8th AIAA/USAF/NASA/ISSMO
Symposium on Multidisciplinary
Analysis and Optimization
6-8 Sept. 2000 Long Beach, CA

A00-40072

AIAA 2000-4809

VERY LARGE SCALE OPTIMIZATION

G. N. Vanderplaats, President
Fellow
Vanderplaats Research & Development, Inc.
1767 S. 8th Street, Suite 100
Colorado Springs, CO 80906, USA

ABSTRACT

As optimization problems grow in size, modern algorithms such as Sequential Quadratic Programming require large amounts of computer memory. Also, the optimizer itself begins to use considerable CPU time, relative to the function and gradient computations.

A method is presented here, based on Sequential Unconstrained Minimization Techniques using an Exterior Penalty Function, which requires very little memory to store information and does not require time consuming optimization calculations. The penalty for these advantages is that the optimization requires 3-5 times as many function evaluations to converge to a solution. Examples are presented to demonstrate the method.

INTRODUCTION

As optimization becomes more widely accepted, the size of the problems being addressed has grown dramatically. In structural optimization, problems with thousands of design variables and over a million constraints are now being addressed. For multidiscipline (MDO) problems, the number of variables and constraints can be even larger. In this case, due to our inability to handle very large numbers of variables, relatively complicated decomposition methods have been resorted to.

While many common methods such as Sequential Quadratic Programming can theoretically handle very large problems, two issues quickly arise. First, these methods require solution of a large and often time consuming sub-optimization problem and, second, they require storage of large amounts of information (both gradients and Lagrangian approximation information). This second issue can be dealt with using spill logic, but this can also be complicated and inefficient.

This suggests a need for methods which will solve very large problems with limited central memory and which avoid large sub-optimization tasks. Such a method will be presented here. It has been developed primarily for structural optimization, but is equally useful for MDO tasks where gradients are available. In either case, it is desirable that high quality approximations are available since the new algorithm requires more function and gradient evaluations than before.

BACK TO THE FUTURE

In the 1960's Sequential Unconstrained Minimization Techniques (SUMT) were popular [1]. Over the years, these methods were largely abandoned in favor of SQP and similar methods. Recently, there has been renewed interest in SUMT [2].

In the present research, several SUMT approaches were studied, with the (somewhat surprising) result that a modified exterior penalty function method achieves the goals of this study best.

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

Minimize

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

Subject to;

$$X_i^L \le X_i \le X_i^U \qquad i = 1, n \quad (2)$$

where $\mathbf{X}$ is the vector of design variables, $F(\mathbf{X})$ is the objective function and $g_j(\mathbf{X})$ 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 [3], but are calculated by a proprietary formula. Equation (2) 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 Eq. (1) is solved by the Fletcher-Reeves conjugate direction method [4], which requires virtually no memory.

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

$$\nabla\Phi(\mathbf{X}) = \nabla F(\mathbf{X}) \qquad (3)$$

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

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 Eq. (3) 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 [5]. This is presented in Table 1, 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 1: 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 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. The number of constraints equals the number of design variables.

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, the search direction is found as;

If q = 1

$$S^q = -\nabla\Phi(X^{q-1}) \qquad (4)$$

If q > 1

$$S^q = -\nabla\Phi(X^{q-1}) + \beta S^{q-1} \qquad (5)$$

where

$$\beta = \frac{|\nabla\Phi(X^{q-1})|^2}{|\nabla\Phi(X^{q-2})|^2} \qquad (6)$$

and q is the iteration number.

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 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.

EXAMPLES

To evaluate the proposed method, a prototype program was created and used to solve a variety of optimization problems. Here, preliminary results are presented for several design examples.

Cantilevered Beam

The cantilevered beam shown in Figure 1 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.

The design task is now defined as;

Minimize

$$V = \sum_{i=1}^{N} b_i h_i l_i \tag{7}$$

Subject to;

$$\frac{\sigma_i}{\bar{\sigma}} - 1 \le 0 \qquad i = 1, N \tag{8}$$

$$h_i - 20b_i \le 0 \qquad i = 1, N \tag{9}$$

$$\frac{y_N}{\bar{y}} - 1 \le 0 \tag{10}$$

$$b_i \ge 1.0 \qquad i = 1, N \tag{11}$$

$$h_i \ge 5.0 \qquad i = 1, N \tag{12}$$

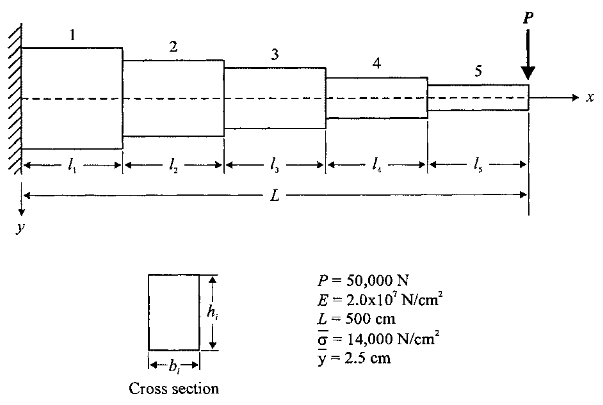

Fig. 1 Cantilevered Beam

Here $\bar{\sigma}$ is the allowable bending stress and $\bar{y}$ is the allowable displacement. σ_i is the bending stress at the left end of element i and y_N is the displacement at the tip of the beam. This is a design problem in $n = 2N$ variables. There are $N + 1$ nonlinear constraints defined by Eqs. (8) and (10), N linear constraints defined by Eq. (9), and $2N$ side constraints on the design variables defined by Eqs. (11) and (12).

Here, N=500, 2500 and 5000 was used, to create problems of n = 1000, 5000 and 10,000 design variables, respectively. The solutions are presented in Table 2.

For the 10,000 variable case sufficient storage was not provided to store the desired number of constraint gradients, so a maximum of 2986 gradients were calculated at any one time. Because more constraint gradients than this were required during optimization, reduced storage logic was used and so the number of gradient evaluations was increased (each time control is returned to the user for gradients, it is considered a gradient evaluation in Table 2). Depending on how gradients are calculated, this can increase the run time, but otherwise will have no effect on the optimum.

Table 2: Optimization Results

	INITIAL VALUE	**OPTIMUM (n=1000)**	**OPTIMUM (n=5000)**	**OPTIMUM (n=10000)**
OBJECTIVE	100,000	63,670	63,650	63,905
MAX g	0.5625	3.73E-5	8.02E-5	-4.34E-5
# CYCLES		6	6	10
FUNCTION EVALS		157	173	363
GRADIENT EVALS		37	40	148

The minimum storage requirements for optimization are approximately 11n + 3m, where n is the number of design variables and m is the number of constraints. Additional storage will be used, if available, up to another n*m. For the 10,000 variable (10,001 constraints) problem given above, a minimum of about 140,000 words and a maximum of just over 100 million words of storage will be used. In this example, 30 million words were specified and the reduced storage logic was used. By comparison, the SQP method contained in the DOT optimizer [5] would require over $1.2\text{X}10^9$ words to store the needed information.

As a second case, this problem is solved by deleting the displacement constraint of Eq. 10. In this case, the optimum should be fully constrained (the number of active constraints equals the number of design variables). The theoretical optimum is 53,714 if the beam height is a continuous function.

Table 3 shows the results for this case, where the number of active constraints within 5% at the optimum are shown.

Table 3: Optimization Results

	INITIAL VALUE	**OPTIMUM (n=1000)**	**OPTIMUM (n=5000)**	**OPTIMUM (n=10000)**
OBJECTIVE	100,000	53,867	53,773	53,826
MAX g	0.3393	1.55E-4	5.38E-4	6.06E-4
# CYCLES	-	6	7	9
# ACTIVE CONSTR.	127	999	4970	9839
FUNCTION EVALS	1	196	189	317
GRADIENT EVALS	-	46	45	205

While a true fully constrained optimum is not achieved for large numbers of variables, it is very close to fully constrained, indicating a high degree of robustness.

Topology Optimization

The GENESIS structural analysis and optimization program [6] will optimize structures using approximation concepts. When solving topology optimization problems, the number of design variables can become very large.

Here, a classical problem known as the Mitchell truss is designed using GENESIS. The design conditions are shown in Figure 2

.

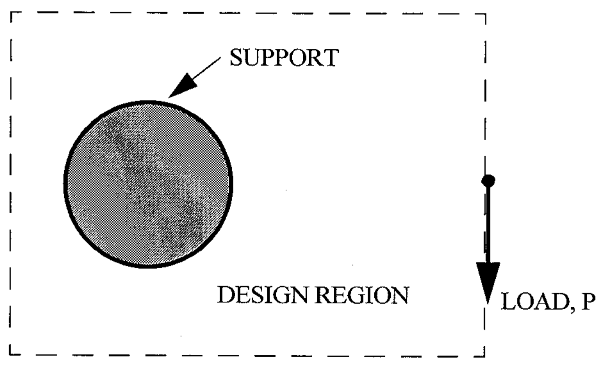

Fig. 2 Topology Design Conditions

The goal is to find the structure supported by the round bar and subject to a single point load on the right, as shown in the figure. The initial design is a planar structure made up of over 47,000 quadrilateral plate elements filling the design region. GENESIS treats the density of each element as an independent design variable, where the "density factor" can range from zero to one. In this case symmetry was imposed about the horizontal mid-plane for the left half of the structure, leaving a total of just over 35,000 independent design variables. Young's modulus is linked to the density, so as the density approaches zero, so does the stiffness.

We wish to find the optimum structure which is as stiff as possible while using only 10% of the original material.

The Figure 3 shows the optimum topology obtained by GENESIS. The approximate optimization phase of GENESIS required about seven percent of the total run time. Of this, the vast majority of time was spent evaluating the approximate functions and their gradients.

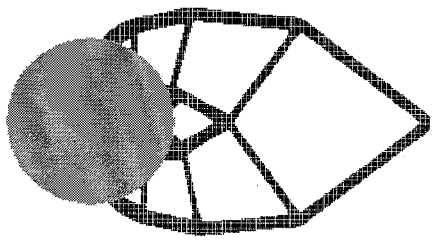

Fig. 3 Optimum Topology

As a final example, consider the simplified wing model shown in Figure 4.

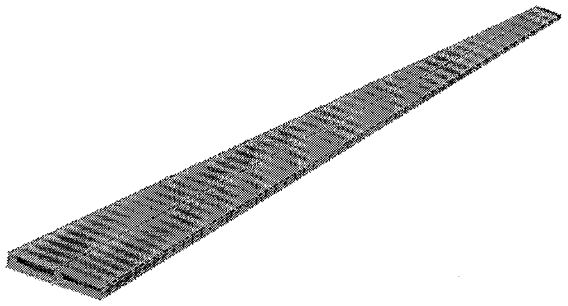

Fig. 4 Wing Model

This example was also solved using the GENESIS structural analysis/optimization program, were the approximate optimization sub-problem was solved using the proposed method.

The structure was modeled with 2400 quadrilateral elements for the skins, spars and ribs and 600 rod elements for the spar caps. The skins, spars and ribs were modeled using composite elements with four thickness design variables per element. The spar caps were isotropic, with a single cross sectional area design variable per element.

A single static load case of uniform lift was imposed, along with an eigenvalue load case.

The objective was to minimize mass. There were a total of 10,200 independent design variables, 232,000 stress and failure index constraints and a lower bound constraint on the first fundamental frequency.

Due to memory requirements of GENESIS, using its constraint deletion feature, only about 41,000 constraints were retained in the approximate optimization phase.

Only about six megawords of storage were allocated to the optimizer, allowing for the calculation of 580 gradients at a time.

GENESIS solves optimization problems by creating high quality approximations which are sent to the optimizer. Thus, several analysis/optimization cycles are used to find the optimum.

Figure 5 shows the optimization progress, where the normalized objective function and maximum constraint violation are plotted.

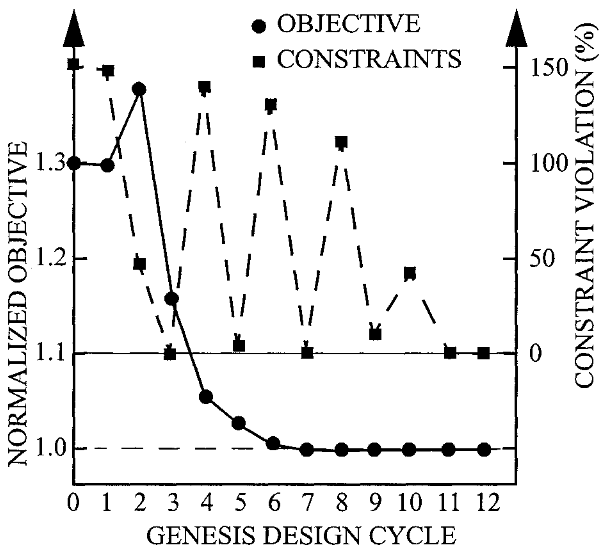

Fig. 5 Optimization History

Because a limited number of constraints were retained for the approximate optimization phase, this led to "constraint switching." Thus, during a given design cycle, the approximate optimum would be feasible, but when an analysis was performed, one or more non-retained constraints were found to be violated. This was overcome during the optimization.

At the optimum, there were 7048 active stress and failure index constraints, one active eigenvalue constraint and 3846 active side constraints.

SUMMARY

An algorithm has been developed for solving very large optimization tasks, which uses limited central memory and which avoids solution of a large sub-optimization task. The memory requirements grow only linearly with problem size. Preliminary results with this algorithm indicate that it can reliably solve very large optimization tasks with reasonable efficiency.

ACKNOWLEDGEMENT

This research was supported by NASA Langley Research Center under a Phase I SBIR Contract NAS1-99026 and is continuing under a Phase II Contract. This support is greatly appreciated.

REFERENCES

1. Fiacco, A. V., and G. P. McCormick: "Nonlinear Programming: Sequential Unconstrained Minimization Techniques," John Wiley and Sons, New York, 1968.
2. Hager, W.W., D. W. Hearn and P. M. Pardalos: "Large Scale Optimization; State of the Art," Kluwer Academic Publishers, 1994, pp. 45-67.
3. 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.
4. Fletcher, R. and C. M. Reeves: Function Minimization by Conjugate Gradients, Br. Computer J., vol. 7, no. 2, pp. 149-154, 1964.
5. DOT Users Manual, Version 5.0: Vanderplaats Research & Development, Inc., Colorado Springs, CO, 1999.
6. GENESIS User's Manual, Version 6.0: Vanderplaats Research & Development, Inc., Colorado Springs, CO, 2000.