CHAPTER III

DESIGN OPTIMIZATION STRATEGY

3.1. Introduction

The flowchart shown in Figure 3.1.1 depicts the general strategy for design optimization used in this research, of which each component will be discussed in subsequent sections. The vector of design variables is denoted $\vec{\beta}_n$, the discretized grid is denoted by χ, the steady-state flow variables are given by Q, and F is the objective function to be optimized. For this research, the flow analysis is performed by a finite element solver of the shallow water equations, the calculation of the derivative of the objective function with respect to design variables is performed by the direct formulation of discrete sensitivity analysis, and the new design variables are determined via a modification of the Gauss-Newton optimization algorithm. Flow prediction is an added benefit of discrete sensitivity analysis which provides a better initial guess for the flow variables for the new design and can be used in linear searches within the optimization algorithm. The grid generation subroutine and the choice of the objective function are problem dependent. Finally, the termination criteria can be based on the change in the objective function, the magnitude of the design space gradient, or the change in the design variables. The choice of each of the components in the flowchart is discussed in detail later in this chapter.

Figure 3.1.1. Flowchart for Design Optimization.

3.2. Grid Generation

An integral part of efficient design optimization is the streamlining of the grid generation process. For an analysis code, a researcher may spend a large amount of time using a grid generation code to develop the grid to represent the discretized domain. Typically, the researcher is interested in the flow for only one design and hence only one grid. But in design optimization, multiple grids are used, one for each

set of design variables. Furthermore, if gradient-based optimization is used, the design variables are slightly perturbed to obtain the derivatives, as discussed in connection with equations (2.1.3), (2.1.5), (2.1.12) and (2.1.21). A grid is generated for each set of perturbed design variables. If the researcher must be involved in each grid generation, the grid generation portion of the design optimization process would be extremely time consuming. To streamline the grid generation process, the researcher wants a black box that takes the design variables and grid parameters as inputs and outputs the grid that is ready for simulation, as shown in Figure 3.2.1.

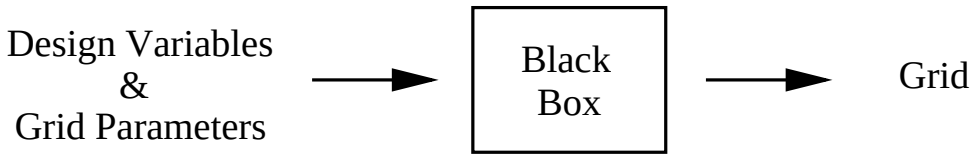

Figure 3.2.1. Streamlined Grid Generation Process.

At least three different alternatives exist for streamlining this process - integration of an existing grid generation code within the optimization code, developing and encoding a set of rules based on the design variables and grid parameters, and inputting and modifying an existing grid based on the design variables. Many existing grid generation codes allow the user to save the steps taken to generate a code as a script that can be modified and re-executed with the grid generator. Thus, once the researcher has developed the initial grid for the design problem, he can modify the script to account for new sets or perturbed sets of design variables. This method requires a low level of coupling between the analysis code and the grid generation code and thus can be rapidly incorporated into the optimization process; however,

the entire grid must be re-generated for each set of design variables.

The grid generation method employed in this research involves developing and encoding a set of rules to generate the grid based on inputted design variables and grid parameters. Many of the design variables and grid parameters control the shape of the boundary, as well as the grid spacing. Once the boundary is determined, a structured, multi-block grid is generated. The rules are different for each design problem, requiring a large amount of development time for setting up the design problem. However, once the rules are encoded, a tremendous amount of flexibility in regards to the grid generation is available to the researcher.

A third streamlined grid generation process inputs a grid as developed by an existing grid generation code, as well as directions for the grid's dependence on the design variables. Since design variables typically control the shape of the boundaries, the amount of perturbation along the boundaries of the original grid are determined from the design variables by using these directions. Once these boundary perturbations are established, they can be propagated into the grid via a Laplace solver. This process is shown in Figure 3.2.2 for a channel contraction.

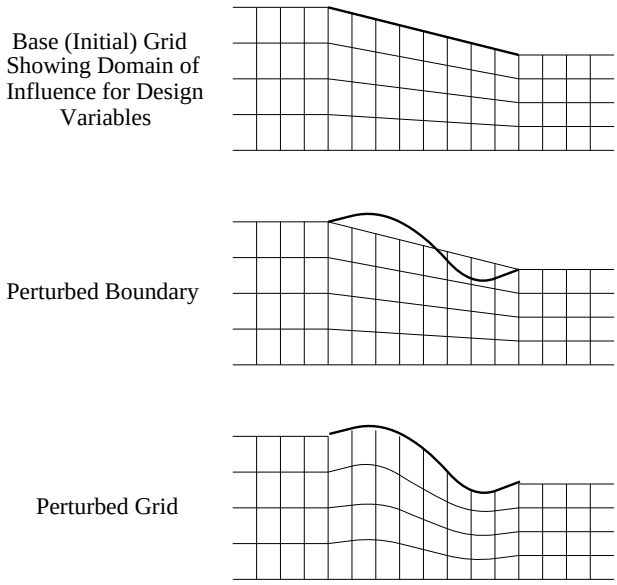

Figure 3.2.2. Boundary Perturbation Propagation Method.

In regards to discrete sensitivity analysis, the change in the grid χ with respect to the design variable β_i, or $\frac{\partial \chi}{\partial \beta_i}$, must be calculated for each design variable. Within this research, this term is calculated via finite differencing the grid locations for the grids associated with the perturbed design variables. (Actually, the residual vector $W(\vec{\beta} + e_i \Delta\beta)$ is evaluated and used to estimate the term $\left.\frac{dW}{d\beta_i}\right|_{textQfixed}$, which incorporates the perturbed grids.) Often the term $\frac{\partial \chi}{\partial \beta_i}$ is calculated by hand-differentiation or by using automatic differentiation to determine the exact value for this term. This technique requires additional work is required to derive this term but the grids do not need to be regenerated and more accurate results can be obtained.

Finally, for open-channel flow, the design variables control the shape of walls and the channel bed, the length and width of channel reaches, the location of channel structures and transitions, the bed slope, the friction coefficient and the inflow and

outflow parameters such as the depth and discharge. B-Spline curves [**65**] are used to define the shape of the channel walls and bed.

3.3. Flow Analysis

As the goal of this research is to study numerical design optimization as applied to high-fidelity, steady-state analysis tools, an existing flow solver HIVEL2D has been chosen to be transformed into an optimization code. This flow solver uses the finite element method to solve the shallow water equations. As the finite element method is an important method for solving systems of partial differential equations and since the shallow water equations display many of the same characteristics as other systems of partial differential equations for modeling fluid flow, the design optimization techniques applied to this analysis code can be similarly applied to other high-fidelity codes. As more physically realistic mathematical models and more efficient numerical codes are developed, the numerical design optimization process presented in this research can hopefully be applied to these tools.

The two-dimensional, depth-averaged, shallow water equations are often used to study the flow properties of open-channels in which the effects of vertical acceleration can be ignored but the effects across the channel are important. Frictional effects along the bed and walls of the channel are estimated to be consistent with Manning's friction equation. Furthermore, Reynolds stresses are included in the equations. These stresses alter the velocity profile across the channel and change the properties of normal flow, as described in Appendix E. The shallow water equations [**1,66**] can be expressed as

$$\frac{\partial h}{\partial t} + \frac{\partial p}{\partial x} + \frac{\partial q}{\partial y} = 0$$

$$\frac{\partial p}{\partial t} + \frac{\partial}{\partial x}\left(\frac{p^2}{h} + \frac{1}{2}gh^2 - h\sigma_{xx}\right) + \frac{\partial}{\partial y}\left(\frac{pq}{h} - h\sigma_{xy}\right) = -gh\frac{\partial z}{\partial x} - \frac{gn^2 p\sqrt{p^2+q^2}}{C_o^2 h^{7/3}}$$

$$\frac{\partial q}{\partial t} + \frac{\partial}{\partial x}\left(\frac{pq}{h} - h\sigma_{yx}\right) + \frac{\partial}{\partial y}\left(\frac{q^2}{h} + \frac{1}{2}gh^2 - h\sigma_{yy}\right) = -gh\frac{\partial z}{\partial y} - \frac{gn^2 q\sqrt{p^2+q^2}}{C_o^2 h^{7/3}} \quad (3.3.1)$$

where h is the depth, p and q are the discharges in the x- and y-directions, respectively, σ_{xx}, σ_{xy}, σ_{yx} and σ_{yy} are the Reynolds stresses, n is Manning's friction coefficient, and z is the bed elevation.

The primary assumptions of the shallow water equations are that vertical acceleration and other variations in the vertical direction are negligible, that the pressure distribution is hydrostatic, and that the bed slope is geometrically mild. Starting with the three-dimensional equations of conservation of mass and momentum, these assumptions are used to derive the shallow water equations. The equations of conservation of mass and x- and y-momentum are integrated in the vertical direction and are simplified by using the assumption that the acceleration in the vertical direction can be ignored and that the velocities in the other directions do not depend on the vertical direction. As a result, depth h is introduced into the equations. The conservation of vertical momentum equation can be reduced to

$$\frac{\partial}{\partial z}\left(\frac{P}{\rho}\right) + g = 0 \quad (3.3.2)$$

which implies that $P(z) = -\rho g z + P_o$, where P_o can be defined as the pressure at the water surface. This equation is the hydrostatic pressure distribution equation.

Thus, the assumption about the velocity components leads to the assumption that the pressure distribution is hydrostatic.

The final assumption in the shallow water equations deals with the bed slope. The vertical direction is typically defined as the direction of gravity. In the shallow water equations, the equations are derived by integrating in the direction that is perpendicular to the bed. For small bed slopes, the angle θ between these two directions is small, and the effects of this difference can be ignored by making the assumption that $sin\theta \approx 0$ and $cos\theta \approx 1$. Typically, the bed slope is considered geometrically mild when it is less than 0.02. When these three assumptions are realistic, the shallow water adequately reflect the physics of fluid flow through open-channels. See the discussions in Chaudhry [**66**] for more details.

To solve the shallow water equations for a particular open-channel geometry, the channel is discretized, and the flow is estimated numerically. The flow solver HIVEL2D uses the Petrov-Galerkin finite element method and is described in greater detail in Appendix F and in the references [**1,2**]. As is typical for most finite element solvers, HIVEL2D uses unstructured grids, even though the open-channel geometries used in this research are expressed as structured grids.

Once the computational domain is discretized, the system of partial differential equations are converted into a system of integral equations, one set for each node in the domain. At node i, there are three conservative variables to be determined, h, p and q. Using the finite element method, the three partial differential equations are converted into three integral equations $W_{i,1}$, $W_{i,2}$ and $W_{i,3}$ by considering the

weight functions associated with node i as shown in Figure 3.3.1. The system of partial differential equations is multiplied by a weight function. These equations are manipulated by using various forms of the divergence theorem, which involves the boundary conditions. The resulting equations are functions of the unknown flow variables and are evaluated by using Gauss Quadrature.

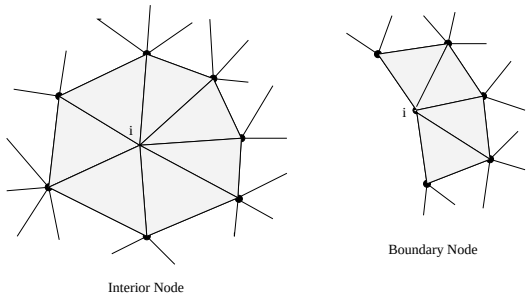

Figure 3.3.1. Domain of Influence for Weight Functions at Node i.

Once this conversion process is completed, $3N$ integral equations are formed, and $3N$ unknowns need to be determined, where N is the number of nodes. The result of this discretization process is a system of nonlinear equations of the form

$$W(Q^{n+1}, Q^n, \cdots, Q^0, \chi, t^{n+1}, t^n, \cdots, t^0) = 0 \tag{3.3.3}$$

where W is called the weighted residual vector, Q is a vector of flow variables at the various time levels and χ is the grid discretization. The only unknown variables at time level $n+1$ are Q^{n+1}, so the system can be expressed more compactly as $W(Q^{n+1}) = 0$. This system of equations is solved within HIVEL2D using Newton-Raphson iterations by solving

$$\frac{\partial W}{\partial Q}(Q^{n,m})\Delta Q^m = -W(Q^{n,m}) \tag{3.3.4}$$

and $Q^{n,m+1} = Q^{n,m} + \Delta Q^m$. This iteration process continues at time level $n+1$ until $\|\vec{W}(Q^{n,m})\| < tolerance$, at which point $Q^{n+1} = Q^{n,m}$. The Jacobian matrix $\frac{\partial W}{\partial Q}$ as implemented within the flow solver HIVEL2D has some approximations dealing with the Reynolds stresses and Petrov-Galerkin trial functions, but due to the Newton-Raphson iteration solution method, the approximate Jacobian matrix is sufficiently accurate to drive the system to convergence.

Because HIVEL2D solves the shallow water equations using an implicit method, subroutines that estimate the Jacobian matrix and solve equation (3.3.4) are available within the code. Thus, this code can be readily changed into a design optimization code where the design space derivatives are calculated via discrete sensitivity analysis. But due to the approximations in the Jacobian, the accuracy of the resulting design space derivatives will be limited.

Other gradient-estimation methods, such as ADIFOR or CTSE, can be applied to HIVEL2D. However, since this open-channel design problem requires the steady-state flow as determined by a high-fidelity, computationally expensive flow solver, discrete sensitivity analysis is used because it is the most computationally efficient for this class of problems.

3.4. Function Evaluation

Once the steady-state flow variables are obtained from the flow solver, the objective function F is evaluated. The objective function can be a function of the flow variables Q, the grid mesh χ and/or the design variable $\vec{\beta}$. In this research, the objective functions are explicitly dependent only on the flow variables, although there are

open-channel design problems where χ or $\vec{\beta}$ can be included in the objective function. For instance, if the bed elevation varies across the channel but a uniform surface is desired, then the depth of the water surface across the channel will depend on both the depth of flow and the bed elevation, and the objective function will include both components. Furthermore, the objective functions in this research are constructed to be continuous functions of the flow variables; otherwise, the design space derivatives would not exist.

Typically, in scale models and actual open-channels, surface elevations are the easiest characteristic to measure. Also, designers of open-channels are often quite concerned about the water depths in the channel so that the channel walls will be high enough to contain the flow. In open-channel flow, velocities are also important design considerations to prevent cavitation and wall scour; however, since HIVEL2D is primarily used by researchers to estimate flow depths, the objective functions in this dissertation are functions of the flow depths and not of the velocities.

The objective function for open-channel design can measure variation in depth so as to produce flow depths that are nearly uniform, depth raised to some power to minimize the maximum depth in a region, and energy loss through a transition, or a combination of these functions. The objective functions used in this research measure the non-uniformity of the flow depths as measured by the variation of the depth from the average depth over the functional domain Ω_F or

$$F(\vec{\beta}) = \sum_{\vec{x}_k \in \Omega_F} \left(h_{ave}(\vec{\beta}) - h_k(\vec{\beta}) \right)^2 \tag{3.4.1}$$

where h_{ave} is the average depth over Ω_F. If the flow depths are uniform, the objective function is at its minimum, which is zero. One of the conclusions of Appendix E is that uniform depth of flow is a valid solution of the governing equations under certain circumstances. Thus, the optimal solution for this objective function is an acceptable solution of the shallow water equations. Finally, because the objective function is in the form of a nonlinear least-squares function, the Gauss-Newton optimization algorithm [**67**] can be used to update the design variables, as described in Section 3.6 and in Appendix C.7.

3.5. Calculation of Design Space Gradients

In gradient-based design optimization, the choice of gradient-estimation method is typically based on the complexity, nature and computational expense of the objective function and the need for accuracy. Because HIVEL2D solves the high-fidelity shallow water equations, the computational expense of determining the steady-state solution demands consideration. For this class of problems, since sensitivity analysis estimates the design space derivatives without additional steady-state solutions, this method is the most efficient computationally of those techniques presented in Chapter 2. Discrete sensitivity analysis is used, rather than continuous sensitivity analysis, because the objective function is evaluated only at steady-state. (See Appendix A for derivations of discrete sensitivity analysis.) Thus, the gradient-estimation subroutines are only needed once the steady-state flow variables are determined; whereas, for continuous sensitivity analysis, the adjoint equations must be solved at each iteration in the solution process.

Since HIVEL2D is an implicit solver, an approximation to the Jacobian matrix and matrix equation solution subroutines are already available within the code. However, since the Jacobian matrix used in the flow solver is only an approximation which does include the terms for the derivatives of the Petrov-Galerkin weight function, the resulting design space derivatives have limited accuracy. For several different test cases, these derivatives were only accurate to within a few percent. By using the complex Taylor's series expansion (CTSE) method to generate the Jacobian matrix, a highly accurate Jacobian is generated, resulting in design space derivatives that are accurate to at least 6 significant digits. According to the work of Frank and Shubin [8], inaccurate design space derivatives can cause the optimization process to fail, so generating accurate design space derivatives is important.

Within discrete sensitivity analysis, either the adjoint variable formulation or the direct formulation can be used. The adjoint variable formulation is more efficient computationally when the number of design variables is larger than the number of objective functions, because the adjoint variable formulation must solve a matrix equation for each function regardless of the number of design variables. On the other hand, since the direct formulation must solve a matrix equation for each design variable, it is more efficient when the number of objective functions is larger than the number of design variables. In this research, there is only one objective function $F(\vec{\beta})$. Thus, the adjoint variable formulation can estimate the design space gradient $\nabla_{\vec{\beta}} F$ more efficiently than the quasi-analytic formulation. However, by considering the objective function as a nonlinear least squares function, the Gauss-Newton method

can be used to estimate the Hessian matrix $\nabla^2_{\vec{\beta}} F$, as well as the design space gradient $\nabla_{\vec{\beta}} F$. The objective function is the sum of squares or

$$F(\vec{\beta}) = \sum_{k=1}^{N} f_k^2(\vec{\beta}) \tag{3.5.1}$$

where $f_k(\vec{\beta})$ are called the residual functions. The derivative of $F(\vec{\beta})$ can be obtained via

$$\frac{dF}{d\beta_i} = \sum_{k=1}^{N} 2 \frac{df_k}{d\beta_i} f_k \tag{3.5.2}$$

which requires the derivative of each residual function. In this case, there are more residual functions than design variables; hence, the direct formulation is more computationally efficient, when used in conjunction with the Gauss-Newton optimization algorithm.

3.6. Updating the Design Variables

In Appendix C, a variety of optimization algorithms are discussed that can be used to determine the new design variables based only on the gradient information available from discrete sensitivity analysis. Since the Hessian, or second derivative, matrix is not available, Newton's method can not be used. The linearly convergent method of steepest descent is not used in this research due to its slow convergence rates. The conjugate gradient method and trust region models have not been chosen as the optimization algorithm because they require additional function evaluations in the gradient direction, which increases the computational expense. Thus, the quasi-Newton methods and the Gauss-Newton methods have been determined to have

much potential as the optimization algorithm, as demonstrated by many researchers [**16,24**].

Because the objective function can be expressed as a nonlinear least-squares function, the Gauss-Newton method is used because it can achieve super-linear convergence due to the special structure of the objective function. Furthermore, the Gauss-Newton method provides information about the appropriate step size, so additional function evaluations in the search direction are not necessary. For more general functional forms, the quasi-Newton methods can be used with the expectation of super-linear convergence rates, as well.

The Gauss-Newton optimization algorithm uses the structure of the objective function to estimate the Hessian matrix. The form of a nonlinear least-squares function is given in equation (3.5.1) and its first derivative is given in equation (3.5.2). The second derivative has the form

$$\frac{d^2F}{d\beta_i d\beta_j} = \sum_{k=1}^{N} 2\frac{df_k}{d\beta_i}\frac{df_k}{d\beta_j} + \sum_{k=1}^{N} 2\frac{d^2f_k}{d\beta_i d\beta_j}f_k \tag{3.6.1}$$

If the second term is small, then the second derivative can be effectively approximated as

$$\frac{d^2F}{d\beta_i d\beta_j} = \sum_{k=1}^{N} 2\frac{df_k}{d\beta_i}\frac{df_k}{d\beta_j} \tag{3.6.2}$$

which only requires first derivative information of the residual functions f_k. Thus, the second derivative matrix can be approximated using only first derivative information, providing the potential of superlinear convergence rates.

From practical experience, however, the resulting matrix is ill-conditioned because many of its eigenvalues are quite close to zero. One of the properties of using an ill-conditioned matrix in the equation $Ax = b$ is that small changes to the vector b can result in large changes to the vector x, which is not desirable, especially when there many be small amounts of error in the vector b. Thus, this ill-conditioning can result in unstable updating of the design variables. See Appendix C.9 for details about this phenomenon. More research is needed to understand the reason for the ill-conditioning, to determine if it is a result of the Gauss-Newton method or of the use of Bezier and B-spline curves to define the boundary geometries.

Since this ill-conditioning is a result of the eigenvalues being near zero, one solution to this difficulty is to add a constant to the diagonal. Thus, the eigenvalues will increase by that constant, the condition number will decrease and the optimization algorithm will be more stable. The constant is often called the Levenberg-Marquardt constant[**67**]. From practical experience, the best choice for this constant may be the magnitude of the largest derivative. Thus, as the design variables converge to the optimal solution, the design space derivatives are driven to zero, and the Levenberg-Marquardt constant vanishes. Due to the efficiency of this method as demonstrated by the examples in this research, it is concluded that this modification of the Gauss-Newton method produces an excellent and robust algorithm to determine the new design variables.

3.7. Flow Prediction

As presented in Appendix A, flow prediction is an added benefit of discrete sensitivity analysis. The Taylor's series expansion of $Q(\vec{\beta}^{n+1})$ is

$$Q(\vec{\beta}^{n+1}) = Q(\vec{\beta}^{n} + \Delta\vec{\beta}^{n}) = Q(\vec{\beta}^{n}) + \frac{\partial Q}{\partial \beta}\Delta\vec{\beta}^{n} \tag{3.7.1}$$

so to first order

$$\Delta Q^{n} = \frac{\partial Q}{\partial \beta}\Delta\vec{\beta}^{n} \tag{3.7.2}$$

The matrix $\frac{\partial Q}{\partial \beta}$ is composed of the vectors $\frac{\partial Q}{\partial \beta_i}$ that are generated in the direct formulation. If the adjoint variable formulation is used, slightly more computational effort is required, but the same information concerning the change in flow variables can still be derived. As a result, the initial conditions for the flow calculations for the new set of design variables are more accurate and may require less computational effort to drive the flow to steady-state. Baysal and Eleshaky [**33,34**] used flow prediction in many papers to reduce the computational cost of the additional function evaluations in the gradient direction required by their optimization algorithm.

Since the flow prediction equations assume that the discretization is the same for the old and new design variables, the flow prediction subroutines can not be used for design problems where the number of grid points is dependent on the design variables. Furthermore, since the flow prediction analysis is only a linear approximation, it may predict a decrease in the water depth for the new design variables that is larger than the depth for the old design variables, resulting in negative flow depth. When this

error occurs, the results of the flow prediction algorithm are ignored, and the steady-state solution for the old design variables is used as the initial conditions for the new design variables. Finally, since the flow solver HIVEL2D allows large time steps due to its implicit nature, the advantages and possible computational efficiencies of the flow prediction subroutine may be unnecessary.

3.8. Termination Criteria

In design optimization, there are at least three different sets of termination criteria, being based on the change in the objective function value, the magnitude of the design space gradient, and the magnitude of the change in the design variables. When the goal of the design process is design improvement rather than design optimization, the optimization process can be terminated once the design has achieved the design objectives.

The termination criterion used in the examples in Chapter 5 is to terminate the design process when

$$\frac{|F(\vec{\beta}^n) - F(\vec{\beta}^{n-1})|}{F(\vec{\beta}^o)} < tolerance_1 \tag{3.8.1}$$

where $tolerance_1$ is a small percentage and β^o is a realistic initial guess for the design variables.

For design optimization, the optimal solution satisfies the condition that the design space gradient is zero. Thus, to guarantee that a local minimum has been achieved, the stopping criterion could be based on the design space gradient such as

$$\frac{\|\nabla_{\vec{\beta}} F(\vec{\beta}^n)\|}{\|\nabla_{\vec{\beta}} F(\vec{\beta}^o)\|} < tolerance_2 \tag{3.8.2}$$

Typically, $tolerance_2$ will be an extremely small number, say 10^{-6}. However, when constraints are placed on the acceptable design, the design space derivatives may not be zero at the optimal solution within the constrained design space. Thus, this stopping criterion must be used judiciously.

When there is a possibility that the optimization algorithm will be unable to update the design variables efficiently and thus not be able to locate the local minimum, a prudent stopping criterion is based on the change in the design variables through a formula such as

$$\|\vec{\beta}^n - \vec{\beta}^{n-1}\| < tolerance_3 \tag{3.8.3}$$

Since acceptable ranges for the design variables are determined a priori, via side constraints, and hence are known prior to the execution of the design optimization process, there is no need to analyze the relative change as in the previous termination criteria. When the termination criterion is based on the change in the design variables, the design variables should be scaled so that their relative values are similar; otherwise, the design process will terminate prematurely when the largest design variables have stabilized.

For the examples in Chapters 4 and 5, the objective function decreases quite rapidly for the first few design iterations and then decreases much more slowly. The

majority of design improvement occurs in the first few design iterations. Thus, if the goal of the design process is to improve the current design rather than to find the optimal design, the design optimization process can terminate once acceptable design improvement is achieved.