NUMERICAL OPTIMIZATION

by

Garret N. Vanderplaats

VMA Engineering
1767 S. 8th Street, Suite M-200
Colorado Springs, CO

ABSTRACT

The purpose of this paper is to describe numerical optimization techniques for application to engineering design. General concepts of numerical optimization are simply outlined to provide a basic understanding of this technology. Following this, several issues relating to problem formulation are discussed to help the user solve design problems more efficiently and reliably. Particular emphasis is placed on tuning parameters in the DOT optimizer contained in the ADAMS program to help the user take full advantage of this recent addition to ADAMS. Finally, several design examples are offered to demonstrate the generality and efficiency of modern optimization methods. It will be seen that addition of optimization to ADAMS can dramatically broaden the applicability of this powerful engineering tool.

INTRODUCTION

Numerical optimization methods provide a uniquely general and versatile tool for design automation. Research has been extensive and today optimization is finding its way into engineering offices. The methods that form the basis of most modern optimization were developed roughly 30 years ago, and the first application to nonlinear structural design was in 1960 [1]. Much of the research in structural design in the past 15 years has been devoted to creating methods that are efficient for design problems where the analysis is expensive. This has resulted in various approximation methods that allow a high degree of efficiency while maintaining the essential features of the original problem. Research in other engineering disciplines has been less intensive, but is rapidly gaining in importance. Addition of optimization to the ADAMS [2] represents a major step in this expanded use of optimization.

Here, we will first define the general design task in terms of optimization. We will briefly discuss several common algorithms for solving this general problem and will identify their key features. Basic issues related to proper problem formulation will then be discussed. Several design examples are offered to demonstrate the power of optimization as a design tool. Because the optimization contained in the ADAMS program is new, the examples given here are necessarily from a variety of disciplines. Extension to the multitude of engineering tasks available in the ADAMS program will be apparent. References are offered for more thorough study.

BASIC OPTIMIZATION CONCEPTS

Mathematical programming (the formal name for numerical optimization) provides a very general framework for scarce resource allocation, and the basic algorithms originate in the operations research community. Engineering applications include chemical process design, aerodynamic optimization, nonlinear control system design, mechanical component design, multidiscipline system design, and a variety of others. Because the statement of the numerical optimization problem is so close to the traditional statement of engineering design problems, the design tasks to which it can be applied are inexhaustible.

In the most general sense, numerical optimization solves the nonlinear, constrained problem, Find the set of design variables, X_i, i=1,N contained in vector **X**, that will

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

Subject to;

$$g_j(\mathbf{X}) \le 0 \qquad j = 1,M \tag{2}$$

$$h_k(\mathbf{X}) = 0 \qquad k = 1,L \tag{3}$$

$$X_i^L \le X_i \le X_i^U \qquad i = 1,N \tag{4}$$

Equation 1 defines the objective function which depends on the values of the design variables, **X**. Equations 2 and 3 are inequality and equality constraints respectively, and equation 4 defines the region of search for the minimum. This provides limits on the individual design variables. The bounds defined by equation 4 are referred to as side constraints. A clear understanding of the generality of this formulation makes the breadth of problems that can be addressed apparent. However, there are some important limitations to the present technology. First, it is assumed that the objective and constraint functions be continuous and smooth (continuously differentiable). Experience has shown this to be a more theoretical than practical requirement and this restriction is routinely violated in engineering design. A second requirement is that the design variables contained in **X** be continuous. In other words, we are not free to chose structural sections from a table. Also, we cannot treat the number of plies in a composite panel as a design variable, instead treating this as a continuous variable and rounding the result to an integer value. It is not that methods do not exist for dealing with discrete values of the variables. It is just that available methods lack the needed efficiency for widespread application to real engineering design. Finally, even though there is no theoretical limit to the number of design variables contained in **X**, if we use optimization as a "black box" where we simply couple an analysis program to an optimization program, the number of design variables that can be considered is limited to the order of fifty. Again, there are many exceptions to this, but it is still a conservative general rule. Also, this is not too great a restriction when we recognize that using graphical methods would limit us to only a few design variables.

The general problem description given above is remarkably close to what we are accustomed to in engineering design. For example, assume we wish to determine the dimensions of a structural member that must satisfy a variety of design conditions. We normally wish to minimize weight, so the objective function, F(**X**), is just the weight of the structure, which is a function of the sizing and shape variables. However, we also must consider constraints on stresses, deflections, buckling and perhaps dynamic response limits. Assuming we model the structure as an assemblage of finite elements, we can calculate the stresses in the elements under each of the prescribed loading conditions. Then a typical stress limit may be stated as

$$\sigma^{l} \le \sigma_{ijk} \le \sigma^{u} \tag{5}$$

where i = element number, j = stress component, k = load condition. The compression and tensile stress limits are σ^{l} and σ^{u}, respectively (if we use a von Mises stress criterion, only σ^{u} would be used). While Equation 5 may initially appear to be different from the general optimization statement, it is easily converted to two equations of the form of Equation 2 as

$$g_1(X) = \frac{\sigma^{l} - \sigma_{ijk}}{|\sigma^{l}|} \le 0 \tag{6}$$

$$g_2(X) = \frac{\sigma_{ijk} - \sigma^{u}}{|\sigma^{u}|} \le 0 \tag{7}$$

Thus, the formal statement of the optimization task is essentially identical to the usual statement of the structural design task. The denominator of Equations 6 and 7 represents a normalization factor. This is important since it places each constraint in an equal basis. For example, if the value of a stress constraint is -0.1 and the value of a displacement constraint is -0.1, this indicates that each constraint is within 10% of it's allowable value. Without normalization, if a stress limit is 20,000, it would only be active (within 10%) if it's value was 19,999.9. This accuracy is probably impossible to achieve on a digital computer. Also, it is not meaningful since loads, material properties, and other physical parameters are not known to this accuracy.

It is often assumed that for optimization to be used, the functional relationships must be explicit. However, *this is categorically untrue.* It is only necessary to be able to evaluate the objective and constraint functions for proposed values of the design variables, **X**. Normally, this is done by a computer program, but we are not limited to this. For example, in [3], jet engine compressor vane settings are determined by passing information between the optimizer and an engine test rig, where the objective and constraint functions are evaluated experimentally. Using optimization, the efficiency was improved compared to the previous trail and error method. Furthermore, the number of experimental data points was reduced by 40% in a very costly study.

Using optimization as a design tool has several advantages; We can consider large numbers of variables relative to traditional methods. In a new design environment such as with composites, as well as many new mechanical applications, we do not have a great deal of experience to guide us and so optimization often gives unexpected results which can greatly enhance the final product.

One of the most powerful uses of optimization is to make early design trade-offs using simplified models. Here we can compare optimum designs instead of just comparing point designs. Furthermore we can obtain the optimal sensitivity with respect to design parameters and use this to guide the decision making process.

On the other hand, optimization has some disadvantages to be aware of. First, the quality of the result is only as good as the underlying analysis. Thus, if we ignore or forget an important constraint, optimization will take advantage of that, leading to a meaningless if not dangerous design. Second, there is a danger that by optimizing we will reduce the hidden factors of safety that now exist. In this context, we should re-think our use of optimization, using it as only a design tool and not as a sole means to an end product.

However, assuming we agree that optimization is useful, it is also important to know how these algorithms solve our design problems. In the following sections we briefly outline several common algorithms to provide some insight into the numerical techniques used. For brevity as well as clarity, we will limit our discussion to a few basic concepts.

The Optimization Process

Most optimization algorithms do just what good designers do. They seek to find a perturbation to an existing design which will lead to an improvement. Thus, we seek a new design which is the old design plus a change so

$$\mathbf{X}^{new} = \mathbf{X}^{old} + \delta\mathbf{X} \tag{8}$$

Optimization algorithms use much the same formula, except it becomes a two step process. Here we update the design by the relationship

$$\mathbf{X}^{q} = \mathbf{X}^{q-1} + \alpha\mathbf{S}^{q} \tag{9}$$

where $\alpha\mathbf{S}^q$ in Equation 9 is equivalent to $\delta\mathbf{X}$ in Equation 8. Here q is the iteration, or design cycle, number. The engineer must provide an initial design, $\mathbf{X}^0$ but it need not be feasible (it may not satisfy the inequalities of Equation 2 or equalities of Equation 3). Optimization will then determine a "Search Direction," $\mathbf{S}^q$ that will improve the design. If the design is initially feasible, the search direction will reduce the objective function without violating the constraints. If the initial design is infeasible (some $g_j(\mathbf{X}) > 0$), the search direction will point toward the feasible region, even at the expense of increasing the objective function.

The next question is how far can we move in direction $\mathbf{S}^q$ before we must find a new search direction. This is called the "One-Dimensional Search" since we are just seeking the value of the scalar parameter, α, to improve the design as much as possible. If the design is feasible and we are reducing the objective function, we seek the value of α that will reduce $F(\mathbf{X})$ as much as possible without driving any $g_j(\mathbf{X})$ positive or violating any bound on the components of $\mathbf{X}$. If the design is initially infeasible, we seek the value of α that will overcome the constraint violations if possible, or will otherwise drive the design as near to the feasible region as possible. Note that this is precisely what a design engineer does under the same conditions. The difference is that optimization does it without the need to study many pages of computer output.

There are a wide variety of algorithms for determining the search direction, **S**, as well as for finding the value of alpha [4]. Determining α is conceptually a simple task. For example, we may pick several values of α and calculate the objective and constraint functions. Then we fit a polynomial curve to each function and determine the value that will minimize F(**X**) or drive some $g_j(\mathbf{X})$ to zero. Since we picked a search direction that will improve the design, we need only consider positive values of α. The minimum positive value of α from among all of these curve fits is the one we want. Other methods for finding α are called by such names as Golden Section, Bisection, and Fibonacci search, as examples.

Unconstrained Minimization

Unconstrained minimization problems are defined by Equation 1 only, where Equations 2 through 4 are omitted. Most engineering problems are not of this form, and so we will only offer some basic concepts here. The motivation for considering unconstrained minimization methods is that they provide a basis for the constrained problem, and that they can sometimes be used indirectly to solve constrained problems. Also, some engineering analysis problems can be posed as unconstrained minimization tasks, an example being to have a linkage follow a defined path. Finally, unconstrained minimization is familiar to us from calculus, where the minimum or maximum of a function is known to be a point where the gradient of the function vanishes.

Perhaps the oldest, best known, and worst unconstrained minimization algorithm is known as the Steepest Descent Method. Here, the search direction, **S**, is calculated as the negative of the gradient of the objective function;

$$\mathbf{S}^q = -\nabla F\left(\mathbf{X}^{q-1}\right) \tag{10}$$

where

$$\nabla F\left(\mathbf{X}^{q-1}\right) = \begin{Bmatrix} \dfrac{\partial F\left(\mathbf{X}^{q-1}\right)}{\partial X_1} \\ \dfrac{\partial F\left(\mathbf{X}^{q-1}\right)}{\partial X_2} \\ \cdots\cdots \\ \cdots\cdots \\ \dfrac{\partial F\left(\mathbf{X}^{q-1}\right)}{\partial X_N} \end{Bmatrix} \tag{11}$$

This search direction is now used in Equation 9 to update the design. It is surprising how often this method is used today and presented as an "advanced" algorithm. However, it must be stated that, this method is one of the worst available and should never be used. The steepest descent method will seldom converge reliably to a solution in even the simplest of nonlinear minimization tasks. Furthermore, it is almost trivial to modify this method to make it efficient, although this simple modification is still not considered to be the best algorithm available.

The conjugate gradient method [5] represents a simple modification to the steepest descent method, but provides dramatic improvements in optimization efficiency. Here, we still use the steepest descent direction on the first iteration (q = 1). On subsequent iterations, we use a conjugate direction defined by

$$S^q = -\nabla F\left(X^{q-1}\right) + \beta S^{q-1} \tag{12}$$

where

$$\beta = \frac{\left|\nabla F(X)^{q-1}\right|}{\left|\nabla F(X)^{q-2}\right|} \tag{13}$$

Equations 12 and 13 have a simple physical interpretation. If we were making progress on the last iteration, it makes sense to move partly in that direction (as defined by β), while including the gradient information at the present design point. Other, more powerful methods, are known as Variable Metric Methods and go by such names as DFP and BFGS methods. Each such method attempts to create second order information using the first order (gradient) information, and can be shown to converge in N or fewer iterations for a quadratic problem. On the other hand, the Steepest Descent Method cannot be guaranteed to converge under any circumstances.

Constrained Minimization

The vast majority of engineering design tasks are constrained problems of the form of Equations 1-4. Just as for unconstrained problems, there are perhaps as many methods available as there are researchers in the field. Here, we will discuss four methods which provide a basic understanding of the concepts. The first method converts the constrained problem to a sequence of unconstrained problems. The second two methods have been shown to be powerful tools for engineering design, even though each method is considered to be "poor" by theoretical standards. The last method has been developed more recently than the others and is considered to be a theoretically "good" method, but is somewhat sensitive to problem parameters.

First, we consider the simplest case of what are referred to as Sequential Unconstrained Minimization Techniques or SUMT. The basic concept is to convert the original problem to an unconstrained problem that can be solved by unconstrained minimization methods. The most direct way to do this is to create a "penalty" function that will increase the objective for any constraint violation. Thus, we define a Pseudo-objective function of the form

$$F'(X) = F(X) + R\left\{\sum_{j=1}^{N} \mathrm{Max}[0, g_j(X)]^2 + \sum_{k=1}^{L} \left[h_k(X)\right]^2\right\} \tag{14}$$

The parameter, R, is referred to as a penalty parameter. Initially, R is taken as a relatively small number and F(X) is minimized as an unconstrained function. Then R is increased, typically by a factor of 10 and the process is repeated. The process is terminated when no improvement in F(X) is found and all constraints are satisfied within a small tolerance. The reason that R cannot just be set to a very large value from the start is that this would create a high degree of nonlinearity, making it impossible to solve the unconstrained problem. This method is called the Exterior Penalty

Function method since it penalizes the objective only if a constraint is violated. This method has the advantages of simplicity and reliability, but the disadvantages that it approaches the optimum from the infeasible region and requires more function evaluations than competing methods. Another implicit advantage is that by coming from the infeasible region, there is an improved likelihood of finding the best optimum for cases where relative minima exist. A variety of other such methods are available and some of them (referred to as interior penalty function methods) approach the optimum from inside the feasible region [6,7]. Also, the Augmented Lagrange Multiplier Method [8] is considered to be a more modern sequential unconstrained minimization method.

The second method we consider is the Method of Feasible Directions [9]. This is referred to as a direct method since if deals with the constraints directly in the optimization process. The basic algorithm is for inequality constraints, although equality constraints may be included in a variety of ways if necessary. Assuming we begin in the feasible design region (there are no active or violated constraints (that is all $g_j(X) < 0$), we begin with a steepest descent search direction. If it takes more than one iteration to encounter a constraint, we use a conjugate direction or variable metric method on subsequent iterations, until a constraint is encountered. Once having encountered a constraint boundary, we must find a "Usable-Feasible" direction, where a usable direction is one that reduces the objective and a feasible direction is one that either follows or moves inside of a constraint boundary. This requires solving the following sub-problem;

Find the components of S^q and β that will

$$\text{Minimize } \nabla F\left(X^{q-1}\right) \bullet S^q \le 0 \tag{15}$$

Subject to;

$$\nabla g_j\left(X^{q-1}\right) \bullet S^q \le 0 \qquad j \in J \tag{16}$$

$$S^q \bullet S^q \le 1 \tag{17}$$

where J is the set of active constraints (all $g_j\left(X^{q-1}\right) \ge \varepsilon$, where ε is a small negative number, say -0.03). Equation 15 attempts to reduce the objective function as much as possible, while Equation 16 prevents the search direction from pointing into the infeasible region. The purpose of Equation 17 is to prevent an unbounded solution to this sub-problem. Methods for solving this problem are beyond the scope of this discussion. However, the resulting search direction will reduce the objective function without violating constraints. Similar methods will actually move away from the presently active constraints. This method, which "follows" constraints requires modifications to the one-dimensional search to bring the design back to the constraint boundaries. This is only one of a class of algorithms based on the feasible directions concept, but serves to define the general method.

The third method we consider is the Sequential Linear Programming (SLP) method. Sequential Linear Programming, also known as Kelley's Cutting Plane Method [10], was developed in the early 1960's. This method is considered unattractive by theoreticians, but has proven to be quite powerful and efficient for engineering design. The basic concept is that we first linearize the objective and constraint functions and then solve this linearized problem by the optimizer of our choice (the obvious choice is the standard linear programming method, but we need not restrict ourselves to this. The Method of Feasible Directions will work very well). Thus, we create the following approximation;

$$\tilde{F}(\delta X) = F\left(X^{q-1}\right) + \nabla F\left(X^{q-1}\right) \bullet \delta X \tag{18}$$

$$\tilde{g}_j(\delta X) = g_j\left(X^{q-1}\right) + \nabla g_j\left(X^{q-1}\right) \bullet \delta X \tag{19}$$

where

$$\delta X = X^q - X^{q-1} \tag{20}$$

We now solve the approximate optimization problem;

$$\text{Minimize } \tilde{F}(\delta X) \tag{21}$$

Subject to;

$$\tilde{g}_j(\delta X) \le 0 \qquad j \in J \tag{22}$$

$$\delta X_i^L \le \delta X_i \le \delta X_i^U \qquad i = 1, N \tag{23}$$

Here, the set of active constraints, J, includes all potentially active constraints (say all $g_j\left(X^{q-1}\right) \ge -0.3$). It is not necessary to approximate all constraints, which may number in the thousands. Usually we can limit the set J to about 3N.

The move limits, δX_i^L and δX_i^U are needed to prevent unlimited moves as well as limiting the design changes to the region of applicability of the approximation. Once the approximate optimum is found, the problem is re-linearized and the process is repeated until it converges to the optimum. During the process, the move limits are adjusted to insure convergence.

Finally, we consider a relatively recent method called Sequential Quadratic Programming [11,12]. While this is a relatively complicated method, it has been found to be quite powerful if reasonable care is taken in formulating the optimization problem. The basic concept is to find a search direction, S, that will minimize a quadratic approximation to the "Lagrangian" function subject to linear approximations to the constraints. That is, Find the components of S that will

$$\text{Minimize } Q(S) = F\left(X^q\right) + \nabla F\left(X^{q-1}\right) \bullet S + \frac{1}{2} S^T [B] S \tag{24}$$

Subject to;

$$g_j\left(X^q\right) + \nabla g_j\left(X^{q-1}\right) \bullet S \le 0 \qquad j = 1, M \tag{25}$$

This is a quadratic programming problem which is solved for the components of **S**. A variety of modifications are available to insure this sub-problem has a feasible and bounded solution. The matrix, **[B]**, is initially the identity matrix, but as the optimization proceeds, **[B]** is updated using gradient information to approximate the Hessian of the Lagrangian function.

Once the search direction is found, a one-dimensional search is performed using an exterior penalty formulation which includes the Lagrange multipliers from the process of finding **S**.

As noted above, this is a relatively complicated algorithm and has been found by experience to be somewhat sensitive to the care with which the overall optimization problem is formulated as well as being sensitive to several internal parameters. However, for those classes of problems where it works, it has been found to be particularly powerful.

In considering the variety of algorithms available, there are no reliable rules to determine which method is best. However, experience shows that the most important thing is to use an algorithm that provides acceptable results on the average problem of interest. The more complicated algorithms are usually considered best by the theoreticians, but also are found to be less reliable for problems that are not carefully formulated. On the other hand, algorithms like the feasible directions method the sequential linear programming are considered "poor" by the theoreticians, but usually perform reliably in a practical design environment.

Using Optimization

Assuming we wish to use optimization in the design environment, the issue now becomes one of implementation. To begin with, it is not necessary for every design group to create its own optimization program. Just as with finite element analysis or mechanical modeling, there is a growing number of optimization software products becoming commercially available. The real issue is how to get these codes used in an already overworked design department. This is primarily a management and training issue. This integration of optimization into the design process is now being greatly facilitated by software vendors who are incorporating optimization directly into their products.

As experience is gained using such integrated programs as ADAMS, users will quickly realize that optimization can be applied to a wide variety of design tasks. In some cases, it will be possible to include other disciplines with ADAMS to perform multidiscipline optimization with this versatile tool. In other cases, the user may wish to couple specialty codes with an optimizer. The key is a very minor amount of standardization.

By writing engineering analysis software in a consistent format, it can be easily coupled with an optimizer and optimization can be done with very little effort beyond that needed for a single analysis. Now, by proposing standardization we suggest neither an additional level of bureaucracy nor a removal of software coding from the engineering office. On the contrary, engineers should be allowed and encouraged to code their own special purpose analysis software. Of course, they shouldn't code what they can buy any more than they should design a bolt that is available from stock. Also, they should not be burdened with a long list of rules on structured programming, documentation and the like.

In order to write software that can be readily coupled to optimization, only three simple rules are needed;

1. Write the analysis as a subroutine with a very simple main calling program. This subroutine can call any number of additional subroutines.
2. Break the analysis into Input, Execution and Output. The optimizer will require that execution be performed repeatedly for different combinations of the design variables.
3. Store every parameter that may be a design variable, objective or constraint function in a single labeled common block or disk file.

Following these rules is good programming practice and will not be offensive except to the most individualistic of engineers/programmers. Yet by following these rules, over 90% of all engineering analysis programs where optimization may be useful can be coupled to an optimizer in a matter of minutes. The main program will be replaced by an optimization control program such as DOC [13] and the input data will define the optimization problem. Furthermore, once the combined program is working, the choice of design variables, objective and constraint functions, and constraint bounds can be changed by only changing a few lines of data. Finally, the issue of optimization is only marginally considered in writing the analysis code. It is only the data to the combined program that makes it a design optimization task.

The motivation for general purpose optimization programs such as DOC is to make it *so easy* to try optimization that there is little excuse not to. More than that, once the engineer/programmer is accustomed to writing design oriented codes instead of standard analysis codes, a library of design programs evolves which can be repeatedly used as the design process goes forward or as new design problems are addressed.

PROBLEM FORMULATION

While optimization technology is becoming reasonably mature, it is not realistic to expect that we can solve any nonlinear constrained problem as will. Optimization is, in many ways, similar to solving nonlinear problems in analysis. It is well understood that, if those problems are not well conditioned, the analysis will not converge to a solution or may converge to the wrong solution. The same is true for nonlinear optimization. However, there are some things we can do to improve our success. Here, we will discuss several items which will help with optimization. This discussion is strongly related to using the DOT optimizer [14] which is used in the ADAMS program.

Basic Assumptions

As stated previously, there are basic theoretical assumptions that underlie the algorithms used in the DOT optimizer. The most important are that the objective and constraints are continuous functions of the design variables, and that they have continuous first derivatives. These assumptions are often violated in practice, but if discontinuities exist near the optimum, the results may be unreliable. In a more practical sense, this suggests we should not try to minimize the absolute value of a function, because this has a discontinuous derivative at zero.

Another assumption is that the problem is convex. For example, a two variable function will be bowl shaped. The mathematics behind this assumption is that such problems are guaranteed to have only one minimum, and not relative minima. It is seldom possible to prove that engineering problems are convex. Thus, it is often useful to begin the optimization from several different starting points to find the best optimum. There are algorithms which improve the probability of finding the global optimum in such cases, but these method are usually less efficient than just re-starting from several points. In fact, a set of designs obtained by using the Design of Experiments capability contained in ADAMS will give a good statistical set.

Often, when the optimization is started from several different points, somewhat different designs are found. This does not prove that relative minima exist. It often only means that the design space is quite "flat." This is especially true if the optimum objective function has about the same value in each case, but the vector of design variables is different. The best way to test if these designs are actually relative minima is to take the arithematic average of two or more of the "optimum" **X** vectors and analyze the resulting design. If the objective is about the same as before and all constraints are satisfied, it is likely that it is just a flat design space and not relative minima. From an engineering point of view, this is actually good, because it means the design is not too sensitive to manufacturing tolerances.

TUNING THE OPTIMIZER

Bad Gradients

To perform the optimization process, gradients of the objective and constraint functions are needed. Presently, ADAMS calculates these gradients by finite difference. Because many analyses in ADAMS are iterative, convergence is assumed when the result is calculated within a specified tolerance, say $\pm\varepsilon$. As shown in Figure 1, this provides a band about the true function, where the result is said to have converged. Now when the design variable, X, is changed, a new result is found within this band. If we take a very small finite difference step, we can get large differences in the estimated gradient, as shown. However, if we take a relative large step, we still get a less than perfect, but improved gradient as shown. If we are solving an unconstrained problem, where we seek a zero gradient, this may still not be good enough to find a precise optimum. However, recognizing that most of our problems are constrained, the actual gradient is less important, especially if the optimum is at a vertex (as many active constraints as there are design variables). In extreme cases, it may be necessary to use central difference gradients (probably available in the next release). This will provide a higher quality gradient, but at the expense of many more analyses.

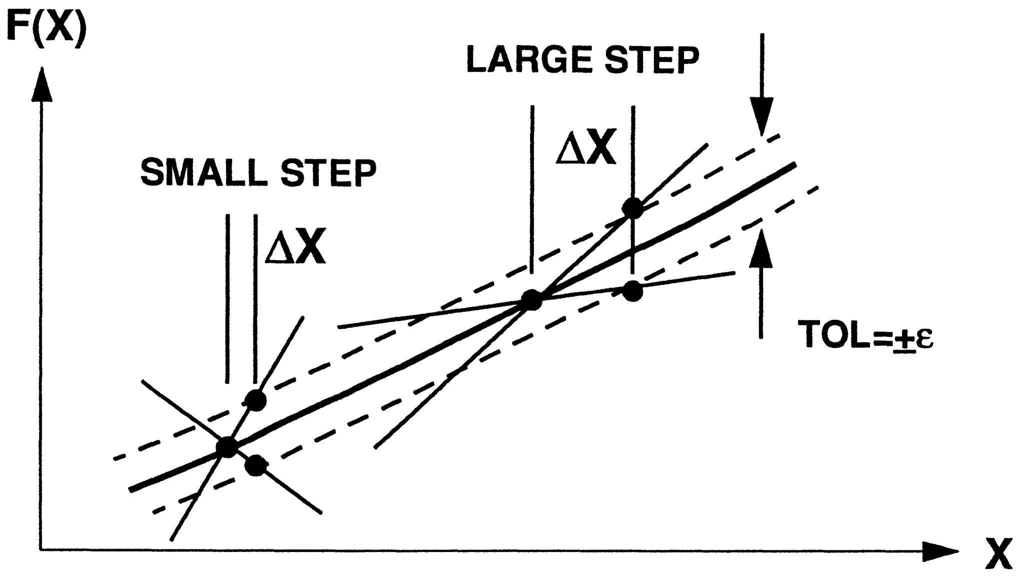


Figure 1. Effect of Finite Difference Step on Gradients

In the DOT optimizer, the finite difference step is controlled by the parameters FDCH and FDCHM, where FDCH is the fractional step and FDCHM is the minimum absolute step taken when X_i is very small.

Premature Convergence

Another situation that sometimes happens is that the optimization seems to have converged when you sense that more progress could be made. You can determine the situation which dictated convergence by looking at the end of the DOT output (parameter IPRINT $\geq$ 1). If the design is feasible and it has converged because the "Kuhn-Tucker" conditions are satisfied, this is probably the best solution you can achieve. If convergence is because the objective does not change by the amount DELFUN or DABFUN for # iterations, this indicates that DOT has reached a point of diminishing returns. If the initial objective is very large and great improvement has been made, the absolute convergence criteria, DABFUN will often control. You may wish to restart from this design with DABFUN set very small, say 0.0001. If the relative change controls, and the optimization is not too expensive, you may set ITRM to say 5 to require that the termination criteria be satisfied five times to "prove" convergence. In either case, you should restart from the best known optimum to avoid excessive cost.

No Progress

A third situation of interest is that the optimization seems to make no progress. If the design is feasible, it is probably optimum. If it is infeasible, it is possible that no feasible solution exists. In either case, set IPRINT=5 to print the gradients that DOT calculates. If the design is infeasible, and if two or more constraints have gradients that are opposite in sign, term by term, no feasible solution exists. If several constraints are violated, this is a bit harder to detect since the combination of gradients will dictate that no search direction can be found to overcome the violations. If the design is feasible and the gradients are quite small, it may help to use a larger finite difference step.

Equality Constraints

While DOT will consider equality constraints by using two equal and opposite inequality constraints, nonlinear optimization generally has difficulty with equality constraints. This is because we are trying to follow a curve, in N-dimensional space, that may be very nonlinear. Say you are seeking a design where a part should have a fundamental frequency of 1 Hz. If this is a true requirement, an equality constraint is appropriate. However, if you only require that the frequency be at least 1 Hz, then use an inequality constraint. You may be pleased to find that, at the optimum, the fundamental frequency is 5 Hz.

Make the Design Variables Independent

Often, polynomials are used to define a shape we want to design or a path we want to follow. While it may be adequate to define the shape by, say, a 5th order polynomial, it is better to use a piecewise quadratic fit. In the 5th order fit, if you change one coefficient, all others may need to change to meet the boundary conditions. However, by using piecwise polynomials, the control points are much less dependent and the optimization is better conditioned. The resulting optimum should be the same. Remember that the objective function and each constraint must be a function of at least one design variable, but need not be a function of all variables. For example, minimization or maximization of one of the design variables is perfectly valid.

Use Intermediate Variables

Consider the following simple problem;

Minimize $F(A) = 3*A1 + 5*A2$

Subject to;

$$G(A) = 2/A1 + 4/A2^2 - 5 \leq 0$$

If A1 and A2 are the design variables, this problem has a linear objective function with a nonlinear constraint. Now consider a simple change in variables so $X1 = 1/A1$ and $X2 = 1/A2^2$. The problem now becomes;

Minimize $F(X) = 3/X1 + 5/SQRT(X2)$

Subject to;

$$G(A) = 2*X1 + 4*A2 - 5 \leq 0$$

Now the objective function is nonlinear, but the constraint is linear. This is a much better conditioned optimization task. The original variables, A1 and A2 are easily recovered. While such a simple change in variables is not always possible, some effort at choosing "good" design variables is worthwhile.

Similarly consider the constraint;

$$X5/X6 - 20 \leq 0$$

This is clearly a nonlinear function. However, we can make it linear with by simply multiplying by X6 to give;

$X5 - 20*X6 \leq 0$

Finally, we should always normalize constraints. In this example, if X5 is expected to be of the order 100, a "properly normalized" constraint could be;

$X5/100 - X6/5 \leq 0$

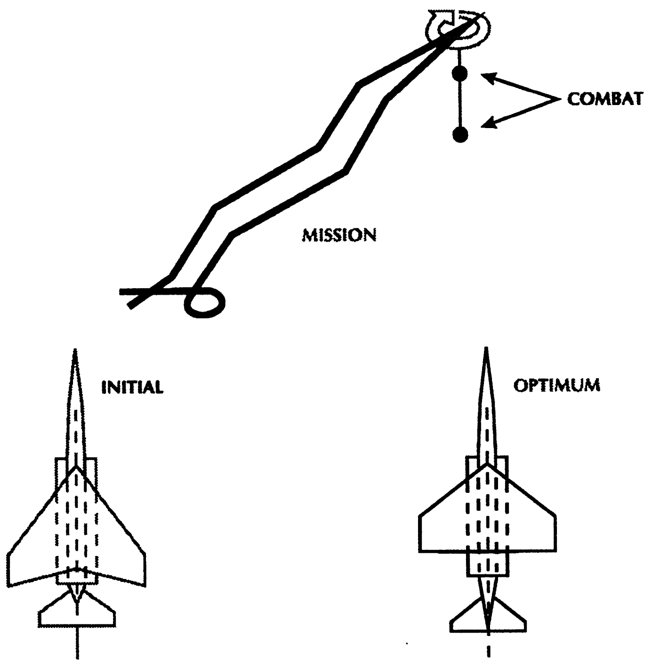
Actually, X5 may end up having a value of perhaps 10 or 1000. However, an order of magnitude difference of 10 in choosing the normalization factor is usually not significant, while a factor of 1000 can have a very detrimental effect on the efficiency and reliability of the optimization.

EXAMPLES

Here several examples are offered to show the breadth of design tasks that amenable to optimization. These are primarily taken from the author's experience, but the extension to problems solved by ADAMS should be apparent.

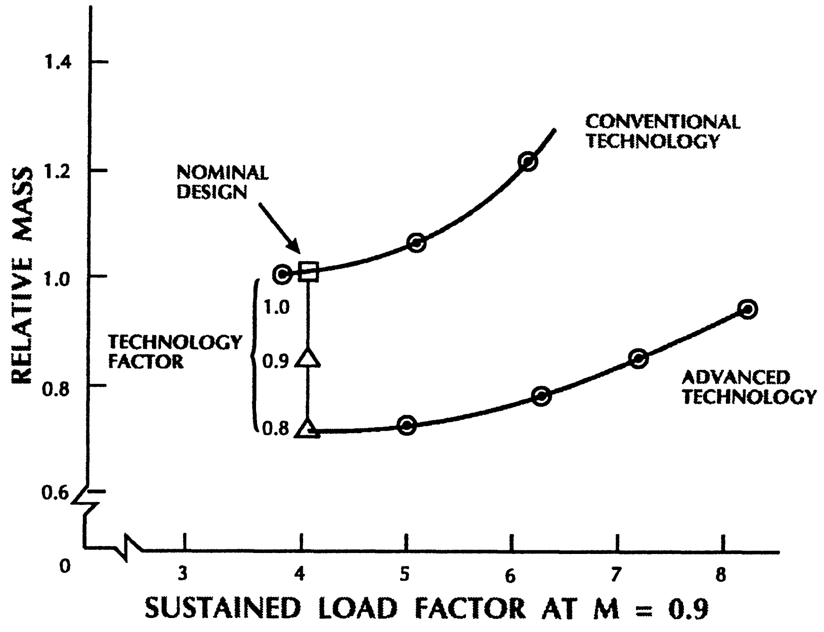
Conceptual Aircraft Design

Figure 2 shows the mission and planforms of a supersonic cruise aircraft, where the conceptual design was performed by optimization [15]. Figure 3 shows a trade-off study performed on the same design. The notable features of this study are that it was performed nearly twenty years ago and required less than two weeks for two engineers. When the effects of technology were studied, as shown in Figure 3, each design was optimized, requiring nine separate optimizations. In other words, this was an optimum trade-off study!



SOLVED BY THE ACSYNT PROGRAM
5 DESIGN VARIABLES, 2 PERFORMANCE CONSTRAINTS

Figure 2: Conceptual Aircraft Design



TRADE - OFF STUDY

Figure 3: Optimum Design Trade-Offs

Airfoil Optimization

Figure 4 shows the initial shape and the optimum shapes for an airfoil designed for two separate conditions [16]. This optimization was performed by the simple coupling of the optimization with an inviscid aerodynamics code. The optimum design was created using a combination of four existing airfoils and each optimization required under 50 analyses. Note that the optimum shape is quite different, depending on the design requirements. This points out the importance of considering the complete flight envelop when using optimization for such problems. If an airfoil is optimized for a single flight condition, it will be far from optimum under other conditions. However, multiple analyses can easily be performed as part of the optimization, leading to a balanced design.

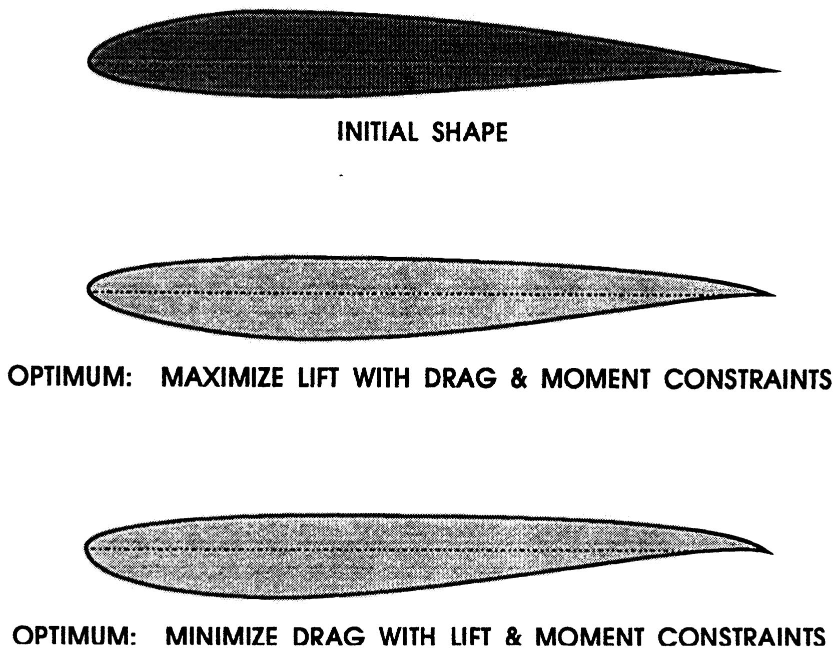

INITIAL SHAPE
OPTIMUM: MAXIMIZE LIFT WITH DRAG & MOMENT CONSTRAINTS
OPTIMUM: MINIMIZE DRAG WITH LIFT & MOMENT CONSTRAINTS

Figure 4: Airfoil Optimization

Structural Optimization

Figure 5 shows a very simple planar truss, where the member dimensions as well as the shape were design variables and constraints were imposed on stresses, member buckling and joint displacements. This structure included 44 design variables and so called "advanced approximation techniques" were used [17]. The optimum was achieved using only 9 structural analyses. This shows the efficiency that can be achieved by careful creation of advanced methods for a particular class of problems.

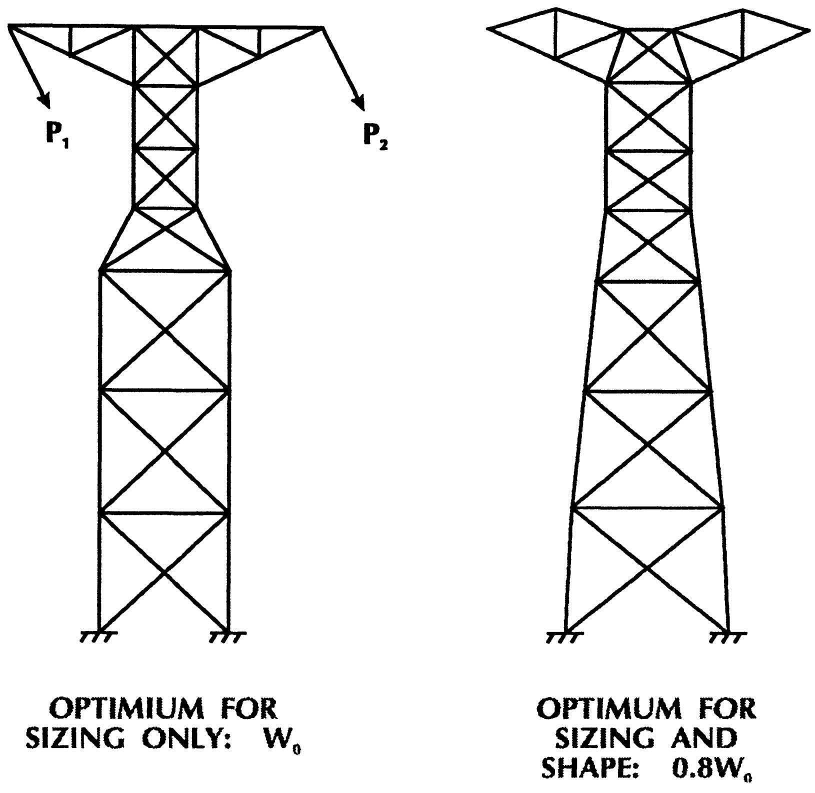

P_1
P_2
OPTIMIUM FOR SIZING ONLY: W_0
OPTIMUM FOR SIZING AND SHAPE: $0.8W_0$

Figure 5: Planar Tower Optimization

Robot Arm

Finally consider the simple robot arm shown in Figure 6. This example is contained in the ADAMS Optimization Guide. The objective is to minimize the total effort, measured as the time integral of the joint torques. The design variables are the proportional and derivative gain factors on the joint motions. Basically, this minimizes the energy required, but may produce torques that are greater than produced by available motors. Note that it is almost trivial to constrain the maximum torque to be below a specified bound. Finally, looking to the future, it is possible to simultaneously design the controls and the actual structural members by coupling ADAMS to a structural optimization program such as GENESIS [17]. ADAMS users are encouraged to advise Mechanical Dynamics as well as this author of desirable future software which can integrate other disciplines with the ADAMS program.

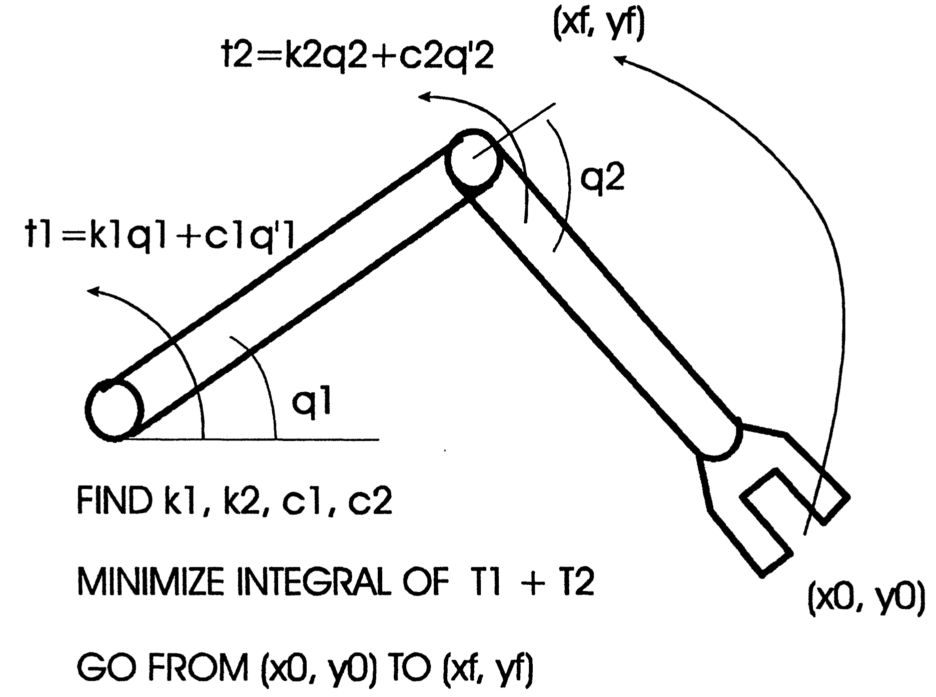


Figure 6: Robot Arm Optimization

SUMMARY

The purpose here has been to present the basic concepts of numerical optimization methods. This technology has been around for a long time but is only now being widely recognized as a valid and efficient design tool. The development of these methods has matured to the point that they are relatively easy to use in modern engineering design.

Key to the increased use of optimization in design is the addition of this capability to existing or new commercial analysis software. In the past ten years, optimization has been added to most major structural analysis programs. Some use very simple "black box" coupling of the optimizer with the analysis, while others use very sophisticated "approximation techniques." Addition of optimization to the ADAMS mechanical dynamics software represents a pioneering step in expanding the use of optimization to this more general field. Similarly, optimization is being added to CFD and other commercial software. In the future, it can be expected that various disciplines will be combined to create a true multidisciplinary optimization capability, and indeed research in the formal integration of multiple disciplines has been intense now for many years.

Optimization by itself doesn't save design time or money. There are fundamental natural laws that state that we will use all of the time and funds available for design. What it does is provide us with a tool to reach a high quality design much faster, allowing us to investigate a wide variety of alternatives. The final result is not a product that costs less in terms of design time and money, but a product that is superior. This is what design is about. What is important is that it is better than what the competition can produce in the same time frame. This is an argument for the use of formal optimization methods that, if true, is compelling. There is now sufficient evidence to indicate that it is indeed true and so it can be safely stated that optimization is a design tool whose time has come!

REFERENCES

1. Schmit, L. A., "Structural Design by Systematic Synthesis," Proc. 2nd Conference on Electronic Computation, American Society of Civil Engineers, New York, 1960, pp. 1249-1263.
2. ADAMS Optimization Guide, Mechanical Dynamics, Ann Arbor, MI, November 1994.
3. Garberoglio, J. E., Song, J. O. and Boudreaux, W. L., "Optimization of Compressor Vane and Bleed Settings," ASME Paper No. 82-GT-81, Proc. 27th International Gas Turbine Conference and Exhibit, London, April 18-22, 1982.
4. Vanderplaats, G. N., Numerical Optimization Techniques for Engineering Design; with Applications, McGraw-Hill, 1984.
5. Fletcher, R. and Reeves, C. M., "Function Minimization by Conjugate Gradients," British Computer Journal, Vol. 7, No. 2, pp. 149-154, 1964.
6. Fiacco, A. V. and McCormick, G. P., Nonlinear Programming: Sequential Unconstrained Minimization Techniques, John Wiley and Sons, New York, 1968.
7. Cassis, J. H. and Schmit, L. A., "On Implementation of the Extended Interior Penalty Function," Int. J. Num. Methods for Engineering, Vol. 10, No. 1, 1976, pp. 3-23.
8. Pierre, D. A. and Lowe, M. J., Mathematical Programming via Augmented Lagrangians, Applied Mathematics and Computation Series, Addison-Wesley, Reading, Mass., 1975.
9. Vanderplaats, G. N., "An Efficient Feasible Directions Algorithm for Design Synthesis," AIAA Journal, Vol. 22, No. 11, Nov. 1984.
10. Kelley, J. E., "The Cutting Plane Method for Solving Convex Programs," J. SIAM, pp. 703-713, 1960.
11. Powell, M. J. D., "Algorithms for Nonlinear Constraints that Use Lagrangian Functions," Math. Prog., Vol. 14, No. 2, 1978, pp. 224-248.
12. Vanderplaats, G. N., and Sugimoto, H. "Application of Variable Metric Methods to Structural Synthesis," Engineering Computations, Vol. 2, No. 2, June 1985.
13. DOC - Design Optimization Control User's Manual, VMA Engineering, Colorado Springs, CO, 1994
14. DOT - Design Optimization Tools User's Manual, VMA Engineering, Colorado Springs, CO, 1994
15. Vanderplaats, G. N. and Gregory, T., "A Preliminary Assessment of the Effects of Advanced Technology on Supersonic Cruise Tactical Aircraft," Proc. Super Cruise Military Aircraft Design Conference, Colorado Springs, CO, Feb. 17-20, 1976.

16. Vanderplaats, G. N., "An Efficient Algorithm for Numerical Airfoil Optimization," AIAA J. Aircraft, Vol. 16, No. 12, Dec. 1979.

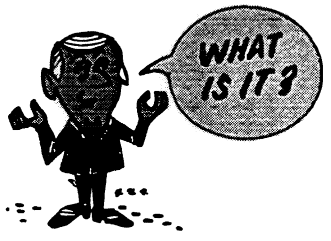
17. GENESIS Structural Optimization Program User's Manual, VMA Engineering, Colorado Springs, CO, 1995.

NUMERICAL OPTIMIZATION

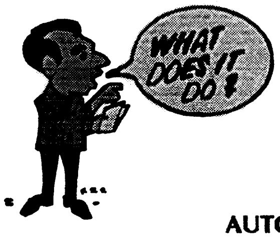
G. VANDERPLAATS

VMA ENGINEERING

COLORADO SPRINGS, CO

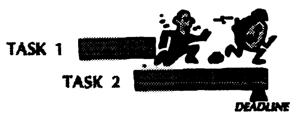


A VERY GENERAL AUTOMATED DESIGN TECHNIQUE



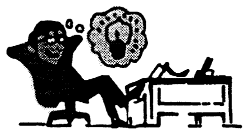
AUTOMATICALLY CHANGES IMPORTANT PARAMETERS TO FIND THE "BEST" DESIGN SATISFYING SPECIFIED CRITERIA

REDUCE DESIGN TIME

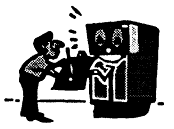


IMPROVE DESIGN QUALITY

FREE THE ENGINEER FOR CREATIVE WORK

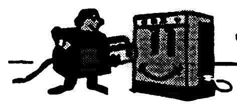


WRITE SOFTWARE IN STANDARD FORM



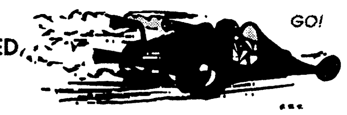
ORGANIZATION

COUPLE WITH OPTIMIZER



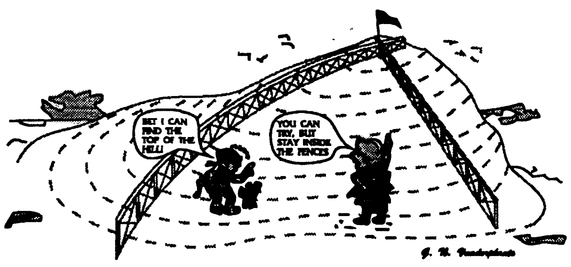
OPTIMIZATION

EXECUTE COMBINED PROGRAM



GO!

THE PHYSICAL PROBLEM


BET I CAN FIND THE TOP OF THE HILL!
YOU CAN TRY, BUT STAY INSIDE THE FENCES

THE ENGINEERING PROBLEM

FENCE NO. 1	HILL	FENCE NO. 2
$F_1 = f_1(X_1, X_2)$	$Y = f(X_1, X_2)$	$F_2 = f_2(X_1, X_2)$
< 0 INSIDE > 0 OUTSIDE		< 0 INSIDE > 0 OUTSIDE

THE OPTIMIZATION PROBLEM

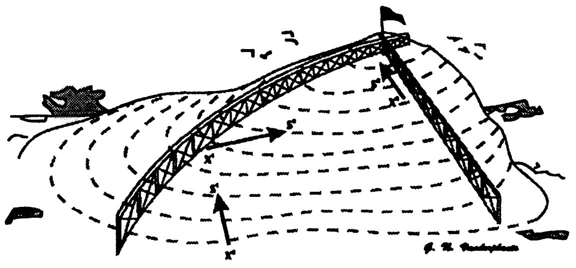
MAXIMIZE $Y = f(X_1, X_2)$ — OBJECTIVE

SUBJECT TO:

$$\left.\begin{aligned} F_1 &= f_1(X_1, X_2) \le 0 \\ F_2 &= f_2(X_1, X_2) \le 0 \end{aligned}\right\}$$ CONSTRAINTS

$$X = \begin{vmatrix} X_1 \\ X_2 \end{vmatrix}$$ DESIGN VARIABLES

THE OPTIMIZATION PROCESS



- BASIC ASSUMPTIONS
  - > FUNCTIONS ARE CONTINUOUS
  - > FUNCTIONS HAVE CONTINUOUS FIRST DERIVATIVES
- THESE ARE "THEORETICAL" REQUIREMENTS
  - > THEY ARE OFTEN VIOLATED IN PRACTICE
  - > THEY USUALLY CREATE DIFFICULTY ONLY IF DISCONTINUITIES EXIST NEAR THE OPTIMUM

- DESIRABLE CHARACTERISTICS
  - > ALL FUNCTIONS ARE "CONVEX"
    - » SHAPED LIKE A BOWL
- CONVEX PROBLEMS HAVE ONLY ONE MINIMUM
  - > THERE ARE NO RELATIVE MINIMUMS
  - > SELDOM TRUE IN ENGINEERING
- HOW TO FIND THE "GLOBAL" OPTIMUM
  - > START OPTIMIZATION FROM SEVERAL INITIAL DESIGNS
  - > NO "GUARANTEE" THAT TRUE OPTIMUM IS FOUND

PROBLEM FORMULATION

- **I START THE OPTIMIZATION FROM SEVERAL DIFFERENT DESIGNS AND GET DIFFERENT RESULTS. WHY?**
  - > **IS THE OBJECTIVE FUNCTION ABOUT THE SAME FOR ALL?**
    - » **YES - IT'S PROBABLY JUST A "FLAT" DESIGN SPACE**
    - » **NO - EITHER THERE ARE RELATIVE MINIMA OR THE PROBLEM IS POORLY POSED**
- **HOW DO I KNOW?**
  - > **AVERAGE TWO DESIGN VECTORS**
    - » **ANALYZE THIS DESIGN**

      IF IT'S FEASIBLE, THERE'S PROBABLY JUST A FLAT DESIGN SPACE

      IF IT'S NOT FEASIBLE, THERE ARE PROBABLY RELATIVE MINIMA

GLOBAL OPTIMIZATION

- **WHAT IS "GLOBAL OPTIMIZATION"?**
  - > **THE THEORETICAL ANSWER**
    - » **A GLOBALLY CONVERGANT ALGORITHM IS ONE THAT WILL FIND THE TRUE OPTIMUM FROM ANY STARTING POINT IF THE PROBLEM IS CONVEX**
  - > **A POPULAR MISCONCEPTION**
    - » **"MY ALGORITHM WILL ALWAYS FIND THE GLOBAL MINIMUM EVEN IF RELATIVE MINIMA EXIST"**
    - » **<u>NO SUCH ALGORITHM EXISTS</u>**

      GENETIC SEARCH AND SIMILAR METHODS WILL IMPROVE THE PROBABILITY OF FINDING THE GLOBAL MINIMUM

      THIS IS BECAUSE THESE ARE BASICALLY RANDOM SEARCH METHODS - THEY REQUIRE HUNDREDS TO THOUSANDS OF FUNCTION EVALUATIONS

GLOBAL OPTIMIZATION

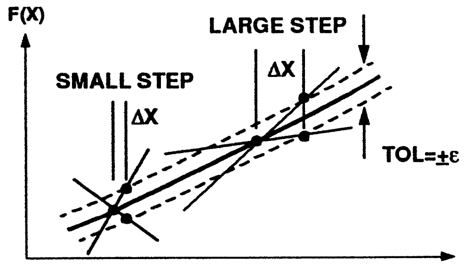
- **WHY CAN'T I CREATE AN ALGORITHM THAT WILL FIND THE GLOBAL MINIMUM?**
  - > **IF YOU CAN PROVE THE PROBLEM IS CONVEX, MOST ALGORITHMS WILL DO THIS**
  - > **IF YOU DON'T KNOW THE CONVEXITY OF THE PROBLEM, YOU CAN ONLY PROMISE A RELATIVE MINIMUM**
- **WHY?**
  - > **TO PROVE AN ABSOLUTE MINIMUM YOU MUST PROVE THAT THE OBJECTIVE AND ALL CONSTRAINTS ARE CONVEX**
    - » **FOR ALL POSIBLE COMBINATIONS OF THE DESIGN VARIABLES, THE HESSON MATRIX OF THE OBJECTIVE AND ALL CONSTRAINTS MUST BE POSITIVE DEFINITE**
  - > **IN ENGINEERING, WE CAN ALMOST NEVER PROVE THIS**
    - » **IF WE KNOW THAT MUCH ABOUT THE DESIGN, WE DON'T NEED OPTIMIZATION ANYWAY!**

WHAT'S OPTIMIZATION GOOD FOR?

- **IF OPTIMIZATION CAN'T GUARANTEE THE TRUE OPTIMUM, WHAT'S IT GOOD FOR?**
  - > **IF THE REAL PROBLEM HAS RELATIVE MINIMA, THE OPTIMIZATION PROBLEM WILL TOO**
  - > **RELATIVE MINIMA ALWAYS EXISTED**
    - » **WE JUST DIDN'T THINK ABOUT THIS BEFORE**
- **OPTIMIZATION ALLOWS US TO INVESTIGATE THE DESIGN SPACE FROM MANY DIFFERENT STARTING DESIGNS MUCH FASTER THAN BEFORE**
  - > **OPTIMIZATION WILL AT LEAST PROVIDE US WITH A SERIES OF RELATIVE MINIMA - NOT JUST A SET OF NON-OPTIMUM POINT DESIGNS**
- **OPTIMIZATION VIRTUALLY *<u>ALWAYS</u>* PROVIDES SOME DESIGN IMPROVEMENT**

TUNING THE OPTIMIZER

- **FINITE DIFFERENCE STEP SIZE FOR GRADIENTS**
  - **IF THE FUNCTIONS ARE CALCULATED VERY PRECISELY**
    - **A VERY SMALL FORWARD DIFFERENCE GRADIENT IS GOOD**
  - **IF FUNCTIONS ARE CALCULATED ITERATIVELY**
    - **USE LARGE FORWARD DIFFERENCE STEP OR USE CENTRAL DIFFERENCE**



TUNING THE OPTIMIZER

- **CONVERGENCE CRITERIA**
  - **DELOBJ AND DABOBJ CONTROL RELATIVE AND ABSOLUTE CONVERGENCE**
  - **IF THE INITIAL OBJECTIVE IS VERY LARGE AND IS REDUCED BY AN ORDER OF MAGNITUDE OR MORE**
    - **SET DABOBJ TO A SMALL VALUE**
      SAY 0.001 TIMES THE "EXPECTED" OPTIMUM
  - **DELOBJ AND DABOBJ MUST BE SATISFIED BY ITRM CONSECUTIVE ITERATIONS**
    - **IF FUNCTION EVALUATIONS ARE VERY CHEAP**
      SET ITRM = 5 TO "BEAT THE PROBLEM TO DEATH"

NO PROGRESS IN OPTIMIZATION

- **IF INITIAL DESIGN IS INFEASIBLE**
  - **SET IPRINT = 5 AND LOOK AT GRADIENTS**
    - **IF GRADIENTS OF ACTIVE/VIOLATED CONSTRAINTS ARE OPPOSITE IN SIGN**
      PROBABLY NO FEASIBLE SOLUTION EXISTS
  - **IF YOU HAVE A FEASIBLE DESIGN BUT NO PROGRESS**
  - **PROBABLY OPTIMUM**
  - **IF IN DOUBT**
    - **SET IPRINT = 5 AND LOOK AT GRADIENTS**
    - **IF GRADIENTS ARE GREATER THEN 10^{-3}**
      PROBABLY "BAD" GRADIENTS
      TRY INCREASING FDCH AND FDCHM FINITE DIFFERENCE STEP SIZES

PROBLEM FORMULATION

- **AVOID EQUALITY CONSTRIANTS**
  - **DO YOU REALLY WANT THE FUNDAMENTAL FREQUENCY TO BE 1 HZ?**
    - **IF YES, AN EQUALITY CONSTRAINT IS JUSTIFIED**
  - **IF > 1HZ WOULD BE NICE**
    - **DON'T IMPOSE YOUR PRECONCEIVED IDEAS ABOUT THE OPTIMUM**

      JUST REQUIRE THAT THE FREQUENCY BE ≥ 1
- **WHY DO WE PREFER INEQUALITIES?**
  - **IT'S EASIER TO STAY INSIDE OF A CURVE THAN TO PRECISELY FOLLOW THE CURVE**

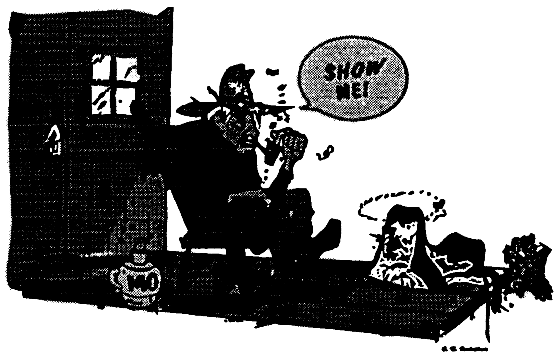
PROBLEM FORMULATION

- **MAKE THE DESIGN VARIABLES AS "INDEPENDENT" AS POSSIBLE**
  - > **E.G. - BETTER TO USE SEVERAL PIECEWISE QUADRATIC POLYNOMIALS TO DEFINE A CURVE THAN TO USE A VERY HIGH ORDER POLYNOMIAL**
- **BE SURE THE OBJECTIVE AND CONSTRAINTS ARE EACH A FUNCTION OF AT LEAST ONE VARIABLE**
- **IT'S OK TO MINIMIZE/MAXIMZE A SINGLE DESIGN VARIABLE; E.G. OBJECTIVE = X(3)**

PROBLEM FORMULATION

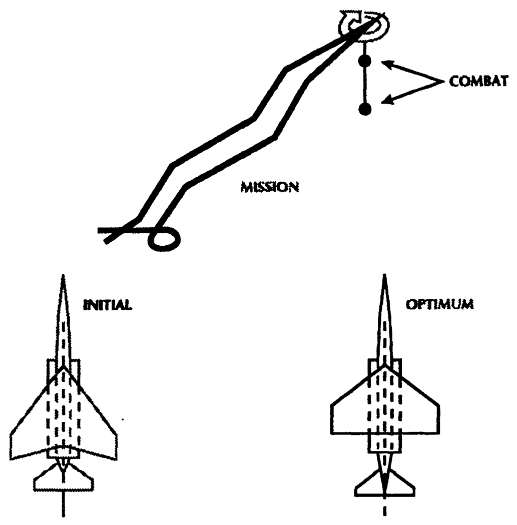
- **CONSIDER A SIMPLE PROBLEM**
  - > **F(A) = 3*A1 + 5*A2**
  - > **G(A) = 2/A1 + 4/A2^2 -5 < 0**
- **IF X^T = { A1 A2 }**
  - > **THE OBJECTIVE IS LINEAR AND THE CONSTRAINT IS NONLINEAR**
- **NOW LET X1 = 1/A1 AND X2 = 1/A2^2**
  - > **F(X) = 3/X1 + 5/SQRT(X2)**
  - > **G(X) = 2*X1 + 4*X2 -5**
- **THE OBJECTIVE IS NOW NONLINEAR AND THE CONSTRAINT IS LINEAR**
  - > **THIS IS MUCH EASIER TO OPTIMIZE**
  - > **THE ORIGINAL VARIABLES, A1 AND A2 ARE EASILY RECOVERED**
- **THIS IS NOT ALWAYS POSSIBLE**
  - > **BUT WE SHOULD TRY**

OPTIMIZATION WORKS



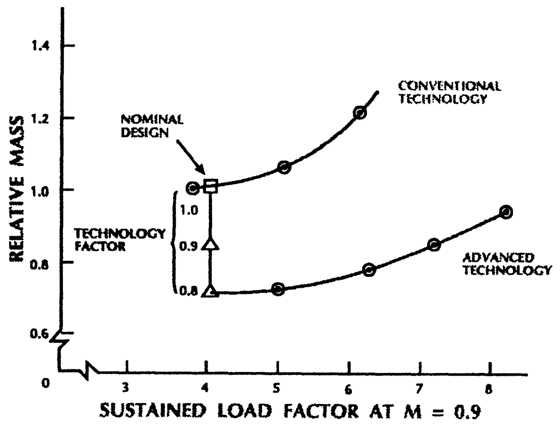
NUMERICAL OPTIMIZATION

SUPERSONIC CRUISE AIRCRAFT



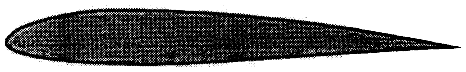
SOLVED BY THE ACSYNT PROGRAM
5 DESIGN VARIABLES, 2 PERFORMANCE CONSTRAINTS

SUPERSONIC CRUISE AIRCRAFT

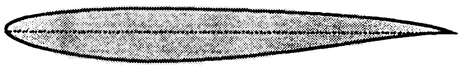


TRADE - OFF STUDY

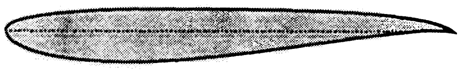
HIGH SPEED AIRFOIL OPTIMIZATION



INITIAL SHAPE



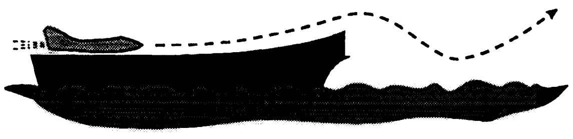
OPTIMUM: MAXIMIZE LIFT WITH DRAG & MOMENT CONSTRAINTS



OPTIMUM: MINIMIZE DRAG WITH LIFT & MOMENT CONSTRAINTS

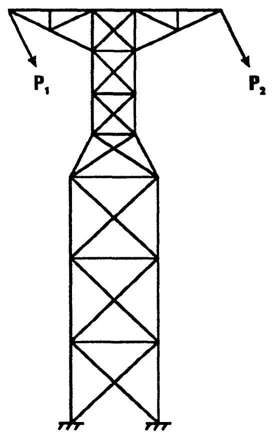
STOL AIRCRAFT TAKEOFF

CONVENTIONAL: $W_G = W^0$

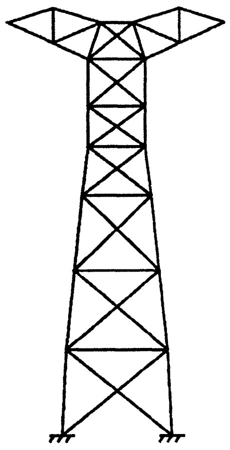


SKI JUMP: $W_G = 1.2W^0$

TOWER OPTIMIZATION



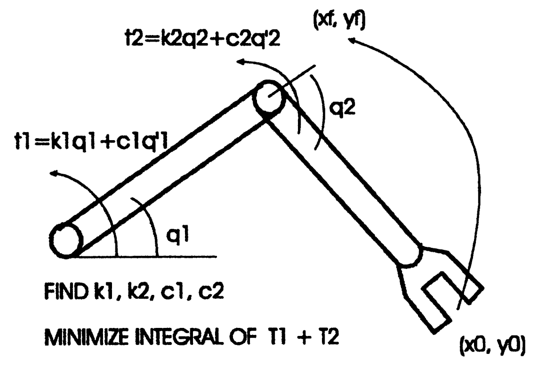
OPTIMIUM FOR SIZING ONLY: W_0



OPTIMUM FOR SIZING AND SHAPE: $0.8W_0$

NUMERICAL OPTIMIZATION

- **ROBOT ARM**



FIND k1, k2, c1, c2

MINIMIZE INTEGRAL OF T1 + T2

GO FROM (x0, y0) TO (xf, yf)

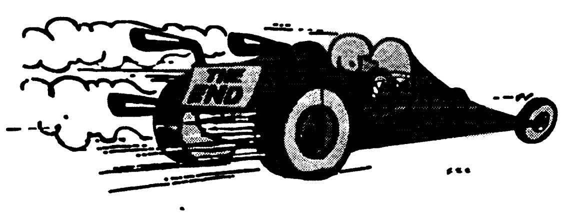
REF. ADAMS OPT. GUIDE, P. 47

- **THIS WILL MINIMIZE THE "AVERAGE" TORQUE**
  - > WE CAN EASILY CONSTRAIN THE MAXIMUM TORQUES AS WELL

THE FUTURE

- **GENERAL PURPOSE OPTIMIZATION IS ESTABLISHED**
  - > COUPLE OPTIMIZER WITH USER'S ANALYSIS
- **STRUCTURAL OPTIMIZATION IS WELL DEVELOPED**
  - > SEVERAL COMMERCIAL CODES AVAILABLE
- **MECHANICAL OPTIMIZATION IS NOW AVAILABLE**
  - > ADAMS IS WELL ESTABLISHED AS PREMIER CODE
    - » ADAMS OPTIMIZATION ADDS A NEW DIMENSION
- **THE NEXT STEP IS FULLER INTEGRATION**
  - > COMBINED MECHANICAL AND STRUCTURAL OPTIMIZATION
    - » INCLUDE FLEXIBLE DYNAMICS WHILE OPTIMIZING BOTH THE STRUCTURE AND THE MECHANICS
  - > SYSTEMATICALLY ADD OTHER DISCIPLINES

NUMERICAL OPTIMIZATION



VMA Engineering

Vanderplaats, Miura & Associates, Inc.

1767 S. 8th Street, Suite M-200
Colorado Springs, CO 80906

Tel. (719) 473-4611 Fax (719) 473-4638

March 31, 1995

Marilyn Lee
Mechanical Dynamics, Inc.
2301 Commonwealth Boulevard
Ann Arbor, MI 48105

Dear Marilyn:

Enclosed is an original of my paper for the ADAMS users conference. I have also enclosed copies of my slides for the presentation. These are more simplistic and general. If you think it is appropriate, please feel free to include these in the proceedings.

I *almost* got this to you by the deadline. Boy am I proud!

Thanks for your assistance with hotels. I got your fax.

I've organized numerous conferences. Tell MDI I said you deserve a few days extra R&R after May 17!

Sincerely,

[illegible]

Garret N. (Gary) Vanderplaats
President