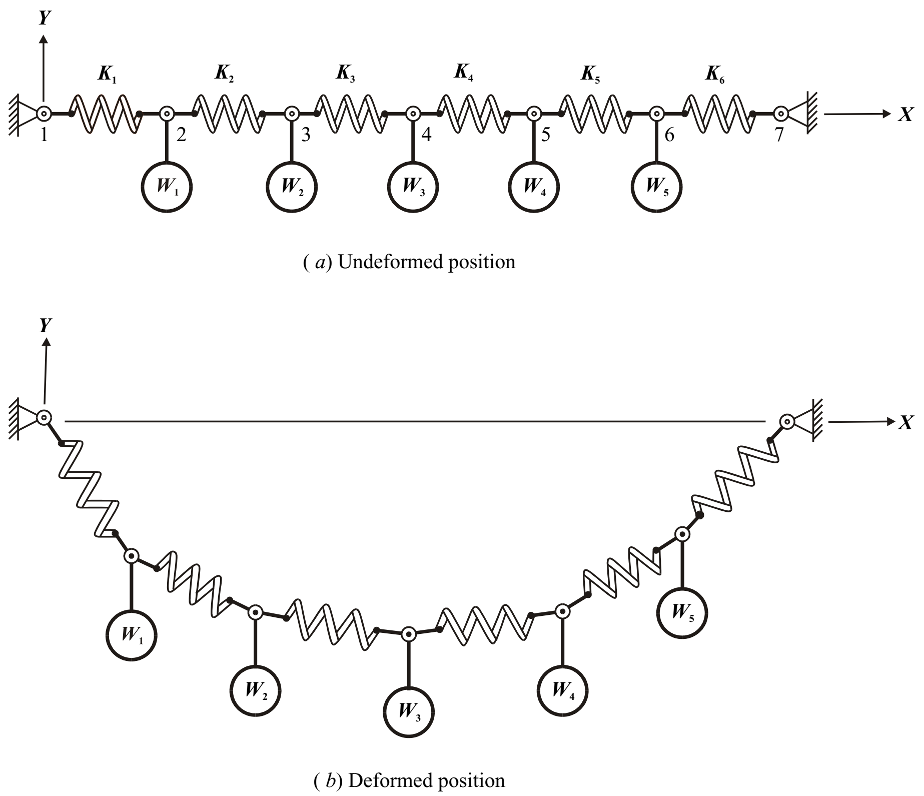


Figure 3 Spring-Mass System

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

The deformation of spring *i* is

$$\Delta L_i = [(X_{i+1} - X_i)^2 + (Y_{i+1} - Y_i)^2]^{1/2} - L_i^0 \tag{55}$$

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

The stiffness of spring i is taken to be

$$K_i = 500 + 200\left(\frac{N}{3} - i\right)^2 \quad \text{N/m} \tag{56}$$

Mass W_j is defined to be

$$W_j = 50j \ \ \text{N} \tag{57}$$

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

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

The total potential energy, PE, is now

$$PE = \sum_{i=1}^{N+1} \frac{1}{2} K_i \Delta L_i^2 + \sum_{j=1}^{N} W_j Y_{j=1} \tag{58}$$

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

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

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

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

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

The BIGDOT solutions are

Table 3 Optimization Results

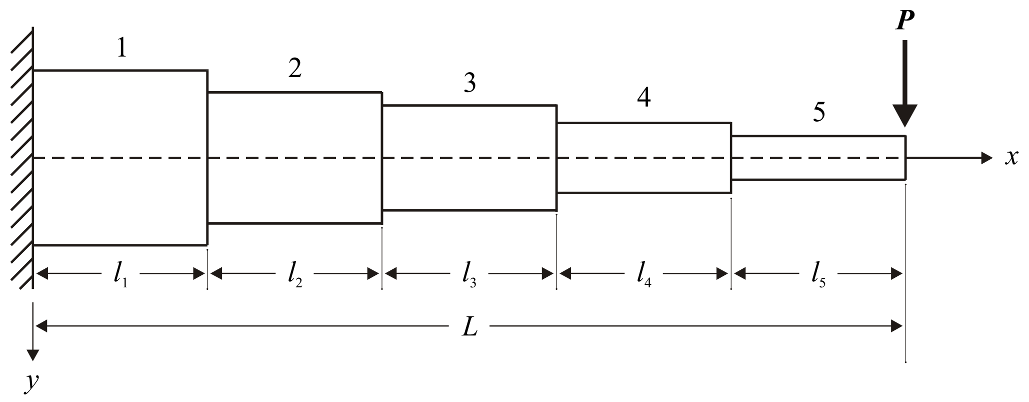
PARAMETER	OPTIMUM N=5,000 (NDV=10,000)	OPTIMUM N=12,500 (NDV=25,000)	OPTIMUM N=25,000 (NDV=50,000)
Continuous Optimization			
OBJECTIVE	$-2.57X10^{9}$	$-1.97X10^{10}$	$-8.76X10^{10}$
NUMBER OF CYCLES	1	1	1
FUNCTION EVALUATIONS	507	507	507
GRADIENT EVALUATIONS	100	100	100
Additional Computations for Discrete Optimization			
OBJECTIVE	$-2.28X10^{9}$	$-1.72X10^{10}$	$-5.12X10^{10}$
NUMBER OF CYCLES	5	6	5
FUNCTION EVALUATIONS	146	215	151
GRADIENT EVALUATIONS	27	41	28

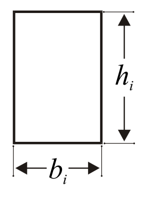
The program provided here (Appendix A) allows the user to input the number of masses, N=NMASS, as well as the print control value. Here, N = 5,000, 12,500 and 25,000 was used, to create problems of 10,000, 25,000 and 50,000 design variables, respectively. You may change NMASS to increase or decrease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

The computer program for this example is given in Listing 1 and the output for the 10,000 variable example is given in Listing 2 in Appendix A.

4.6 Cantilevered Beam

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





Cross section

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

Figure 4 Cantilevered Beam

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

$$M_i = P\left[L + l_i - \sum_{j=1}^{i} l_j\right] \tag{59}$$

and the corresponding maximum bending stress is

$$\sigma_i = \frac{M_i h_i}{2I_i} \tag{60}$$

where

$$I_i = \frac{b_i h_i^3}{12} \tag{61}$$

The design task is now defined as

$$\text{Minimize: } V = \sum_{i=1}^{N} b_i h_i l_i \tag{62}$$

Subject to:

$$\frac{\sigma_i}{\bar{\sigma}} - 1 \le 0 \qquad i = 1, N \tag{63}$$

$$h_i - 20b_i \le 0 \qquad i = 1, N \tag{64}$$

$$b_i \ge 1.0 \qquad i = 1, N \tag{65}$$

$$h_i \ge 5.0 \qquad i = 1, N \tag{66}$$

$$X_i \in \{0.1, 0.2, 0.3, ...\} \tag{67}$$

Here $\bar{\sigma}$ is the allowable bending stress. This is a design problem in $n = 2N$ variables. There are N nonlinear constraints defined by Equations 63, N linear constraints defined by Equation 64, and $2N$ side constraints on the design variables defined by Equations 65 and 66. The design variables are required to take on discrete values in increments of 0.1 as given in Equation 67.

The program provided here allows the user to input the number of beam segments (elements), NSEG, as well as the print control value.Here, NSEG=500, 12,500 and 25,000 was used, to create problems of 1,000, 25,000 and 50,000 design variables, respectively. You may change NSEG to increase or decrease the problem size. The program is dimensioned to allow a value of NSEG up to 25,000, to yield 50,000 design variables. If you wish to solve even larger problems, be sure to increase the array sizes.

The BIGDOT solutions are shown in Table 4, where a constraint is considered active if its value is greater than -0.05 (five percent tolerance). In each case, a discrete solution was found with minor increase in the objective functions.

Note that the number of function and gradient evaluations does not change significantly with problem size. This is a very significant feature of the algorithm developed here because it suggests that there is no particular limit to the size of optimization problems we may consider.

Table 4 Optimization Results

PARAMETER	INITIAL VALUE	OPTIMUM NSEG=5,000 (NDV=10,000)	OPTIMUM NSEG=12,500 (NDV=25,000)	OPTIMUM NSEG=25,000 (NDV=50,000)
Continuous Optimization				
OBJECTIVE	100,000	53,816	53,737	53,747
MAX G	0.3393	$4.5X10^{-4}$	$8.4X10^{-4}$	$9.0X10^{-4}$
# OF INITIALLY ACTIVE/VIOLATED CONSTRAINTS		1267	3,167	6,334
CYCLES		7	7	7
FUNCTION EVALUATIONS		235	242	248
GRADIENT EVALUATIONS		44	45	47
ACTIVE CONSTRAINTS		9,995	24,986	49,972
ACTIVE SIDE CONSTRAINTS		12	30	46
Additional Computations for Discrete Optimization				
OBJECTIVE		54,865	54,859	54,866
MAX G		$1.0X10^{-3}$	$9.2X10^{-9}$	$9.5X10^{-9}$
NUMBER OF CYCLES		6	6	6
FUNCTION EVALUATIONS		85	79	83
GRADIENT EVALUATIONS		15	10	13
ACTIVE CONSTRAINTS		9,584	23,967	47,826
ACTIVE SIDE CONSTRAINTS		22	54	82

The computer program for this example is given in Appendix A as Listing 3 and the BIGDOT output for a 10,000 variable case is shown in Listing 4. Note that sufficient storage was not provided to store the desired number of constraint gradients, so NGMAX was set to 3,484. Because BIGDOT required more constraint gradients than this during optimization, reduced storage logic was used. This increased the run time, but otherwise had no effect on the optimum.

4.6.1 Test Case - Topology Optimization Using GENESIS

The Michell truss problem is commonly used to verify topology design algorithms. Figure 5 shows the entire design domain including the circular non-designable region, whose perimeter is completely fixed.

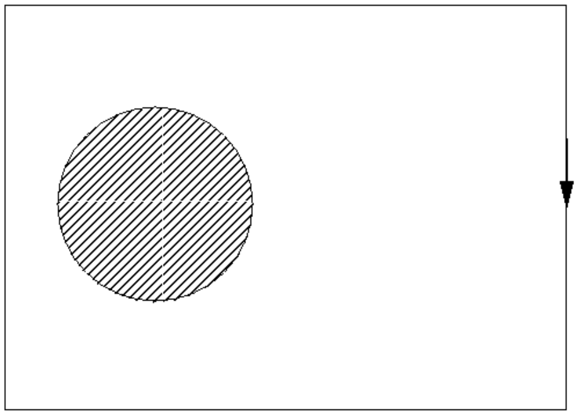


Figure 5 Designable Region and Boundary Conditions

The optimization task is to minimize strain energy, subject to a maximum mass constraint of 20%. The BIGDOT optimizer was incorporated into the GENESIS structural optimization program to solve this problem. The topology optimization task consisted of 7,588 design variables, but only one nonlinear constraint. At the optimum, almost all variables were at their lower or upper bound.

The solution shown in figure 6 appears to be a truss-like structure and is similar to the theoretical solution.

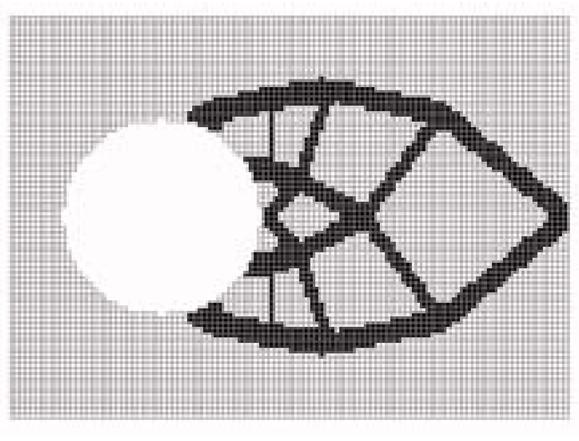


Figure 6 Optimum Topology

Figure 7 is the theoretical Michell truss solution.

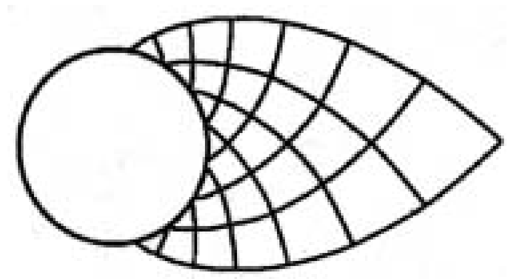


Figure 7 Mitchell Truss

The initial value of the objective function was 2,917 and the optimum was 102.06. This problem is of the class where there is only one constraint, but where almost all design variables end at their lower or upper bound. It should be noted that the DOT optimizer would require approximately 500 Mb of storage to solve this problem.

This same problem was solved again, but with a much finer mesh, consisting of 47,335 quadrilateral elements and 142,824 degrees of freedom. This resulted in a total of 35,017 design variables. For this case, the initial objective function value was 24,762 and the optimum was 225.02. The optimum topology is essentially the same as that shown in Figure 6.

4.6.2 Test Case - Wing Design Using GENESIS

As a final example, a wing structure was designed using both DOT and the BIGDOT code within GENESIS. This structure was modeled with almost 50,000 degrees of freedom and was subjected to 8 load cases. There were 1,251 independent and 1,384 dependent design variables, designing isotropic rod and composite membrane elements, and a total of 8,400 constraints. This problem uses numerous "synthetic" functions to impose manufacturing constraints, and the gradients of these responses are relatively time consuming in the approximate optimization.

Table 5 summarizes the results using DOT as well as BIGDOT. The initial value of the objective function was normalized to 1000.00 for each case, to protect potentially confidential information. It is significant that the solution obtained by DOT was found only after considerable effort to "fine tune" the control parameters. This is a particularly difficult problem for DOT because many variables have very little effect on the objective but, if they are changed enough, have a significant overall effect on the optimum. The SUMT method seem to deal with this situation well so that BIGDOT achieved an optimum very close to the best known solution. Also, note that the design time is dramatically reduced using BIGDOT because the DOT optimizer required a great deal of CPU time within the optimization algorithm itself while BIGDOT required a fraction as much time.

Table 5 Objective Function and Computation Time

Method	Optimum	CPU Time per GENESIS Design Cycle	Time in the Approximate Optimization
DOT	955.07	2,870	2,643
BIGDOT	953.98	345	147

4.7 Conclusions

A new optimization algorithm has been created along with its implementation into a commercial software product. While the basic concept is an exterior penalty function approach, enhancements have been made to improve efficiency and reliability, as well as to deal with discrete variable problems.

The BIGDOT software created here overcomes the key problems encountered with existing software while offering some very attractive features.

1. It requires very little computer memory.
2. It does not solve a complex and time intensive direction finding problem.
3. It handles redundant constraints (constraints with the same values and gradients) easily.
4. It deals with equality constraints with no loss in efficiency or reliability.
5. It solves discrete variable problems to efficiently produce a "good" discrete solution.
6. It scales very well. That is, the efficiency is about the same regardless of problem size.

With the BIGDOT software, we are now able to address problems of very large scale. While the methods implemented here are not as efficient as existing modern methods, as measured by the number of function and gradient evaluations, they are much more efficient in terms of memory and internal CPU requirements.

5.0 Potential Applications

The immediate application of the software developed here will be within our own structural optimization software, GENESIS. The other companies who presently license the DOT optimizer for use in their structural optimization software are likely short term users of this software. Finally, as topology optimization software becomes more commonplace, this is expected to generate a market opportunity.

BIGDOT has already been incorporated into the AFT Mercury software from Applied Flow Technology for optimizing pumping systems [39].

Additionally, BIGDOT is being incorporated into Version 3 of the VisualDOC [40] software from VR&D. VisualDOC is a general purpose, graphics based, program for adding optimization to almost any analysis.

The second short term potential application is by research organizations, both academic and government. An important research area is in methods to solve ever larger structural optimization problems. One possible approach, which has been the subject of research dating back to 1965, is the concept of simultaneous analysis and design. Because the number of degrees of freedom in analysis commonly exceeds 500,000 in industrial applications, the ability to efficiently perform the optimization task is clear.

As multidiscipline design optimization becomes more accepted, the need to deal with large numbers of design variables and constraints will increase. This software will provide the needed research tools as well as the necessary optimizer for commercial applications in MDO, without the necessity of decomposing the overall problem into smaller optimization tasks.

Finally, as we consider computers of the future, massive parallelization becomes an important technology. While the present proposed software does not address this issue explicitly, it provides a research tool for studying concepts for partitioning of the optimization process. Because the proposed algorithms essentially operate in vector space (as opposed to matrix space), it may be possible to partition the vector operations and solve them in parallel.

6.0 Commercialization Plans

VR&D has an established clientele using the GENESIS, DOT and VisualDOC software of over 100 companies and universities. Also, eight companies presently use the DOT optimizer embedded in their own commercial software.

It is significant that the BIGDOT software is the second VR&D product funded by a Phase II SBIR contract. The first product, VisualDOC, already has an established clientele and BIGDOT will immediately enhance its capabilities.

Our existing client base will provide the basis for marketing of the new software. Additionally, specific marketing strategies include the following:

1. VR&D has and will publish the theory and capabilities at conferences as well as professional journals which emphasize software products and applications.
2. VR&D will promote the software on our Internet Home Page. This medium has already demonstrated itself to be a powerful means of advertising to engineers who seek information on optimization capabilities. Also, our newsletter will be used as a marketing tool. While its distribution is limited to about 2,000, the international recipients are predisposed to use our products, based on our strong reputation for advanced technology.
3. VR&D markets Dr. Vanderplaats' textbook on numerical optimization, which is now used by several universities. Together with the text, we provide a CD-ROM containing the VisualDOC, DOT and GENESIS software for educational use. BIGDOT will be added to this.
4. Finally, VR&D will target professional publications which emphasize CAE applications as a mechanism for informing candidate clients. For example, the Computer Aided Engineering magazine has published articles complimentary to the GENESIS program. This and similar magazines reach a very wide audience that is receptive to new engineering software.

It may be noted here that we do not propose an advertising campaign in professional journals. This is simply because the cost of such marketing is very high and the benefit has not been demonstrated. Past efforts with this approach have demonstrated that the benefit/cost ratio is quite low as compared on relying on our reputation to produce sales via direct contacts and "word of mouth."

VR&D markets its products worldwide. In the U.S. and many other countries, we distribute our products directly. In Japan, we have three distributors. Also, we have distributors in Korea, France, Australia, India, Spain and China. Each of these will be responsible for marketing in neighboring countries as well. Each distributor will market the BIGDOT software together with our other software.

In summary, our principal marketing strategy is to 1) work with our existing clients to promote our new software offerings, 2) target engineers who can use such software with their applications, 3) develop an aggressive distributor network, 4) provide software in Dr. Vanderplaats' textbook and 5) promote the new software capabilities via the Internet, professional conferences, and professional journals. Our experience has shown that this creates a committed clientele which will further use and promote our products.

7.0 References

1. Schmit, L.A., “Structural Design by Systematic Synthesis,” Proceedings, 2nd Conference on Electronic Computation, ASCE, New York, 1960, pp. 105-132.

2. Schmit, L.A. and Farshi, B., “Some Approximation Concepts for Efficient Structural Synthesis,” AIAA Journal, Vol. 12, May 1974, pp. 692-699.

3. Schmit, L.A. and Miura, H., “Approximation Concepts for Efficient Structural Synthesis,” NASA CR 2552, March 1976.

4. Vanderplaats, G. N., “Structural Design Optimization - Status and Direction,” AIAA J. Aircraft, Vol. 13, No. 1, 1999, pp. 11-20.

5. GENESIS User’s Manual, Version 7.0: Vanderplaats Research & Development, Inc., Colorado Springs, CO, 2001.

6. Vanderplaats, G. N., Numerical Optimization Techniques for Engineering Design - With Applications, McGraw-Hill, 1984 (3rd edition published by VR&D, 1999).

7. DOT - Design Optimization Tools, Users Manual, Version 5, Vanderplaats Research & Development, Inc., Colorado Springs, CO, 2001.

8. Hager, W.W., D. W. Hearn and P. M. Pardalos: “Large Scale Optimization; State of the Art,” Kluwer Academic Publishers, 1994, pp. 45-67.

9. Kelley, J. E.: The Cutting Plane Method for Solving Convex Programs, J. SIAM, vol. 8, pp. 702-712, 1960.

10. Thomas, H. T., G. N. Vanderplaats and Y-K Shyy: A Study of Move Limit Adjustment Strategies in the Approximation Concepts Approach to Structural Synthesis, Proc. Fourth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, Cleveland, OH, Sept. 21-23, 1992, pp. 507-512.

11. Fiacco, A. V., and G. P. McCormick: “Nonlinear Programming: Sequential Unconstrained Minimization Techniques,” John Wiley and Sons, New York, 1968.

12. Polyak, R., “Modified Barrier Functions (Theory and Methods),” Mathematical Programming, 54 (1992), pp. 177-222.

13. Polyak, R. A., Griva, I. and Sobieski, J., “The Newton Log-Sigmoid Method in Constrained Optimization,” Proc. 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MI, Sept. 2-4, 1998, AIAA Paper No. AIAA-98-4797.

14. Kavlie, D., and J. Moe: Automated Design of Frame Structures, ASCE J. Struct. Div., vol. 97, no. ST1, pp. 33 – 62, January 1971.

15. Cassis, J. H., and L. A. Schmit: On Implementation of the Extended Interior Penalty Function, Int. J. Num. Meth. Engin. vol. 10, no. 1, pp. 3 – 23, 1976.

16. Cassis, J. H.: Optimum Design of Structures Subjected to Dynamic Loads, Ph.D. thesis, University of California, Los Angeles, 1974.

17. Cassis, J. H. and Schmit, L. A., “On Implementation of the Extended interior Penalty Function,” Int. J. Num. Meth. Engin., Vol. 10, No. 1, 1976, pp. 3-23.

18. Haftka, R. T., and J. H. Starnes, Jr.: Applications of a Quadratic Extended Interior Penalty Function for Structural Optimization, AIAA J., vol. 14, no. 6, pp. 718 – 724, June 1976.

19. Prasad, B.: A Class of Generalized Variable Penalty Methods for Nonlinear Programming, J. Optim. Theory Appl., vol. 35, no. 2, pp. 159 – 182, October 1981.

20. Rockafellar, R. T.: The Multiplier Method of Hestenes and Powell Applied to Convex Programming, J. Optim. Theory Appl., vol. 12, no. 6, pp. 555 – 562, 1973.

21. Pierre, D. A., and M. J. Lowe: “Mathematical Programming via Augmented Lagrangians,” Applied Mathematics and Computation Series, Addison-Wesley, Reading, Mass., 1975.

22. Imai, K.: Configuration Optimization of Trusses by the Multiplier Method, Ph.D. Thesis, University of California, Los Angeles, 1978.

23. Imai, K., and L. A. Schmit: Configuration Optimization of Trusses, ASCE, J. Struct. Div., vol. 107, no. ST5, pp. 745 – 756, May 1981.

24. Rockafellar, R. T.: The Multiplier Method of Hestenes and Powell Applied to Convex Programming, J. Optim. Theory Appl., vol. 12, no. 6, pp. 555 – 562, 1973.

25. Powell, M. J. D.: Optimization Algorithms in 1979, Committee on Algorithms Newsletter, no. 5, Mathematical Programming Society, pp. 2 – 16, February 1981.

26. Schmit, L. A., and C. Fleury: Structural Synthesis by Combining Approximation Concepts and Dual Methods, AIAA J., vol. 18, pp. 1252 – 1260, October 1980.

27. Schmit, L. A., and C. Fleury: Discrete-Continuous Variable Structural Synthesis Using Dual Methods, AIAA J., vol. 18, pp. 1515-1524, December 1980.

28. Fletcher, R. and C. M. Reeves: Function Minimization by Conjugate Gradients, Br. Computer J., vol. 7, no. 2, pp. 149-154, 1964.

29. Broydon, 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.

30. Fletcher, R.: A New Approach to Variable Metric Algorithms, Computer J., vol. 13, pp. 317 – 322, 1970.

31. Goldfarb, D.: A Family of Variable Metric Methods Derived by Variational Means, Math. Comput., vol. 24, pp. 23 – 36, 1970.

32. Shanno, D. F.: Conditioning of Quasi-Newton Methods for Function Minimization, Math. Comput., vol. 24, pp. 647 – 656, 1970.

33. Thanedar, P.B. and Vanderplaats, G. N., “Survey of Discrete Variable Optimization for Structural Design,” ASCE Journal of Structural Engineering, Vol. 121, No. 2, February 1995, pp. 301-306.

34. Olsen, G. and Vanderplaats, G. N., “A Method for Nonlinear Optimization with Discrete Variables,” AIAA Journal, Vol. 27, No. 11, Nov. 1989, pp. 1584-1589.

35. Shin, D. K., Gurdal, Z. and Griffin, O. H., Jr., “A Penalty Approach for Nonlinear Optimization with Discrete Design Variables,” Eng. Opt., Vol. 16, pp. 29-42, 1990.

36. Vanderplaats, G. N., “Very Large Scale Optimization,” AIAA Paper No. 2000-4809, Proc. 8th AIAA/USAF/NASA/ISSMO Symposium at Multidisciplinary Analysis and Optimization, Long Beach, CA September 6-8, 2000.

37. Rastogi, N., Ghosh, D. K., and Vanderplaats, G. N., "Discrete Optimization Capabilities in GENESIS Software", Proc. of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, September 4-6, 2002 Atlanta, Georgia, AIAA-2002-5646.

38. BIGDOT Very Large Scale Design Optimization Tool, Users Manual, Version 1.0, Vanderplaats Research & Development, Inc., Colorado Springs, CO, 2002.

39. Hodgson, J. and Walters, T., “Optimizing Pumping Systems to Minimize First or Life/Cycle Cost,” Proc. 19th International Pump Users Symposium, Turbomachinery Laboratory, Texas A&M University, College Station, TX, Feb. 2002, pp. 1-8.

40. VisualDOC Users Manual, Version 3.0, Vanderplaats Research & Development, Inc., Colorado Springs, CO, 2002.

APPENDIX A

SAMPLE PROGRAMS AND OUTPUT

This appendix contains the program for solving the unconstrained spring-mass system shown in Figure 3 as Listing 1. The output for a 1,000 variable problem is shown in Listing 2.

Listing 3 is the program for solving the cantilevered beam of Figure 4 and Listing 4 is the output for a 10,000 variable beam optimization.

LISTING 1: SPRING-MASS SYSTEM FORTRAN PROGRAM

```
C
C     SAMPLE PROGRAM.  NMASS HANGING MASS ANALYSIS.
C
      DIMENSION X(50000),XL(50000),XU(50000),G(1),
     *WK(40000000),IWK(300000),RPRM(20),IPRM(20),DUMY(50000)
      DIMENSION IDISCR(101000),DISCRT(10000)
      COMMON/JUNK/IJUNK
C     DEFINE NRWK, NRIWK.
      NRWK=40000000
      NRIWK=300000
C
C     NUMBER OF MASSES IS NMASS
C
10    WRITE(*,*)' INPUT NMASS, IPRINT, METHOD, NDSCRT'
      READ(*,*)NMASS,IPRINT,METHOD,NDSCRT
      IF(NMASS.EQ.0) STOP
C
      INFO=0
C
C     DEFINE NDV,NCON.
C     TWO TIMES NMASS DESIGN VARIABLES.
      NDV=2*NMASS
C     NO CONSTRAINTS.
      NCON=0
C
C --- TEMP FOR TESTING DISCRETE VARIABLES
C
      IF(METHOD.EQ.4.AND.INFO.EQ.0) THEN
C
C     DISCRETE VARIABLE INFORMATION
C
         DO 20 I=1,NDV
            IDISCR(I)=1
20          IDISCR(NDV+I)=6000
         DO 30 I=1,6000
            DISCRT(I)=.1*FLOAT(I)-300.
30       CONTINUE
      ENDIF
C
C --- END TEMP
C
C     DEFINE INITIAL DESIGN.
      DO 40 I=1,NMASS
         X(I)=0.0
         X(I+NMASS)=0.0
40    CONTINUE
      NA=NDV+1
C     MINIMIZE
      MINMAX=-1
C     OPTIMIZE.
      CALL EVAL(OBJ,X,G,NMASS,WK,WK(NA),NDV,INFO)
```

```
C     DEFINE BOUNDS.
      DO 50 I=1,NMASS
C     LOWER BOUNDS.
         XL(I)=-5000.
         XL(I+NMASS)=-5000.
C     UPPER BOUNDS
         XU(I)=5000.
         XU(I+NMASS)=5000.
50    CONTINUE
C     INITIALIZE INFO TO ZERO.
      INFO=0
C     ZERO RPRM AND IPRM.
      DO 60 I=1,20
         RPRM(I)=0.0
60       IPRM(I)=0
      IPRM(16)=NDSCRT
70    CONTINUE
      CALL ALLDOT(INFO,METHOD,IPRINT,NDV,NCON,X,XL,XU,OBJ,MINMAX,
     1G,RPRM,IPRM,WK,NRWK,IWK,NRIWK,DISCRT,IDISCR)
C     FINISHED?
      NGT=0
      CALL EVAL(OBJ,X,G,NMASS,WK,WK(NA),NDV,INFO)
C     FINISHED?
      IF(INFO.EQ.0) GO TO 10
      GO TO 70
      END
      SUBROUTINE EVAL (OBJ,X,G,NMASS,DF,A,NDV,INFO)
      DIMENSION X(*),G(*),DF(*),A(NDV,*)
C
      IF(INFO.GT.1) THEN
         DO 10 I=1,NDV
            DF(I)=0.
10       CONTINUE
      ENDIF
C     NMASS MASSES.
C
      AL=60.
      ALI=AL/(FLOAT(NMASS)+1.)
      PE1=0.
      PE2=0.
      XI=0.
      YI=0.
      DO 20 I=1,NMASS
         J=I-1
         WI=50.*FLOAT(I)
         PE1=PE1+WI*X(I+NMASS)
         IF(INFO.EQ.2) DF(I+NMASS)=DF(I+NMASS)+WI
         XIP1=X(I)
         YIP1=X(I+NMASS)
         AKI=500.+200.*((FLOAT(NMASS)/3.-FLOAT(I))**2)
         DLI1=SQRT((ALI+XIP1-XI)**2+(YIP1-YI)**2)
         DLI=DLI1-ALI
         IF(INFO.EQ.2) THEN
           DF(I)=DF(I)+AKI*DLI*(ALI+XIP1-XI)/DLI1
           DF(I+NMASS)=DF(I+NMASS)+AKI*DLI*(YIP1-YI)/DLI1
           IF(J.GT.0) THEN
             DF(J)=DF(J)-AKI*DLI*(ALI+XIP1-XI)/DLI1
             DF(J+NMASS)=DF(J+NMASS)-AKI*DLI*(YIP1-YI)/DLI1
           ENDIF
         ENDIF
         PE2=PE2+0.5*AKI*(DLI**2)
         XI=XIP1
         YI=YIP1
20     CONTINUE
C
C      LAST SPRING
C
       AKI=500.+200.*((FLOAT(NMASS)/3.-FLOAT(NMASS+1))**2)
       DLI1=SQRT((ALI-XI)**2+YI**2)
       DLI=DLI1-ALI
       PE2=PE2+0.5*AKI*(DLI**2)
       IF(INFO.EQ.2) THEN
         DF(NMASS)=DF(NMASS)-AKI*DLI*(ALI-XI)/DLI1
         DF(NDV)=DF(NDV)+AKI*DLI*YI/DLI1
       ENDIF
       OBJ=PE1+PE2
       RETURN
       END
```

LISTING 2: SPRING-MASS SYSTEM OUTPUT: NMASS = 5,000

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

                   DESIGN OPTIMIZATION TOOLS



                      (C) COPYRIGHT, 2001

                       VANDERPLAATS R&D

                ALL RIGHTS RESERVED, WORLDWIDE

                         BETA VERSION




 CONTROL PARAMETERS

 NUMBER OF DECISION VARIABLES,          NDV =     10000
 NUMBER OF CONSTRAINTS,                NCON =         0
 PRINT CONTROL PARAMETER,            IPRINT =         2
 THE OBJECTIVE FUNCTION WILL BE MINIMIZED


 -- SCALAR PROGRAM PARAMETERS

 REAL PARAMETERS
   1) PENALT =  1.00000E+00           6) DABOBJ =  1.00000E-10
   2) PMULT  =  5.00000E+00           7) DELOBJ =  1.00000E-03
   3) CTMIN  =  3.00000E-03           8) PENLTD =  1.00000E-02
   4) DABSTR =  1.00000E-10           9) PMULTD =  5.00000E+00
   5) DELSTR =  1.00000E-03

 INTEGER PARAMETERS
   1) NGMAX  =        0              6) ITRMOP =      2
   2) ISCAL  =        1              7) IWRITE =      6
   3) JTMAX  =       50              8) JWRITE =      0
   4) ITRMST =        2              9) MAXINT =  2000000000
   5) ITMAX  =      100             10) NDISCR =      1
```

```
    STORAGE REQUIREMENTS
ARRAY  DIMENSION        USED
  WK    40000000      110040
 IWK      300000       30071


-- INITIAL FUNCTION VALUES

OBJ =   0.0000


-- BEGIN OPTIMIZATION


-- BEGIN CONTINUOUS CYCLE    1

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  0.00000E+00 OBJECTIVE =  0.00000E+00

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE = -2.57357E+09   OBJECTIVE = -2.57357E+09


-- OPTIMIZATION IS COMPLETE

NUMBER OF UNCONSTRAINED MINIMIZATIONS =    1

THERE ARE         0 ACTIVE SIDE CONSTRAINTS

TERMINATION CRITERIA


-- OPTIMIZATION RESULTS


OBJECTIVE, F(X) =   -2.57357E+09

FUNCTION CALLS =     507

GRADIENT CALLS =     100



-- BEGIN DISCRETE VARIABLE OPTIMIZATION


-- INITIAL FUNCTION VALUES

OBJ = -2.57357E+09
```

```
-- BEGIN DISCRETE CYCLE    1

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE = -2.57357E+09 OBJECTIVE = -2.57357E+09

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE = -2.62031E+09   OBJECTIVE = -2.62031E+09

THERE ARE        21 DISCRETE VALUES


-- BEGIN DISCRETE CYCLE    2

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE = -2.62031E+09 OBJECTIVE = -2.62031E+09

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE = -2.62102E+09   OBJECTIVE = -2.62102E+09

THERE ARE        24 DISCRETE VALUES


-- BEGIN DISCRETE CYCLE    3

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE = -2.62102E+09 OBJECTIVE = -2.62102E+09

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE = -2.27744E+09   OBJECTIVE = -2.27744E+09

THERE ARE     10000 DISCRETE VALUES


-- BEGIN DISCRETE CYCLE    4

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE = -2.27744E+09 OBJECTIVE = -2.27744E+09

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE = -2.27745E+09   OBJECTIVE = -2.27745E+09

THERE ARE     10000 DISCRETE VALUES


-- BEGIN DISCRETE CYCLE    5

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE = -2.27745E+09 OBJECTIVE = -2.27745E+09

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE = -2.27745E+09   OBJECTIVE = -2.27745E+09

THERE ARE     10000 DISCRETE VALUES
```

```
-- OPTIMIZATION IS COMPLETE

  NUMBER OF UNCONSTRAINED MINIMIZATIONS =    5

  THERE ARE         0 ACTIVE SIDE CONSTRAINTS

  THERE ARE     10000 DISCRETE VALUES

  TERMINATION CRITERIA

  RELATIVE CONVERGENCE CRITERION WAS MET FOR  2 CONSECUTIVE ITERATIONS


  -- OPTIMIZATION RESULTS


  OBJECTIVE, F(X) =   -2.27745E+09

  FUNCTION CALLS =     653

  GRADIENT CALLS =     127
```

LISTING 3: CANTILEVER BEAM ANALYSIS FORTRAN PROGRAM

```
C
C     SAMPLE PROGRAM.  NSEG ELEMENT BEAM DESIGN.
C
      DIMENSION X(50000),XL(50000),XU(50000),G(50001),
     *WK(40000000),IWK(300000),RPRM(20),IPRM(20)
      DIMENSION IDISCR(101000),DISCRT(10000)
      COMMON/JUNK/IJUNK
C     DEFINE NRWK, NRIWK.
      NRWK=40000000
      NRIWK=300000
C
C     NUMBER OF BEAM SEGMENTS IS NSEG
C
10    CONTINUE
      WRITE(*,*)' INPUT NSEG, IPRINT, METHOD, NDSCRT'
      READ(*,*)NSEG,IPRINT,METHOD,NDSCRT
      IF(NSEG.EQ.0) STOP
C
      INFO=0
C
C     DEFINE NDV,NCON.
C     TWO TIMES NSEG DESIGN VARIABLES.
      NDV=2*NSEG
C     TWO TIMES NSEG + 1 CONSTRAINTS.
      NCON=2*NSEG
C
C --- DISCRETE VALUES
C
      IF(METHOD.EQ.4.AND.INFO.EQ.0) THEN
C
C     DISCRETE VARIABLE INFORMATION
C
         DO 20 I=1,NDV
            IDISCR(I)=1
20          IDISCR(NDV+I)=6000
         DO 30 I=1,6000
            DISCRT(I)=.1*FLOAT(I)
30       CONTINUE
      ENDIF
C
C     DEFINE INITIAL DESIGN.
      DO 40 I=1,NSEG
C     INITIAL VALUES.
         X(I)=5.0
         X(I+NSEG)=40.0
40    CONTINUE
      NA=NDV+1
C     MINIMIZE
      MINMAX=-1
```

```
C     OPTIMIZE.
      CALL EVAL(OBJ,X,G,NSEG,WK,WK(NA),NDV,INFO,NGT,IWK)
C     DEFINE BOUNDS.
      DO 50 I=1,NSEG
C     LOWER BOUNDS.
         XL(I)=0.5
         XL(I+NSEG)=5.
C     UPPER BOUNDS
         XU(I)=100.
         XU(I+NSEG)=100.
50    CONTINUE
C     INITIALIZE INFO TO ZERO.
      INFO=0
C     ZERO RPRM AND IPRM.
      DO 60 I=1,20
         RPRM(I)=0.0
         IPRM(I)=0
60    CONTINUE
      IPRM(16)=NDSCRT
70    CONTINUE
      CALL ALLDOT(INFO,METHOD,IPRINT,NDV,NCON,X,XL,XU,OBJ,MINMAX,
     1G,RPRM,IPRM,WK,NRWK,IWK,NRIWK,DISCRT,IDISCR)
C
      NGT=IPRM(20)
      NA=NDV+1
      CALL EVAL(OBJ,X,G,NSEG,WK,WK(NA),NDV,INFO,NGT,IWK)
C     FINISHED?
      IF(INFO.EQ.0) GO TO 10
      GO TO 70
      END
      SUBROUTINE EVAL (OBJ,X,G,NSEG,DF,A,NDV,INFO,NGT,IC)
      DIMENSION X(*),G(*),DF(*),A(NDV,*),IC(*)
C
      IF(INFO.GT.1) THEN
         DO 20 I=1,NDV
            DF(I)=0.
            IF(NGT.GT.0) THEN
               DO 10 J=1,NGT
                  A(I,J)=0.
10             CONTINUE
            ENDIF
20       CONTINUE
      ENDIF
C     NSEG-ELEMENT BEAM.
      P=50000.
      E=2.0E+7
      AL=500.
      ALI=AL/FLOAT(NSEG)
      SIG=14000.
      YMX=2.5
```

```
C     VOLUME, STRESS CONSTRAINTS, H/B CONSTRAINTS.
      VOL=0.
      ALA=0.
      Y=0.
      YP=0.
      DO 40 I=1,NSEG
      BI=X(I)
      HI=X(I+NSEG)
      VOL=VOL+ALI*BI*HI
      AI=BI*(HI**3)/12.
      ALA=ALA+ALI
      AM=P*(AL+ALI-ALA)
      SIGI=.5*AM*X(I+NSEG)/AI
C     STRESS CONSTRAINTS.
      G(I)=1.*(SIGI/SIG-1.)
C     GRADIENTS
      IF(INFO.GT.1) THEN
C     DF
         DF(I)=HI*ALI
         DF(I+NSEG)=BI*ALI
         IF(NGT.GT.0) THEN
            DO 30 J=1,NGT
               IF(IC(J).EQ.I) THEN
                  A(I,J)=-6.*AM/(BI**2*HI**2*SIG)
                  A(I+NSEG,J)=-12.*AM/(BI*HI**3*SIG)
               ENDIF
               IF(IC(J).EQ.I+NSEG) THEN
                  A(I,J)=-.2
                  A(I+NSEG,J)=0.01
               ENDIF
30          CONTINUE
         ENDIF
      ENDIF
C     H/B CONSTRAINTS.
      G(I+NSEG)=X(I+NSEG)-20.*X(I)
      G(I+NSEG)=.01*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)
40    CONTINUE
      OBJ=VOL
      RETURN
      END
```

LISTING 4: CANTILEVER BEAM OUTPUT: NSEG = 5,000

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

                   DESIGN OPTIMIZATION TOOLS



                      (C) COPYRIGHT, 2001

                       VANDERPLAATS R&D

                ALL RIGHTS RESERVED, WORLDWIDE

                         BETA VERSION





 CONTROL PARAMETERS

 NUMBER OF DECISION VARIABLES,          NDV =     10000
 NUMBER OF CONSTRAINTS,                NCON =     10000
 PRINT CONTROL PARAMETER,            IPRINT =         2
 THE OBJECTIVE FUNCTION WILL BE MINIMIZED


 * * * REQUIRED STORAGE EXCEEDS USER INPUT VALUE FOR NRWK OR NRIWK

 STORAGE       INPUT     DESIRED
  NRWK       40000000    100150040
  NRIWK        300000        40071

 NGMAX WILL BE ADJUSTED IF POSSIBLE
 INPUT OR DEFAULT VALUE OF NGMAX =     10000

 * * * WILL TRY OPTIMIZATION WITH NGMAX =      3984
```

```
-- SCALAR PROGRAM PARAMETERS

REAL PARAMETERS
  1) PENALT =  1.00000E+00           6) DABOBJ =  1.00000E+01
  2) PMULT  =  5.00000E+00           7) DELOBJ =  1.00000E-03
  3) CTMIN  =  3.00000E-03           8) PENLTD =  1.00000E-02
  4) DABSTR =  1.00000E+01           9) PMULTD =  5.00000E+00
  5) DELSTR =  1.00000E-03

INTEGER PARAMETERS
  1) NGMAX  =      3984              6) ITRMOP =      2
  2) ISCAL  =         1              7) IWRITE =      6
  3) JTMAX  =        50              8) JWRITE =      0
  4) ITRMST =         2              9) MAXINT =  2000000000
  5) ITMAX  =       100             10) NDISCR =      1


    STORAGE REQUIREMENTS
ARRAY  DIMENSION        USED
  WK    40000000    39990040
 IWK      300000       40071


-- INITIAL FUNCTION VALUES

OBJ =  1.00000E+05

MAXIMUM CONSTRAINT VALUE =  0.33929     IS CONSTRAINT NUMBER         1


-- BEGIN OPTIMIZATION


-- BEGIN CONTINUOUS CYCLE    1


MAXIMUM CONSTRAINT VALUE   = 3.39286E-01 IS CONSTRAINT         1

NUMBER OF CRITICAL CONSTRAINTS =      1267

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  1.00049E+05 OBJECTIVE =  1.00000E+05

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  4.50190E+04   OBJECTIVE =  3.89634E+04


MAXIMUM CONSTRAINT VALUE   = 5.24156E-01 IS CONSTRAINT      4929
```

```
-- BEGIN CONTINUOUS CYCLE    2


MAXIMUM CONSTRAINT VALUE   = 5.24156E-01 IS CONSTRAINT      4929

NUMBER OF CRITICAL CONSTRAINTS =      9952

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  6.92413E+04 OBJECTIVE =  3.89634E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.19065E+04   OBJECTIVE =  5.03371E+04


MAXIMUM CONSTRAINT VALUE   = 1.11826E-01 IS CONSTRAINT      4965


-- BEGIN CONTINUOUS CYCLE    3


MAXIMUM CONSTRAINT VALUE   = 1.11826E-01 IS CONSTRAINT      4965

NUMBER OF CRITICAL CONSTRAINTS =      9970

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.81841E+04 OBJECTIVE =  5.03371E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.33602E+04   OBJECTIVE =  5.30002E+04


MAXIMUM CONSTRAINT VALUE   = 1.48099E-02 IS CONSTRAINT      4976


-- BEGIN CONTINUOUS CYCLE    4


MAXIMUM CONSTRAINT VALUE   = 1.48099E-02 IS CONSTRAINT      4976

NUMBER OF CRITICAL CONSTRAINTS =      9971

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.48000E+04 OBJECTIVE =  5.30002E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.36579E+04   OBJECTIVE =  5.35722E+04


MAXIMUM CONSTRAINT VALUE   = 5.48092E-03 IS CONSTRAINT      9875
```

```
-- BEGIN CONTINUOUS CYCLE    5


MAXIMUM CONSTRAINT VALUE   = 5.48092E-03 IS CONSTRAINT      9875

NUMBER OF CRITICAL CONSTRAINTS =      9698

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.40005E+04 OBJECTIVE =  5.35722E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37180E+04   OBJECTIVE =  5.37014E+04


MAXIMUM CONSTRAINT VALUE   = 1.64821E-03 IS CONSTRAINT      9945


-- BEGIN CONTINUOUS CYCLE    6


MAXIMUM CONSTRAINT VALUE   = 1.64821E-03 IS CONSTRAINT      9945

NUMBER OF CRITICAL CONSTRAINTS =      9250

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37845E+04 OBJECTIVE =  5.37014E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37444E+04   OBJECTIVE =  5.37290E+04


MAXIMUM CONSTRAINT VALUE   = 1.43720E-03 IS CONSTRAINT      9945


-- BEGIN CONTINUOUS CYCLE    7


MAXIMUM CONSTRAINT VALUE   = 1.43720E-03 IS CONSTRAINT      9945

NUMBER OF CRITICAL CONSTRAINTS =      4977

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.38058E+04 OBJECTIVE =  5.37290E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37413E+04   OBJECTIVE =  5.37406E+04


MAXIMUM CONSTRAINT VALUE   = 4.47931E-04 IS CONSTRAINT      9974
```

```
-- OPTIMIZATION IS COMPLETE

NUMBER OF UNCONSTRAINED MINIMIZATIONS =    7

CONSTRAINT TOLERANCE, CT =-5.00000E-02   CTMIN = 3.00000E-03

THERE ARE      9995 ACTIVE CONSTRAINTS AND         0 VIOLATED CONSTRAINTS

THERE ARE        12 ACTIVE SIDE CONSTRAINTS

TERMINATION CRITERIA

RELATIVE CONVERGENCE CRITERION WAS MET FOR  2 CONSECUTIVE ITERATIONS


-- OPTIMIZATION RESULTS


OBJECTIVE, F(X) =    5.37406E+04

MAXIMUM CONSTRAINT VALUE =  4.47931E-04  IS CONSTRAINT NUMBER      9974

FUNCTION CALLS =     235

GRADIENT CALLS =      44



-- BEGIN DISCRETE VARIABLE OPTIMIZATION


-- INITIAL FUNCTION VALUES

OBJ =   53741.

MAXIMUM CONSTRAINT VALUE =  4.47931E-04 IS CONSTRAINT NUMBER      9974


-- BEGIN DISCRETE CYCLE    1


MAXIMUM CONSTRAINT VALUE   = 4.47931E-04 IS CONSTRAINT      9974

NUMBER OF CRITICAL CONSTRAINTS =      1092

AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37945E+04 OBJECTIVE =  5.37406E+04

AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
PSEUDO-OBJECTIVE =  5.37879E+04   OBJECTIVE =  5.37378E+04

MAXIMUM CONSTRAINT VALUE   = 8.48126E-05 IS CONSTRAINT      9938

THERE ARE        66 DISCRETE VALUES
```

```
-- BEGIN DISCRETE CYCLE    2


  MAXIMUM CONSTRAINT VALUE   = 8.48126E-05 IS CONSTRAINT      9938

  NUMBER OF CRITICAL CONSTRAINTS =      1114

  AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
  PSEUDO-OBJECTIVE =  5.39882E+04 OBJECTIVE =  5.37378E+04

  AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
  PSEUDO-OBJECTIVE =  5.39871E+04   OBJECTIVE =  5.37370E+04


  MAXIMUM CONSTRAINT VALUE   = 6.10089E-05 IS CONSTRAINT      9938

  THERE ARE        73 DISCRETE VALUES


  -- BEGIN DISCRETE CYCLE    3


  MAXIMUM CONSTRAINT VALUE   = 6.10089E-05 IS CONSTRAINT      9938

  NUMBER OF CRITICAL CONSTRAINTS =      1023

  AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
  PSEUDO-OBJECTIVE =  5.49873E+04 OBJECTIVE =  5.37370E+04

  AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
  PSEUDO-OBJECTIVE =  1.10861E+07   OBJECTIVE =  5.37493E+04


  MAXIMUM CONSTRAINT VALUE   = 9.05131E-02 IS CONSTRAINT      4970

  THERE ARE     10000 DISCRETE VALUES


  -- BEGIN DISCRETE CYCLE    4


  MAXIMUM CONSTRAINT VALUE   = 9.05131E-02 IS CONSTRAINT      4970

  NUMBER OF CRITICAL CONSTRAINTS =      4825

  AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
  PSEUDO-OBJECTIVE =  5.52156E+07 OBJECTIVE =  5.37493E+04

  AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
  PSEUDO-OBJECTIVE =  5.51171E+04   OBJECTIVE =  5.48645E+04

  MAXIMUM CONSTRAINT VALUE   = 1.00003E-03 IS CONSTRAINT      6720

  THERE ARE     10000 DISCRETE VALUES
```

```
-- BEGIN DISCRETE CYCLE    5

  MAXIMUM CONSTRAINT VALUE   = 1.00003E-03 IS CONSTRAINT      6720

  NUMBER OF CRITICAL CONSTRAINTS =        88

  AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
  PSEUDO-OBJECTIVE =  5.61272E+04 OBJECTIVE =  5.48645E+04

  AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
  PSEUDO-OBJECTIVE =  5.61296E+04   OBJECTIVE =  5.48645E+04


  MAXIMUM CONSTRAINT VALUE   = 1.00003E-03 IS CONSTRAINT      6720

  THERE ARE     10000 DISCRETE VALUES


  -- BEGIN DISCRETE CYCLE    6


  MAXIMUM CONSTRAINT VALUE   = 1.00003E-03 IS CONSTRAINT      6720

  NUMBER OF CRITICAL CONSTRAINTS =        88

  AT START OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
  PSEUDO-OBJECTIVE =  6.11899E+04 OBJECTIVE =  5.48645E+04

  AT END OF UNCONSTRAINED MINIMIZATION SUB-PROBLEM
  PSEUDO-OBJECTIVE =  6.11899E+04   OBJECTIVE =  5.48645E+04


  MAXIMUM CONSTRAINT VALUE   = 1.00003E-03 IS CONSTRAINT      6720

  THERE ARE     10000 DISCRETE VALUES


  -- OPTIMIZATION IS COMPLETE

  NUMBER OF UNCONSTRAINED MINIMIZATIONS =    6

  CONSTRAINT TOLERANCE, CT =-5.00000E-02   CTMIN = 3.00000E-03

  THERE ARE      9584 ACTIVE CONSTRAINTS AND         0 VIOLATED CONSTRAINTS

  THERE ARE        22 ACTIVE SIDE CONSTRAINTS

  THERE ARE     10000 DISCRETE VALUES

  TERMINATION CRITERIA

  RELATIVE CONVERGENCE CRITERION WAS MET FOR  2 CONSECUTIVE ITERATIONS

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

```
-- OPTIMIZATION RESULTS

  OBJECTIVE, F(X) =    5.48645E+04

  MAXIMUM CONSTRAINT VALUE =  1.00003E-03  IS CONSTRAINT NUMBER      6720

  FUNCTION CALLS =     320

  GRADIENT CALLS =      59

  FINAL OBJ =   54864.53
```

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 08-2002	**2. REPORT TYPE** Contractor Report	**3. DATES COVERED** *(From - To)*

4. TITLE AND SUBTITLE Very Large Scale Optimization	**5a. CONTRACT NUMBER** NAS1-00102
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Vanderplaats, Garrett	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER 253-02-98-01
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Vanderplaats Research and Development, Inc. 1767 S. 8th Street Colorado Springs, Colorado 80906	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001	**10. SPONSOR/MONITOR'S ACRONYM(S)** NASA
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) NASA/CR-2002-211768

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 61
Availability: NASA CASI (301) 621-0390 Distribution: Standard

13. SUPPLEMENTARY NOTES
Vanderplaats, Vanderplaats Research and Development
An electronic version can be found at http://techreports.larc.nasa.gov/ltrs/ or http://techreports.larc.nasa.gov/cgi-bin/NTRS
Langley Technical Monitor: James C. Townsend

14. ABSTRACT

The purpose of this research under the NASA Small Business Innovative Research program was to develop algorithms and associated software to solve very large nonlinear, constrained optimization tasks. Key issues included efficiency, reliability, memory, and gradient calculation requirements. This report describes the general optimization problem, ten candidate methods, and detailed evaluations of four candidates. The algorithm chosen for final development is a modern recreation of a 1960s external penalty function method that uses very limited computer memory and computational time. Although of lower efficiency, the new method can solve problems orders of magnitude larger than current methods. The resulting BIGDOT software has been demonstrated on problems with 50,000 variables and about 50,000 active constraints. For unconstrained optimization, it has solved a problem in excess of 135,000 variables. The method includes a technique for solving discrete variable problems that finds a "good" design, although a theoretical optimum cannot be guaranteed. It is very scalable in that the number of function and gradient evaluations does not change significantly with increased problem size. Test cases are provided to demonstrate the efficiency and reliability of the methods and software.

15. SUBJECT TERMS
Optimization; Structural optimization; Large-scale optimization; Multidisciplinary design optimization; Discrete optimization

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			STI Help Desk (email: help@sti.nasa.gov)
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	55	(301) 621-0390