MY CAREER IN ENGINEERING OPTIMIZATION

G. N. Vanderplaats
Founder and Chief Executive Officer
Vanderplaats Research & Development, Inc.
1767 S. 8th Street, Suite 200
Colorado Springs, CO 80905, USA

ABSTRACT

The purpose here is to offer a narrative of my experience in developing gradient based optimization algorithms. This is not intended to be a review of the technology in general but a review of my personal views, motives and experience. Thus, it is written in the first person.

The various algorithms I have developed, modified or used will be described and outlined. Perhaps more importantly, my rational for often raining havoc on theory will be given. Also, I will offer comments on what I've learned about problem formulation, critique of other algorithms and general musings. Since much of this is in my textbook [1], I've taken the liberty of copying from it where possible (plagiarizing myself). While the non-technical reader may find some useful philosophy here, I am assuming the reader is familiar with basic matrix theory.

The overall goal in offering this narrative is to encourage developers in engineering optimization to treat this technology as a tool for design and not a mathematical exercise. This requires a solid background in optimization theory as well as engineering design. It is concluded that optimization algorithms and software should mimic the thought process of a good designer while taking advantage of the immense power of today's computers.

INTRODUCTION

For some time, our company president, Juan Pablo Leiva has been encouraging me to write a narrative of my experience developing gradient based optimization algorithms and software. Once I started on it, I quickly realized that gradient based optimization is only part of the story. As I have been writing, I naturally fell into the mode of talking philosophy because that forms the basis of much of my work. Therefore, this has become more the technical story of my career with sometimes rambling sidelines into personal stories, motives and views.

My first exposer to numerical optimization (mathematical programming) was in May 1967. Professor Louis Hill was my mentor at Arizona State University, where I was about to graduate with a degree in Civil Engineering. He had been encouraging me to get a Masters degree and stopped me on the sidewalk to tell me he had gotten a fellowship for me. He gave me copies of Lucien Schmit's landmark paper of 1960 [2], as well as others (including his own) and suggested I do research in optimization of concrete footings. I browsed the papers and concluded that "This is mathematical gibberish." However, I politely accepted his fellowship and did my masters thesis on finite element analysis with Professor Scholler. Hill was not dissuaded and stopped me the next May, same sidewalk, same time of day, and possibly the same day in May. He said, "I have a NSF fellowship for you at Case. Now get a PhD." Lucien (I'll take the liberty of calling him Lucien in some cases here as we became close friends over the years) was the chairman at Case and was instrumental in getting me the fellowship. When he welcomed me to Case, I neglected to tell him that I thought optimization was nonsense. However, once I took a class from him on "Structural Synthesis" as he called it, I was hooked. Later, he was instrumental in my being hired by NASA and I became the only student of structural synthesis at that time who was able to spend his entire career in the optimization field.

When I was beginning my thesis research, Professor Schmit called me to his office and advised me that he was moving to UCLA. We investigated the possibility that I transfer but, in the end, he said "I think you should stay here and finish with Professor Moses. That way you'll be back in California sooner with a degree." That was prophetic because the following June, people from NASA Ames Research Center visited UCLA looking for someone specializing in optimization. Professor Schmit gave them my name as well as my roommate's name. I was doing a thesis in truss configuration optimization. My roommate had a masters degree in aerospace engineering and was doing his research in supersonic wing optimization. I got the job because I was the U.S. citizen!

An important aspect of my studies at Case is that Professor Schmit sent us to the Operations Research Department to take theoretical courses on linear and nonlinear programming. The professor there was Leon Lasdon who is renown in his field. While I didn't like studying theory much, it has provided an invaluable background for developing algorithms.

I always thought of myself as a designer, not an analyst. Therefore, once I understood the numerical optimization process, I was naturally attracted to algorithm development. In other words, "How can I create an algorithm that will perform the design process more efficiently and reliably?" Designers and analysts think differently. While analysts often take pride in their ability to account for all conceivable nonlinearities, designers tend to focus on first principals, believing that simple is better. Of course, with the sophistication of our best analysis tools, together with the power of modern computers, designers can utilize the analyst's tools early in the design process.

This background forms the basis of my research and development in optimization algorithms. In the following sections, I will discuss this avocation which has consumed my career for nearly five decades. Much of the descriptions given here are brief but, as one may expect, more details may be found in Ref. 1.

To provide the basis for this discussion, the following statement of the nonlinear programming problem will be used;

Minimize: $F(\mathbf{X})$ objective function (1)

Subject to:

$$g_j(\mathbf{X}) \le 0 \qquad j=1,m \quad \text{inequality constraints} \tag{2}$$

$$h_k(\mathbf{X}) = 0 \qquad k=1,l \quad \text{equality constraints} \tag{3}$$

$$X_i^l \le X_i \le X_i^u \qquad i=1,n \quad \text{side constraints} \tag{4}$$

where $$\mathbf{X} = \begin{Bmatrix} X_1 \\ X_2 \\ X_3 \\ . \\ . \\ . \\ X_n \end{Bmatrix}$$ design variables

The vector **X** is referred to as the vector of design variables. The objective function $F(\mathbf{X})$ given by Eq. 1, as well as the constraint functions defined by Eqs. 2 and 3 may be linear or nonlinear functions of the design variables **X.** These functions may be explicit or implicit in **X** and may be evaluated by any analytical or numerical techniques we have at our disposal. Indeed, these functions could actually be measured experimentally. However, except for special classes of optimization problems, it is important that these functions be continuous and have continuous first derivatives in **X.** Eq. 4 defines the lower and upper bounds on the design variables and are referred to as side constraints.

Most optimization algorithms require that an initial set of design variables, $\mathbf{X}^0$**,** be specified. Beginning from this starting point, the design is updated iteratively. Probably the most common form of this iterative procedure is given by

$$\mathbf{X}^q = \mathbf{X}^{q-1} + \alpha^* \mathbf{S}^q \tag{5}$$

where q is the iteration number and **S** is a vector search direction in the design space. The scalar quantity α^* defines the distance that we wish to move in direction **S**.

Note that $\alpha^* \mathbf{S}^q$ represents a perturbation on **X** so Eq. 5 could be written as;

$$\mathbf{X}^q = \mathbf{X}^{q-1} + \delta\mathbf{X} \tag{6}$$

Therefore, Eq. 5 is very similar to the usual engineering approach of Eq. 6 where we perturb an existing design to achieve some improvement. Understanding this is what enamoured me with this technology. I realized that I could automate the “cut and try” approach to design. More importantly, mathematical programming offered a logical method to search for an improved design.

Figures 1 and 2 are an example I often use to describe the optimization process.

The idea is to imagine you want to climb a hill blindfolded. The reason for the blindfold is that a computer has no way to look ahead but can only evaluate things where you are.

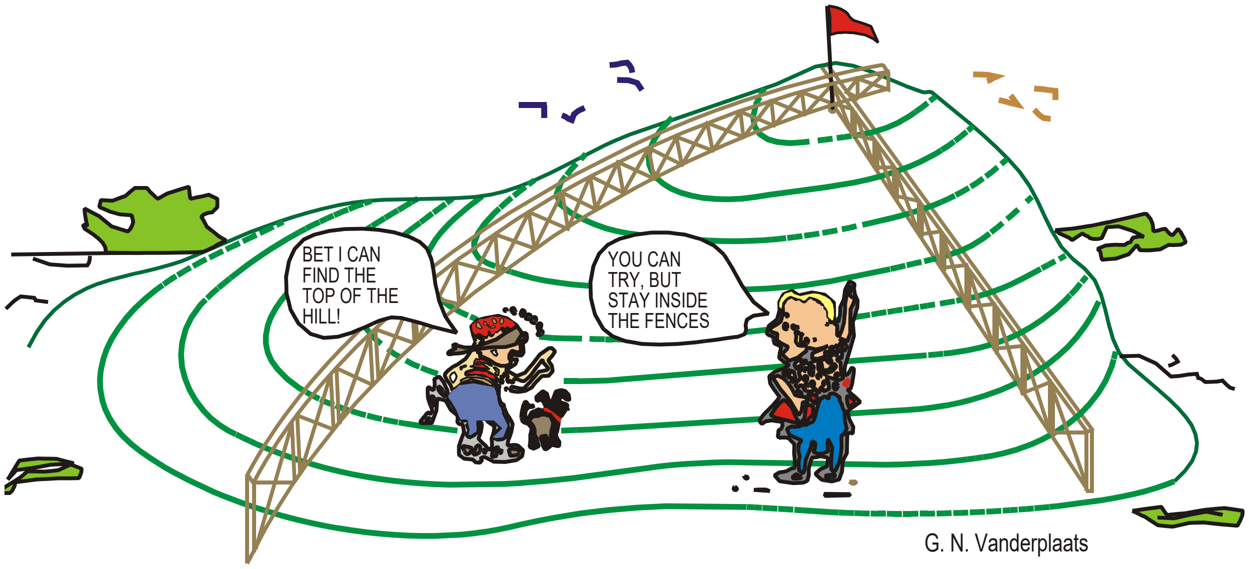
BET I CAN FIND THE TOP OF THE HILL!
YOU CAN TRY, BUT STAY INSIDE THE FENCES
G. N. Vanderplaats

Figure 1. The Physical Problem

Now ask, how you may climb the hill. This process is shown in Figure 2.

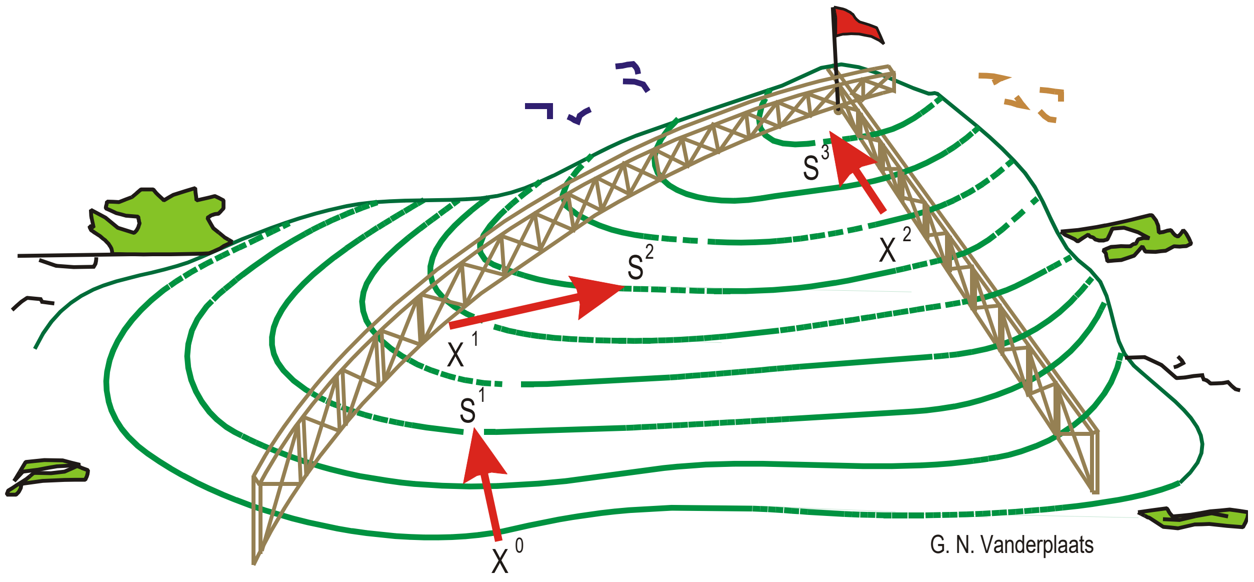

Figure 2. The Search Process

Here the design variables are the longitude and latitude, the objective function is to find the highest the elevation on the hill inside the fences and the constraints are the two fences. Note that the fence on the right is straight so is straight and so a linear function of the design variables.

Beginning at $\mathbf{X}^0$ we may take a small step in the north-south direction and a small step in the east-west direction to determine the slope of the hill. In other words, we calculate the gradient of the objective by taking finite difference steps. This gives us the direction of steepest ascent, $\mathbf{S}^1$.

We now search in direction $\mathbf{S}^1$ until we cross the crown of the hill or encounter a fence; in this case, we encounter a fence. This process is called a line search or one-dimensional search. This leads us to design point $\mathbf{X}^1$.

Now, we again calculate finite difference gradients. Here, we calculate the gradient of the objective function and the "active" constraint. We now determine search direction, $\mathbf{S}^2$, and continue the search process. Note that we chose our search direction to "push away" from the fence. Otherwise, a small step in that direction will move us outside the fence. In other words, it would violate this constraint. At the second fence, we can find our search direction tangent to the fence, assuming we knew in advance that it is strait (linear).

This description of the optimization process is essentially Zoutendijk's method of feasible directions. However, the search directions calculated this way are not unique. Any search that improves the objective (usable) and stays inside the fences (feasible) is acceptable. This is why virtually every researcher in optimization algorithms creates his or her "new" method. In my case, it was the "Modified Method of Feasible Directions" or MMFD which will be discussed later.

THE EARLY YEARS

This actually begins in September of 1968 with my arrival at Case Western Reserve University in Cleveland. Lucien Schmit was Chairman of the Engineering Mechanics Department and there were five or six doctoral students focused on optimization. Other professors who supported this were Richard Fox, Thomas Kicker and Fred Moses. Fox's expertise was dynamics, Kicker's was plates and shells and Moses's specialty was reliability. Schmit's fame in structural synthesis is widely known. He also made fundamental contributions in finite element analysis, most notably the use of Hermite Interpolation Polynomials. Professor Schmit taught finite element analysis and structural synthesis. He did not remember it years later when I reminded him that I failed the finite element analysis part of my qualifiers. I was at home in California for the summer and received a note from him saying something to the effect "You've passed the dynamics and materials parts of the qualifiers and will be encouraged to repeat the finite element analysis portion next year." I guess this didn't diminish his faith in me because we became very close friends and always reserved an evening at the annual AIAA Structures, Structural Dynamics and Materials Conference for a private dinner together.

As mentioned above, we took mathematical programming classes from Professor Lasdon in the Operations Research Department. When we studied linear programming, I wrote a linear programming code. When we studied nonlinear programming, I coded the various one-dimensional search algorithms and went on to code virtually every nonlinear programming algorithm we studied.

In the late 1950s, random search methods were popular. They did not require gradients, were easy to program and virtually always improved the design, though at the cost of many function evaluations (analyses). Various people published enhanced random search methods intended to improve efficiency. My observation of these enhancements was simple. They observed that, after numerous function evaluations, they could identify the worst and best design. They then drew a line (mathematically) from the worst to best and searched in that direction. This is logical and is what good engineers do. However, it struck me that they were just calculating a finite difference gradient, albeit often a rather poor one. I reasoned that, if we're going to get gradients anyway, why not do it up front and more accurately. This is at the heart of why I've always focused on gradient based methods.

Also, in the late 1950s and continuing through the 1960s, Sequential Unconstrained Minimization Techniques (SUMT) were the "state of the art" and the book by Fiacco and McCormick [3] was the bible. This represented a logical progression because, in the late 1950s, Powell [4], Fletcher and Reeves [5] and Davidon, et al [6] offered powerful methods for solving unconstrained minimization problems defined by Eq. 1 only.

Above, I described calculating the steepest ascent direction. Generally, I will assume we are minimizing so we need the steepest descent direction. If we need to maximize a function, simply minimize its negative. Now, considering unconstrained optimization, we could simply search repeatedly in the steepest descent direction for each iteration. If we do this, we cannot guarantee ever reaching the optimum. The unconstrained minimization methods developed in this period had the feature that convergence could be guaranteed for quadratic functions. While this, of course, does not prove convergence for general nonlinear functions, it provides a solid foundation for the more general applications.

This ability to prove convergence for quadratic problems forms the basis of much of the theoretical work in optimization. This seems to be lost on many researchers who have never taken a theoretical course on the subject. However, it is the reason that when using response surface approximations that I prefer quadratic approximations instead of any of the other "advanced" methods. This will be discussed in more detail later.

With Sequential Unconstrained Minimization Techniques, the original constrained optimization problem is converted to a sequence of unconstrained problems of the form;

Minimize

$$\Phi(\mathbf{X}) = F\mathbf{X} + r_p \sum_{j=1}^{m} q_j^p \{MAX[0, g_j(\mathbf{X})]\}^2 \tag{7}$$

Subject to;

$$X_i^L \le X_i \le X_i^U \qquad i = 1, n \tag{8}$$

Eq. 7 defines the "Exterior Penalty Function." 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 unconstrained minimization 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.

Professor Schmit preferred the "Interior Penalty Function Method" of the form;

Minimize

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

Here, the penalty parameter, r'_p is initially set to a large value and sequentially reduced during the optimization process. Schmit preferred the interior method because it begins with a feasible design (one that satisfies all constraints) and approaches the optimum while retaining feasibility. This was important in those days where computers were, by today's standards, pitifully slow. Therefore, we always ran out of computer time before the optimization converged. However, we at least had a feasible design which was better than what we started with.

During the 1970s, considerable research was continued in penalty function methods, leading to concepts such as extended interior penalty functions. Also, a theoretically attractive method known as the Augmented Lagrange Multiplier Method was developed [7].

Equality constraints are included above for completeness but will be ignored hereafter except where needed for clarity. In general, it can be assumed that equality constraints can be handled by some form of penalty term or by creating two equal and opposite inequality constraints.

In 1960 Zoutendijk [8] published a textbook on his feasible directions method. The book was difficult reading and was out of print by the mid-1960s. The theoreticians didn't seem to think much of his method, citing the problem of "zig-zagging" where a constraint would become active on one iteration, inactive on the next and then active again on following iterations. Also, no proof of convergence could be made, a fact that is offensive to a purest in the field.

I was attracted to Zoutendijk's method because it is the way designers think. Find a way to perturb the design to improve it. If the design does not satisfy the constraints, find a search direction that will at least reduce the constraint violation. It also has the advantage that we only need gradients of active or violated constraints. This is important because, in areas such as structural synthesis, we may have only a few variables but thousands of constraints (stress in each finite element for each of many load cases, for example).

The Optimization Algorithm

Zoutendijk's method required that we start with a feasible design and remain feasible. However, one of the most powerful uses of optimization is to simply find a design that is feasible.

My research was focused on two areas. First was to find a way to design trusses including both the member sizes and the coordinates of the joints. This led to a multilevel technique that also became part of my MDO (Multidiscipline Design Optimization) research in later years. The second area was the optimization algorithm and I chose Zoutendijk's method as the starting point for that.

The key to Zoutendijk's method is in calculating the search direction, **S**. In Figure 3, consider a design $\mathbf{X}^0$ on constraint boundary $g_1(\mathbf{X})$. We first calculate the gradient of the objective function and of the active constraint to yield the gradient vectors shown.

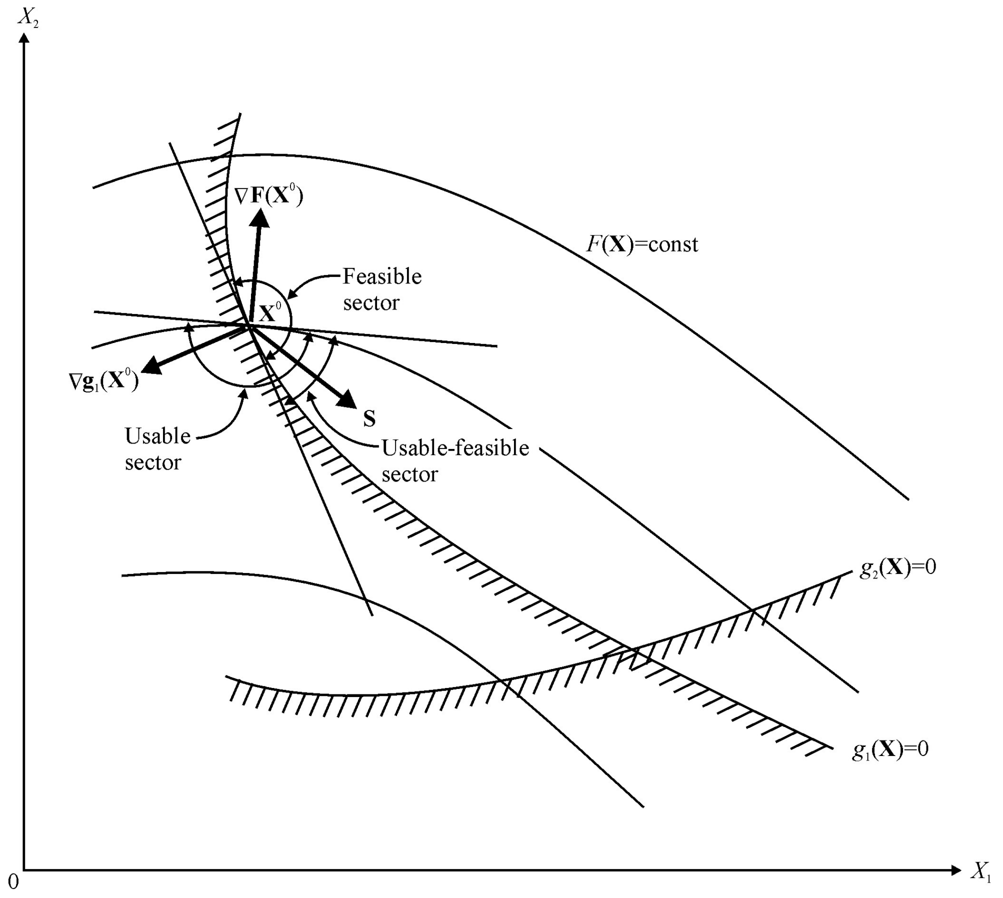

Figure 3. Usable-feasible search direction.

The lines (hyperplanes in *n* dimensions) tangent to the line of constant objective and tangent to the constraint boundary are now the linear approximation to the problem at this point. The task is to find a search direction, **S,** which will reduce the objective function without violating the active constraint for some finite move.

Any **S** vector which reduces the objective function is called a usable direction. Clearly, such a search direction will make an angle greater than 90° with the gradient vector of the objective function. This suggests that the dot product of $\nabla \mathbf{F}(\mathbf{X})$ and **S** should be negative, since the dot product is the product of the magnitudes of the vectors and the cosine of the angle between them, and this angle must be between 90° and 270° for the cosine to be negative. The limiting case is when the dot product is zero, in which case the **S** vector is tangent to the plane of constant objective function. Mathematically, the usability requirement becomes

$$\nabla \mathbf{F}(\mathbf{X}^0)^T \mathbf{S} \le 0 \tag{10}$$

A direction is called feasible if, for some small move in that direction, the active constraint will not be violated. That is, the dot product of $\nabla g_1(\mathbf{X})$ with **S** must be non-positive. Thus,

$$\nabla \mathbf{g}_1(\mathbf{X}^0)^T \mathbf{S} \leq 0 \tag{11}$$

Observe that the greatest reduction in $F(\mathbf{X})$ can be achieved by finding the **S** which will minimize the quantity in Eq. 10, with Eq. 11 met with precise equality. This would give a search direction precisely tangent to the constraint boundary. However, assuming the constraint is nonlinear and convex, a small move in this direction would violate the constraint. This is not desirable, so we need a way to "push" away from the constraint boundary. We can do this by adding a positive "push-off" factor to Eq. 11 to give

$$\nabla \mathbf{g}_1(\mathbf{X}^0)^T \mathbf{S} + \theta_1 \leq 0 \tag{12}$$

where θ_1 is not a geometric angle but is simply a nonnegative constant. This ensures that the cosine of the angle between $\nabla g_1(\mathbf{X})$ and **S** is strictly negative so that the angle between the two vectors will exceed 90^o.

In practice, we would like the push-off factor to be affected by the direction of $\nabla \mathbf{F}(\mathbf{X})$. Thus, remembering that $\nabla \mathbf{F}(\mathbf{X})^T \mathbf{S}$ is negative for usability, we can modify Eq. 12 to be

$$\nabla \mathbf{g}_1(\mathbf{X}^0)^T \mathbf{S} - [\nabla \mathbf{F}(\mathbf{X}^0)^T \mathbf{S}]\theta_1 \leq 0 \tag{13}$$

Now minimizing the quantity in Eq. 10 is equivalent to maximizing β in the following inequality.

$$\nabla \mathbf{F}(\mathbf{X}^0)^T \mathbf{S} + \beta \leq 0 \tag{14}$$

and Eq. 14 becomes the usability condition. Clearly for β to be maximum, Eq. 14 will be satisfied with equality so $\beta = -\nabla \mathbf{F}(\mathbf{X}^0)^T \mathbf{S}$. Therefore, the feasibility requirement of Eq. 13 becomes, in general

$$\nabla \mathbf{g}_j(\mathbf{X}^0)^T \mathbf{S} + \theta_j \beta \leq 0 \tag{15}$$

Figure 4 shows the effect that the push-off factor θ has on determining the possible **S** vectors. With $\theta = 0$, the **S** vector is tangent to the constraint boundary, while as θ becomes large, a direction tangent to the line of constant objective function results, since the usability requirement of Eq. 10 prevents a search direction which increases $F(\mathbf{X})$. That is, we wish to minimize the quantity in Eq. 10 [maximize β in Eq. 14] while insisting the result be non-positive. Assuming we have normalized both $\nabla \mathbf{F}(\mathbf{X})$ and $\nabla \mathbf{g}_1(\mathbf{X})$, a push-off factor of $\theta = 1.0$ roughly bisects the angle between the line of constant objective and the constraint boundary.

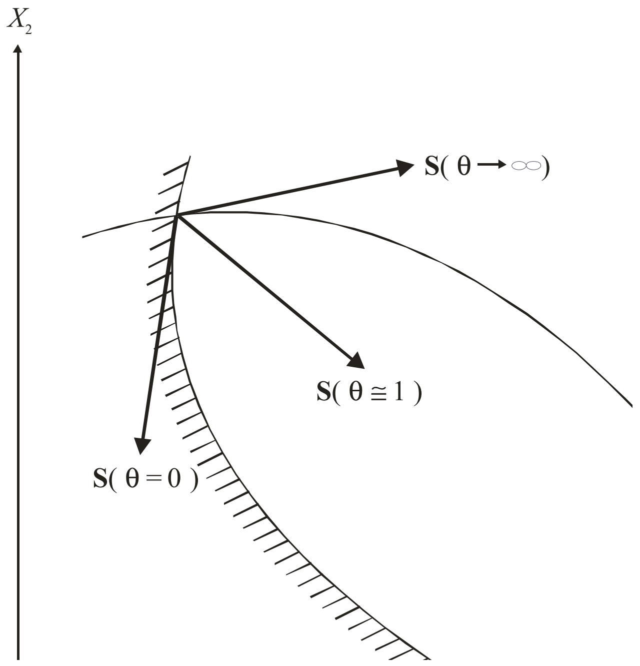
X_2
$\mathbf{S}(\theta \rightarrow \infty)$
$\mathbf{S}(\theta \cong 1)$
$\mathbf{S}(\theta = 0)$

Figure 4. Effect of θ on the search direction.

At this point, we can make a simple mathematical observation. If we have an **S** vector such that β is maximized in Eq. 14 and the first term in Eq. 15 is negative, we have met the conditions of usability and feasibility. However, if we multiply the **S** vector by any positive number, we still satisfy these conditions. Therefore, if we wish to maximize β, the solution will be unbounded with the magnitude of **S** increasing without bound. To prevent this, we must impose bounds on the **S** vector. Also, in general, there may be more than one active constraint for which the feasibility condition of Eq. 15 must be met. Thus, we can write the general direction-finding problem as, find the values of the components of **S** and β

Maximize: β (16)

Subject to:

$$\nabla \mathbf{F}(\mathbf{X}^0)^T \mathbf{S} + \beta \le 0 \tag{17}$$

$$\nabla \mathbf{g}_j(\mathbf{X}^0)^T \mathbf{S} + \theta_j \beta \le 0 \qquad j \in J \tag{18}$$

$$\mathbf{S} \text{ bounded} \tag{19}$$

where J is the set of currently active constraints $g_j(\mathbf{X}) = 0$.

Calculating the Push-Off Factors

From Eqs. 16 to 18, several questions arise. First, when do we define a constraint as active? Second, how can we choose reasonable values for the push-off factors θ_j? And finally, how do we impose bounds on **S**?

The first two issues are interdependent in a practical optimization algorithm. From an engineering viewpoint, we may consider a constraint to be active if we are near the boundary, say $g_j(\mathbf{X}) \geq \varepsilon$, where $\varepsilon = -0.1$ at the beginning of the optimization and is reduced near the end to, say, -0.001. This numerically large "constraint boundary" has the effect of "trapping" constraints quickly, and it helps considerably in alleviating a problem known as "zig-zagging," whereby a constraint alternately becomes active and inactive on successive iterations, a situation which reduces the efficiency of the method.

Coupled with our use of a numerically wide constraint boundary is the choice of the push-off factor θ_j. While it is often recommended that $\theta_j = 1.0$ be used for all nonlinear constraints and $\theta_j = 0$ for all linear constraints, it has been found that θ_j is better defined by the following expression [9, 10]:

$$\theta_j = \left[1 - \frac{g_j(\mathbf{X})}{\varepsilon}\right]^2 \theta_0 \tag{20}$$

Equation 20 has the feature that, as a constraint just becomes active [$g_j(\mathbf{X}) = \varepsilon$], the push-off factor is zero. As $g_j(\mathbf{X})$ approaches zero (from the negative side), θ_j increases quadratically until $\theta_j = \theta_0$ at $g_j(\mathbf{X}) = 0$. This has the effect of pushing harder as the constraint becomes more critical and is consistent with engineering judgment. Usually $\theta_0 = 1.0$ is adequate for the nominal value of θ_j. Also, $\theta_j = 0$ is used if the constraint is known to be linear, and we will wish to use a smaller value of ε, say, -0.001, for linear constraints. This choice of ε is because interpolation on linear constraints is numerically easy relative to interpolation on nonlinear constraints.

When we consider violated constraints we will allow θ_j to become large to "push" the design back to the feasible region.

Bounds on the S Vector

The remaining problem now is to devise a way of imposing bounds on the **S** vector. Probably the most popular technique is to bound the components of **S** as

$$-1 \leq S_i \leq 1 \qquad i = 1, n \tag{21}$$

Now Eqs. 16 to 19 and 21 define a linear problem in which the objective to be maximized is β and the decision variables are the components S_i and β itself. By simple transformation,

$$S_i = S_i' - 1 \qquad i = 1, n \tag{22}$$

Equation (21) becomes

$$0 \leq S_i' \leq 2 \qquad i = 1, n \tag{23}$$

Now, substituting Eq. 22 into Eqs. 17 and 18 gives the problem in standard Linear Programming form for solution by the simplex method, where the decision variables are S_i' , $i = 1, n$ and β. Having solved this problem, Eq. 22 is used to define the final search direction **S**.

While this approach is easy to incorporate into the program, it has a significant drawback, as seen from Eqs. 17 and 18 and Figure 5, where the square box represents the bounding given by Eq. 21. Referring to Eqs. 17 and 18, we see that if we maximize β, we are implicitly making the first terms in these equations as large and negative as possible. One way to do this is to make the magnitude of the **S** vector as large as possible. Referring again to Figure 5, the largest magnitude of **S** is obtained if **S** points toward the corners of the bounding square (hypercube in *n* dimensions). In other words, this choice of bounding the **S** vector actually biases the search direction. This is undesirable, since we wish to use the push-off factors θ_j as our means of controlling the search direction. A method which avoids this bias in the search direction is also shown in Figure 5 as a circle (hypersphere in n dimensions). This is mathematically equivalent to bounding **S** by the Euclidian norm

$$\mathbf{S}^T\mathbf{S} \le 1 \tag{24}$$

Unfortunately, using Eq. 24 with Eqs. 17 to 18 creates a problem for finding **S**, which is linear except for one quadratic constraint, so that linear programming cannot be used directly. Some authors suggest that this can be converted to a problem in which the objective is quadratic and the constraints are linear. However, using the Kuhn-Tucker conditions, Zoutendijk [8] provides a much more direct approach.

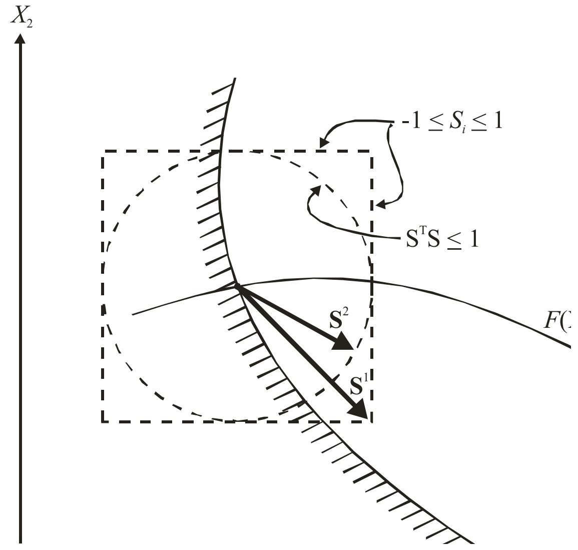

Figure 5. Bounding of the S vector.

Zoutendijk's book describes a convenient way to impose the quadratic bounds via the Kuhn-Tucker conditions. In my research, I first coded the method with the simple + or - bounds on the components of **S**. I finished the thesis in April, 1971 and was sitting around waiting for my mother to come and see me graduate when I read that portion of the book for perhaps the fifth time. It finally got through to me and required less than four hours to code and test.

It is noteworthy that if all push-off factors θ_j. were taken as zero, the resultant **S** vector is the projection of the negative of the gradient of the objective onto the active constraint surfaces. Two observations follow from this. First is that if this projection is zero, then the gradient of the objective is some linear combination of the gradients of the active constraints. This is equivalent to the second and third Kuhn-Tucker conditions. In fact, the **S** vector obtained here may be viewed as the residual in a direct computation of the Kuhn-Tucker multipliers. Therefore, if the magnitudes of the components of **S** are small, say less than 0.001, this indicates that the Kuhn-Tucker conditions for local optimality are met and the optimization process can be stopped. In the usual case where the push-off factors are nonzero, this same argument applies because, at the optimum, the usable-feasible sector in Figure 3 goes to zero so the value of θ_j has no effect. The second observation is that if the Kuhn-Tucker conditions are met, the objective β of this suboptimization problem will be zero, or at least numerically very small. This is actually equivalent to the first observation in that it indicates that the Kuhn-Tucker conditions have been satisfied and thus the optimization can be terminated. In either case, if the constraint tolerance ε is large in magnitude, it should be reduced to see if any constraints cease to be active. If so, it may now be possible to find a search direction. In other words, the optimization process should not be stopped unless all active constraints are suitably close to zero.

Dealing with Initially Infeasible Designs

The method of feasible directions assumes that we begin with a feasible design and maintain feasibility throughout the optimization process. If the initial design is not feasible, a search direction pointing toward the feasible region can be found by a simple modification to the direction-finding problem.

The basic approach here is to delete the usability requirement, Eq. 17, from the direction-finding problem. If a constraint is badly violated, the value of θ_j given by Eq. 20 may become very large and should be limited, for numerical reasons, to a value of, say, 50. That is, if $\theta_j > 50$ from Eq. 20, θ_j is arbitrarily set to 50. Now a direction is found which will be away from the violated constraints and toward the feasible region and which will very likely increase the value of the objective function.

It is noteworthy that a design situation can exist in which the violated constraints are strongly dependent on part of the design variables, while the objective function is primarily dependent on other design variables. For example, in a composite laminate, the strength may be highly dependent on the direction of the fibers, but the mass is not a function of fiber orientation at all. This suggests a further modification whereby we seek a search direction which will simultaneously minimize the objective while overcoming the constraint violations.

These considerations lead to the following statement of the direction-finding problem:

Maximize: $$-\nabla \mathbf{F}(\mathbf{X})^T \mathrm{S} + \Phi\beta \tag{25}$$

Subject to:

$$\nabla \mathbf{g}_j(\mathbf{X})^T \mathbf{S} + \theta_j \beta \le 0 \qquad j \in J \tag{26}$$

$$\mathbf{S}^T \mathbf{S} \le 1 \tag{27}$$

Now the set *J* contains all active and violated constraints, where a constraint is considered violated if its value is more positive than some parameter ε_{min}, say, 0.005 for nonlinear constraints and 0.0001 for linear constraints. Here, if a linear constraint is violated, we use a nonzero push-off factor until the constraint violation is overcome. The scalar Φ in Eq. 25 is a weighting factor determining the relative importance of the objective and the constraints. Usually a value of $\Phi \geq 10^5$ will ensure that the resulting **S** vector will point toward the feasible region.

If the result of the subproblem of finding **S** is that the components of **S** are all very small in magnitude or that β is very small (zero), this indicates the second and third Kuhn-Tucker conditions are satisfied. However, since the current design is not feasible, the first condition (feasibility) is not satisfied. If such a situation exists, it indicates that no feasible solution can be found and so the optimization process should be terminated. This argument can be proved for convex design spaces. If we suspect that the design space is non convex, it is good practice to restart the optimization from a different point. If the same result is found from several starting points, we can be reasonably sure that no feasible design exists. However, our effort has not been wasted because we have found the design which most nearly satisfies our design criteria and we have identified the constraints which must be relaxed (something which is often acceptable in practical design) in order to achieve a meaningful design.

The one-dimensional search algorithm becomes somewhat complicated for this method. If no constraints are violated, we find the largest α^* in Eq. 5 from all possible values that will minimize the objective, encounter some new constraint, or re-encounter a currently active constraint. If one or more constraints are violated, we find the α^* which will minimize the maximum constraint violation or which will minimize the objective if the design corresponding to this is feasible. Figure 6 depicts several possibilities which exist for the one-dimensional search process. The methods of Chapter 2 in Ref. [1] may be used to create the one-dimensional search algorithm to be used for this general problem.

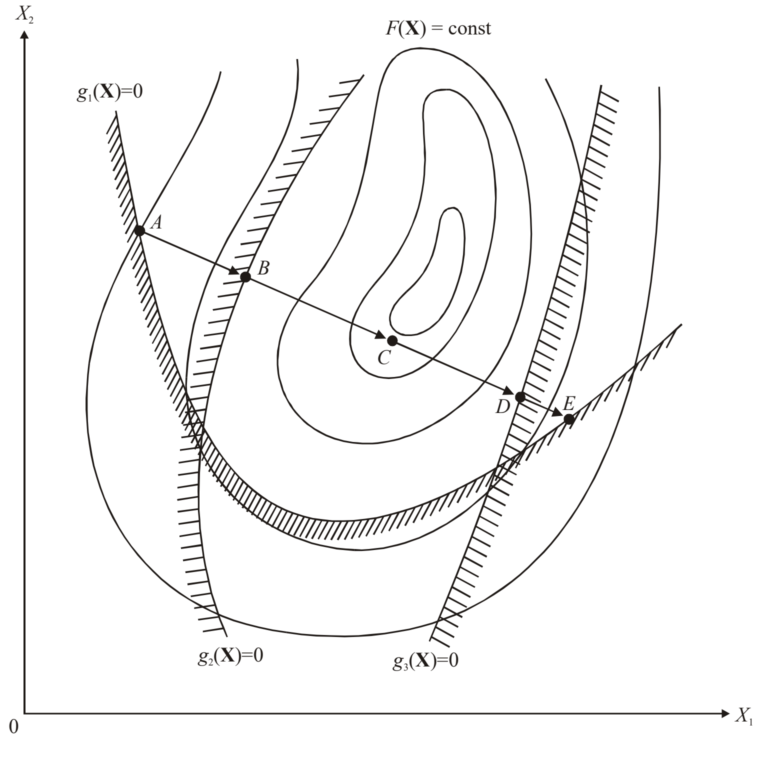

Figure 6. Possible solutions to the one-dimensional search.

Search Direction Modification

Often in design optimization, the one-dimensional search is limited by the α at which a side constraint X_i^l or X_i^u becomes critical. Assume α^* corresponds to encountering the lower or upper bound on some variable X_k. Because this constraint is a function of X_k only, it may be possible to fix this value at its bound and repeat the one-dimensional search from this point with respect to the remaining variables. To achieve this, the *k*th component of **S** is set to zero. If other constraints are violated and a feasible design is being sought, the one-dimensional search is restarted from this point. If no constraints are currently violated, the scalar product $\nabla F(\mathbf{X})^T\mathbf{S}$ is calculated. If this product is negative, the objective is assumed to be decreasing for a move in this modified direction.

This is an assumption, since $\nabla F(\mathbf{X})$ has not been reevaluated for the current **X**. If the objective is, in fact, increasing in this new direction, the resulting α^* will be zero.

The advantage of this modification is that, on designs where many side constraints are active, the one-dimensional search can continue without the necessity of calculating new gradient information, usually a time-consuming process. For cases where this modification applies, it often considerably improves the efficiency of the optimization process and is recommended when using this and other direct search algorithms.

An alternative here would be to simply set any X_i to its bound if, during the one-dimensional search, that bound is violated. This usually works well in practice, even though the objective and constraints now have discontinuous first derivatives with respect to α.

Truss Configuration Optimization

The second part of my doctoral research was configuration optimization of trusses, where the design variables included the member cross-section areas as well as the joint coordinates. I wrote an analysis code using the force method (as opposed to the standard displacement method used today). The reason for the choice of analysis method was that trusses tend to be almost statically determinate so that re-analysis requires only the solution of the compatibility conditions.

It may be assumed that, with both sizing and shape variables, we just need to include both in the **X**-vector. However, for trusses, this tends to be poorly conditioned. It’s interesting, parenthetically, to note that in those days we focused our research on truss structures. This was based on the assumption that trusses are the simplest structure. Ironically, when considering shape variables, solid structures are more easily solved.

Here, I divided the vector of design variables into two, being geometry and sizing variables so,

$$\mathbf{X} = \begin{Bmatrix} X_G \\ X_S \end{Bmatrix} \tag{28}$$

Now, to calculate the search direction, I used the following feasible direction formulation;

$$\text{Minimize} \qquad \nabla F(\mathbf{X}^0)^T\mathbf{S} \tag{29}$$

Subject to;

$$\nabla \mathbf{g}_j(\mathbf{X}^0)^T \mathbf{S} \leq 0 \qquad j \in J \tag{30}$$

$$\mathbf{S} \text{ bounded} \tag{31}$$

Note the similarity to Eqs. 16 - 19 with the push-off factor set to zero. In other words, we seek a search direction tangent to the active constraints.

Now break the search direction into two parts as,

$$\mathbf{S} = \begin{Bmatrix} S_G \\ S_S \end{Bmatrix} \tag{32}$$

and perform a one-dimensional search in direction S_G.

At each step in the geometry space, solve a sub-optimization problem in terms of the sizing variables. Here, two options were considered. If only stress constraints were included and the structure was statically determinate, the traditional stress ratio method was used. If displacement constraints were present, or if the structure was indeterminate, the sub-optimization task was solved with the feasible directions algorithm. In either case, after the first fixed geometry optimization, we have a very good starting point for the sub-optimization task.

I think this was the first work in multilevel optimization and it later formed the basis of some of my research in multidiscipline optimization.

Summary of Early Work

At Case, I had a bit of a reputation for starting my thesis research in Sept. and having it finished in under six months. The facts are that I had coded both the analysis and optimization algorithms while taking finite element analysis and mathematical programming courses. The separation of design variables was a minor modification to the already written codes.

As an aside, I might note that I had become an avid FORTRAN programmer but it didn’t start that way. AT ASU, I took a structural steel design class from Professor Hill. There were about 15 of us and we routinely stayed up all night on Sundays doing the week’s project. We were all good friends and this was the most competitive class we had together. Lectures often became noisy debates but, due to this extreme competitiveness, everyone excelled and earned a grade of A. To the best of my knowledge, that is the only time Professor Hill (who was generally considered a hard grader) ever gave A’s to an entire class.

During that class, Professor Hill took a week to teach us FORTRAN (three lectures). The university had IBM 1620, GE 225 and CDC 3400 computers. He said, computers are the future in engineering and we should all have basic programming skills. He taught by using examples and, after the first lecture, I commented (rather loudly) that this stuff was incomprehensible. His response was “Vanderplaats, each of these examples was created by my twelve year old son.” After that, I became and enthusiastic FORTRAN programmer and even became his research assistant my final semester. Later, he created a programming course for engineers and told me he always used me as an example to those who thought it was too complicated. Today, I consider myself to be the best FORTRAN II programmer I know (probably the only one still alive).

At Case, if there were contributions in optimization algorithms they were the creation of the variable push-off factor and the method of dealing with infeasible designs. In the structural synthesis arena the separation of the design space into two parts was new. This work was published in Refs. 9 - 11. Ref. 12 was a short study that demonstrated that the quadratic constraint on the search direction provided not only a better search direction but required much less CPU time. Ref. 13 provided a generalization of the multi-level approach as an optimization algorithm.

THE NASA YEARS

1971 was just a few years after the cancellation of the Boeing SST as well as a general decline in opportunities for engineers with Doctorate degrees. Therefore, I was very fortunate to have two opportunities. One was to work with Drs. Venkayya and Berke at Wright Patterson Air Force Base in Ohio and the other was at NASA Ames Research Center in Mountain View, California. As a native Californian, the decision was easy. Also, by that time I had developed a healthy competition with my friends at WPAFB since I was from the Math Programming community and they were developing Optimality Criteria methods.

On my way to California, I stopped at Arizona State University to thank Professor Hill for all his help and guidance and then went on to UCLA to thank Lucien. I showed up, unannounced, and knocked on his door around 3:00 P.M. I might note that, while at Case, we had an arm's length Professor - Student relationship. Now, he enthusiastically invited me into his office. After a few minutes of small talk, he excused himself and called his wife. He said something to the effect "Eli, guess who's here..." When he hung up, he invited me to their home for dinner and I ended up staying the night. I was no longer a student but a colleague. Over time, we became close and shared many research ideas as well as personal views.

When I was hired, NASA was having a Reduction in Force (RIF) so I was actually hired by the U.S. Army Air Mobility Lab. I transferred to NASA a year or two later.

I was hired for my expertise in structures and optimization. When I arrived, my new boss gave me a report on the AESOP optimization program [14]. This was the result of a project funded by his best friend and he was very proud of it (as I soon learned). I studied it and told him it was nice work but really 1950s technology, and there were much better methods available. He concluded that I really didn't understand optimization and he ordered me not to write my own optimizer as it was unnecessary.

Our branch was called the Advanced Vehicle Concepts Branch and it was an outgrowth of a previous similar branch that had written a program called ACSYNT. The idea of ACSYNT was that it would include semi-analytic analysis covering all parts of the design process; geometry, trajectory, aerodynamics, propulsion, structures and mass properties.

My job was structures and mass properties. The structures module modelled wings and fuselages as beams and did a simple fully stressed design. It included composites based on a demonstrably incorrect failure criteria. I had gotten the thirty volume composites documents from Wright Patterson Air Force Base and had studied composites in some detail. I even wrote a program I called COMAND (Composite Analysis and Design) which optimized panels for minimum weight, including thickness and fibre orientation [15]. When I pointed out the error, the branch chief concluded that, not only do I not know optimization but I don't know structural design either. Fortunately for me, even then, it was almost impossible to fire a government employee. Note that these comments are not intended to be negative toward the person. He was a fine man and always allowed me to argue my case. There were about 6 engineers in the branch, each with his own specialty.

When I arrived at NASA, we had a IBM 360 computer in house and access to CDC machines at Lawrance Livermore Labs. Those were the punched card days and the computer lab was upstairs, complete with an attractive operator. When I arrived, I learned the definition of the aspect ratio of a wing, bypass ratios on engines, etc., all the aerospace terms that I had never been introduced to. Also, I met a guy who worked in the wind tunnels and we became friends so I could wander around out there. I found two pieces of equipment that were useful. One was a teletype, which I was able to connect to the computer. Others around me weren't that excited about the noise, but it was otherwise great. The other thing I found was a dictating machine. I was writing papers and learned to use the machine and have the secretary type the papers. That made a big improvement on my writing style, which became more conversational. I learned that I, as well as most engineers, tend to write sentences many lines long. However, we don't speak this way and dictating makes the technical writing much more conversational.

Shortly after my arrival at NASA, Texas Instruments came out with a "portable" computer modem. It weighed 35 pounds, was like a suitcase, had a thermal printer and a 300 baud modem. I got one and used it in place of my teletype, much to my office mates' delight. Also, I packed it home and spend evenings and weekends writing an optimizer, based on my dissertation. I had many parts, but it needed a lot of clean up. Also, I needed to incorporate what I'd learned my first time through writing the feasible directions algorithm.

About this time, another engineer was studying Remotely Piloted Vehicles (RPVs, today called drones or UAVs). He had a version of the ACSYNT program that would evaluate the performance of a conventional configuration as well as a flying (oblique) wing. I asked if I could couple his code to my optimizer and he agreed. The next morning, when he arrived at work, I was waiting with two optimized designs. He was quite excited and said he would show it to the branch chief. I persuaded him to wait. I have a student intern working for me and I had her couple his code with the AESOP optimizer. It took about a month and produced mediocre results (to my delight). He then showed my results to the boss, who responded that we had a program called AESOP which should be used. The engineer then pulled out the AESOP results. Soon after that, the branch chief came to my office and said "Well your optimizer seems to work, at least for this problem." My optimizer was called CONMIN (CONstrained function MINimization) [16] and is used by several of our competitors even now.

Everyone was becoming quite frustrated with the ACSYNT program that had been handed down to us. They knew by then that I was fearless in talking to the boss and asked me to go to him and demand that we be allowed to start over and write the program from the ground up. I dutifully did so and, to his credit, he called a meeting. He explained that I had registered their concerns and wanted to write the program over and he asked for other people's input. DEAD SILENCE! I was shocked and hurt and apologized profusely for making such a fool of myself.

Later, the others came to me one by one and explained that I was single and could mouth off all I want because I didn't have their obligations, but they couldn't afford to be laid off in the next reduction in force (RIF).

We continued for perhaps another year with the old program and the others again asked me to speak to the branch chief. I impolitely declined. This time, they went as a group. He agreed and appointed me to be the ACSYNT Manager!

We re-wrote the ACSYNT program in a modular form. Figure 7 shows the general organization of the program showing the various modules. The process begins with the basic layout of the aircraft and an estimate of its gross take-off weight. The modules are then cycled through. During the Mission Analysis, the aircraft numerically “Flys” the mission, calling aerodynamics and propulsion to get the lift, drag, sfc, etc. needed to estimate the fuel usage. At the end, all weights are added and compared to the estimate. If they do not match, a new estimate is made and the process is repeated to convergence. This is called a “Point Design.”

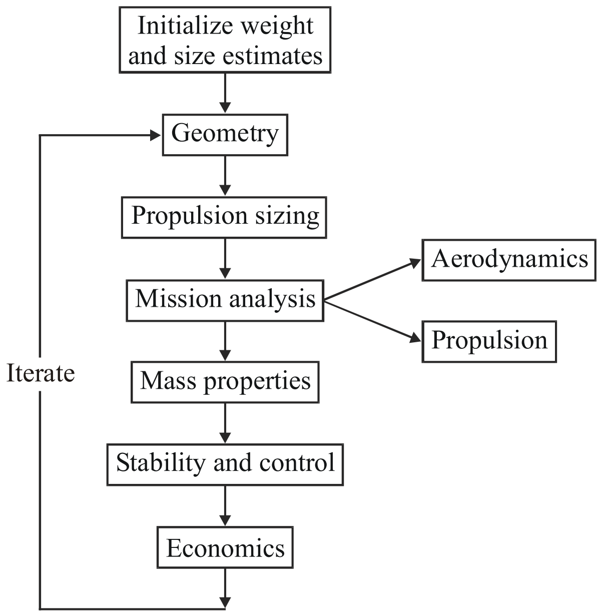

Figure 7. ACSYNT Program Organization.

Figure 8 shows a modern interpretation of the ACSYNT program using our VisualDOC software[17] Version of ACSYNT.

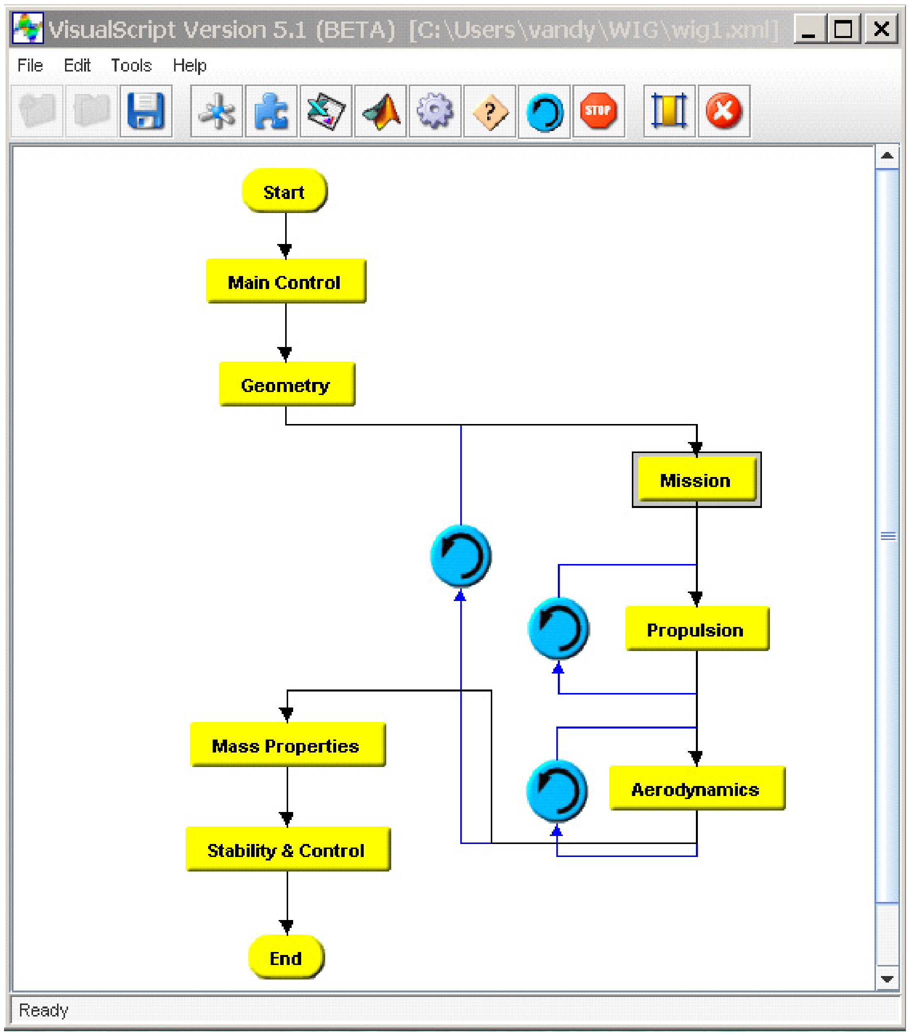

Figure 8. Aircraft Synthesis.

In re-writing the program, we began by identifying the input variables and output responses for each of the program modules. The engineer in charge of a given module was instructed to write his/her module however they chose. The only rule was that the module be broken into Input, Execution and Output phases. The input variables and output responses were stored in a single array in a named common block. In this way, the engineer could write a very simple main program to test his module, independently of the others.

I wrote a program called CATLOG which transferred data from each module to a global array and perform the iterative process to converge to an estimated gross take-off weight. This provided a simple mechanism to test individual modules. For example, during the input phase, the geometry module would define parameters such as wing area, aspect ratio, sweep, etc. However, during execution and output phases, these parameters were not to be changed. I simply wrote some routines to test to be sure the module developers did not break these rules when we put everything together. This dramatically reduced data transfer problems and simplified program debugging.

Some important lessons were learned in the process of writing the new program. The one mentioned above, where each developer did his/her work independently, greatly simplified the process of integrating the various modules. It had the added (and surprising) benefit of raising morale because no one was telling them how to write their module. I've often used this lesson as I tell people that, at VR&D, I always try to hire the very best and then stay out of their way.

Another lesson was in the choice of design variables. Aircraft designers like to work with wing loading, W/S, as a main variable. By treating this as a design variable, the geometry module would use the current estimated aircraft weight and the wing loading to define the wing area, S. However, when coupled with an opimizer, I soon learned that treating the wing area, not wing loading, as a design variable produced a much better conditioned optimization task. At the optimum, the wing loading is the same. It is just much easier to get there with the proper choice of design variables.

Also, the mass properties module is very fast, being a series of equations for component masses based on historical data. The individual masses are added to the fuel and structural mass to estimate the calculated gross weight. We found that, by iterating on mass properties several times, the overall convergence rate was improved.

Finally, being the self appointed optimization expert, CONMIN was a key part of the new program. Recognizing that we were performing an internal loop to make the estimated and calculated gross weights the same, it occurred to me that this could easily be avoided. Simply treat the estimated gross weight as a design variable and constrain the calculated weight to be equal to the estimated weight. Actually, if we are minimizing the gross weight, we can require that the calculated weight be lower than or equal to the estimated weight. This makes it an inequality constraint that is easier to deal with and which is always active at the optimum.

To further emphasize the importance of problem formulation, consider the remotely piloted oblique wing vehicle shown in Figure 9. The objective is to minimize the gross weight subject to a volume constraint which allows space for fuel and equipment. Only two design variables are considered here, being the thickness-to-chord ratio, t/c, and wing loading, W/S.

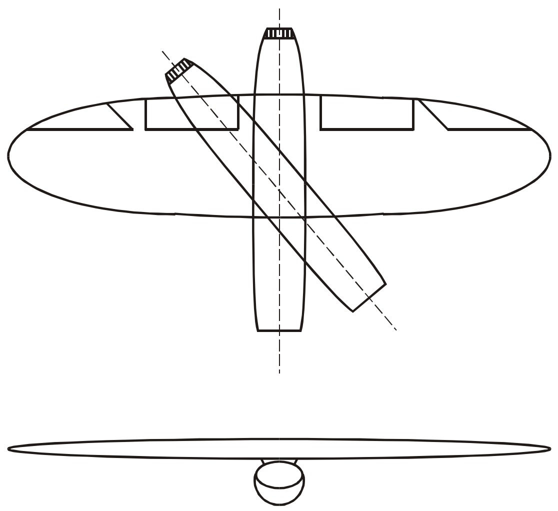

Figure 9. Remotely Piloted Vehicle.

Figure 10 is a plot of the design space in terms of the values of the design variables relative to their optimum values.

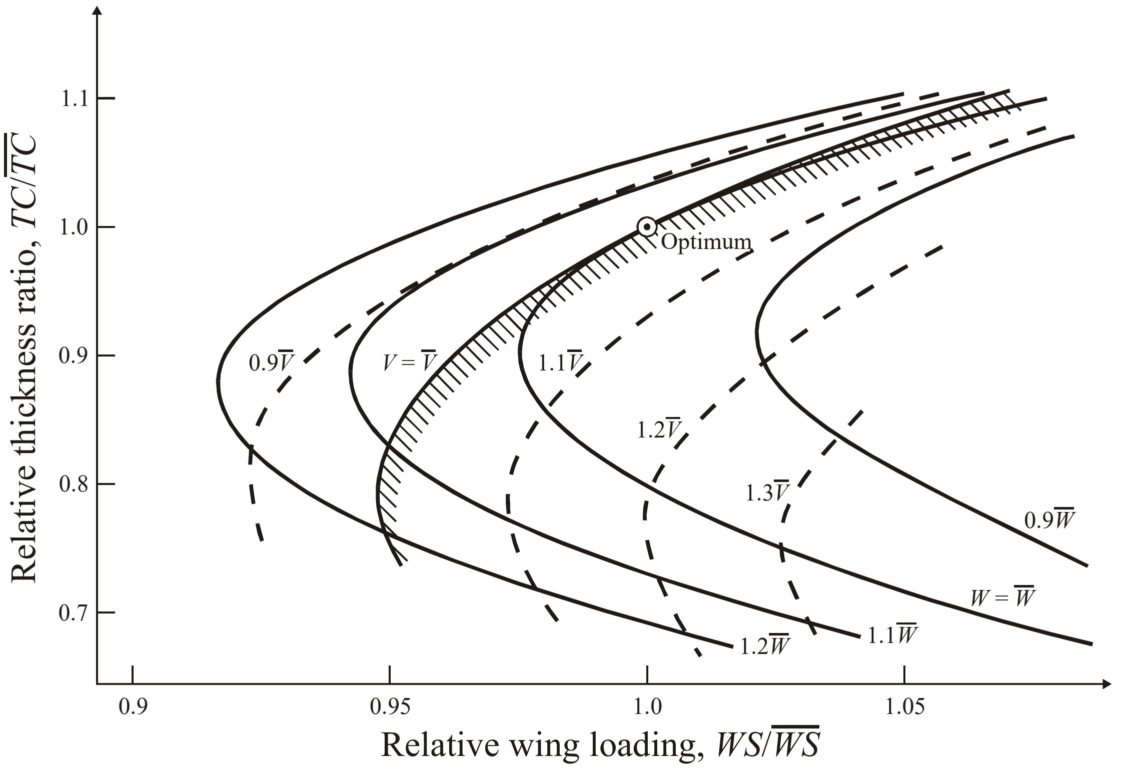

Figure 10. Design Space in Terms of t/c and W/S.

Note that the contours of the objective function and constraint are nearly parallel and that the design space is theoretically non-convex.

Now, consider Figure 11 which is the design space in terms of t/c and the wing area, S. This is the only change in the problem statement. However, now the design space is convex, the optimum is clearly defined and the result is the same.

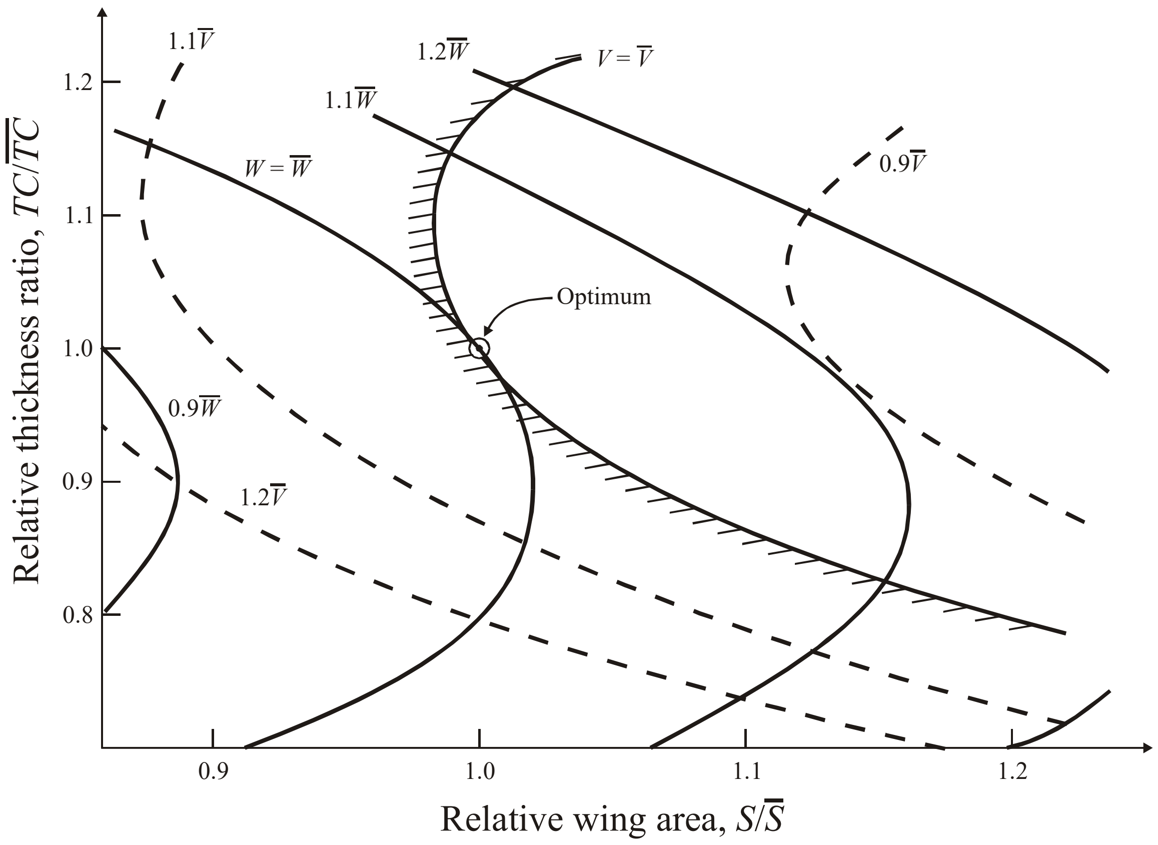

Figure 11. Design Space in Terms of t/c and S.

Over time, we studied Oblique Wing transport aircraft, Hydrogen Fueled Aircraft, Supersonic Cruise Aircraft [18], Remotely Piloted Vehicles and numerous others. During this effort, CONMIN became established and I was routinely sending it to other research labs, universities and companies.

We had an interesting experience when studying the Oblique Wing Aircraft, the brainchild of our Chief Scientist R. T. Jones. This aircraft has a high aspect ratio, straight wing. On take-off and landing, the wing is at right angles to the fuselage. At speed, the wing is rotated, one side forward and the other side aft. Typically, we would fund an aircraft company to do a parallel study. In this case, at the mid-point, they estimated the wing would weight 130,000 pounds for a 260,000 pound aircraft, clearly not feasible. This is due to divergence of the forward swept wing. Divergence is easy to understand by putting your arm forward out a car window at 60 miles per hour. If you're really weak, the wind will push your arm up and, break it. They were studying a ten percent thick wing with an aspect ratio of fourteen. Our optimization study predicted the wing should have a thickness-to-chord ratio of twelve percent and an aspect ratio of twelve. We predicted the wing would weigh about 30,000 pounds, similar to a Boeing 727 aircraft. Their chief engineer said words to the effect that "We won't waste our time on that." When I asked why, he explained that "I'm a Design Engineer and you're just a Research Scientist." At that point, I suggested we could pay half and terminate the study or they could study our design and get full payment. They decided to humor us and study our design. Six months later, they returned with a

30,000 pound wing. Over time, their chief engineer and I became good friends. However, he would never admit that optimization produced this result.

During these years, I learned a great deal about the importance of problem formulation. Finally, the importance of correlating the analysis with existing successful aircraft cannot be over-emphasized. During that time, numerous people attempted aircraft conceptual design using optimization and many concluded optimization was of no value. They universally argued that it gave unrealistic results. I'm convinced that this was a direct result of bad analysis tools. For example, we had a collection of more than a dozen wing weight equations, collected from various experts. They all showed correlations with many aircraft (I still have a large collection of aircraft weight statements for everything from the Sopwith Camel to the Boeing 747). The best of them predicted that, as the aspect ratio of a wing is increased it gets lighter. While this clearly contradicts physics, it is none the less an historical fact. That is because we moved from cloth to panels to stressed skins. Each technology allowed for larger aspect ratios at lighter weight. Using this equation, I optimized a F-5 fighter aircraft with an upper bound on 100 on the aspect ratio. The result was a fighter with an aspect ratio of 100 and negative gross weight (large negative wing weight, according to the equation). This is why we went to great effort to use physics based simplified analysis. Given proper care, a great deal can be achieved at the conceptual level to sort out the many design concepts we encounter in the early design phases [19].

During that time, I also wrote a structural optimization research code I called SAD (Structural Analysis and Design). This didn't go very far, perhaps due to a poor choice of name. I did have a series of National Science Foundation Post Doctoral researchers who used parts of this program in their research. On one occasion, Lucien Schmit visited and he, I and the post-doc went to the Officers Club at Moffett Field for lunch. The post-doc asked Lucien for research ideas, saying "Can you suggest a research topic? Something simple, like the 3-bar truss that will make me famous." Lucien didn't hit him, but I suspect he wanted to. In any case, the post-doc didn't become famous. He became an optometrist, never to be heard from again.

The ACSYNT program was several boxes of punched cards long. We had access to a CDC 3600 mainframe at the Lawrance Livermore Labs and would submit jobs over a 300 baud modem. Each time we changed the optimization task, we had to read the entire program again, a very time consuming process. This led me to create the COPES program (COntrol Program for Engineering Synthesis) [20]. Now, we could store the entire executable program on the system and just execute it by reading the input data. When the optimization task changed (for example, change from minimizing mass to maximizing range), we only had to read a short data deck. In the ACSYNT program, the main common block contained all input and calculated data that needed to be transferred between modules or to and from the optimizer. The input simply pointed to locations in this array to put data or retrieve results. This is the same technique that virtually all multidiscipline design optimization software uses today, though we now do it in a graphical environment.

In the mid-1970s, I was dabbling in airfoil optimization, working with two aerodynamicists at Ames. Our first effort was to model the airfoil using polynomials. The typical problem was to minimize wave drag of a transonic airfoil with limits on cross-sectional volume and pitching moment [21, 22]. The analysis was an inviscid code from Lockheed.

During this period, researchers in structural synthesis were developing two key ideas that would be useful here. The first was the concept of basis vectors by Pickett, Rubinstein and Nelson [23]. Here, the idea is that we can create a new design as a linear combination of existing or proposed designs, so

$$\mathbf{X} = x_1\mathbf{X}^1 + x_2\mathbf{X}^2 + \ldots + x_n\mathbf{X}^n \tag{33}$$

where the x_i are the design variables and $\mathbf{X}^i$ are arrays containing the candidate designs. It occurred to me that there are a large number of published airfoils that could be used as candidate designs. Not being confused by any particular knowledge of the subject, I went into the NACA reference [24] and picked four airfoils that were noticeably different.

Figure 12 shows these airfoils, where the last two shapes are used to adjust the trailing edge thickness.

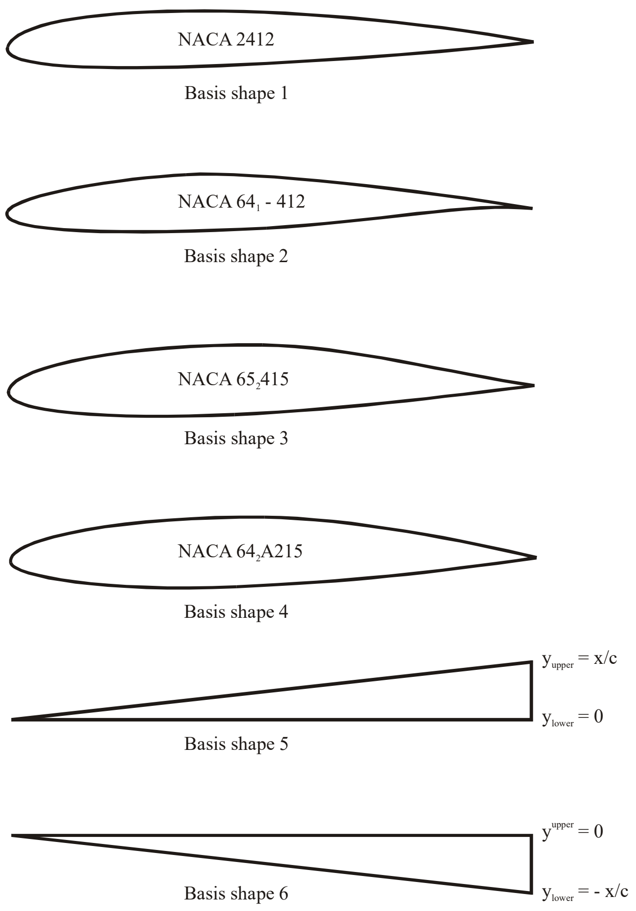

Figure 12. Basis Shapes.

Figure 13 shows the optimum shape where wave drag was minimized with a pitching moment constraint [25]. It's fascinating to note that the optimum airfoil looks nothing like the basis airfoils used to create it. This is because the optimizer said it's just fine to use airfoil #1 *minus* airfoil #2 plus one half of airfoil #3, etc. It is also fascinating to note that, at this time, Dr. Whitcomb at NASA Langley was developing his super critical airfoils. This airfoil is virtually identical to his GAW-2 airfoil.

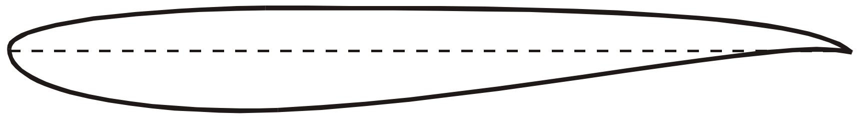

Figure 13. Optimum Airfoil.

The second key concept from the structural synthesis community was approximation concepts developed by Schmit and Farshi [26]. Here, the idea is to study the physics of the problem and create approximations that model the physics, instead of just blind linearization. Here, for a truss structure, the stress in a rod is defined as

$$\sigma = \frac{F}{A} \tag{34}$$

where F is the force in the member and A is the cross-sectional area. Normally, we treat A as the design variable. Now, they said to let the design variable be the reciprocal of A.

$$X = \frac{1}{A} \tag{35}$$

so

$$\sigma = FX \tag{36}$$

Now, if we linearize in terms of X, the approximation is exact for statically determinate structures and is an excellent approximation for statically indeterminate structures. This led to our being able to optimize structures using a fraction of the finite element analyses need without these approximations. Indeed, this “simple” concept is what makes structural synthesis feasible today.

In aerodynamics, we did not have similar physical interpretations but it occurred to me that the approximation concepts idea may be valid here. The airfoil problem was defined by only a view variables but a very expensive analysis. This led to the idea of creating several candidate designs and analyzing them. Then fit curves to this data and optimize using the curves. In this case, I actually started with one design and just perturbed it. This goes back to the concept defined earlier that my philosophy in optimization is to model the thought process of a good design engineer. Now, with just two designs, we can create a (very crude) approximation to the gradients of the objective and constraint functions. We then optimize using this approximation with move limits. That leads to a new, hopefully improved, candidate design which we analyze and add to the data set. We repeat this process until we have n+1 data points, at which we can estimate relatively good first order gradients. I say “relatively good” because the designs may be pretty far apart (as opposed to taking small finite difference steps). Once we have the first order approximation, we use additional designs to create second derivatives, calculating the off-diagonals of the Hessian matrix first and then the diagonal terms. We create the off-diagonal terms first because they are the coupling terms. The diagonals are just the curvature with respect to the individual variables. When we have more data than needed to create the second order approximation, we do a least squares fit, rather than going to higher order approximations.

Consider a Taylor series expansion

$$F(\mathbf{X}) \approx F(\mathbf{X}^0) + \nabla \mathbf{F}(\mathbf{X}^0)^T \delta \mathbf{X} + \frac{1}{2} \delta \mathbf{X}^T [\mathbf{H}(\mathbf{X}^0)] \delta \mathbf{X} + \ldots \tag{37}$$

where $F(\mathbf{X})$ is taken to be any objective or constraint function, $\mathbf{X}^0$ is the point at which we create the approximation, $\nabla \mathbf{F}(\mathbf{X}^0)$ is the gradient, and $\mathbf{H}(\mathbf{X}^0)$ is the Hessian matrix. The perturbation vector $\delta\mathbf{X}$ is

$$\delta \mathbf{X} = \mathbf{X} - \mathbf{X}^0 \tag{38}$$

If we use only the first two terms on the right-hand side of Eq. 37, we create a linear approximation to the problem. If we solve this, reanalyze and solve again repetitively, we have the Sequential Linear Programming method. Adding the third term gives a second-order expansion. We wish to use these approximations for both the objective and constraint functions. Assuming the **H** matrix is available for the objective function and $\nabla \mathbf{F}$ is available for each constraint, this becomes a sequential "quadratic" programming problem. This may be solved using formal quadratic programming methods or by any of the methods available for constrained optimization. The distinction here is that a complete optimization is performed using Eq. 37, rather than updating the gradient information at each step during the optimization.

When updating our approximation, it may not always be necessary to update all terms. If we update only the constant term, this requires one function evaluation. The linear term requires an additional *n* function evaluation if finite-difference gradients are used. Finally, calculation of the symmetric Hessian matrix requires an additional *n*(*n*+1)/2 function evaluations. Therefore, depending on the ease with which the information can be obtained, we may wish to perform several approximation cycles only updating the constant term. Then we update the linear terms and repeat, updating the Hessian matrix only when no progress can be made otherwise.

Following this line of reasoning, it may be desirable to begin with only a small amount of information and create the best approximation possible based on this. We then optimize using this approximation, with reasonable move limits on the design variables. This result is analyzed precisely and the corresponding function values used to improve the approximation. The important concept here is that we need not use small finite-difference steps to approximate $\nabla F(\mathbf{X})$ and $\mathbf{H}(\mathbf{X})$ but may instead use whatever design information we have at a given time. To clarify this, expand Eq. 37 using only the linear and quadratic terms shown

$$\begin{aligned} \Delta F = {} & \nabla F_1 \delta X_1 + \nabla F_2 \delta X_2 + \ldots + \nabla F_n \delta X_n \\ & + \frac{1}{2}(H_{11}\delta X_1^2 + H_{22}\delta X_2^2 + \ldots + H_{nn}\delta X_n^2) \\ & + H_{12}\delta X_1 \delta X_2 + H_{13}\delta X_1 \delta X_3 + \ldots + H_{1n}\delta X_1 \delta X_n \\ & + H_{23}\delta X_2 \delta X_3 + \ldots + H_{n-1,n}\delta X_{n-1}\delta X_n \end{aligned} \tag{39}$$

where for brevity

$$\Delta F = F(\mathbf{X}) - F(\mathbf{X}^0) \equiv F - F^0 \tag{40}$$

$$\nabla F_i = \nabla F_i(\mathbf{X}^0) \qquad H_{ij} = H_{ij}(\mathbf{X}^0) \tag{41}$$

Now assume a "nominal" design $\mathbf{X}^0$ has been analyzed to yield F^0. Also, numerous other designs, $\mathbf{X}^1$, $\mathbf{X}^2$,..., $\mathbf{X}^k$ have been analyzed to give F^1, F^2,..., F^k.
Let

$$\delta\mathbf{X}^i = \mathbf{X}^i - \mathbf{X}^0 \qquad i = 1, k \tag{42}$$

and

$$\Delta F^i = F^i - F^0 \qquad i = 1, k \tag{43}$$

Now we can write k equations in the form of Eq. 39. The unknowns are ∇F_1, ∇F_2, ..., ∇F_n and H_{11}, H_{12}, ...,H_{nn} for a total of $l = n + n(n+1)/2$ unknowns. Remembering that one analysis is required for the nominal design, a total of $l + 1$ designs is required. Thus if $k \geq \ell + 1$, the unknowns can be determined.

Equation 39 is linear in the unknown coefficients. Writing this equation for each of the k designs yields equations which can be solved directly. If $k \geq \ell$, a weighted least-squares fit can be used in which those designs with the smallest magnitude $|\delta\mathbf{X}^i|$ (designs nearest the nominal) are weighted most heavily. If less than l designs are used, fewer coefficients are calculated. In general, it is desirable (but not mandatory) that $k \geq n + 1$ initial designs be specified to provide at least an initial linear approximation to each function. Also, as long as $k \leq \ell$, the designs must be linearly independent to avoid a singular matrix in the solution for the coefficients. If all designs are very close to $\mathbf{X}^0$, Eq. 39 gives the finite difference form of a second-order Taylor series expansion. If the designs are more randomly spaced throughout the design space, this is in fact not a Taylor series approximation but is simply a quadratic polynomial approximation to the design.

Note that if we have analytic first derivatives, $\nabla F(\mathbf{X})$, the components, ∇F_i, of these may be substituted into Eq. 39. These components may now be multiplied by their corresponding δX_i to yield constant terms which are moved to the other side of the equation. This reduces the number of function evaluations needed by n. Now, letting

$$\nabla F^i(\mathbf{X}) = \nabla F(\mathbf{X}^i) - \nabla F(\mathbf{X}^0) \tag{44}$$

$$\begin{aligned} \Delta F_i - (\nabla F^i)^T \delta X^i = {} & \\ & \tfrac{1}{2}[H_{11}(\delta X_1^i)^2 + H_{22}(\delta X_2^i)^2 + \ldots + H_{nn}(\delta X_n^i)^2] \\ & + H_{12}\delta X_1^i \delta X_2^i + H_{13}\delta X_1^i \delta X_3^i + \ldots + H_{1n}\delta X_1^i \delta X_n^i \\ & + H_{23}\delta X_2^i \delta X_3^i + \ldots + H_{n-1,n}\delta X_{n-1}^i \delta X_n^i \end{aligned} \tag{45}$$

Equation 45 can be used in place of Eq. 39 for each design where we have both function values and gradients. If gradients are available at every design, the total number of needed equations will be reduced from $k = 1 + n + n(n+1)/2$ to $k = n(n+1)/2$. While this may not appear to be a major reduction, the quality of the approximation will probably be improved by including the gradient information in the calculations.

It is important to recognize that the quadratic approximation given here is not the only possible approach. Any curve fits can be used which reasonably approximate the functions. Also, the functions need not be defined analytically or numerically. We could use experimental data, using approximate optimization to identify an improved proposed design to test experimentally. In this fashion, optimization can be used to dramatically reduce the number of test points in an experimental study.

For example, Consider the minimization of a simple linear function subject to one nonlinear constraint

Minimize: $$F(\mathbf{X}) = X_1 + X_2 \tag{46}$$

Subject to:

$$g = \frac{1}{X_1} + \frac{1}{X_2} - 2 \le 0 \tag{47}$$

$$0.1 \le X_i \le 5.0 \qquad i = 1, 2 \tag{48}$$

The design space for this problem is shown in Figure 14, where the optimum is clearly at $X_1 = X_2 = 1.0$.

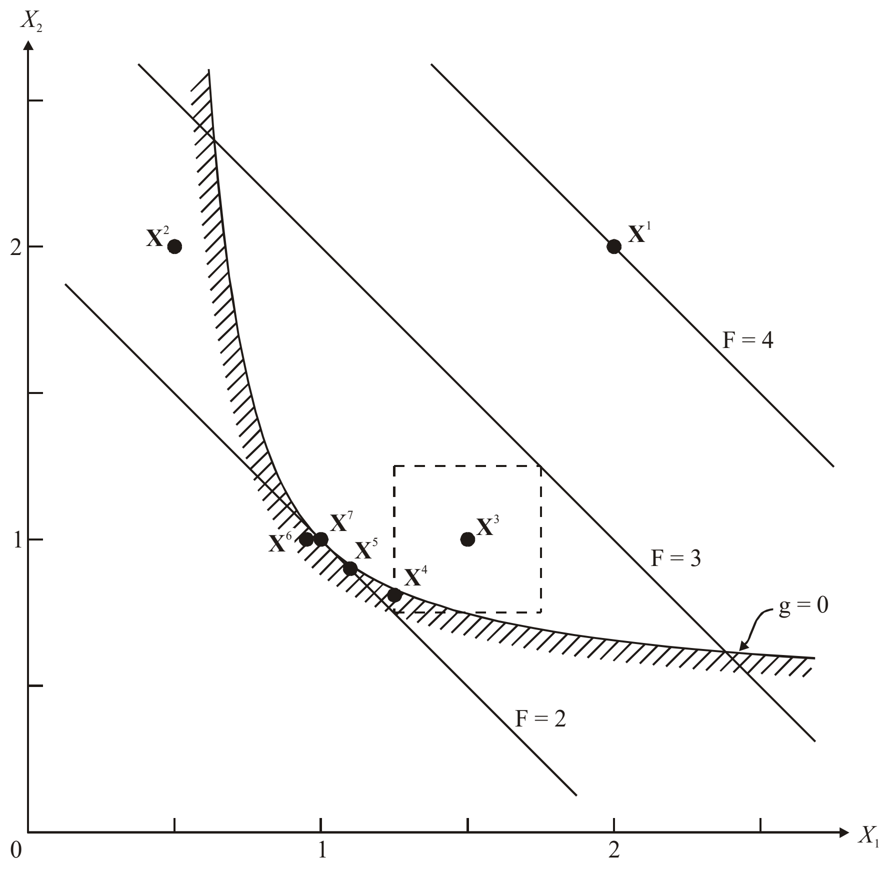

Figure 14. Sequential approximate optimization.

Starting at $X_1 = X_2 = 2.0$, the sequence of designs is shown in the figure. In this example, we began with the first three designs and created a linear approximation at $\mathbf{X}^3$. This point was chosen because it is the best available design. Now, move limits are included and the sequence of design is as shown. Note that the optimum was achieved using only seven analyses. An interesting feature of building the approximation as we go, instead of creating the full second order approximation at the beginning, is that we consistently make progress while gathering information. Thus, by the time a full second order approximation is available, we're already in the vicinity of the optimum. We could have just started with one design and perturbed it to create a crude first order approximation. In practice, this is my preferred approach.

When I proposed this idea to the aerodynamicists, they rejected it out of hand. Their response was that, as a structures guy, I simply did not understand how nonlinear aerodynamics is. This has been a recurring experience in my career when very talented analysts decide they are designers. Analysts brag that they solve more nonlinear problems than their competitors. Designers don't like nonlinearities in "design space" because it makes the design sensitive to imperfections or poor quality manufacturing. In other words, if we need to solve the Navier Stokes equations when we perform an analysis, that's fine. However, when we formulate the design problem, we want to choose our variables to make the design task easy and well defined and so the results are predictable if small changes are made.

This combination of basis vectors and approximations dramatically improved the efficiency of the optimization process as well as leading to much better results [27, 28]. I simply called it an approximation method using curve fits. Today, it's called response surface approximations and there are a myriad of methods that have been published. My own preference is for quadratic approximations and I've seen very little to suggest they are not the best overall approach in creating reliable optimization software.

During this period, I took a year leave from NASA and was an Adjunct Professor at the Naval Postgraduate School in Monterey, California. I thought that was a great title until I looked up the definition and found it meant "unnecessary" or "nonessential"! There, I did some research on optimizing an aircraft canopy to shoot a laser beam with minimum phase distortion. I learned about Zernike Polynomials but otherwise not much. Also, I wrote a couple of papers on windmill optimization. Finally, I created a new course on engineering optimization.

From there, I spent a year in Washington DC at the Naval Ship R&D Center. There, I continued ACSYNT development and worked on Vertical and Short Take-off and Landing aircraft (VSTOL). Also, I did a bit of aerodynamic optimization with the local aerodynamicist.

Upon return to NASA Ames, I was in a new branch, but still doing conceptual aircraft design. About then, the aerodynamicists were proposing their approach to aerodynamic optimization. One of their key points was that finite difference gradients were not useful because they gave bad gradients. Thus, they argued, numerical optimization was probably not useful.

Their key mistake was the use of very small finite difference steps (remember they think like mathematicians). The problem is that the aerodynamic analysis is iterative, with its own convergence criteria. Therefore, the analysis is noisy and small finite difference steps are meaningless. That is why the large steps used by the curve fit approach worked well. Since they rejected that approach, I told my branch chief that I preferred to bow out and not attend the meeting he'd asked me to attend with them and the Deputy Director. His response was "You don't have that right. If you have an idea, you have an obligation to present it." And so I did, leaving a document outlining my ideas with the Deputy Director.

At that time, the business class aircraft industry was proposing that NASA create a new series of transonic airfoils, complete with wind tunnel verification. My proposal was to optimize a design using the approximation techniques that they would build and we would test. I also made a strong argument that we must not design for just the cruise condition but must consider the entire flight envelop. I didn't hear for some time but later got the document back with a note to all of us to pursue my ideas together.

At this time (1979), I had already decided to move to the Naval Postgraduate School in Monterey as an Associate Professor. I didn't have to confront the task of trying to work with people who rejected my ideas. I later learned that they designed an airfoil for transonic cruise and tested it with good wind tunnel results. The company built a wing and put it on their test aircraft, only to learn that the stall speed was so high they couldn't take off from their runway. I felt vindicated that design must consider the full flight envelop, not just focus on efficient cruise.

At the Naval Postgraduate School (NPS), I taught the optimization course that I had created when I was an adjunct there. Also, I taught Machine Design, which I thoroughly enjoyed. I taught continuum mechanics, which I had never studied but managed to stay a day ahead of the students. I had started writing a book some years earlier and now worked on that in ernest. I made a Heathkit PC and used it in my laundry room to write the book. My wife, Ginny, proofread it for spelling and grammar.

At NPS, my research was mainly funded by NASA Langley Research Center and I was writing a new optimization program with numerous algorithms. This was a good match to the text because I coded all of the algorithms I described in the book. The program was called ADS (Automated Design Synthesis) and the user could pick many combinations of strategy, optimizer and one-dimensional search [29]. Today, like CONMIN, ADS is used by our competitors.

When I studied nonlinear programming, I was taught that Sequential Linear Programming (SLP) was a terrible algorithm that should never be used. When I coded it, I soon realized that an adjustable move limit feature was needed. When I did that, I found that the algorithm was surprisingly efficient and reliable, particularly if the optimum was fully constrained (as many active constraints as design variables). Also, because SLP does not use a one-dimensional search, it parallelizes with near theoretical speedup.

When I wrote the CONMIN program, I used Zoutendijk's method of feasible directions, with the modifications I had created during my dissertation research. This method uses "push-off" factors to push the design away from the constraint boundaries. Another algorithm known as the Generalized Reduced Gradient (GRG) method found a search direction tangent to the constraint boundaries [30]. Then, during the one-dimensional search, the problem was broken into independent and dependent variables. As the independent variables were changed, the dependent variables were updated as a sub-problem to drive the design to the constraint boundaries. The number of dependent variables equaled the number of active constraints, so this just required the solution of a set of linear equations. I observed that the GRG search direction could easily be calculated by simply setting the push-off factors to zero in the method of feasible directions. Also, it was not necessary to break the problem into independent and dependent variables. Instead, I searched in terms of all variables. At each step, I solved a sub-problem to drive the design back to the constraint boundaries. This sub-problem required minimizing the distance back to the constraints. This sub-problem has a quadratic objective function with linear equality constraints. However, with some slight of hand, utilizing the Kuhn-Tucker conditions, this becomes a simple problem of solving simultaneous equations. I called this algorithm the Modified Method of Feasible Directions (MMFD) [31] and it became the first algorithm in my first commercial optimization program, DOT (Design Optimization Tools) [32]. The second algorithm in DOT was the SLP method.

Another interesting algorithm developed during that time was Sequential Quadratic Programming (SQP) [33]. This method creates a quadratic approximation to the objective function with linear approximations to the constraints. It uses a sub-problem that is solved with quadratic programming. The difficulty with the quadratic programming algorithm is that, if there is no feasible solution, it simply dies. Various techniques are used to deal with this situation but I found that simply calling the MMFD algorithm works reliably. This is because the MMFD algorithm will find a solution as nearly feasible as possible. SQP became the third algorithm in the DOT program.

In 1984, I moved the University of California at Santa Barbara (UCSB) as a full professor. This was poetic justice because as a high school graduate, I was not eligible to be admitted to the University of California (UC) system.

In high school, I always worked just hard enough to be eligible for UC, which was a 3.0 grade point average (GPA). As the youngest of seven kids, the others had all been high achievers in high school. My first day, the little old lady teaching algebra said “Oh, you’re a Vanderplaats. I hope you’re as smart as your brothers and sisters.” I told her I intended to flunk algebra and, if I hadn’t been a Vanderplaats, I’m sure I would have. I think she gave me a C. One brother was an outstanding track star. The other was an outstanding trombone player. A sister was a star singer, and so goes the list. I committed myself to cars, girls and basketball. Mediocre at basketball, very successful at autos. Girls seemed to like me because I was as shy as a stump and they considered me harmless. My main problem there was working up the courage to ask out a popular girl (though I finally did date the homecoming queen!).

My last semester in high school, I was doing fine but got a B in PE (yes, that is possible). I was not a favorite of the coach and I’ve always suspected that he looked at my transcripts and calculated that he could keep me out of UC by giving me a B. Anyway, I had no money so I stayed with my friend, Jerry Woodring, and went to a nearby Junior College. The first semester, I got a 1.15 GPA on a 4 point system. I then moved to my brother’s house and went to a different Junior College. Over the next two years, I worked my way up to a 2.37 GPA and applied to UC Davis as a transfer student. I received a polite letter saying that, if I had a 3.0 GPA in high school, I would only need a 2.0 to transfer. However, because I did not have a 3.0 in high school, I would need a 2.4 to transfer.

At that point, I went to the Library and looked at college catalogs. Starting with the A’s, I found Alaska, Alabama, Arkansas, and Arizona (closest). The University of Arizona at Tucson was further than Arizona State University in Tempe, so ASU got the nod. I could drive the 700 miles to ASU in a day and they only required a 2.0 GPA for admission. The choice of ASU is pretty much the way I have always made decisions. I’m a lot like my dog. If I see a rabbit, I chase it, often neglecting to think of what I'll do if I catch it. Fortunately, at ASU, I grew up a bit, graduating Magna Cum Something or Other and continuing on to the PhD.

At UCSB, I taught the only required undergraduate optimization course in the country to about 150 students each year. I also taught graduate courses in Finite Element Analysis, General Optimization and Structural Optimization. In 1984, my book was published by McGraw-Hill. Also, I started the company, Engineering Design Optimization (EDO). The purpose of the company was to create an optimization Users Group and we would meet each summer at a hotel on the beach. One day, I got a letter from a company named EDO saying they optimize everything so my use of EDO violated their trademark. I spoke to an attorney and he said I could fight and win but it would probably cost thousands of dollars. I decided to switch instead and changed the name to Vanderplaats, Miura & Associates (VMA). Some years later, I created Vanderplaats Research and Development, Inc. (VR&D) and abolished VMA.

At UCSB, most of my research was funded by NASA Langley and NASA Lewis.

During my time in aerospace, engineers generally talked about internal forces in structures while I thought in terms of stress. The approximation techniques developed by Professor Schmit used stress approximations. These worked well for rods and membrane structures but not for beams, plates and other elements. This is because, for rod and membrane elements, the element stiffness matrix is the area or thickness times other information. For elements such as beams, this is not the case so the approximation is not as simple. It dawned on me one day that aerospace engineers are actually pretty smart so I tried to approximate element forces instead of stresses [34]. This worked surprisingly well and applied to virtually any element. A student, Scott Hansen, was working on a Master's Degree with me and looking for a topic. I suggested he use force approximations and solve some examples. The MS degree did not require original research and this was a quite adequate topic. He solved a lot of problems in a week or so and came asking for more. I suggested he try configuration variables and solve the problems I had solved on my dissertation. He did so very quickly and demonstrated remarkable efficiencies [35]. Having demonstrated that it works, we then studied it more closely and were able to show that, while it was not as clean as for member sizing, it did uncouple the problem, leading to its efficiency. I later learned that Bofang in China had published the idea earlier [36]. I had visited China in 1984 and may have gotten the idea from him so I want to give him credit here. Upon graduation, Scott came to work for our company. He later left and attended law school. He is now our intellectual property attorney.

When I sent our paper on force approximations to a journal for publication, a reviewer declined publication saying the idea was too simple to merit publication. Today, it's a standard method in our GENESIS structural optimization software [37] as well as that of most competitors.

In 1984, a friend had a consulting job to analyze a submarine gear housing. He performed the analysis and then went on to couple the CONMIN program to NASTRAN and optimize the structure [38]. He asked me to attend the MSC (MacNeal-Schwendler) conference and present the results and I arrived in Pasadena the evening before my presentation. Each evening they had a hospitality party with food and drinks. When I arrived, Dick MacNeal saw me from across the room and literally ran over to me. He said he had read the paper and wanted us to put optimization in NASTRAN. I had known him for years and he had always joked that optimization was nothing but a research toy for professors. His turn about amazed me until I had time to think it over. In those days, people paid for their NASTRAN license by the CPU seconds used. He realized that we had just called NASTRAN six times!

In 1986, MSC hired us to add optimization to NASTRAN and that became Solution 200. We used the approximation techniques developed by Schmit because the structure of the NASTRAN program made it almost impossible to use the force approximations I was working on. The new capability was released in 1989. Initially, the optimizer was ADS but later, we licensed DOT to them. Many years later, they decided they didn't like to pay for DOT and so reverted to ADS, which they continue to use.

By 1987, I was growing bored with academic life. I had pretty much proven whatever I wanted to prove, at least to myself. I'd written a book, gotten teaching awards, published numerous research papers, etc. I felt that I needed a new challenge so, in a moment of temporary insanity, I walked away from my nice lifetime tenure with an ocean view office to devote full time to the company. This had actually been coming for some time because I was quite active as a consultant to various major companies and several were encouraging me to commercialize this technology. Several promised that I would have more business than I could handle but most were hard to find after I made the move. Indeed, one company research department cancelled my consulting agreement of many years, explaining that I was now their competition. This led to some very lean years but I've never regretted the move. Today, I can say with confidence that we have the best software in the optimization world and the best staff of experts to support it. I often say I never expected to get rich and I haven't been disappointed.

In 1987, I released our first commercial optimizer, DOT [32]. This optimizer was written to be maintainable and expandable as I continued research in algorithms. I have added or modified numerous aspects of DOT and it is now at Version 7.

In 1990, we released the DOC (Design Optimization Control) program [39]. This was a commercial version of the technology I had developed for the COPES program [20] and was a batch program. In the mid-1990s, we received a Small Business Innovative Research grant (SBIR) from Wright Patterson Air Force Base to modernize this. The resulting program is VisualDOC [17] and included a companion program we called Visual Script. This was a graphics based program with all of the features of DOC. Since then, VisualDOC has been continually enhanced so, today, it contains a multitude of advanced features for Multidiscipline Design Optimization.

In 1989, we decided to create our own structural optimization code using our latest research. The idea was to create an OK analysis with exceptional optimization capabilities. It was to be a companion to NASTRAN and not a competitor. Most of the funding came from Toyota and, in the three years it took to create Version 1, Toyota continually added things they wanted. As a result, the first version, with statics and normal modes analysis and sizing and shape optimization, was quite competitive with NASTRAN. Over the years, we've added a multitude of capabilities to GENESIS, making it the premier structural optimization code.

As a researcher and academician, I enjoyed publishing my work. Over time, I found that our competitors were always in the front row for my presentations and so I decided to keep things more proprietary. For this reason, most of the enhancements in DOT as well as some features of VisualDOC and GENESIS are not published in the open literature. One exception is the BIGDOT program [40], though the key parts that make it work were not published. BIGDOT was funded by a SBIR from NASA Langley and is intended to solve very large constrained optimization problems. DOT works well for problems up to perhaps 5000 variables, as long as there not too many active constraints. If there are many active constraints, the direction finding sub-problem becomes prohibitively time consuming. However, by late 1990s, GENESIS needed an optimizer capable of solving problems in the tens of thousands of variables and possibly that many active constraints.

In the 1960's Sequential Unconstrained Minimization Techniques (SUMT) were popular [3]. Over the years, these methods were largely abandoned in favor of SQP and similar methods. In the early 1990s, there was renewed interest in SUMT [41].

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

However, after about two and a half years, I was not able to solve problems over a few thousand variables, and then not reliably. One day I was riding my ATV, trying to kill myself in the mountains and was constantly thinking about the research when I had an idea. I stopped on the side on the mountain for nearly an hour thinking it through to be sure I wouldn't forget. The next day, I coded it and by noon was solving 10,000 variable problems with 10,000 active constraints.

Here, the original constrained optimization problem is converted to a sequence of unconstrained problems of the form given by Eqs. 7 and 8 and repeated here for clarity.;

Minimize

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

Subject to;

$$X_i^L \le X_i \le X_i^U \qquad i = 1, n \tag{50}$$

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

The unconstrained penalized function defined by Eq. 49 is solved by the Fletcher-Reeves conjugate direction method [5], which requires virtually no memory.

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

$$\nabla\Phi(\mathbf{X}) = \nabla F(\mathbf{X}) \tag{51}$$

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

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. (51) is a simple addition of gradient vectors, in the limit, we can calculate only one gradient (objective or constraint) at a time.

If the q_j^p parameters are thought of as the Lagrange Multipliers, λ_j this looks very much like the Augmented Lagrange Multiplier method. The ALM method is in the ADS program but can't solve large problems. Also, when I reprogrammed it, I still had no success on large problems. The idea I had while riding my ATV was a different way of calculating the q_j^p. This worked like magic and the program became BIGDOT.

When I presented BIGDOT at a conference, a presenter ahead of me (a competitor) said we can solve problems with many variables but only a few constraints and we can solve problems with only a few variables but many constraints (since only a few will be active). However, we cannot solve problems with many variables and many active constraints. Shortly after that, I got up and presented BIGDOT. My largest example was a 30,000 variable problem with 30,000 active constraints. Since then, I've solved a 500,000 variable problem with 500,000 active constraints and GENESIS has solved problems of 2.5 million variables with a few active constraints.

After the conference, the competitor emailed me saying it can't work to calculate only a few gradients at a time because constraints will be violated, inactive, violated, etc. on succeeding iterations. I responded that this will not happen with my method and he responded that it surely will and how was I storing all the needed gradients. Finally, I wrote him that it's our M-cubed method that prevents problems; Magic Memory Management. He figured out that I wasn't going to tell secrets and stopped asking.

While I continue research that I think may be useful to VR&D, I no longer publish it. I've finally learned to avoid helping our competitors.

One idea is how to solve discrete variable problems. One method is included in BIGDOT and published as it is based on a published work of another researcher. Another method is contained in the Discrete DOT program, DSCDOT. DSCDOT does a better job but is not published in the public domain.

A second idea is how to deal with very large numbers of active stress constraints in structural optimization. In the early 1970's we were very proud that we could calculate gradients analytically. At a conference, Dr. Roy Levi from JPL said "You think gradients are cheap? Wait until you need 100." Calculating gradients is basically like adding a load case and can become costly, especially in those days when 10,000 degrees of freedom in the finite element model was considered large. Today, we want to solve problems with 10 million degrees of freedom and perhaps a million design variables and 50 million stress constraints. If we think in terms of only 100,000 variables and 100,000 active constraints, gradient calculations are already prohibitive. This lead to development of the Stress DOT or STRDOT optimizer which is not published in the public domain. This algorithm creates good approximations to the stress constraints at almost no cost or memory requirement. It works well for common automotive structures but can tend to oscillate near the optimum. I've told Juan Pablo Leiva how to overcome that difficulty and I hope he remembers.

A third idea is how to improve reliability in combined sizing and shape problems, since these can be ill-conditioned. This led to Version 7 of DOT and is based on my thesis.

A fourth idea is how to do probabilistic structurally optimization in a very efficient manner. I published this in an internal memo some time ago. JP has added probability of failure calculations to GENESIS and I'll be pursuing this idea in the near future.

Hopefully, I'll have some more ideas from time to time.

REFERENCES

1. Vanderplaats, G. N., Multidiscipline Design Optimization, Vanderplaats Research & Development, Inc., Colorado Springs, CO, 2007.
2. Schmit, L. A., "Structural Design by Systematic Synthesis," Proc. 2nd Conference on Electronic Computation, ASCE, New York, 1960, pp. 105-122.
3. Fiacco, A. V., and G. P. McCormick: "Nonlinear Programming: Sequential Unconstrained Minimization Techniques," John Wiley and Sons, New York, 1968.
4. Powell
5. Fletcher, R. and C. M.Reeves: Function Minimization by Conjugate Gradients, Br. Computer J., vol. 7, no. 2, pp. 149-154, 1964.
6. Davidon, Fletcher, powell
7. Rockafeller, 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.
8. Zoutendijk, G, Methods of Feasible Directions, Elsevier, Amsterdam, 1960.
9. Vanderplaats, G. N., "Automated Design of Elastic Trusses for Optimum Geometry," Report #45, Dept. Engineering Mechanics, Case Western Reserve University, June 1971.
10. Vanderplaats, G. N. and Moses, F., "Automated Design of Trusses for Optimum Geometry," J. Str. Div., ASCE, No. ST3, March 1972.
11. Vanderplaats, G. N. and Moses, F., "Structural Optimization by Methods of Feasible Directions," J. Computers and Structures, VOl. 2, Pergamon Press, July 1973.
12. Gwin, L. B. and Vanderplaats, G. N., "The Auxiliary Problem for Feasible Directions," Tech. Note, AIAA Journal, Vol. 13, No. 5, May 1975.
13. Vanderplaats, G. N., "Structural Optimization Via a Design Space Hierarchy," Tech. Note, Int. J. Numerical Methods in Engineering, Vol. 10, No. 3, 1976.
14. Hague, D. S., "An Introduction to Multivariable Search Techniques for Parameter Optimization (and Program AESOP)," Boeing Co., Space Division, January 1, 1968.
15. Vanderplaats, G. N., "COMAND - A FORTRAN Program for Simplified Composite Analysis and Design, User's Manual," NASA TM X-73,104, Feb. 1976.
16. Vanderplaats, G. N., "CONMIN _ A FORTRAN Program for Constrained Function Minimization, User's Manual," NASA TM X-62,282, Aug. 1973.
17. VisualDOC, a Graphics Based Program for Multidiscipline Design Optimization, Vanderplaats Research & Development, Inc., Colorado Springs, CO, 2014.
18. Vanderplaats, G. N. and T. Gregory: A Preliminary Assessment of the Effects of Advanced Technology on Supersonic Cruise Tactical Aircraft, *Proceedings, Super Cruise Military Aircraft Design Conference*, Colorado Springs, Col., February 17 – 20, 1976.
19. Vanderplaats, G. N.: Automated Optimization Techniques for Aircraft Synthesis, AIAA Paper 76-909, *Proceedings, AIAA Aircraft Systems and Technology Meeting*, Dallas, Texas, September 27-29, 1976.
20. Madsen, L. E. and G. N. Vanderplaats: COPES – A FORTRAN Control Program for Engineering Synthesis, Report NPS69-81-003, Naval Postgraduate School, Monterey, CA, March 1982.

21. Hicks, R. M., E. M. Murman and G. N. Vanderplaats: An Assessment of Airfoil Design by Numerical Optimization, NASA TM-X 3092, July 1974.

22. Vanderplaats, G. N., R. M. Hicks and E. M. Murman: Application of Numerical Optimization Techniques to Airfoil Design, *NASA Conference on Aerodynamic Analysis Requiring Advanced Computers*, Langley Research Center, Va., NASA SP-347, part II, 1975.

23. Pickett, R. M., Jr., M. F. Rubinstein and R. B. Nelson: Automated Structural Synthesis Using a Reduced Number of Design Coordinates, *AIAA J.*, vol. 11, no. 4, pp. 489 – 494, 1973.

24. NACA Report No. 460, "The Characteristics of 78 Related Airfoil Sections from Tests in the Variable-Density Wind Tunnel," NASA 1933.

25. Vanderplaats, G. N. and R. M. Hicks: Numerical Optimization Using a Reduced Number of Design Coordinates, NASA TM X-73, 151, 1976.

26. Schmit, L. A. and B. Farshi: Some Approximation Concepts for Structural Synthesis, *AIAA J.*, Vol. 12(5), 1974, pp. 692-699.

27. Vanderplaats, G. N.: Approximation Concepts for Numerical Airfoil Optimization, NASA Technical Paper 1370, March 1979.

28. Vanderplaats, G. N.: Efficient Algorithm for Numerical Airfoil Optimization, *AIAA J. Aircraft*, vol. 16, no. 12, pp. 842 – 847, December 1979.

29. Vanderplaats, G. N., Sugimoto, H. and Sprague, C. M., "ADS-1: A General-Purpose Optimization Program," Synoptic, AIAA Journal, Vol. 22, No. 10, Oct. 1984.

30. Abadie, J. and J. Carpentier: "Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraints, in Optimization," R. Fletcher (ed.), Academic Press, New York, pp. 37-47, 1969.

31. Vanderplaats, G. N., "An Efficient Feasible Directions Algorithm for Design Synthesis," AIAA Journal, Vol. 22, No. 11, Nov. 1984, pp. 633-640.

32. DOT Design Optimization Tools, Vanderplaats Research & Development, Inc., Colorado Springs, CO.

33. Powell, M. J. D.: Algorithms for Nonlinear Constraints that use Lagrangian Functions, *Math. Prog.*, vol. 14, no. 2, pp. 224-248, 1978.

34. Vanderplaats, G. N. and E. Salajegheh: A New Approximation Method for Stress Constraints in Structural Synthesis, *AIAA J.*, Vol. 27 No. 3, 1989, pp. 352-358.

35. Hansen, S.R. and G. N. Vanderplaats: An Approximation Method for Configuration Optimization of Trusses, AIAA Journal, Vol 28, No. 1, 1990, pp. 161-172.

36. Bofang, Z.: Shape Optimization of Arch Dams, *Water Power and Dam Construction*, March 1987, pp. 43-51.

37. GENESIS User's Manual,: Vanderplaats Research & Development, Inc., Colorado Springs, CO.

38. Vanderplaats, G. N.,Chargin and H. Miura: Structural Optimization with MSC/NASTRAN Applied to Gear Housing, Proc. MSC/NASTRAN User's Conference, Pasadena, CA, March 1984.

39. Vanderplaats, G. N., "DOC Users Manual," Vanderplaats Research & Development, Inc., Colorado Springs, CO, 1995.

40. Vanderplaats, G. N., "Very Large Scale Continuous and Discrete Variable Optimization," 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, NY, Aug. 30 - Sept. 1, 2004.

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