KSME Journal, Vol. 1, No. 2, pp. 167~171, 1987.

THE GAUSS OPTIMIZATION METHOD FOR PROBLEMS WITH GENERAL NONLINEAR CONSTRAINTS

T. E. Potter*, K. D. Willmert and M. Sathyamoorthy****

(Received November 2, 1987)

A new algorithm is presented for finding the minimum of a nonnegative objective function subject to general nonlinear constraints. This algorithm, based of Gauss' method for unconstrained problems, is developed as an extension to the Gauss constrained technique for linear constraints. The derivation of the algorithm, using a Lagrange multiplier approach, is based on the Kuhn-Tucker conditions so that, when the iteration process terminates, these conditions are automatically satisfied. A feasible design is maintained throughout the iteration process. The solution of preliminary examples indicate excellent results in terms of the number of objective function evaluations required by the algorithm to obtain an optimal design.

Key Words: Gauss Optimization, Nonlinear Constraint, Computational Time, Mechanism Design

1. INTRODUCTION

The optimal design of many complex structural and mechanical systems is hindered by the large computational times involved. Most currently available optimization techniques require a large number of analyses to obtain the optimal design. For small problems, or ones in which the analysis is simple, these methods are adequate; however, for large problems, or where a time consuming analysis is required, more efficient optimization methods are needed. The goal of this research was to develop such methods, particularly techniques applicable to mechanical mechanism design where the members are deforming because of high speed motion and large external forces. Computational times to perform an analysis are enormous for problems of this type where large deformations are occurring involving nonlinear material characteristics. Thus the goal of the methods developed was to reduce the number of analyses, even at the expense of increased computational effort in the optimization technique itself, i. e. additional effort in finding new candidate design points to analyze.

The methods generated in this research take advantage of the special characteristics of the optimization problem, similar to optimality criterion techniques. This greatly improves their efficiency. For most mechanism design problems, the objective function can be formulated as a sum of squared quantities such as the difference between the desired performance and the actual performance of the mechanism at specified points during its motion. Thus the techniques were developed specifically to handle problems of this type, although the methods are applicable to objective functions which are general sums of nonnegative quantities, such as weight. Many mechanism problems have constraints which are only linear functions of the design variables. Thus a special method was developed for problems of this type. Other problems have constraints which are linear or quadratic, and another method was developed for this case. Some mechanism design problems have more general nonlinear constraints. Methods to handle these cases are currently being investigated.

All of the techniques developed in this work have been based on Gauss' method (Himmelblau, 1972) which is applicable to problems without constraints. Wilde (1981) has shown this method to be particularly efficient on simple mechanism design problems. The research presented in this paper has extended this method to handle various types of constraints common to more complex mechanisms.

2. FORMULATION

For an unconstrained sum-of-squares objective functions

$$f(\vec{x}) = \vec{\phi}^T \vec{\phi}, \tag{1}$$

where $\vec{\phi}$ is a vector of linear or nonlinear functions ϕ_1 thru ϕ_p in $\vec{x}$, the Gauss method for calculating the next iteration of the design variables, $\vec{x}_{k+1}$, given a current design, $\vec{x}_k$, is

$$\vec{x}_{k+1} = \vec{x}_k - [J(\vec{x}_k) J^T(\vec{x}_k)]^{-1} J(\vec{x}_k) \vec{\phi}(\vec{x}_k), \tag{2}$$

where

$$J(\vec{x}) = \begin{bmatrix} \vdots & & \vdots \\ \nabla\phi_1(\vec{x}) & \cdots & \nabla\phi_p(\vec{x}) \\ \vdots & & \vdots \end{bmatrix} = \nabla\vec{\phi}^T. \tag{3}$$

*Mechanical Engineering Technology Department, Pennsylvania State University at Harrisburg, the Capital College, Middletown, PA 17057, USA

**Mechanical and Industrial Engineering Department, Clarkson University, Potsdam, NY 13676, USA

It is observed that only first derivatives of the ϕ functions are required and that the new design point is calculated directly from the current design without using a step length determination with associated one dimensional minimization. Himmelblau (1972) has shown this method to be very efficient for unconstrained minimization problems.

This technique has been extended to handle linear inequality constraints of the form

$$g_i(\vec{x}) = \vec{b}_i{}^T\vec{x} - c_i \leq 0, \quad i=1,\cdots,M \tag{4}$$

as well as equality constraints

$$h_i(\vec{x}) = \vec{d}_i{}^T\vec{x} - e_i = 0, \quad i=1,\cdots,L. \tag{5}$$

In the derivation of the optimization method, the ϕ_i functions are assumed to be linear in $\vec{x}$ of the form

$$\vec{\phi} = J^T\vec{x} + \vec{u}. \tag{6}$$

where J is a constant matrix. However the resulting technique is applicable to problems in which the ϕ_i's are general nonlinear functions of $\vec{x}$.

At iteration k, the L equality constriants and any of the inequality constraints that are active can be combined and written in the form

$$B^T\vec{x} - \vec{C} = 0. \tag{7}$$

If at the next iteration, $k+1$, the variables $\vec{x}_{k+1}$ are at the optimum design, then the Kuhn-Tucker conditions will be satisfied

$$\nabla f(\vec{x}_{k+1}) + B\vec{\lambda} = 0 \tag{8}$$

$$B^T\vec{x}_{k+1} - \vec{C} = 0 \tag{9}$$

and

$$\vec{\lambda} \geq 0 \tag{10}$$

where $\vec{\lambda}$ is a vector of Lagrange multipliers. The gradient of f is given by

$$\nabla f(\vec{x}) = 2J\vec{\phi}(\vec{x}). \tag{11}$$

Expanding $\vec{\phi}(\vec{x})$ in a Taylor series results in

$$\vec{\phi}(\vec{x}_{k+1}) = \vec{\phi}(\vec{x}_k) + J^T[\vec{x}_{k+1} - \vec{x}_k] + (\textit{higher order terms}) \tag{12}$$

It is noted that the higher order terms are equal to zero if $\vec{\phi}$ is linear. If $\vec{\phi}$ is not linear then these terms will be neglected and the expansion is only approximate. If Eq. (12) is substituted into Eq. (11), and evaluated at $\vec{x}_{k+1}$, the result is

$$\nabla f(\vec{x}_{k+1}) = 2J[\phi(\vec{x}_k) + J^T[\vec{x}_{k+1} - \vec{x}_k]]. \tag{13}$$

This may be substituted into the first Kuhn-Tucker condition, Eq. (8), and then solved for $\vec{x}_{k+1}$

$$\vec{x}_{k+1} = \vec{x}_k - [2JJ^T]^{-1}[2J\vec{\phi}(\vec{x}_k) + B\vec{\lambda}]. \tag{14}$$

Plugging this equation for $\vec{x}_{k+1}$ into the second Kuhn-Tucker condition, Eq. (9), yields

$$B^T\vec{x}_k - C - B^T[2JJ^T]^{-1}[2J\vec{\phi}(\vec{x}_k) + B\vec{\lambda}] = 0. \tag{15}$$

If the same set of constraints that are active at $\vec{x}_{k+1}$ were also active at $\vec{x}_k$, then $B^T\vec{x}_k - \vec{C} = 0$. Using this result, Eq. (15) can be solved for $\vec{\lambda}$ producting

$$\vec{\lambda} = -[B[2JJ^T]^{-1}B]^{-1}B^T[2JJ^T]^{-1}2J\vec{\phi}(\vec{x}_k). \tag{16}$$

Substituting this back into Eq. (14) and simplifying yields an iterative expression for $\vec{x}_{k+1}$ which will give the optimum solution in one iteration if the constraints that are active at the optimum point (iteration $k+1$) are active at iteration k

$$\vec{x}_{k+1} = \vec{x}_k - [I - [JJ^T]^{-1}B[B^T[JJ^T]^{-1}B]^{-1}B^T][JJ^T]^{-1}J\vec{\phi}(\vec{x}_k). \tag{17}$$

This expression is equivalent to that derived by Paradis and Willmert (1983) using a Gradient Projection method as a foundation. The technique converges to the optimal design in one iteration if the objective function, f, is quadratic and the starting point is on the constraints which are active at the optimal design. If f is not quadratic, the technique can still be applied, but it will generally require several iterations to reach the optimal design. When the technique terminates, the Kuhn-Tucker conditions will be satisfied independent of the form of the objective function.

Paradis and Willmert deminstrated the efficiency of this method by solving several examples. One example presented was the optimal design of a four-bar mechanism to generate a desired coupler point path. The Gauss constrained technique was compared with the Davidon-Fletcher-Powell method using an interior penalty function approach to handle constraints. Using four different starting points, the Gauss constrained method required from 23 to 33 objective function evaluations whereas the Davidon-Fletcher-Powell method required from 209 to 622. While not all starting points yielded the same optimal design, both methods reached the same local minimum from each starting point. Other examples also showed considerable improvement over existing methods.

The Gauss method has recently been extended to include quadratic inequality constraints or quadratic approximations to higher order nonlinear constraints. In this work the constraints are assumed to have the form

$$g_i(\vec{x}) = \frac{1}{2}\vec{x}^T A_i\vec{x} + \vec{B}_i{}^T\vec{x} - C_i \leq 0, \quad i=1,\cdots,M. \tag{18}$$

If at iteration $k+1$ there are r active constraints ($r \leq M$), the Kuhn-Tucker conditions will be

$$\nabla f(\vec{x}_{k+1}) + \sum_{j=1}^{r}[A_j\vec{x}_{k+1} + \vec{B}_j]\lambda_j = 0 \tag{19}$$

$$\frac{1}{2}\vec{x}^T{}_{k+1}A_j\vec{x}_{k+1} + \vec{B}_j^T\vec{x}_{k+1} - C_j = 0, \quad j=1,\cdots,r \tag{20}$$

and

$$\vec{\lambda} \geq 0 \tag{21}$$

where the summation in Eq. (19) and the j subscript in Eq. (20) refer to the set of active constraints only.

Using a derivation similar to that for linear constraints, substituting the expression for the gradient of f, Eq. (13), into the first Kuhn-Tucker condition, Eq. (19), and solving for $\vec{x}_{k+1}$ produces

$$\vec{x}_{k+1}=\left[2JJ^T+\sum_{j=1}^{r} A_j\lambda_j\right]^{-1}\left[2JJ^T\vec{x}_k-\nabla f(\vec{x}_k)-\sum_{j=1}^{r}\vec{B}_j\lambda_j\right]. \tag{22}$$

This expression for $\vec{x}_{k+1}$ in terms of $\vec{\lambda}$ is now substituted into the second Kuhn-Tucker condition, Eq. (20), to obtain

$$\begin{aligned}&\frac{1}{2}\left[\left[2JJ^T+\sum_{j=1}^{r} A_j\lambda_j\right]^{-1}\left[2JJ^T\vec{x}_k-\nabla f(\vec{x}_k)-\sum_{j=1}^{r}\vec{B}_j\lambda_j\right]\right]^T\\&A_i\left[\left[2JJ^T+\sum_{j=1}^{r} A_j\lambda_j\right]^{-1}\left[2JJ^T\vec{x}_k-\nabla f(\vec{x}_k)-\sum_{j=1}^{r}\vec{B}_j\lambda_j\right]\right]\\&+B_i\left[\left[2JJ^T+\sum_{j=1}^{r} A_j\lambda_j\right]^{-1}\left[2JJ^T\vec{x}_k-\nabla f(\vec{x}_k)-\sum_{j=1}^{r}\vec{B}_j\lambda_j\right]\right]\\&-C_i=0,\ i=1,\cdots,\ r.\end{aligned} \tag{23}$$

These r nonlinear equations in terms of the unknowns, λ_1, thru λ_r, and the old values of the design variables, $\vec{x}_k$, are solved by an iterative process for the values of λ_1 thru λ_r. The lambda values are then substituted into Eq. (22) which will yield new values for the design variables. It is observed that the matrix $2JJ^T$ is the matrix of second partial derivatives, G, of the objective function if it is quadratic. Thus, by replacing $2JJ^T$ in Eq. (22) and (23), this technique becomes a modification of the second order method rather than the Gauss method.

At the optimum design, all constraints will either be satisfied (less than zero) or active (equal to zero) and each active constraint will have a corresponding lambda whose value is greater than or equal to zero. If, at some iteration, the set of design variables yields a violated constraint, then obviously the optimum point has not been reached. in this case, the newly violated constraint is added to the set of active constraints and the procedure allowed to continue. If at some iteration, the set of design variables yields all active or satisfied constraints, but one or more of the active constraints has a corresponding negative lambda, then the optimum design has also not been reached. The negative lambda implies that the iteration process would like to move away from the corresponding constraint boundary toward the feasible region where the constraint is satisfied. Thus, the constraint is dropped from the set of active constraints and the process allowed to continue. If more than one negative lambda existed, then constraints are dropped one at a time starting with the constraint with the most negative lambda.

A constraint is added to the set of active constraints if it should become either active (equal to zero) or violated (greater than zero) when the step is taken from $\vec{x}_k$ to $\vec{x}_{k+1}$. In the case where a constraint becomes violated, a line is "drawn" between $\vec{x}_k$ and $\vec{x}_{k+1}$ and the actual step is taken to the farthest point along the line so that no constraints are violated. In effect, this procedure is the same as stepping back from $\vec{x}_{k+1}$ toward $\vec{x}_k$ until the newly violated constraint is just active (equal to zero). The constraint is the added to the set of active constraints for the next iteration.

An example problem with quadratic constraints given by Boston, Willmert and Sathyamoorthy (1984) shows this method to be very efficient when compared to the generalized reduced gradient method (GRG). The problem consisted of finding the optimal design of a four-bar mechanism (minimizing the error between the actual coupler point path and a given path) subject to several linear constraints on link length and movability. Additionally, constraints were placed on the crank pin to limit its location to the intersection of two circular (quadratic) regions. The program was run for a four by four matrix of problems which included four different starting points and four different conditions on the quadratic constraints. For all sixteen runs, the number of objective function evaluations for this new method ranged from 8 to 32 (average was 15), while the GRG method required from 303 to 699 (average was 502) evaluations.

The interesting information here is that the solution to this quadratically constrained four-bar mechanism problem used no more objective function evaluations than the linearly constrained four-bar mechanism example considered by Paradis and Willmert. While these two examples are necessarily different, this tendency toward requiring similar numbers of function evaluations for different classes of problems is very desirable. The net result is that we now have an optimization procedure for objective functions which are the sum of squared quantities, subject to linear and quadratic constraints that not only requires relatively few function evaluations, but seems to be constraint order independent. Now the need is to develop a method which will also work for higher order constraints.

Boston, et al. (1984) attempted to apply this method to higher order problems, but met with mixed results. The problems encountered seemed to be caused by the higher order constraints rather than with the higher order objective functions. There are several limitations implicit in the algorithm which appear to be the source of the problems encountered. The first limitation has to do with the application of the constraints, Eq. (18), to the first Kuhn-Tucker condition, Eq. (19). When approximating a higher order function by a quadratic Taylor series expansion about some point $\vec{x}_0$, not only is the A_i matrix a functin of $\vec{x}_0$, but so is the $\vec{B}_i$ vector and the C_i scalar. Thus the constraint approximation, Eq. (18), should be written as

$$g_i(\vec{x})\cong\frac{1}{2}\vec{x}^T A_i(\vec{x}_0)\vec{x}+[\vec{B}_i(\vec{x}_0)]^T\vec{x}-C_i(\vec{x}_0)\le 0,\quad i=1,\cdots,M \tag{24}$$

where

$$\vec{x}=\vec{x}_0+\Delta\vec{x}. \tag{25}$$

As $\vec{x}$ approaches $\vec{x}_0$ (or $\Delta\vec{x}$ approaches 0), this approximation approaches the exact value of the constraint. Thus as the algorithm progresses along and constraints are added and dropped, the constraints must be reapproximated at the latest design to keep the step size small. This can be achieved by taking the new values of $\vec{x}$ as generated by Eq. (22) and

substituting them into the actual constraint equations to get improved values for the $A_i(\vec{x}_0)$, $\vec{B}_i(\vec{x}_0)$ and $C_i(\vec{x}_0)$ terms in Eq. (24) with resspect to the current design point.

The second limitation involves the second Kuhn-Tucker condition, Eq. (20), which is used to obtain the equation for the new values of $\vec{\lambda}$, Eq. (23). This is simply the equation for the active constraints. In the original formulation, an attempt was made at obtaining a linear approximation in $\vec{\lambda}$ for this constraint equation. This would allow Eq. (23). This is simply the equation for the active constraints. In the original formulation, an attempt was made at obtaining a linear approximation in $\vec{\lambda}$ for this constraint equation. This would allow Eq. (23) to be solved explicitly for $\vec{\lambda}$. However, failing this, an iterative procedure was employed to find the values for $\vec{\lambda}$. Now that an iterative process is required, there is no advantage in keeping a quadratic approximation when the actual constraint will work just as well. Replacing Eq. (20) with the active nonlinear constraint equations will remove any errors due to the approximation process.

The stepping back procedure for violated constraints, described above, can also be a source of problems. With non-convex programming problems this procedure may lead to a situation where the algorithm cannot move away from a non-optimum design. Because the stepping back procedure assumed a straight line path between the two design points, it is possible, when backing out of a newly violated constraint, to move into the violated region of the constraint that was active at the beginning of the step. The procedure would then step back still further until all constraints are satisfied. It is poossible to end up with the same set of active constraints as at the start of the iteration. In this case the next iteration will produce the same design, which may be non-optimal.

Two alternatives are readily apparent which may solve this problem. The first one is that when a constraint becomes violated, repeat the step but include the newly violated constraint in the set of active constraints. The second alternative is to move to the point where the constraint is violated, and then iterate from there without stepping back. Of course, the violated constraint is added to the set of active constraints. Boston, et al. (1984) looked into this second alternative to some extent. They reported that it did not always work. However, it is not clear if it was the "no stepping back" that was the cause of the problems or if the second order approximations to the constraints contributed to the difficulty.

In summary, the Gauss nonlinearly constrained technique is very effective at solving quadratically constrained problems. No major difficulties appear to exist which would preclude it from solving problems with higher order constraints once the modifications discussed above are implemented. This method with the proposed modifications is currently the leading candidate as the best method for solving highly nonlinear mechanism design problems.

3. RESULTS

A verification of the effectiveness of the Gauss constrained method applied to problems with quadratic constraints is obtained by solving the Rosen-Suzuki test problem (Rqsen and Suzuki, 1965)

minimize $F(\vec{x}) = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4$

subject to :

$$g_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 + x_3 - x_4 - 8 \leq 0$$

$$g_2(\vec{x}) = x_1^2 + 2x_2^2 + x_3^2 + 2x_4^2 - x_1 - x_4 - 10 \leq 0$$

$$g_3(\vec{x}) = 2x_1^2 + x_2^2 + x_3^2 + 2x_1 - x_2 - x_4 - 5 \leq 0$$

The optimum design for this problem is at $\vec{x} = [0, 1, 2, -1]$.

Two versions of the Gauss nonlinearly constrained technique and the generalized reduced gradient method, identified as GRG, were used from four different starting points. One version of the Gauss nonlinearly constrained technique, identified as GNLC, uses the stepping back procedure, requires a feasible starting design and will always maintain a feasible design. The other version, identified an GNLC. NS, does not use the stepping back procedure and has no requirement no the feasibility of the design at any stage of the optimization. The results are summarized in Table 1. It can easily be seen that the Gauss nonlinearly constrained technique is much more efficient with respect to number of function evaluations than the generalized reduced gradient method.

Table 1 Comparision of algorithms

Algorithm	Starting design, $\vec{x}_0$	Number of iterations	Function evaluations
GNLC. NS	[0, 0, 0, 0]	2	3
GNLC. NS	[1, 1, 1, 1]	2	3
GNLC. NS	[2, 2, 2, 2]	2	3
GNLC. NS	[0, 0, $\sqrt{5}$, 0]	3	4
GNLC	[0, 0, 0, 0]	5	6
GNLC	[1, 1, 1, 1]	5	6
GNLC	[0, 0, $\sqrt{5}$, 0]	3	4
GRG	[0, 0, 0, 0]	11	106
GRG	[1, 1, 1, 1]	11	133
GRG	[2, 2, 2, 2]	11	144
GRG	[0, 0, $\sqrt{5}$, 0]	9	83

4. CONCLUSIONS

The optimization techniques developed in this research as extensions of the Gauss method to handle various types of constraints are effective approaches to reducing the number of analyses required to obtain an optimal design. As a result, the computational time for large problems should be reduced significalty.

ACKNOWLEDGEMENTS

This research was sponsored by the Air Force Office of Scientific Research, Air Force Systems Command, USAF, under Grant Numbers AFOSR-84-0076, and AFOSR-85-0103 and the National Science Foundation under Grant No. DNC-8500627 and INT-8616036. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notiwithstanding any copyright notation theron.

REFERENCES

Boston, D. R., Willmert, K. D. and Sathyamoorthy, M.,

1984, "Gauss Nonlinearly Constrained Optimization Method", Proceedings of the 5th ASCE Engineering Mechanics Division Speciality Conference, Laramie, WY. pp. 82~85.

Himmelblau, D. M., 1972, "Applied Nonlinear Programming", McGraw-Hill, New York.

Paradis, M.J. and Willmert, K. D., 1983, "Optimal Mechanism Design using the Gauss Constrained Method", Trans. of ASME, Journal of Mechanism, Transmissions and Automation in Design, Vol. 105, pp. 187~196.

Rosen, J.B. and Suzuki, S. 1965, "Construction of Nonlinear Programming Test Problems", Communications of the ACM, Vol. 8, p. 113.

Wilde, D.J., 1981, "Error Linearization in the Least-squares Design of Function Generating Mechanisms", Progress in Engineering Optimization, ASME, pp. 33~37.