AIAA JOURNAL
Vol. 36, No. 10, October 1998

New Adaptive Move-Limit Management Strategy for Approximate Optimization, Part 1

B. A. Wujek*
Engineous Software, Inc., Morrisville, North Carolina 27560
and
J. E. Renaud†
University of Notre Dame, Notre Dame, Indiana 46556-5637

Approximations play an important role in multidisciplinary design optimization by offering system behavior information at a relatively low cost. Most approximate optimization strategies are sequential, in which an optimization of an approximate problem subject to design variable move limits is iteratively repeated until convergence. The move limits are imposed to restrict the optimization to regions of the design space in which the approximations provide meaningful information. To ensure convergence of the sequence of approximate optimizations to a Karush–Kuhn–Tucker solution, a move-limit management strategy is required. Issues of move-limit management are reviewed and a new adaptive strategy for move-limit management is developed. With its basis in the provably convergent trust region methodology, the trust region ratio approximation method (TRAM) strategy utilizes available gradient information and employs a backtracking process using various two-point approximation techniques to provide a flexible move-limit adjustment factor. In a companion paper, the new strategy is tested in comparison to several existing strategies using a variety of multidisciplinary design optimization test problems. Those implementation studies highlight the ability of the TRAM strategy to control the amount of approximation error and efficiently manage the convergence to a Karush–Kuhn–Tucker solution.

I. Introduction

THE use of approximations to represent the design space is essential to the efficiency of multidisciplinary design optimization (MDO) algorithms. Approximations provide information about the system necessary for the optimization process without the cost of executing CPU-intensive analysis tools. Moreover, the use of approximations allows for the temporary decoupling of disciplines, which avoids the constant transfer of information among disciplines required during an iterative system analysis. Through the use of design space approximations, optimization of large, complex systems is made more practicable. It is important that these approximations accurately portray the design space so that the infeasible region is avoided and the design objective is continuously improving. These approximations will tend to stray from the actual system response surface as the design moves away from the data point(s) about which the approximation was formed. Therefore, design variable move limits are imposed to restrict the approximate optimization to regions of the design space in which the approximations are accurate. After each sequence of approximate optimization, the approximations of system behavior are updated with new information about the current design. Thus, many iterations of such algorithms may be required before convergence of the optimization process is achieved, and every additional iteration adds to the cost of the process. In light of this, a primary concern in developing an approximate optimization strategy is the proper choice of a move-limit management strategy.

The standard form of a nonlinear optimization process is

$$\text{minimize}_{wrtx} \quad f(\mathbf{x}), \qquad \mathbf{x} = [x_1, \ldots, x_n] \tag{1}$$

$$\text{subject to:} \quad g_j(\mathbf{x}) = 0, \qquad j = 1, \ldots, J \tag{2}$$

$$h_k(\mathbf{x}) \geq 0, \qquad k = 1, \ldots, K \tag{3}$$

$$\mathbf{x}_{(L)} \leq \mathbf{x} \leq \mathbf{x}_{(U)} \tag{4}$$

The lower and upper bounds ($\mathbf{x}_{(L)}$ and $\mathbf{x}_{(U)}$) in Eq. (4) are global variable bounds imposed by the designer.

Approximate optimization algorithms typically build approximations of the objective $\tilde{f}$ and the constraints $\tilde{\mathbf{g}}$ and $\tilde{\mathbf{h}}$ and then solve a sequence of approximate optimizations

$$\text{minimize}_{wrtx} \quad \tilde{f}(\mathbf{x}), \qquad \mathbf{x} = [x_1, \ldots, x_n] \tag{5}$$

$$\text{subject to:} \quad \tilde{g}_j(\mathbf{x}) = 0, \qquad j = 1, \ldots, J \tag{6}$$

$$\tilde{h}_k(\mathbf{x}) \geq 0, \qquad k = 1, \ldots, K \tag{7}$$

$$\max\left[\mathbf{x}_{(L)}, \mathbf{x}_{(l)}^{(t)}\right] \leq \mathbf{x} \leq \min\left[\mathbf{x}_{(u)}^{(t)}, \mathbf{x}_{(U)}\right] \tag{8}$$

In Eq. (8) additional move limits ($\mathbf{x}_{(l)}^{(t)}$ and $\mathbf{x}_{(u)}^{(t)}$) about the current design iterate $\mathbf{x}^{(t)}$ are placed on the design variables in an attempt to ensure approximation accuracy. The superscript t denotes the current design iterate within the sequential approximate optimization process. These limits are temporary bounds applied at each design iterate and may change as the optimization proceeds, but they are always restricted by the original global bounds of the problem ($\mathbf{x}_{(L)}$ and $\mathbf{x}_{(U)}$). If the allowed changes in the design variables are too liberal, the discrepancy between the approximations and the actual response surface may eventually become unacceptable and adversely affect the optimization process. Common maladies are cycling about a minimum or about a constraint boundary and the inability to recover from an infeasible region. If the allowed changes in the design variables are too stringent, the overall progress of the optimization will be unnecessarily slowed, and more algorithm iterations will be required, increasing the cost of the process.

It can be reasoned that the amount that design variables should be allowed to change at any given time is related to the nature of the design space at the current location, the accuracy of the current approximation, and/or possibly even the history of previous movements. Thus, the development of a strategy to account for the aforementioned concerns in setting design variable move limits is a worthy task. Several existing move-limit strategies are briefly reviewed, and their strengths and weaknesses are highlighted. A new move-limit strategy, which avoids many of the deficiencies of existing methods, is developed in this research for use in approximate optimization procedures commonly employed in MDO algorithms.

Received Feb. 6, 1998; presented as Paper 98-1965 at the AIAA/ASME/ASCE/AHS/ASC 39th Structures, Structural Dynamics, and Materials Conference, Long Beach, CA, April 20–23, 1998; revision received June 1, 1998; accepted for publication June 2, 1998. Copyright © 1998 by B. A. Wujek and J. E. Renaud. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

*Advanced Technology Developer. Member AIAA.

†Assistant Professor, Department of Aerospace and Mechanical Engineering. Member AIAA.

In a companion paper[1] the strategy is tested in the coordination module of the concurrent subspace optimization (CSSO).[2]

II. Review of Move-Limit Strategies

Move limits on design variables $\boldsymbol{x}$ are most often expressed in terms of a percentage of the current design variable values $\boldsymbol{x}^{(t)}$ such that

$$x_{i,(u)}^{(t)} = x_i^{(t)} + A\left|x_i^{(t)}\right|, \qquad x_{i,(l)}^{(t)} = x_i^{(t)} - A\left|x_i^{(t)}\right| \tag{9}$$

The most straightforward approach to setting move limits is to assign a constant percentage A for the entire duration of the algorithm execution. The inefficiency of this procedure is obvious considering that, as the design optimization process progresses, different move limits would be more appropriate based on new approximations and different regions of the design space. Recognizing the need for an effective method of setting and adjusting design variable move limits, many researchers have employed different strategies in an attempt to improve efficiency in their approximate optimization algorithms. A thorough review of many of these strategies can be found in Ref. 3; here they will be briefly described, noting relevant strengths and weaknesses of each method.

Pourazady and Fu[4]

In Ref. 5, it is claimed that "In general, move-limits should be gradually shrunk as the design approaches the optimum," suggesting a prescribed reduction scheme. Such a reduction method is offered by Pourazady and Fu,[4] in which the move limits are reduced from the initially prescribed values in an exponential fashion based on the iteration number

$$A^{(t)} = \left(A^{(0)}\right)^t, \qquad \text{if} \qquad A^{(0)} < 1 \tag{10}$$

$$A^{(t)} = \left(A^{(0)}\right)^{1/t}, \qquad \text{if} \qquad A^{(0)} > 1 \tag{11}$$

Methods employing prescribed reduction will ensure termination of the design optimization process, but suboptimal designs may be reached due to premature termination. Moreover, the efficiency of these strategies is questionable because move-limit reduction is often enforced when unnecessary.

Thomas et al.[6]

A strategy was offered by Thomas et al.[6] that accounts for the history of the design. All design variable move limits are reduced by 50% whenever the maximum constraint violation has increased during the last iteration, and individual move limits are increased by 33% whenever a design variable hits the same upper or lower move-limit bound in two consecutive iterations. The constant percentages by which the move limits are decreased (50%) and increased (33%) are heuristic factors, set based solely on experience, and different factors would most likely be more appropriate for different design variables, different regions of the design space, and different problems. Also, because the objective function is not considered, the move-limits settings are driven by the bounds rather than overall design improvement.

Bloebaum et al.[7]

A strategy based on design space sensitivity is developed by Bloebaum et al. in Ref. 7. The theory is that those design variables with the most impact on design improvement should be given less restrictive move limits, whereas the less critical ones should be restricted because they have little effect on the design. This strategy is carried out by defining an effectiveness coefficient e_i (first defined in Ref. 8) for each design variable x_i using the known sensitivities at the current design point

$$e_i = \begin{cases} \left(\dfrac{\mathrm{d}f}{\mathrm{d}x_i}x_i\right) \Big/ \left(\dfrac{\mathrm{d}C}{\mathrm{d}x_i}\right), & \text{if} \quad C \geq 0 \\[2ex] \left(\dfrac{\mathrm{d}C}{\mathrm{d}x_i}\right) \Big/ \left(\dfrac{\mathrm{d}f}{\mathrm{d}x_i}x_i\right), & \text{if} \quad C < 0 \end{cases} \tag{12}$$

where C is a cumulative constraint in the form of the Kreisselmeier–Steinhauser[9] function necessary to determine the overall effect of a design variable on all of the constraints simultaneously. Upper and lower bounds are set on the effectiveness by a one standard deviation limit

$$e^u = \bar{e} + \sigma_e, \qquad e^l = \bar{e} - \sigma_e \tag{13}$$

where $\bar{e}$ and σ_e are the mean and standard deviation of the efectiveness coefficients, respectively. Maximum and minimum move-limit percentages (A^u and A^l) are then chosen, so that any variable with an effectiveness coefficient above e^u is assigned the maximum move limit, and any variable with an effectiveness below e^l is assigned the minimum move limit. For those variables with effectiveness between e^u and e^l, the move limits are assigned based on a linear distribution of move limits between e^u and e^l as

$$A_i = \frac{\left(e_i - e^l\right)}{2\sigma_e}(A^u - A^l) + A^l \tag{14}$$

In this strategy, the use of the cumulative constraint C is a point of concern because accurate depiction of the extent of feasibility is highly dependent on the setting of the parameter ρ within the Kreisselmeier–Steinhauser function.[9] Also, the upper and lower move-limit percentage limits (A^u and A^l) are set heuristically. In Ref. 7, several test cases are run, and different values are chosen for these quantities in each case, with no reasoning behind the choices. More importantly, this strategy does not account for approximation error.

Chen[10]

Chen[10] proposed a number of different methods for calculating design variable move limits for use in the sequential linear programming (SLP) algorithm. These methods utilize linear approximations of the constraints to determine when a bound will be reached. Whereas each of the methods to be described is applicable to both equality and inequality constraints, only the equations depicting the inequality constraints will be shown here. It is important to note that Chen defines normalized values (with respect to the nominal design point $\boldsymbol{x}^0$) of both the design variable change

$$\Delta X_i \equiv \left(x_i / x_i^0\right) - 1, \qquad i = 1, \ldots, n_x \tag{15}$$

and of the constraint sensitivities

$$\frac{\mathrm{d}g_j}{\mathrm{d}x_i} \equiv x_i^0 \left.\frac{\mathrm{d}g_j}{\mathrm{d}x_i}\right|_{x = x^0}, \qquad j = 1, \ldots, n_g \tag{16}$$

for use in these methods. Essentially, ΔX is the percentage A defined earlier.

Chen Method I

In Chen's first strategy, the assumption is made that all of the design variable changes are the same percentage

$$\Delta X = \Delta X_1 = \Delta X_2 = \cdots = \Delta X_{n_x} \tag{17}$$

so that the $\Delta X^{(j)}$ to enforce activity of each constraint $g_j \geq b_j$ can be calculated as

$$\Delta X^{(j)} = \left|g_j(\boldsymbol{x}^0) - b_j\right| \Bigg/ \left(\sum_{i=1}^{n_x}\left|\frac{\mathrm{d}g_j}{\mathrm{d}x_i}\right|\right) \tag{18}$$

This quantity is calculated for each constraint g_j, and the largest $\Delta X^{(j)}$ calculated is taken as the move limit. The absolute value of the numerator is justified because the only concern is enforcing the constraint to be active, regardless of whether it is from the feasible or the infeasible side. In the denominator, the sum of the absolute values of the constraint sensitivities is taken as a conservative measure. Because positive and negative sensitivities may cancel out and sum to a very small number, the sum of the absolute values avoids calculation of very large move limits. Although this technically does not adhere to the linear approximation, it is at least conservative.

This method has many obvious faults associated with it. First of all, restricting all of the ΔX_i to be equal basically predetermines the direction in which the constraints are sought. Essentially, this restriction limits the search directions to a set of 2^{n_x} vectors formed from combinations of positive and negative percent changes in each design variable. The result of this assumption is that the true minimum distance to the constraint bound is not obtained. Even more illogical is that the largest $\Delta X^{(j)}$ is taken as the move limit. The largest $\Delta X^{(j)}$ denotes the distance to the farthest bound so that, if that step is taken, there is the possibility that closer constraint bounds will be violated. The farthest constraint bound is desired only when all constraints are violated, in which case the goal is to satisfy the farthest constraint to guarantee satisfaction of all constraints. But this method does not differentiate between feasibility and infeasibility, which is another fault. Why should one be concerned with feasible constraints when other constraints are infeasible? It is possible that feasibility is assumed for this method. In that case, the logical choice for the move limit as calculated by Eq. (18) should be the smallest $\Delta X^{(j)}$ calculated so that the closest bound is the one considered. Clearly, Chen holds a different philosophy concerning move limits, one which is not concerned with the maladies of poor approximations but concentrates on fast convergence alone by reaching bounds as soon as possible. This is evidenced in a statement that appears later in Ref. 10 in which mention is made of the Thomas et al.[6] strategy of reducing move limits by 50% when the maximum constraint violation has increased from the past iteration. In the Chen method the move limits should be reduced when the maximum constraint violation has decreased.

The most glaring deficiency in this method is that the linearizations for which the move limits are required are themselves used to calculate the move limits. If the linearization underpredicts the constraint value in calculating the distance to the constraint bound, it will also do so during the optimization, and the design will go infeasible if that distance calculated is taken as the move limit.

Chen Method II

In this method, the strategy of method I is employed for the first iteration only. For subsequent iterations, the move limits are successively reduced by a user designated reduction factor, which Chen denotes to be between 0.7 and 1. Therefore, this method is really a combination of an approximation-based strategy with a constant reduction strategy.

Because this method incorporates method I, all of the faults of that method apply to this one. Similarly, the deficiencies of a constant reduction strategy have been discussed. Another problem with this method is that the active constraint set will change at some time so that the relationship between the move limits and constraint satisfaction will change. Thus, there is no reason why one should relate the move limits in the first iteration to those in subsequent iterations, even by a constant reduction factor.

Chen Method III

This method provides different move limits for the different design variables. The magnitude of each constraint gradient is calculated as

$$\|\nabla g_j\| = \sqrt{\sum_{i=1}^{n_x}\left(\frac{\mathrm{d}g_j}{\mathrm{d}x_i}\right)^2} \tag{19}$$

and the step needed to reach each constraint along the direction of the gradient is linearly approximated as

$$d^{(j)} = \frac{\left|g_j(\boldsymbol{x}^0) - b_j\right|}{\|\nabla g_j\|} \tag{20}$$

A move limit for each design variable is then set as the component of the step d in the direction of that design variable as

$$\Delta X_i^{(j)} = \left(\left|\frac{\mathrm{d}g_j}{\mathrm{d}x_i}\right| \bigg/ \|\nabla g_j\|\right) d^{(j)} \tag{21}$$

As in method I, because there may be more than one constraint, the largest move limit calculated from Eq. (21) is chosen for each design variable.

This method overcomes many of the deficiencies of method I. Its direction of search for the constraint bounds is logical, and it sets individual design variable move limits. However, the ramifications of using the largest $\Delta X^{(j)}$ as the move limit have been discussed. Also, as in method I, the move limits are calculated using the linearizations for which they are needed, with no regard for any error which may exist in those linearizations.

Chen Method IV

This method incorporates a quantity that has been overlooked by the preceding three methods proposed, the objective function. The whole point of the optimization process is to optimize the objective function; thus, a given design variable's effect on the objective should necessarily be considered in determining its move limit. Chen defines a weighting term for each x_i as

$$w_i = \left|\frac{\mathrm{d}f}{\mathrm{d}x_i}\right| \bigg/ \|\nabla f\| \tag{22}$$

A larger w_i means that x_i has more influence on f so that it should be given more freedom. The weighting term is then applied to method III so that Eq. (20) becomes

$$d^{(j)} = \left|g_j(\boldsymbol{x}^0) - b_j\right| \bigg/ \sqrt{\sum_{k=1}^{n_x}\left(w_k\frac{\mathrm{d}g_j}{\mathrm{d}x_k}\right)^2} \tag{23}$$

and the equation for the move limit becomes

$$\Delta X_i = \left|d^{(j)}\left(w_i\frac{\mathrm{d}g_j}{\mathrm{d}x_i}\right)w_i\right| \bigg/ \sqrt{\sum_{k=1}^{n_x}\left(w_k\frac{\mathrm{d}g_j}{\mathrm{d}x_k}\right)^2} \tag{24}$$

The addition of the extra weighting term in the numerator is used to compensate for enlargement of the step d in Eq. (23) caused by the reduction in the constraint gradients by the weighting terms in the denominator. Again, the largest $\Delta X^{(j)}$ is taken as the move limit.

This method has the same merits and demerits as method III. It is an improvement in that the influence of the objective function is considered in determining the move limits.

The methods proposed by Chen are all based on using gradient information to estimate the distances to constraint boundaries. To avoid large move limits, Chen suggests that only those constraints that are close to their bounds be included in the move-limit determination. Closeness to bounds is a vague and subjective concept for which Chen offers no definition. Also, after all of the technical development and reasoning behind the methods described, Chen finds it necessary to include a move-limit enlargement factor of 1.2–2.0 to accelerate convergence and avoid infeasibility. There is certainly no assurance that either of those results would be realized by randomly enlarging the move limits.

Fadel and Cimtalay[11]

Another strategy which relies on gradient information is developed in Ref. 11, which is based on the two-point exponential approximation (TPEA).[12] This form of approximation is an extension of the Taylor series, which accounts for a matching of the derivatives at consecutive design points through an exponential correction factor. The linear approximation is

$$\tilde{g}_j^{(t+1)} = g_j^{(t)} + \sum_{i=1}^{n_x}\frac{\mathrm{d}g_j}{\mathrm{d}x_i}\left[\left(\frac{x_i}{x_i^{(t)}}\right)^{p_i} - 1\right]\left(\frac{x_i^{(t)}}{p_i}\right) \tag{25}$$

$$p_i = 1 + \left\{\left[\ln\left(\left.\frac{\mathrm{d}g_j}{\mathrm{d}x_i}\right|_{\boldsymbol{x}^{(t)}}\right) - \ln\left(\left.\frac{\mathrm{d}g_j}{\mathrm{d}x_i}\right|_{\boldsymbol{x}^{(t-1)}}\right)\right] \bigg/ \left[\ln\left(x_i^{(t)}\right) - \ln\left(x_i^{(t-1)}\right)\right]\right\} \tag{26}$$

In essence, this is a linear approximation with respect to the design variables raised to some exponent p_i. Upper and lower bounds of 1 and −1, respectively, are placed on the exponents, limiting the approximation to some range between a linear and a reciprocal approximation with respect to each design variable. This is done to ensure that a given design variable does not dominate the approximation, although this seems to restrict any natural curvature that may exist. It seems that some sort of scaling could be imposed based

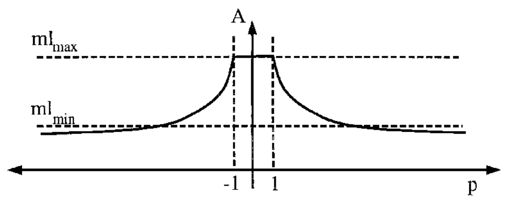


Fig. 1 Relationship between calculated exponent and assigned move limit.

on the actual calculated exponent to meet this same goal. Also note that this approximation takes into account the history of the design (information from the previous design point) allowing for a progressive updating of the approximation based on its past trend.

Because any exponent calculated outside of the range of 1 to -1 is adjusted to meet these bounds, an error is introduced into the approximation by forcing the restriction of the exponent. If the calculated exponent is greater than 1, the approximation is forced to be linear:

$$\tilde{g}_j = g_j^{(t)} + \sum_{i=1}^{n_x} \left(x_i - x_i^{(t)}\right) \frac{\mathrm{d}g_j}{\mathrm{d}x_i} \tag{27}$$

Assuming some move-limit percentage A so that the bounds of Eq. (9) are enforced, the difference (error) between the actual exponential approximation of Eq. (25) and the forced linear approximation of Eq. (27) due to a given variable x_i can be calculated as

$$\Delta g_j \Big/ \left(x_i^{(t)} \frac{\mathrm{d}g_j}{\mathrm{d}x_i}\right) = \frac{(1+A)^{p_i} - 1}{p_i} - A \tag{28}$$

The right-hand side is essentially the error term E, which is a constant. A similar procedure can be followed for those exponents below -1 that are forced to -1 (a reciprocal approximation), giving

$$\Delta g_j \Big/ \left(x_i^{(t)} \frac{\mathrm{d}g_j}{\mathrm{d}x_i}\right) = \frac{(1+A)^{p_i} - 1}{p_i} - \frac{A}{A+1} \tag{29}$$

Fadel and Cimtalay[11] proposes that by selecting some acceptable level of error E, an exponential plot of p_i vs A_i can be formed for each design variable. A typical plot of this sort is shown in Fig. 1.

Thus, for every actual calculated exponent, an associated move-limit percentage A_i can be assigned. Observing that $\boldsymbol{p}$ is actually a matrix of exponents p_{ij} corresponding to each design variable x_i for each output g_j, the question becomes which p_{ij} controls the move-limit settings for each variable. Because the move limits are to be set for each design variable, the most nonlinear function approximation with respect to a given design variable should be the one that controls the size of the move limit. For example, if an exponent of 2 is calculated for the relationship of x_i with g_j, and an exponent of 3 is calculated for the relationship of x_i with g_k (and this is the highest exponent), then $p_{ik} = 3$ should be used in Eq. (28) in determining the move limit for this design variable. If none of the exponents calculated for a given design variable are outside the allowed range, then some maximum move limit should be set for this design variable.

Fadel and Cimtalay's[11] approach is somewhat appealing in that it accommodates individual move-limit adjustment and is based somewhat on the accuracy of the approximation. Yet, it seems that the limitation on the exponent is enforced merely to develop the method; the calculated exponent is a representation of the curvature, and to restrict the approximation to be between linear and reciprocal degrades the approximation. Moreover, the method is restricted to use with the TPEA method, greatly limiting its application generally. Also, note that the acceptable error term E is set heuristically, as are maximum and minimum move-limit settings. Different choices could significantly affect the resulting move limits and, thus, the efficiency of the algorithm.

Grignon and Fadel[13]

In Ref. 13, a fuzzy logic technique is used to assign move limits to the design variables. Essentially, a fuzzy logic table is developed to incorporate the TPEA strategy and the effectiveness strategy that were just discussed. The table, which is made up of different combinations of exponents and effectiveness coefficients, returns a value of low, medium, or high, which is in turn defuzzified by interpolation between some prescribed minimum and maximum move-limit multipliers. Reference 13 is not clear as to how this interpolation actually works, but states that the product of the exponent p and the effectiveness coefficient e is used and that the intent is to increase move-limit bounds when the exponent is between -1 and 1 and e is large. However, the simple product of quantities does not seem to be consistent with this goal because exponents outside the -1 and 1 bounds would be larger in the absolute sense and would provide a higher product. Again, prescription of the maximum and minimum multipliers is a heuristic process; actually, it seems that the whole idea of fuzzy logic is based on user-prescribed rules and is, thus, heuristic in nature. Moreover, because the method incorporates the two methods stated, it also introduces all of the problems associated with them as already discussed.

Trust Region Approach

A strategy that falls into a class by itself is the trust region approach. This is a classical method prevalent in nonlinear programming,[14] which provides a framework for adaptively managing the amount of movement allowed in the design space using approximate models. Trust region methods were originally introduced as a way of ensuring global convergence for Newton-like methods. It is so named because the trust region defines the region in which one may trust the approximate model to accurately portray the actual design space. Quite simply, the trust region is merely another name for the region of the design space defined by the design variable move limits.

The trust region approach is based on the use of a reliability index to monitor how well the current approximation is found to represent the actual design space. Considering an unconstrained problem with an objective function f which is approximated by a function $\tilde{f}$, a trust region ratio is defined as

$$\rho^{(t)} = \frac{f\left(\boldsymbol{x}^{(t)}\right) - f\left(\boldsymbol{x}^{(t)} + \Delta\boldsymbol{x}\right)}{f\left(\boldsymbol{x}^{(t)}\right) - \tilde{f}\left(\boldsymbol{x}^{(t)} + \Delta\boldsymbol{x}\right)} \tag{30}$$

This is simply the ratio of the actual change in the function to the change predicted by the approximation. After each optimization iteration t, the trust region radius is updated according to the following principles:

1) If the ratio is negative or small, the iteration is considered unsuccessful because either the actual objective increased (it is known that $\tilde{f}$ will not increase) or it did decrease, but not nearly as much as predicted by the approximation. In either case, the approximation is certainly poor, and the trust region must be reduced. For the case of a negative ratio, most algorithms[15–17] actually reset the design to the previous iterate and repeat that optimization iteration. This is done to guarantee convergence of the algorithm.

2) Conversely, if the ratio is large, a reasonable decrease in the objective function has been observed relative to the approximate decrease, and the iteration is considered successful. Note that if the ratio is significantly larger than one, the objective function actually decreases more than had been predicted, and the approximation is actually a poor representation of the design space. However, because this scenario is actually favorable as more reduction is gained than expected, an increase in the trust region radius is justified.

3) Finally, if the ratio is an intermediate value, the wisest choice of action may be to leave the size of the trust region as it is.

Mathematically, the preceding rules for updating the trust region radius may be described by choosing constants to define the ranges of the ratio value for which reduction or enlargement are necessary. The positive constants $R_1 < R_2 < 1$ and $c_1 < 1$, $c_2 > 1$ are chosen so that the trust region radius (percent change allowed for each design variable) is updated as

$$A^{(t+1)} = \begin{cases} c_1 A^{(t)} & \text{if} \quad \rho^{(t)} < R_1 \\ c_2 A^{(t)} & \text{if} \quad \rho^{(t)} > R_2 \\ A^{(t)} & \text{otherwise} \end{cases} \tag{31}$$

Typical values used[15,16] for the limiting range values are $R_1 = 0.25$ and $R_2 = 0.75$. The trust region multiplication factors c_1 and c_2 are chosen in Ref. 16 to be 0.25 and 2, respectively, whereas Ref. 15 suggests 0.5 and 2 and offers a method of adjusting these factors that accounts for steps $\|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)}\|$, which are smaller than the current trust region radius $\rho^{(t)}$.

In Ref. 14, an additional mechanism for regulating the trust region is given to adjust the size of the radius to be consistent with the magnitude of the steps taken. When the step taken to solve the approximate optimization problem is a Newton step (a step to the minimum of the quadratic approximation) that is shorter than the current trust region radius, the radius is immediately reduced to the length of the Newton step.

A choice for the initial trust region size is left for the user to determine. It may be based on knowledge of the problem or on some other criteria, such as the length of the Cauchy step, a step to the minimum of the approximation in the steepest descent direction, as suggested by Ref. 18. Another option offered by Ref. 15 is to choose the initial radius to be proportional to the norm of the gradient of the objective, $A^{(0)} = \alpha\|(\mathrm{d}f/\mathrm{d}\boldsymbol{x})|_{x^0}\|$, although proper choice of the proportionality constant α must still be dealt with. Although an algorithm may recover from a bad initial trust region radius, this value has an effect on the efficiency of the algorithm because extra iterations may be required.

One can see that the trust region method encompasses many of the ideas of the move-limit strategies already discussed. It uses a constant reduction/enlargement scheme based on the history of the previous iteration and on how well the convergence of the approximation conformed to the actual convergence. This information might also be used for model management as alluded to in Ref. 19. That is, the model used to approximate the design space might be found to perform exceptionally well, so that use of a model with less accuracy but lower computational cost could be considered. In CSSO,[2] for example, it might be found that the quadratic formed from the subspace data is extremely accurate in the coordination procedure. As a result, it may be decided that either less data is required in the next iteration or the subspace optimizations may be bypassed altogether and a sequential linear approximation (SLA) method could be performed until it is determined that a quadratic model is again required. Thus, it can be seen that a tradeoff exists between reducing the required level of approximation or expanding the trust region for use with a highly accurate model.

Besides the arbitrary setting of the limiting range constants R_1 and R_2 and the adjustment factors c_1 and c_2, one obvious drawback to this method is that it is defined for unconstrained optimization. It necessarily compares the change in a single quantity to the predicted change in that quantity. For use in constrained optimization, this is often overcome by the use of a penalty function, such as the augmented Lagrangian function,[20] in which the constraints of the problem are included with the original objective function to form a new objective function of the form

$$L = f + \boldsymbol{\lambda}^T \boldsymbol{g} + \tfrac{1}{2}\boldsymbol{g}^T \boldsymbol{W}\boldsymbol{g} \tag{32}$$

where $\boldsymbol{\lambda}$ is a vector of Lagrange multipliers and $\boldsymbol{W}$ is a diagonal matrix of penalty weighting terms. This type of problem modification is acceptable as long as the optimization algorithm may be adapted to approximate and use the desired penalty function. In many cases, however, the problem is formulated such that approximations are available only for the objective function and constraints individually. These approximations can be combined to form an approximate penalty function, with the only additional burden being the calculation and updating of penalty parameters (Lagrange multipliers, etc.).

The most promising aspect of the trust region approach is its excellent global convergence properties. Global convergence is defined as the ability to converge to a local optimum from any starting point. In Ref. 21, a proof of global convergence is offered for constrained approximate optimization algorithms in which design variable movement from iteration to iteration is governed by the trust region. This characteristic is important to avoid cycling and to provide steady convergence in the design process. The fact that trust region algorithms are provably convergent is exploited in the move-limit strategy developed in the following section.

III. Move-Limit Strategy Development

In developing a strategy to accomplish any task, it is a good idea to first define the primary objectives of the strategy. Limits on the movement of design variables in system level approximate optimization procedures are necessary to remain nominally feasible and to converge on a minimum. Thus, the primary goals of a move-limit strategy for the system coordination procedure are fairly clear: 1) maintain a decrease in the actual system merit and 2) maintain feasibility. These objectives define what is desired for any given iteration of the algorithm to guarantee improvement in the design and to avoid cycling about the minimum. However, considering that the primary goal from an algorithm standpoint is to reach the optimum as quickly as possible, we can add a third and most important objective, which is to 3) allow as much change in the design variables as can be tolerated to attain objectives 1 and 2. This final objective is the one that determines the efficiency of the overall optimization process. Thus, the extent to which this objective is met can be considered a measure of the effectiveness of a move-limit strategy. Implicit in these objectives is the ability to control the movement such that the error in the approximations is maintained at a reasonable level. With these goals in mind, the following strategy is developed to automatically manage the adjustment of move limits in system level approximate optimization procedures.

The trust region approach just discussed is a well-established methodology, which has been shown to guarantee convergence of algorithms[21] that make use of approximations matching the actual function and gradient values at a given point. It allows for expansion and reduction of the move limits based on the history of the objective convergence and the accuracy of the system approximation. However, its main detractions are that the acceptable trust region ratio limits R_1 and R_2 are arbitrarily set, as are the enlargement/reduction factors c_1 and c_2. These discrete limits result in an undesirable discontinuity of the move-limit adjustment factor. As can be seen in Fig. 2, the difference between the move limits being held at their current value (when $R_1 < \rho \le R_2$) or multiplied/divided by a prescribed constant such as 2.0 (when $\rho > R_2$ or $\rho \le R_1$) may be very slight, and this difference is difficult to justify. A continuous relationship between ρ and the adjustment factor would provide a more consistent and flexible mechanism for controlling the error in the approximations.

The goal of the strategy is to provide a logical quantitative measure for enlarging/reducing the size of the trust region based on gradient information. In essence, it is desired to determine the amount of change in the design variables that would have still been acceptable under the conditions of the trust region enlargement/reduction rules already discussed. That is, if $\rho \le R_1$, the design has progressed beyond a region in which the approximation error is considered acceptable. We might then ask ourselves the question: At what point should the design variables have been restricted so that $\rho = R_1$? Similarly, if $\rho > R_2$, it seems that the design could have been allowed to change further because the approximation is still quite acceptable in this region. How much further could we have allowed the design to progress until $\rho = R_2$? Viewing the acceptable size of the trust region in this manner will allow for a continuous distribution of the trust region adjustment factor. Moreover, by allowing for this type of continuous distribution, it is believed that the arbitrary trust region ratio limits ($R_1 = 0.25$ and $R_2 = 0.75$) may be done away with in favor of a simple median value, $R = 0.5$. These issues are discussed in the following strategy development.

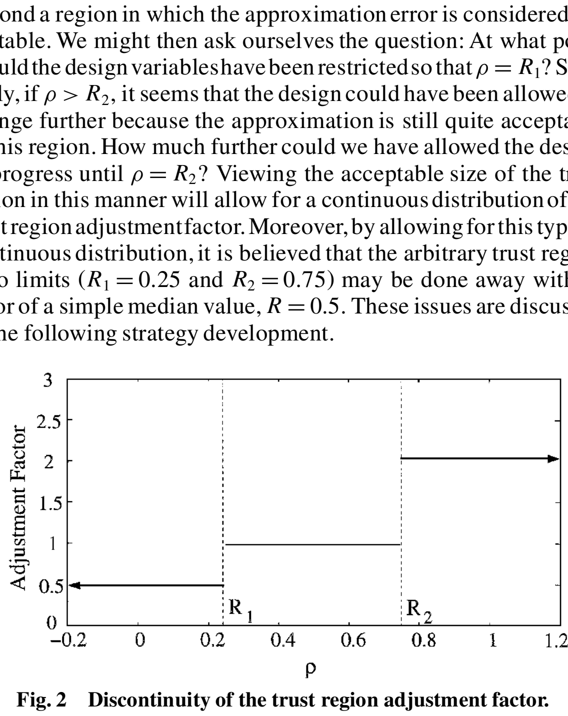


Fig. 2 Discontinuity of the trust region adjustment factor.

Developing a Model for the Trust Region Ratio

To answer the question of allowable movement just posed, it is necessary to characterize the behavior of ρ as a function of the design variables. A true representation of $\rho(\boldsymbol{x})$ could only be obtained by solving Eq. (30) for ρ at any given instance of $\boldsymbol{x}$, a procedure that is obviously too expensive to justify simply to obtain a move-limit adjustment factor because it involves performing system analyses. However, making use of the known values of ρ and sensitivity information $(\mathrm{d}f/\mathrm{d}\boldsymbol{x}, \mathrm{d}\tilde{f}/\mathrm{d}\boldsymbol{x})$ at the new design iterate $\boldsymbol{x}^{(t+1)}$, an adequate model of ρ can be formed as follows.

For brevity, herein any quantity $[\,]^{(t+1)}$ referring to the $(t+1)$st iteration will be denoted by $[\,]^{+}$. Equation (30) can be written as

$$\rho^{(t)} = \frac{f\big(\boldsymbol{x}^{(t)}\big) - f(\boldsymbol{x}^{+})}{f\big(\boldsymbol{x}^{(t)}\big) - \tilde{f}(\boldsymbol{x}^{+})} = \frac{\Delta f}{\Delta \tilde{f}} \tag{33}$$

and the change in each design variable can be defined as

$$\Delta x_i = x_i^{+} - x_i^{(t)} \tag{34}$$

An important quantity that is desired is the sensitivity of ρ with respect to the design vector. This sensitivity information can be calculated using the sensitivities of the actual and approximate functions at the new design point, taking the function value at the previous point to be a constant. Thus,

$$\left.\frac{\mathrm{d}\rho}{\mathrm{d}x_i}\right|_{x^{+}} = \left[\Delta\tilde{f}\left(-\left.\frac{\mathrm{d}f}{\mathrm{d}x_i}\right|_{x^{+}}\right) - \Delta f\left(-\left.\frac{\mathrm{d}\tilde{f}}{\mathrm{d}x_i}\right|_{x^{+}}\right)\right]\bigg/(\Delta\tilde{f})^2 \tag{35}$$

where, when using a second-order Taylor series approximation for $\tilde{f}$,

$$\frac{\mathrm{d}\tilde{f}}{\mathrm{d}x_i} = \left.\frac{\mathrm{d}f}{\mathrm{d}x_i}\right|_{x^{(t)}} + \frac{\mathrm{d}^2\tilde{f}}{\mathrm{d}x_i^2}\Delta x_i + \frac{\mathrm{d}^2\tilde{f}}{\mathrm{d}x_i\mathrm{d}x_j}\Delta x_j, \qquad j \neq i \tag{36}$$

At this point, one could construct a linear approximation to ρ as a function of $\boldsymbol{x}$ about the point $\boldsymbol{x}^{+}$ simply using the derivatives calculated from Eq. (35) in a first-order Taylor series. However, from Eq. (33), it is obvious that ρ is a nonlinear function of $\boldsymbol{x}$ and that a linear model could perform very poorly over the range of $\Delta\boldsymbol{x}$. For the sake of example, consider a function of one variable, as shown in Fig. 3a, along with the corresponding approximation. Figure 3b shows the actual variation in ρ over the entire range of the change in x for the function approximation shown in Fig. 3a. Note that a linear approximation formed for ρ at the new design point does a poor job of representing ρ over the range of the previous step. In fact, if the goal were to estimate the allowable step to ensure $\rho \geq 0.5$ (as it is in this research), it would be determined that almost no movement from the previous point is allowed, when in fact a step of approximately $\Delta x = 3.5$ would be acceptable. It should also be noted that even if higher-order information could be obtained at the new point, this single-point information is not enough to ensure a good approximation of ρ over the entire range of Δx.

To obtain a better approximation for ρ, one can make use of the information at the previous point $\boldsymbol{x}^{(t)}$. Although ρ is actually undefined at this point because $\boldsymbol{x}^{(t)} = \boldsymbol{x}^{+}$ and $f(\boldsymbol{x}^{(t)}) = \tilde{f}(\boldsymbol{x}^{+})$ so that

$$\rho^{(t)} = \frac{f\big(\boldsymbol{x}^{(t)}\big) - f(\boldsymbol{x}^{+})}{f\big(\boldsymbol{x}^{(t)}\big) - \tilde{f}(\boldsymbol{x}^{+})} = \frac{0}{0} \tag{37}$$

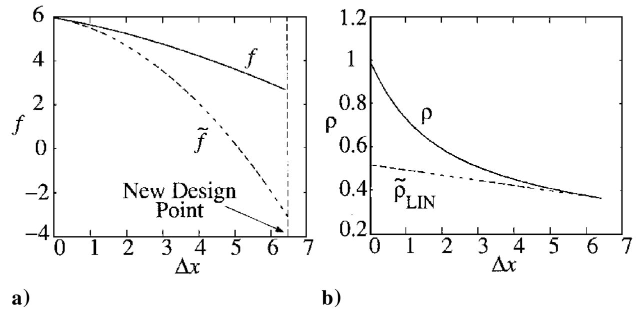


Fig. 3 Modeling the trust region ratio.

it can be defined as $\rho \equiv 1$ because this is the asymptotic value as $\Delta\boldsymbol{x} \to 0$. This can be shown to be true for any relation between a function and its approximation at the point about which the approximation was formed as long as $f(\boldsymbol{x}^{(t)}) = \tilde{f}(\boldsymbol{x}^{(t)})$ is enforced.

One would think that use of sensitivity information for f and $\tilde{f}$ at the previous point would also give the sensitivity of ρ at that point, allowing more information to be used for an even better approximation. However, from Eq. (35) it can be seen that $\Delta\tilde{f} = 0$ at $\boldsymbol{x}^{(t)}$ deems $\mathrm{d}\rho/\mathrm{d}\boldsymbol{x}$ undefined. Moreover, because

$$\left.\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{x}}\right|_{x^{(t)}} = \left.\frac{\mathrm{d}\tilde{f}}{\mathrm{d}\boldsymbol{x}}\right|_{x^{(t)}}$$

is enforced, even l'Hôpital's rule does not yield a finite $\mathrm{d}\rho/\mathrm{d}\boldsymbol{x}$, and so just as ρ was undefined using Eq. (33), so is $\mathrm{d}\rho/\mathrm{d}\boldsymbol{x}$. Unlike $\rho^{(t)}$, $(\mathrm{d}\rho/\mathrm{d}\boldsymbol{x})|_{x^{(t)}}$ has no common value applicable to all situations because it depends on the curvature error, i.e., the error in the Hessian terms. Therefore, for the purpose of this strategy, the information used in forming an approximation for $\rho(\boldsymbol{x})$ is limited to $\rho^{(t)}$, ρ^{+}, and $(\mathrm{d}\rho/\mathrm{d}\boldsymbol{x})|_{x^{+}}$ [from Eq. (35)].

One approximation technique that relies on only the aforementioned information is the two-point adaptive nonlinear approximation (TANA) method developed in Ref. 22. The basis of the TANA method is the introduction of intervening variables

$$s_i = x_i^{r}, \qquad i = 1, \ldots, n_x \tag{38}$$

where x_i are the original variables and r represents a nonlinearity index allowing the inclusion of curvature but avoiding the need for sensitivity information beyond first order. Thus, a first order expansion of a general function $z(\boldsymbol{s})$ about the $(t+1)$st iterate can be written as

$$\tilde{z}(\boldsymbol{s}) = z(\boldsymbol{s}^{+}) + \sum_{i=1}^{n_x} \left.\frac{\mathrm{d}z}{\mathrm{d}s_i}\right|_{x^{+}} \left(s_i - s_i^{+}\right) \tag{39}$$

which, on substitution from relation (38), becomes

$$\tilde{z}(\boldsymbol{x}) = z(\boldsymbol{x}^{+}) + \frac{1}{r}\sum_{i=1}^{n_x} x_i^{+^{1-r}} \left.\frac{\mathrm{d}z}{\mathrm{d}x_i}\right|_{x^{+}} \left(x_i^{r} - x_i^{+^{r}}\right) \tag{40}$$

To determine the nonlinearity index r, the value of the approximation evaluated at the point $\boldsymbol{x}^{(t)}$ is forced to equal the known actual function value at that point

$$z\big(\boldsymbol{x}^{(t)}\big) = \tilde{z}\big(\boldsymbol{x}^{(t)}\big) \tag{41}$$

which, on substitution into Eq. (40), results in the feedback formula

$$z\big(\boldsymbol{x}^{(t)}\big) = \left[z(\boldsymbol{x}^{+}) + \frac{1}{r}\sum_{i=1}^{n_x} x_i^{+^{1-r}} \left.\frac{\mathrm{d}z}{\mathrm{d}x_i}\right|_{x^{+}} \left(x_i^{(t)^{r}} - x_i^{+^{r}}\right)\right] \tag{42}$$

From Eq. (42), r can be solved for using any iterative technique. In terms of the trust region ratio, $\rho(\boldsymbol{x})$ is approximated using the TANA method as

$$\tilde{\rho}(\boldsymbol{x}) = \rho(\boldsymbol{x}^{+}) + \frac{1}{r}\sum_{i=1}^{n_x} x_i^{+^{1-r}} \left.\frac{\mathrm{d}\rho}{\mathrm{d}x_i}\right|_{x^{+}} \left(x_i^{r} - x_i^{+^{r}}\right) \tag{43}$$

where r is solved for as in Eq. (42) with $\rho(\boldsymbol{x}^{(t)}) \equiv 1$.

One might immediately recognize that design variables with negative values will result in the introduction of complex numbers in the TANA approximation. That is, a negative number x_i raised to a noninteger exponent (r or $1 - r$) is complex, a result that is useless in the present context. This limits the application of the model in Eq. (43) to problems in which the design variables cannot take on nonpositive values. One might argue that the design vectors of most engineering problems are composed of quantities that are inherently positive. Yet, this assumption cannot be accepted for the strategy to be generally applicable. Negative variables can always be posed as the difference of two variables restricted to be positive; however, this increases the dimensionality of the problem and requires the problem to be reformulated in its design vector.

A solution to this problem is realized by revisiting the goal of the strategy: to backtrack along the vector $(\alpha\boldsymbol{S}^{(t)})$ joining the current and previous points to find the step size α_{acc} that would have resulted in an acceptable amount of model error, where α is used to denote the step length along the unit vector $\boldsymbol{S}^{(t)}$,

$$\boldsymbol{S}^{(t)} = \frac{\boldsymbol{x}^{+} - \boldsymbol{x}^{(t)}}{\left\| \boldsymbol{x}^{+} - \boldsymbol{x}^{(t)} \right\|} \tag{44}$$

Thus, all that is really necessary is a model for $\rho(\alpha)$, not $\rho(\boldsymbol{x})$. This realization actually simplifies the formation of the model because it is reduced to a two-dimensional curve (in α–ρ space) rather than a $(n_x + 1)$-dimensional surface (in $\boldsymbol{x}$–ρ space). The desired model is then

$$\tilde{\rho}(\alpha) = \rho\left(\alpha^{(t)}\right) + \frac{1}{r}\alpha^{(t)^{1-r}} \left.\frac{\mathrm{d}\rho}{\mathrm{d}\alpha}\right|_{\alpha^{(t)}} \left(\alpha^{r} - \alpha^{(t)^{r}}\right) \tag{45}$$

where $\alpha^{(t)} = \|\boldsymbol{x}^{+} - \boldsymbol{x}^{(t)}\|$ and

$$\left.\frac{\mathrm{d}\rho}{\mathrm{d}\alpha}\right|_{\alpha^{(t)}} = \left[\Delta\tilde{f}\left(-\left.\frac{\mathrm{d}f}{\mathrm{d}\alpha}\right|_{\alpha^{(t)}}\right) - \Delta f\left(-\left.\frac{\mathrm{d}\tilde{f}}{\mathrm{d}\alpha}\right|_{\alpha^{(t)}}\right)\right] \Big/ (\Delta\tilde{f})^{2} \tag{46}$$

and

$$\frac{\mathrm{d}f}{\mathrm{d}\alpha} = \sum_{i=1}^{n_x} \frac{\mathrm{d}f}{\mathrm{d}x_i} S_i, \qquad S_i = \frac{x_i^{+} - x_i^{(t)}}{\alpha^{(t)}} \tag{47}$$

at any given design point $\boldsymbol{x}$. One final transformation must be made with respect to α. Because α^0, i.e., α at the start of the iteration, is zero, a singularity can be encountered in the solution of the exponent r from Eq. (42). Moreover, $\alpha^{(t)}$ can be quite large if the design variable changes are of considerable magnitude, leading to numerical difficulties in the TANA approximation. Both of these problems can be solved by the combined shift/scale transformation

$$\hat{\alpha} = \frac{\alpha + \alpha^{(t)}}{\alpha^{(t)}} \tag{48}$$

so that $\hat{\alpha}^0 = 1$ and $\hat{\alpha}^{(t)} = 2$. To account for the scaling, the sensitivity $\mathrm{d}\rho/\mathrm{d}\alpha$ must also be scaled as $(\mathrm{d}\rho/\mathrm{d}\alpha)\alpha^{(t)}$ (the shift does not affect the sensitivity). These transformations do not alter the basic formulation of the approximation. Herein the $\hat{[\,]}$ will be removed, and the transformation will be assumed.

In the following, the steps described to calculate a nonlinearity index r, a TANA approximation $\tilde{\rho}_{\mathrm{TANA}}(\alpha)$ is formed for $\rho(\alpha)$. As can be seen in Fig. 4, the TANA approximation adheres to the trend of ρ as it is forced to match values and slopes at the two endpoints as closely as possible. Such an approximation allows for a quantitative estimate of the behavior of ρ that can be used to adjust the move limits as described in the following section.

Determination of the Adjustment Factor

With an approximation for ρ at our disposal, we revisit our original question: How far could we have stepped in the previous search direction to exactly meet the trust region ratio limit? The answer to this question will provide us with a good estimate of the extent to which we should increase or decrease the size of the trust region for maximum efficiency in relation to the prescribed limit ($\rho = 0.5$) defining successful and unsuccessful moves. This is best illustrated by example. Consider a given iteration of an MDO algorithm (such as the CSSO algorithm used in this research), which involves the solution of an approximate optimization problem. Figure 5 shows two types of situations for $\rho(\alpha)$ with a step $\alpha^{(t)}$ taken in the search direction $\boldsymbol{S}$ such that

$$x_i^{+} = x_i^{(t)} + \alpha^{(t)} S_i \tag{49}$$

Figure 5a shows a situation in which $\rho < 0.5$, i.e., less than the acceptance limit, at the end of the step. By back tracking in α using the model $\tilde{\rho}_{\mathrm{TANA}}$, the step size α_{acc}, which would have resulted in a successful step ($\rho_{\mathrm{acc}} = 0.5$) can be determined. On the other hand, Fig. 5b shows a situation in which $\rho > 0.5$ at the end of the step, so that the model $\tilde{\rho}_{\mathrm{TANA}}$ is used to search forward, or explore, to determine how large of a step would have been acceptable. Mathematically, the acceptable step α_{acc} can be estimated by setting $\tilde{\rho}(\alpha) = \rho_{\mathrm{acc}}$ in Eq. (45) and solving for the corresponding α_{acc} to get

$$\alpha_{\mathrm{acc}} = \left[\left(r\left(\rho_{\mathrm{acc}} - \rho^{(t)}\right) \Big/ \alpha^{(t)^{1-r}} \left.\frac{\mathrm{d}\rho}{\mathrm{d}\alpha}\right|_{\alpha^{(t)}}\right) + \alpha^{(t)^{r}}\right]^{1/r} \tag{50}$$

A move-limit adjustment factor c is then calculated based on the ratio of this acceptable step size to the actual step size (accounting for the shift/scale transformation)

$$c = \frac{\alpha_{\mathrm{acc}} - \alpha^{(t)}/\alpha^{(t)}}{\left(\alpha^{(t)} + \alpha^{(t)}\right)/\alpha^{(t)} - \alpha^{(t)}/\alpha^{(t)}} = \alpha_{\mathrm{acc}} - 1 \tag{51}$$

The move-limit percentages are then adjusted as

$$A_i^{+} = c A_i^{(t)} \tag{52}$$

In effect we have developed a strategy that provides an adjustment factor that is continuous in nature (Fig. 6) and that provides a more intelligent/flexible move-limit management scheme. Note that the move-limit adjustment factor c is a function of $\rho^{(t)}$, $\alpha^{(t)}$, $(\mathrm{d}\rho/\mathrm{d}\alpha)|_{\alpha^{(t)}}$, and r, and, therefore, its shape as shown in Fig. 6 changes with each iteration.

Adaptation for Constrained Optimization

To this point, the strategy has been developed by focusing on the minimization of a single function f without consideration of other possible influencing quantities, namely, constraints $(\boldsymbol{g}, \boldsymbol{h})$. It has been mentioned, however, that extension of the strategy to include

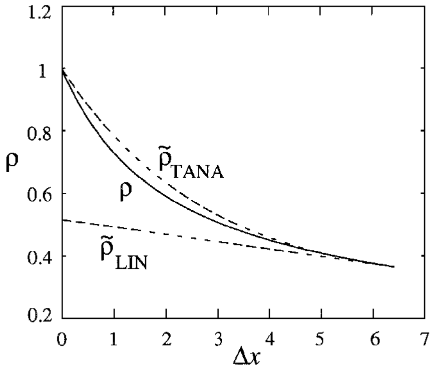


Fig. 4 TANA approximation for the trust region ratio.

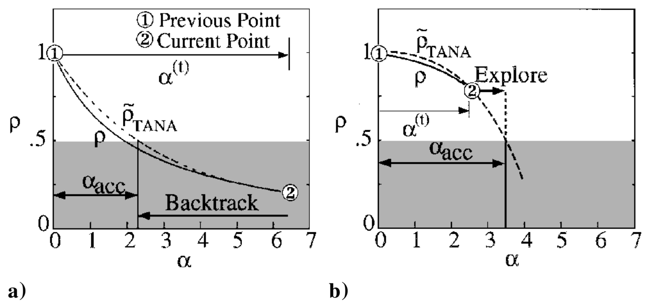


Fig. 5 Determination of the trust region adjustment factor.

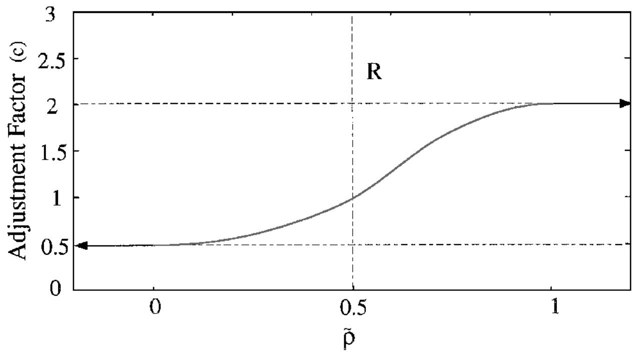


Fig. 6 Continuous trust region adjustment.

constraints is possible by the introduction of a penalty function.[20] In this research, a modified form of the Lagrangian penalty function

$$\Phi(\boldsymbol{x},\boldsymbol{\lambda},\nu,r_p)=f(\boldsymbol{x})+|\boldsymbol{\lambda}^T\boldsymbol{g}(\boldsymbol{x})|+|\nu^T\boldsymbol{h}(\boldsymbol{x})|$$
$$+r_p[\boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{g}(\boldsymbol{x})+\boldsymbol{h}(\boldsymbol{x})^T\boldsymbol{h}(\boldsymbol{x})] \qquad (53)$$

is employed in the strategy. To supply reasoning for the modification to be described, a closer inspection of the components of the Lagrangian is required.

Significance of the Lagrange Multipliers

Recall that the Lagrangian in its original form is

$$L(\boldsymbol{x},\boldsymbol{\lambda},\nu)=f(\boldsymbol{x})+\boldsymbol{\lambda}^T\boldsymbol{g}(\boldsymbol{x})+\nu^T\boldsymbol{h}(\boldsymbol{x}) \qquad (54)$$

where $\boldsymbol{\lambda}$ and ν are vectors of Lagrange multipliers corresponding to active (or violated) inequality and equality constraints, respectively (the multipliers are set to 0 for inactive constraints). Note that when the design is feasible the Lagrangian reduces to the objective of the original problem. For simplicity, only inequality constraints and their multipliers will be included in the following formulation, although the formulation also holds for equality constraints. The Lagrange multipliers are not just arbitrary constants to impose a penalty, they actually have a meaning in relation to the design space. A necessary condition of optimality defined by the Kuhn–Tucker conditions[23] is that

$$\nabla_x L=\nabla_x f+\boldsymbol{\lambda}\nabla_x\boldsymbol{g}=0 \qquad (55)$$

In other words, if $\nabla_x L$ is minimized, then a minimizer of the original optimization problem is obtained; no further improvement can be made starting from this design point. From Eq. (55), one can see that the multipliers are constants that result in a linear combination of the constraint gradients (the feasible direction) that exactly offsets the objective gradient (the usable direction). For the purpose of example, consider a one-dimensional problem with one constraint. In that case, Eq. (55) can be solved for $\boldsymbol{\lambda}$ as

$$\boldsymbol{\lambda}=-\frac{\mathrm{d}f}{\mathrm{d}x}\bigg/\frac{\mathrm{d}g}{\mathrm{d}x}=-\frac{\mathrm{d}f}{\mathrm{d}g} \qquad (56)$$

Thus, $\boldsymbol{\lambda}$ provides a measure of the sensitivity of the objective function with respect to the constraint. In other words, if the bound of a given constraint g_i were shifted by a certain amount Δg_i (which can be interpreted as the amount of constraint violation), $\boldsymbol{\lambda}_i$ can be used to give a linear estimate of the change in the objective function $\Delta f_{(i)}$ through a corresponding shift of the objective contour by

$$\Delta f_{(i)}=-\boldsymbol{\lambda}_i\Delta g_i \qquad (57)$$

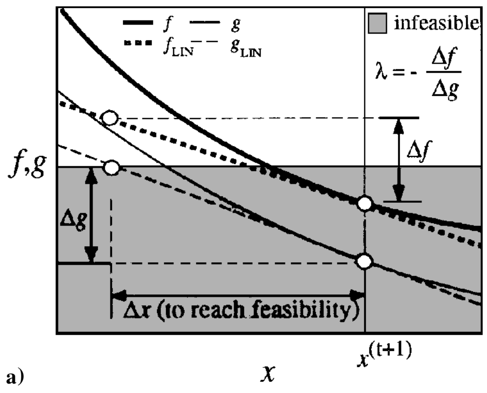
This same meaning holds true in a design space of any dimension with any number of constraints, although $\boldsymbol{\lambda}$ must then be solved for in a least-squares sense, as described in the following section. Thus, the rationale behind the multiplier terms $|\boldsymbol{\lambda g}|$ in the modified Lagrangian of Eq. (53) is that they penalize the merit function by linear estimates of the amount of objective function gain realized in violation of each constraint. This is shown in Fig. 7a. The absolute value is a conservative measure and is required to enforce consistency in the sign of the penalty because the method used to calculate $\boldsymbol{\lambda}$ (described in the following section) does not restrict the sign.

Calculation of Lagrange Multipliers

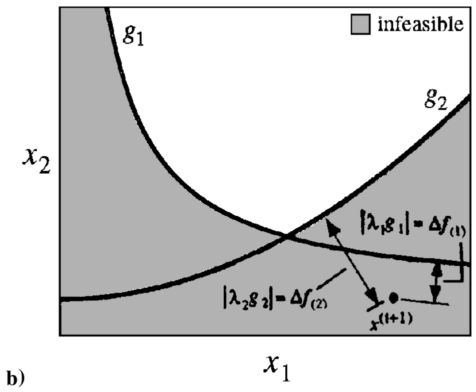
Direct calculation of the Lagrange multipliers $\boldsymbol{\lambda}$ via Eq. (56) is, in general, not possible because Eq. (55) is an overdetermined linear system unless the number of constraints equals the number of design variables. Therefore, a solution of the multipliers is sought by solving the normal equations of the linear least-squares problem

$$\boldsymbol{\lambda}=-[\nabla\boldsymbol{g}^T\nabla\boldsymbol{g}]^{-1}\nabla\boldsymbol{g}^T\nabla f \qquad (58)$$

This method of solving for the multipliers was first introduced in Ref. 24 and is a straightforward execution of matrix operations on the readily available Jacobians of the problem. Equation (58) is also referred to as a projection formula as it arises when ∇f is projected onto the tangent subspace of the constraints. In this research, a slightly different approach is taken to preserve the meaning of the



f — g
f_{LIN} — g_{LIN}
▨ infeasible
$\lambda=-\dfrac{\Delta f}{\Delta g}$
Δf
f,g
Δg
Δx (to reach feasibility)
$x^{(t+1)}$
a)
x



g_1
▨ infeasible
g_2
x_2
$|\lambda_1 g_1|=\Delta f_{(1)}$
$|\lambda_2 g_2|=\Delta f_{(2)}$
$x^{(t+1)}$
b)
x_1

Fig. 7 Penalty concept of the Lagrange multipliers.

multipliers in relation to each individual constraint. Rather than solving for the multipliers of all active constraints simultaneously, Eq. (58) is solved considering each active constraint individually so that

$$\boldsymbol{\lambda}=-\left[\nabla g_i^T\nabla g_i\right]^{-1}\nabla g_i^T\nabla f \qquad (59)$$

The multipliers obtained by solving Eq. (59) for each g_i do not necessarily satisfy the system of Eq. (58). Yet, the way in which the multipliers are used in the present application does not require this satisfaction. Instead, it is more important that the multipliers retain the interpretation of a penalty based on the amount of function gain realized in violation of each constraint. This concept is shown in Fig. 7b.

Additional Penalty

The multiplier terms in Eq. (53) have been described as penalties proportional to the amount of function gain realized in violation of the constraint bounds. In other words, they are a way of returning the merit function to the value it would have had if it had stopped at the constraint bound (by linear approximation). As a means of further penalizing the function based on the extent of constraint violation, the additional term in Eq. (53) is supplied as a pure penalty proportional to the amount of constraint violation, fashioned after the bracket operator.[23] The constant penalty parameter r_p is typically set to be $\mathcal{O}(10)$, although an updating scheme may be used to adjust this parameter as the optimization proceeds.[25]

Application to the Trust Region Ratio

Making use of the modified Lagrangians at the current and previous design iterates, the trust region ratio of Eq. (33) can now be redefined as

$$\rho^{(t)}=\frac{\Phi^{(t)}-\Phi^{+}}{\Phi^{(t)}-\tilde{\Phi}^{+}} \qquad (60)$$

where

$$\Phi^{(t)}=\tilde{f}^{(t)}+\left|\boldsymbol{\lambda g}^{(t)}\right|+\left|\nu\boldsymbol{h}^{(t)}\right|+r_p\left(\boldsymbol{g}^{(t),T}\boldsymbol{g}^{(t)}+\boldsymbol{h}^{(t),T}\boldsymbol{h}^{(t)}\right)$$
$$\Phi^{+}=\tilde{f}^{+}+|\boldsymbol{\lambda g}^{+}|+|\nu\boldsymbol{h}^{+}|+r_p(\boldsymbol{g}^{+,T}\boldsymbol{g}^{+}+\boldsymbol{h}^{+,T}\boldsymbol{h}^{+})$$
$$\tilde{\Phi}^{+}=\tilde{f}^{+}+|\boldsymbol{\lambda}\tilde{\boldsymbol{g}}^{+}|+|\nu\tilde{\boldsymbol{h}}^{+}|+r_p(\tilde{\boldsymbol{g}}^{+,T}\tilde{\boldsymbol{g}}^{+}+\tilde{\boldsymbol{h}}^{+,T}\tilde{\boldsymbol{h}}^{+})$$

Note that, when no constraints are active, Eq. (60) reduces to the standard trust region ratio for unconstrained optimization as defined in Eq. (33). Implementation with this form of the trust region ratio requires care in active constraint set determination; that is, the active constraint set may switch from the beginning of an iteration to the end, requiring different constraints to be included in the calculation of the current (Φ^+, $\tilde{\Phi}^+$) and previous ($\Phi^{(t)}$) modified Lagrangian values. For consistent comparison of Φ values in the calculation of ρ, it is necessary to use a common multiplier for any constraint that is active in both the current and previous designs. This is done to ensure that improvement in feasibility is accurately portrayed; if different λ are calculated, the nonlinearity of the design space can affect the relationship portrayed in comparing the linearizations for penalties as described earlier. Thus, for consistency, when a given constraint is active in more than one design, the most violated version of the constraint is used to determine λ.

Point Rejection Scheme

Recall that when $\rho < 0$, the iteration is considered a complete failure; in such a case, to guarantee convergence, the new point must be rejected, the trust region must be reduced, and the approximate optimization must be repeated. In fact, the proof of convergence for trust region algorithms is reliant on the rejection of bad iterates (in which the merit function increased). The most straightforward approach, which is commonly used, is to reduce the move limits by one-half in the case of failure. However, as shown in Fig. 8, this halving may result in a number of successive rejections, depending on the proximity of the initial point to the minimum (or constraint bound) and the amount of error in the approximations (faulty direction of search). Each additional rejection increases the cost of the algorithm by one system analysis (one iteration).

The ideal reduction factor to guarantee $\rho > 0$ may actually be significantly less than 0.5. Moreover, temporary cycling may still occur even when $\rho > 0$ because the minimum may still be overstepped. In fact, at this point it is logical to attempt to determine the minimum of the merit function Φ in the current search direction because we are sure that it is bounded by the previous point $\boldsymbol{x}^{(t)}$ and the new candidate design $\boldsymbol{x}^{(t+1)}$ (which is being rejected). Such a strategy is offered in Ref. 14. In the research presented here, a similar but arguably more accurate approach is taken, which utilizes an approximation of the function based on sensitivity information at $\boldsymbol{x}^{(t)}$ and $\boldsymbol{x}^{(t+1)}$. This point rejection scheme is detailed in the Appendix.

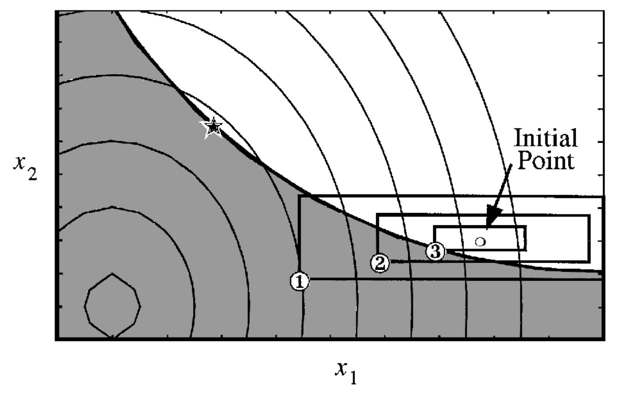


Fig. 8 Inefficiency of the halving method for point rejection.

IV. Implementation Details

Because the move-limit management strategy described in the preceding sections is based on approximating the trust region ratio ρ, herein it will be referred to as the trust region ratio approximation method (TRAM) strategy. Having presented the theory behind the TRAM move-limit management strategy developed in this research, the steps of the strategy can be clearly defined with a detailed discussion of the pertinent implementation issues. A flowchart for implementing the TRAM strategy is given in Fig. 9. In each iteration of an approximate optimization algorithm, the following steps are executed to adjust the design variable move limits. For brevity, certain references are made to equations presented in the earlier sections.

1) Estimate the design progress/model behavior: First, calculate lagrange multipliers. For the actual and approximate new designs [$+$ and ($\tilde{+}$), respectively] and the actual previous design (t), calculate the Lagrange multipliers $\boldsymbol{\lambda}$ from Eq. (58) for the active constraints of each design. For a constraint that is active in more than one design, use a consistent λ calculated at the design for which that constraint is most infeasible. Second, calculate the TRAM merit functions. For the designs $+$, ($\tilde{+}$), and (t), calculate the corresponding TRAM merit functions Φ^+, $\tilde{\Phi}^+$, and $\Phi^{(t)}$ according to Eq. (53). Last, calculate the trust region ratio. Using the calculated TRAM merit functions, evaluate the trust region ratio $\rho^{(t)}$ as in Eq. (60).

2) Analyze the situation: Determine the course of action according to the following guidelines. For $\rho \leq 0$, reject the point and implement the point-rejection scheme (approximate Φ^+ with a TANA-2 approximation, find the minimum of $\Phi_{\text{TANA-2}}$, and calculate the adjustment factor c). Go to step 5. For $\rho > 1$, set $c = 2$. Go to step 5. For $0 < \rho \leq 1$, continue with step 3 to implement the TRAM strategy.

3) Model the trust region ratio: Calculate the TANA exponent for $\rho(\alpha)$. Use $\rho^{(t-1)} \equiv 1$, $\rho^{(t)}$, and $(\mathrm{d}\rho/\mathrm{d}\alpha)|_{\alpha^t}$ to determine a TANA exponent r [apply the shift/scale transformation of Eq. (48) to avoid numerical difficulties]. In this research, it was found that for robustness, i.e., to ensure global convergence in the solution of r, it was necessary to implement a hybrid root solving algorithm. This algorithm, as described in Ref. 26, takes a bisection step whenever Newton–Raphson provides a value beyond some prescribed bounds, or whenever Newton–Raphson is not reducing the size of the bracketed region rapidly enough.

4) Backtrack/explore: Solve for the acceptable step size α_{acc}. Using the model for $\rho(\alpha)$ calculated in step 3, backtrack (if $\rho < 0.5$) or explore forward (if $\rho > 0.5$) along the previous search direction to find α_{acc} for which $\rho_{\text{TANA}}(\alpha) = 0.5$ (use a root solving algorithm). To avoid divergence, place limits of $1.5 \leq \alpha_{\text{acc}} < 3$ on α_{acc} corresponding to restriction of the adjustment factor to $0.5 \leq c \leq 2$. Then solve for the adjustment factor. Use α_{acc} to calculate the move-limit adjustment factor $c = \alpha_{\text{acc}} - 1$.

5) Adjust the move limits: Calculate the move limits for the subsequent approximate optimization procedure as

$$A_i^+ = cA_i^{(t)} \tag{61}$$

Steps 1–5 formally define the TRAM strategy for automatically managing the design variable move limits in an approximate optimization algorithm. The quantities that must be obtained prior to executing the TRAM strategy are listed preceding the flowchart in Fig. 9. These quantities require that a system analysis and a sensitivity analysis be carried out on the current design iterate and that similar information be available for the previous design iterate. These

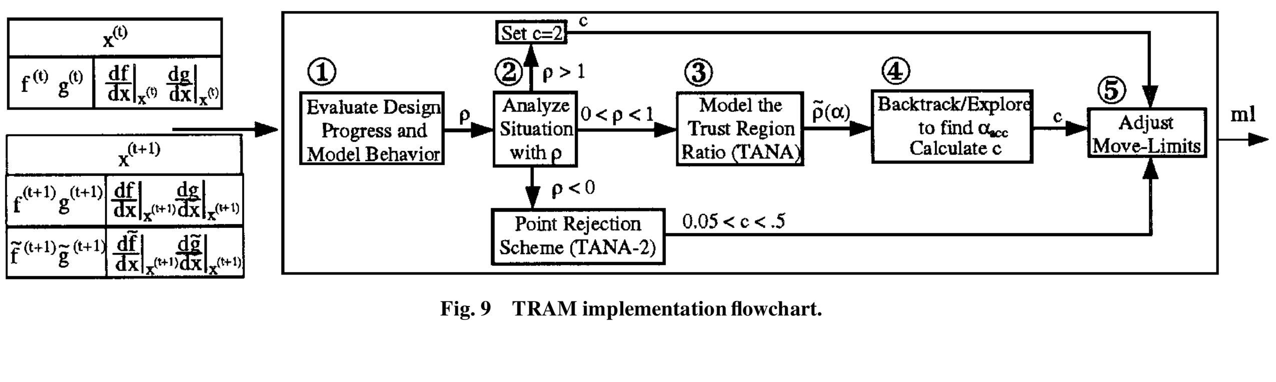

$x^{(t)}$
$f^{(t)}$ $g^{(t)}$ $\left.\frac{df}{dx}\right|_{x^{(t)}}$ $\left.\frac{dg}{dx}\right|_{x^{(t)}}$
$x^{(t+1)}$
$f^{(t+1)}$ $g^{(t+1)}$ $\left.\frac{df}{dx}\right|_{x^{(t+1)}}$ $\left.\frac{dg}{dx}\right|_{x^{(t+1)}}$
$\tilde{f}^{(t+1)}$ $\tilde{g}^{(t+1)}$ $\left.\frac{d\tilde{f}}{dx}\right|_{x^{(t+1)}}$ $\left.\frac{d\tilde{g}}{dx}\right|_{x^{(t+1)}}$
1
Evaluate Design Progress and Model Behavior
ρ
2
Analyze Situation with ρ
Set c=2
c
$\rho > 1$
$0 < \rho < 1$
$\rho < 0$
3
Model the Trust Region Ratio (TANA)
$\tilde{\rho}(\alpha)$
4
Backtrack/Explore to find α_{acc} Calculate c
c
5
Adjust Move-Limits
ml
Point Rejection Scheme (TANA-2)
$0.05 < c < .5$

Fig. 9 TRAM implementation flowchart.

procedures are common to approximate optimization algorithms so that no extra analysis burden is placed on the designer. The only additional cost associated with implementing the TRAM strategy is in executing the root solving algorithms to obtain both r (in step 3) and α_{acc} in step 4. However, because these are one-dimensional problems of simple analytic functions with available analytic gradients, the cost can be considered rather insignificant relative to the cost of other MDO procedures (contributing analyses, subspace optimizations, etc.).

V. Conclusions

In this research, issues of move-limit management are reviewed and a new adaptive strategy for move-limit management is developed. With its basis in the provably convergent trust region methodology, the TRAM strategy utilizes available gradient information and employs a backtracking process using various two-point approximation techniques to provide a flexible move-limit adjustment factor. The TRAM strategy provides a more flexible move-limit adjustment factor than direct trust region methods through the use of trust region ratio sensitivities. Coupled with a new approximation-based point rejection scheme, use of the TRAM strategy results in less approximation error and leads to faster algorithm convergence as observed in a companion paper. The development of the TRAM strategy provides improved efficiency in the execution of approximate optimization algorithms.

Appendix: Point Rejection Scheme

A new strategy for reducing the size of the trust region in the case of design point rejection ($\rho < 0$) is formulated here. Essentially, the problem is reduced to a univariate search in α in the direction defined for the current iteration,

$$S^{(t)} = \frac{x^+ - x^{(t)}}{\|x^+ - x^{(t)}\|} \tag{A1}$$

First, an approximation to the merit function Φ must be formed in the region defined by $\alpha \in (0, \alpha^{(t)})$, where $\alpha^{(t)} = \|x^+ - x^{(t)}\|$. The derivative of Φ with respect to α is defined as

$$\frac{d\Phi}{d\alpha} = \sum_{i=1}^{n_x} \frac{d\Phi}{dx_i} S_i \tag{A2}$$

or more specifically, at the previous and candidate points, respectively,

$$\left.\frac{d\Phi}{d\alpha}\right|_{x^{(t)}} = \sum_{i=1}^{n_x} \left.\frac{d\Phi}{dx_i}^T\right|_{x^{(t)}} S_i \quad \text{and} \quad \left.\frac{d\Phi}{d\alpha}\right|_{x^+} = \sum_{i=1}^{n_x} \left.\frac{d\Phi}{dx_i}^T\right|_{x^+} S_i \tag{A3}$$

This provides first-order information describing how Φ will change with the step size α at both the previous and candidate design points. However, it is obvious that a linear approximation is not suitable for the present purpose. The main decision to be made then is the type of approximation that should be employed. In Ref. 14, the gradients of Eq. (64) are used in a simple fashion to approximate curvature via a divided difference technique. This neglects any cross terms, which could have been used to model the effects of design variable interaction; thus, the accuracy of the approximation is questionable in functions that contain design variable interactions, a typical characteristic of multidisciplinary systems. This inaccuracy is evident in Fig. A1, where for simplification Φ is represented by the function $f = (x-1)^2(y-2)^2$ along a given search direction, i.e., $f(\alpha)$. Attempting to estimate the curvature using merely slope differences not only neglects cross terms, but also does not guarantee a function match at the candidate point x^+. However, note in the plots formed using actual second-order information at the previous point $x^{(t)}$ that the inclusion of cross term Hessian components does not ensure any greater accuracy because the curvature may change over the range of the step. The ideal approximation would match both the function value and the slope at the current design and at the candidate design to provide an accurate approximation over the entire step.

Such an approximation is offered by a variant of the TANA method described earlier for modeling ρ. The TPEA developed in

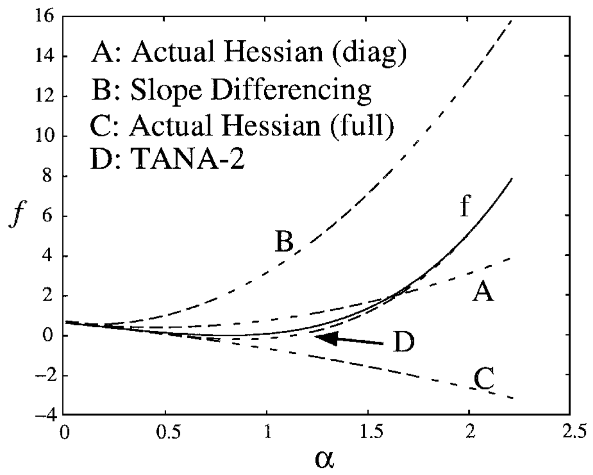


Fig. A1 Various approximations for f within the current trust region.

Ref. 12 (and described earlier in the review of move-limit strategies) and the TANA method are combined in Ref. 27 to produce an improved two-point approximation, TANA-2, which matches both the function values and the gradients at the points $x^{(t)}$ and $x^{(t+1)}$. Just as in the TANA method, the approximation is based on the introduction of intervening variables; however, in this method a coefficient is determined for each variable such that

$$s_i = x_i^{p_i}, \qquad i = 1, 2, \ldots, n_x \tag{A4}$$

The TANA-2 approximation is formed about the candidate design as

$$\tilde{z}(x) = z(x^+) + \sum_{i=1}^{n_x} \left.\frac{dz}{dx_i}\right|_{x^+} \frac{x_i^{+1-p_i}}{p_i}\left(x_i^{p_i} - x_i^{+p_i}\right) + \frac{1}{2}\epsilon_2 \sum_{i=1}^{n_x}\left(x_i^{p_i} - x_i^{+p_i}\right)^2 \tag{A5}$$

or for the desired univariate search in α

$$\tilde{z}(\alpha) = z\left(\alpha^{(t)}\right) + \left.\frac{dz}{d\alpha}\right|_{\alpha^{(t)}} \frac{\alpha^{(t)^{1-p_\alpha}}}{p_\alpha}\left(\alpha^{p_\alpha} - \alpha^{(t)^{p_\alpha}}\right) + \frac{1}{2}\epsilon_2\left(\alpha^{p_\alpha} - \alpha^{(t)^{p_\alpha}}\right)^2 \tag{A6}$$

The inclusion of ϵ_2 is identical to adding second-order Taylor series effects, in which the Hessian matrix has only diagonal elements of value ϵ_2. However, because this approximation is expanded in terms of the intervening variables s, the error from the approximate Hessian is accounted for by adjusting the nonlinearity indices p. The $n_x + 1$ unknown quantities (p_i and ϵ_2) in Eq. (A5) require $n_x + 1$ equations for solution; however, in Eq. (A6), only two unknowns (p_α and ϵ_2) exist requiring two equations. Thus, it can be seen that reduction to a univariate search in α greatly reduces the computational burden of forming the approximation. By matching the derivatives at the previous point $x^{(t)}$ (or α^0), a first equation is obtained of the form

$$E_1 = \left.\frac{dz}{d\alpha}\right|_{\alpha^0} - \left(\frac{\alpha^0}{\alpha^{(t)}}\right)^{p_\alpha - 1} \left.\frac{dz}{d\alpha}\right|_{\alpha^{(t)}} + \epsilon_2\left(\alpha^{p_\alpha} - \alpha^{(t)^{p_\alpha}}\right)\left(\alpha^0\right)^{p_\alpha - 1} p_\alpha = 0 \tag{A7}$$

The other equation required is obtained by forcing the approximate function value to meet that of the actual function at $x^{(t)}$ (or α^0), that is,

$$E_2 = z(\alpha^0) - z\left(\alpha^{(t)}\right) + \left.\frac{dz}{d\alpha}\right|_{\alpha^{(t)}} \frac{\alpha^{(t)^{1-p_\alpha}}}{p_\alpha}\left(\alpha^{0,p_\alpha} - \alpha^{(t)^{p_\alpha}}\right) + \frac{1}{2}\epsilon_2\left(\alpha^{0^{p_\alpha}} - \alpha^{(t)^{p_\alpha}}\right)^2 = 0 \tag{A8}$$

As in the approximation for ρ, the shift/scale transformation of the α axis is required to avoid $\alpha^0 = 0$, i.e., at the start of the iteration, and to avoid numerical difficulties. Solution of the set of nonlinear equations

$$E_1(p_\alpha, \epsilon_2) = 0, \qquad E_2(p_\alpha, \epsilon_2) = 0 \tag{A9}$$

can be carried out using various existing algorithms. In this research, the International Mathematical and Statistical Library (IMSL)[28] routine DNEQBJ was utilized for this purpose. This routine uses a secant algorithm to solve a nonlinear set of equations provided that the Jacobian is given. For the problem at hand, the Jacobian terms are derived and listed in Ref. 3.

The solution of Eq. (A9) provides an approximate Hessian term ϵ_2 and a curvature correcting exponent p_α to form the TANA-2 approximation to the objective function. As seen in Fig. A1, this approximation does an excellent job of modeling a function in one dimension because it attempts to match both function and gradient values at the current and previous points.

In the case of point rejection, the TANA-2 model of the objective is used to estimate the bypassed minimum of the objective. This can be carried out using any minimization or root-solving algorithm. This research makes use of the IMSL routine DUVMID, which uses the secant method combined with cubic interpolation to find the minimum of the function Φ^* along the previous search direction. This algorithm requires calculation of the gradient of the function, which for the TANA-2 approximation is

$$\widetilde{\frac{\mathrm{d}\Phi}{\mathrm{d}\alpha}} = \left(\frac{\alpha}{\alpha^{(t)}}\right)^{p_\alpha - 1} \frac{\mathrm{d}\Phi}{\mathrm{d}\alpha}\bigg|_{\alpha^{(t)}} + \epsilon_2\left(\alpha^{p_\alpha} - \alpha^{(t)^{p_\alpha}}\right)(\alpha)^{p_\alpha - 1} p_\alpha \tag{A10}$$

Once an estimate for Φ^* is obtained, the corresponding α^* can be used to calculate the move-limit reduction factor by

$$c = \alpha^* - 1 \tag{A11}$$

Because this rejection scheme is based on an approximation, to guard against reduction of the move limits to zero, a lower limit of 0.05 is placed on the reduction factor. The move limits for the next iteration are calculated by substituting the reduction factor into Eq. (52).

Acknowledgments

This multidisciplinary research effort was supported in part by the following grants and contracts: NASA Research Grant Award NAG-1-1561, National Science Foundation (NSF) Grant DMI93-08083, and NSF Grant DMI94-57179.

References

[1]Wujek, B. A., and Renaud, J. E., "New Adaptive Move-Limit Management Strategy for Approximate Optimization, Part 2," *AIAA Journal*, Vol. 36, No. 10, 1998, pp. 1922–1934; also AIAA Paper 98-1966, April 1998.

[2]Wujek, B. A., Renaud, J. E., Batill, S. M., and Brockman, J. B., "Concurrent Subspace Optimization Using Design Variable Sharing in a Distributed Computing Environment," *Proceedings of the 21st ASME Design Automation Conference*, Vol. 1, DE-Vol. 82, American Society of Mechanical Engineers, New York, 1995, pp. 181–188.

[3]Wujek, B. A., "Automation Enhancements in Multidisciplinary Design Optimization," Ph.D. Dissertation, Dept. of Aerospace and Mechanical Engineering, Univ. of Notre Dame, Notre Dame, IN, 1997.

[4]Pourazady, M., and Fu, Z., "Integrated Approach to Structural Shape Optimization," *Computers and Structures*, Vol. 60, No. 2, 1996, pp. 279–289.

[5]Haftka, R. T., Gurdal, Z., and Kamat, M. P., *Elements of Structural Optimization*, 2nd ed., Kluwer, Dordrecht, The Netherlands, 1990.

[6]Thomas, H. L., Vanderplaats, G. N., and Shyy, Y.-K., "A Study of Move Limit Adjustment Strategies in the Approximation Concepts Approach to Structural Synthesis," *Proceedings of the AIAA/USAF/NASA/OAI 4th Symposium on Multidisciplinary Analysis and Optimization* (Cleveland, OH), AIAA, Washington, DC, 1992, pp. 507–512 (AIAA Paper 92-4750).

[7]Bloebaum, C. L., Hong, W., and Peck, A., "Improved Move Limit Strategy for Approximate Optimization," *Proceedings of the AIAA/USAF/NASA/ISSMO 5th Symposium on Multidisciplinary Analysis and Optimization* (Panama City, FL), AIAA, Washington, DC, 1994, pp. 843–850 (AIAA Paper 94-4337).

[8]Hajela, P., and Sobieszczanski-Sobieski, J., "The Controlled Growth Method—A Tool for Structural Optimization," *AIAA Journal*, Vol. 20, No. 10, 1982, pp. 1440, 1441.

[9]Kreisselmeier, G., and Steinhauser, R., "Systematic Control Design by Optimizing a Vector Performance Index," *Proceedings of the International Federation of Active Control Symposium on Computer Aided Design of Control Systems* (Zurich, Switzerland), 1979, pp. 113–117.

[10]Chen, T.-Y., "Calculation of the Move Limits for the Sequential Linear Programming Method," *International Journal for Numerical Methods in Engineering*, Vol. 36, No. 15, 1993, pp. 2661–2679.

[11]Fadel, G. M., and Cimtalay, S., "Automatic Evaluation of Move-Limits in Structural Optimization," *Structural Optimization*, Vol. 6, No. 4, 1993, pp. 233–237.

[12]Fadel, G. M., Riley, M. F., and Barthelemy, J. F. M., "Two Point Exponential Approximation Method for Structural Optimization," *Structural Optimization*, Vol. 2, No. 2, 1990, pp. 117–124.

[13]Grignon, P., and Fadel, G. M., "Fuzzy Move Limit Evaluation in Structural Optimization," AIAA Paper 94-4281, Sept. 1994.

[14]Dennis, J. E., and Schnabel, R. R. B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice–Hall, Englewood Cliffs, NJ, 1983.

[15]Conn, A. R., Gould, N. I. M., and Toint, P. L., "LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization," Springer Series in Computational Mathematics, Vol. 17, Springer–Verlag, Heidelberg, Germany, 1992.

[16]Sunar, M., and Belegundu, A. D., "Trust Region Methods for Structural Optimization Using Exact Second Order Sensitivity," *International Journal for Numerical Methods in Engineering*, Vol. 32, No. 2, 1991, pp. 275–293.

[17]Lewis, R. M., "A Trust Region Framework for Managing Approximation Models in Engineering Optimization," *Proceedings of the AIAA/NASA/ISSMO 6th Symposium on Multidisciplinary Analysis and Optimization* (Bellevue, WA), AIAA, Reston, VA, 1996, pp. 1053–1055 (AIAA Paper 96-4101).

[18]Powell, M. J. D., "A Hybrid Method for Nonlinear Equations," *Numerical Methods for Nonlinear Algebraic Equations*, edited by P. Rabinowitz, Gordon and Breach, London, 1970, pp. 87–114.

[19]Alexandrov, N., Dennis, J. E., Lewis, R. M., and Torczon, V., "A Trust Region Framework for Managing the Use of Approximation Models in Optimization," *Structural Optimization*, Vol. 15, No. 1, 1998, pp. 16–23.

[20]Bachem, A., Grötschel, M., and Korte, B., *Mathematical Programming: The State of the Art*, Springer–Verlag, Berlin, 1982.

[21]Conn, A. R., Gould, N. I. M., and Toint, P. L., "Global Convergence of a Class of Trust Region Algorithms for Optimization with Simple Bounds," *SIAM Journal of Numerical Analysis*, Vol. 25, No. 2, 1988, pp. 433–464.

[22]Wang, L. P., and Grandhi, R. V., "Efficient Safety Index Calculation for Structural Reliability Analysis," *Computers and Structures*, Vol. 52, No. 1, 1994, pp. 103–111.

[23]Reklaitis, G. V., Ravindran, A., and Ragsdell, K. M., *Engineering Optimization: Methods and Applications*, Wiley, New York, 1983.

[24]Rosen, J. B., "The Gradient Projection Method for Non-Linear Programming, I.—Linear Constraints," *Journal of the Society of Industrial and Applied Mathematics*, Vol. 8, No. 1, 1960, pp. 181–217.

[25]Arora, J. S., Chahande, A. I., and Paeng, J. K., "Multiplier Methods for Engineering Optimization," *International Journal for Numerical Methods in Engineering*, Vol. 32, No. 7, 1991, pp. 1485–1525.

[26]Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes in FORTRAN*, Cambridge Univ. Press, New York, 1992.

[27]Wang, L. P., and Grandhi, R. V., "Improved Two-Point Function Approximation for Design Optimization," *AIAA Journal*, Vol. 33, No. 9, 1995, pp. 1720–1727.

[28]"User's Manual, Fortran Subroutines for Mathematical Applications," Vol. 3, International Mathematical and Statistical Libraries, Houston, TX, 1997.

A. D. Belegundu
Associate Editor