APPENDIX C

OPTIMIZATION ALGORITHMS

APPENDIX C

OPTIMIZATION ALGORITHMS

C.1. Introduction

For the objective functions and design problems within this dissertation, all the optimization problems were in the class of unconstrained optimization, with the goal of minimizing an objective function $F(\vec{x})$ over a multi-dimensional vector space spanned by the vector $\vec{x}$. For a more detailed discussion of these methods, see Gill, etal [**67**] and Dennis and Schnabel **68**]. Within this class of problems, at least six gradient-based optimization algorithms are commonly used - the method of steepest descent, conjugate gradient methods, Newton's method, Quasi-Newton methods, trust region models and Gauss-Newton methods. In this appendix, each of these methods are reviewed briefly. For a more detailed discussion, see the references [**71,72**]. In general, these optimization algorithms determine the search direction $\vec{p}^n$ from available gradient information and possibly Hessian information. Then, the optimal step size Δx, which is a scalar, is calculated from one or more function evaluations in the search direction or from the algorithm itself. Given the search direction and the optimal step size , the location for the next iterate $\vec{x}^{n+1}$ is determined via

$$\vec{x}^{n+1} = \vec{x}^n + \vec{p}^n \Delta x \tag{C.1.1}$$

C.2. The Method of Steepest Descent

When the only available information for an optimization problem is the function value at a particular location $F(\vec{x}^n)$ and the gradient $\nabla F(\vec{x}^n)$, the method of steepest

descent is often used to determine the next iterate x^{n+1}. The primary concept is that optimal search direction $\vec{p}_n$ at a location $\vec{x}^n$ is the gradient or steepest descent direction $\nabla F(\vec{x}^n)$. Thus, the optimization process is guaranteed to decrease the value of the objective function by moving in this direction, for small enough step sizes. Unfortunately, as the step size increases, there is no guarantee that the objective function value will decrease.

By evaluating the function at various locations in the search direction, the optimal step size Δx can be determined, and the next iterate is chosen via

$$\vec{x}^{n+1} = \vec{x}^n + \Delta x \frac{\nabla F(\vec{x}^n)}{\|\nabla F(\vec{x}^n)\|} \tag{C.2.1}$$

Typically, the optimal step size Δx is determined via a quadratic approximating function using the function value at $\vec{x}^n$, the magnitude of the gradient $\|\nabla F(\vec{x}^n)\|$ and the function value at a location in the search direction $\vec{x}^n + \alpha\vec{p}^n$, where α is some problem dependent value. The minimum of the quadratic approximating function is chosen as the potential next iterate. If this location is not satisfactory, then further function evaluations are used.

The primary drawbacks of this method are the need for one or more additional function evaluations per design iteration and the linear convergence rates of this technique. Due to the poor convergence rates, this method may not be efficient enough for realistic design optimization algorithms because it may require many design iterations to significantly reduce the value of the objective function.

C.3. The Conjugate Gradient Method

This method uses the same information as the method of steepest descent, with an additional constraint - the current search direction p^n must be conjugate, or perpendicular, to the previous NDV search directions, where NDV is the number of design variables. Once the algorithm has searched in the direction $\vec{p}^n$, then there is no need to search in that direction again. For quadratic functions, the conjugate gradient method is able to identify the optimal solution exactly within NDV iterations. However, for general functions, "the traditional conjugate-gradient method with exact linear searches and exact arithmetic is *n-step superlinearly convergent*"[1]. In fact, according to Gill, etal, in practical applications, the conjugate-gradient method yields only linear convergence.

A primary drawback of this method is that the optimal step size is required to achieve efficient convergence. Thus, multiple function evaluations are necessary to determine the optimal step size, and for high-fidelity design optimization, the conjugate gradient method is infeasible due to the high computational cost of the function evaluations.

Another drawback of the conjugate gradient method is that the convergence rate scales with the number of design variables. Since superlinear convergence is achieved after the optimization algorithm has searched once in each conjugate direction, then the number of design iterations to achieve superlinear convergence is the number of

[1]p. 149, Gill, P. E., Murray, W., and Wright, M. H., Practical Optimization, Academic Press, New York, 1981.

design variables. For complicated designs, the number of design variables can be quite large; therefore, the convergence rate is probably unsatisfactory for large problems.

C.4. Newton's Method

Newton's method uses the Hessian, or second derivative matrix, to update the design variables. Since the goal of the minimization process is to find $\vec{x}^*$ such that $\nabla F(\vec{x}^*) = 0$, consider a Taylor's series expansion of $\nabla F(\vec{x}^*)$,

$$\nabla F(\vec{x}^*) = \nabla F(\vec{x}^* - \vec{x}^n + \vec{x}^n) = \nabla F(\vec{x}^n) + \nabla^2 F(\vec{x^n})(\vec{x}^* - \vec{x}^n) + O((\vec{x}^* - \vec{x}^n)^2) \tag{C.4.1}$$

Assuming that $\nabla F(\vec{x}^*) = 0$ and ignoring higher order terms, one obtains

$$\nabla F(\vec{x}^n) = -\left[\nabla^2 F(\vec{x^*})\right](\vec{x}^* - \vec{x}^n) \tag{C.4.2}$$

or

$$(\vec{x}^* - \vec{x}^n) = -\left[\nabla^2 F(\vec{x^*})\right]^{-1} \nabla F(\vec{x}^n) \tag{C.4.3}$$

The result of Newton's method contains both the search direction $\vec{p}^n = \frac{\vec{x}^* - \vec{x}^n}{\|\vec{x}^* - \vec{x}^n\|}$ and the step size $\|\vec{x}^* - \vec{x}^n\|$, so that additional function evaluations are not needed. For quadratic functions, this method yields exact results in one iteration because the higher order terms are zero. In general, Newton's method yields quadratic convergence once the iterates have moved within the convergence region for the function.

The main drawback of this method is the need for the Hessian matrix. Unfortunately, the current implementations of discrete sensitivity analysis and the other methods to estimate the design space derivatives efficiently have not been extended

to the estimation of the second derivative matrix. As a result, Newton's Method is rarely used to update the design variables.

C.5. Trust Region Models

For highly nonlinear functions, the Newton search direction as given in equation (C.4.3) may not be the best search direction. Using the knowledge that the best search direction as the step size tends toward zero is the gradient direction, trust region models approximate the search direction as a combination of the gradient direction and the Newton search direction. Once the optimal step size in the Newton direction is determined, assuming that it is less than the Newton step size, a new search direction is calculated from the gradient and Newton directions. The next iterate is calculated by using the new search direction and the optimal step size. Of course, this process could be repeated by searching along the new search direction for the optimal step size.

The method has received much attention in current literature, especially as a means to improve convergence of implicit flow solvers. But because of the computational cost of function evaluations and the need to calculate the Hessian matrix, trust region models have been rarely used within design optimization.

C.6. Quasi-Newton Methods

Quasi-Newton methods gradually build an approximation to the Hessian matrix based on the information at previous iterations and use this approximate Hessian in equation (C.4.3) to determine the search direction and step size. One large class of methods updates the approximate Hessian matrix B_{n+1} via a variety of formula.

These formula use the previous approximate Hessian matrix B_n, the change in the gradient direction $y_n = \nabla F(\vec{x}^{n+1}) - \nabla F(\vec{x}^n)$, the change in the iterates $s_n = \vec{x}^{n+1} - \vec{x}^n$ and the search direction for the previous iterate p^n. The conditions for these update formulae include the requirement that the Hessian matrix remain symmetric and positive definite and that the new Hessian matrix satisfy the secant equation

$$B_{n+1} s_n = y_n \tag{C.6.1}$$

The Hessian update formula that is generally accepted as the "best" is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update [**69**] or

$$B_{n+1} = B_n - \frac{1}{s_n^T B_n s_n} B_n s_n s_n^T B_n + \frac{1}{y_n^T s_n} y_n y_n^T \tag{C.6.2}$$

This approximate Hessian matrix is used in place of the exact Hessian matrix within Newton's method. As with Newton's method, the Hessian update methods do not require additional function evaluations, since equation (C.4.3) contains the step size information. However, additional improvements in convergence are available when exact linear searches are performed. When this optimization algorithm was used for the preliminary example, in Chapter 3.9.3, one linear search was performed.

Another issue for the update methods is the choice of the initial Hessian approximating matrix B_o. For the preliminary example, the initial Hessian matrix B_o was chosen as a diagonal matrix whose entries where the magnitude of the largest derivative for the first design iteration. The choice of the initial Hessian matrix is a topic beyond the scope of this research.

In practice, BFGS update methods are often used to achieve super-linear convergence rates for design optimization problems. For general objective functions, these update methods provide for more efficient use of the gradient information than the method of steepest descent or the conjugate gradient method, because these methods do not require additional function evaluations and still provide super-linear convergence rates. However, the theoretical developments assume that the derivative information is exact. Huddleston **[15]** used the BFGS method and concluded that the derivatives used in his research were not accurate enough for the BFGS method.

C.7. Gauss-Newton Methods

Given a function of the form

$$F(\vec{x}) = \sum_{i=1}^{N} r_i^2(\vec{x}) = R^T(\vec{x})R(\vec{x}) \tag{C.7.1}$$

where $r_i(\vec{x})$ are N residual functions, $\vec{x}$ is a vector of variables and $R(\vec{x})$ is a vector of residual functions, the Gauss-Newton method takes advantage of the structure of the function $F(\vec{x})$ to approximate the Hessian matrix and thus can produce super-linear convergence.

The first derivative of the function is

$$\frac{\partial F}{\partial x_k} = \sum_{i=1}^{N} 2\frac{\partial r_i}{\partial x_k} r_i = 2\frac{\partial R}{\partial x_k}^T R \tag{C.7.2}$$

and the gradient is

$$\nabla F(\vec{x}) = 2\nabla R^T R \tag{C.7.3}$$

The second derivative is

$$\frac{\partial^2 F}{\partial x_k \partial x_j} = \sum_{i=1}^{N} 2 \frac{\partial r_i}{\partial x_k}^T \frac{\partial r_i}{\partial x_j} + \sum_{i=1}^{N} 2 \frac{\partial^2 r_i}{\partial x_k \partial x_j} r_i = 2 \frac{\partial R}{\partial x_k}^T \frac{\partial R}{\partial x_j} + 2 \frac{\partial^2 R^T}{\partial x_k \partial x_j} R \tag{C.7.4}$$

and the Hessian Matrix is

$$\nabla^2 F(\vec{x}) = 2 \nabla R^T \nabla R + 2S \tag{C.7.5}$$

where S is a matrix whose entries are $\frac{\partial^2 R^T}{\partial x_k \partial x_j} R$.

The Gauss-Newton method assumes that the term S is relatively small compared to $(\nabla R^n)^T \nabla R^n$ near the local extrema and hence can be ignored. Using this assumption, $\vec{p}^n$ is obtained via

$$2(\nabla R^n)^T \nabla R^n \Delta \vec{p}^n = -2(\nabla R^n)^T R^n \tag{C.7.6}$$

The assumption that S can be ignored is reasonable if $R^n(\vec{x}^*) = 0$ at the local extrema $\vec{x}^*$, or if $R^n(\vec{x})$ is linear, in which case $\frac{\partial^2 R^n}{\partial x_k \partial x_j} = 0$. For the work presented herein, neither of these qualifications are true: for general optimization problems, the function value for the optimal set of design variables is not zero, and the optimization problem is not a linear least-squares problem. Nevertheless, since the Gauss-Newton method was able to obtain good design improvement results, the assumption that S can be ignored appears to be reasonable.

C.8. Calculating the Derivative in the Gauss-Newton Method

If the function F is not a nonlinear least-squares function or is considered as a single function, the computational cost of estimating the gradient of the function via

finite differences, automatic differentiation and the complex Taylor's series expansion (CTSE) method scales with the number of variables, which is NDV. Because there are probably more than one design variable, the adjoint variable formulation of discrete sensitivity analysis is more computational efficient than the direct formulation, because the adjoint variable formulation inverts one matrix equation for each function, whereas the quasi-analytic formulation inverts a matrix equation for each variable.

However, if the function is considered as a nonlinear least-square function, the computational cost of estimating the derivatives of the residuals functions f_i via finite differences and CTSE is almost the same as before, because the function F and the residual functions f_i are determined from the same steady-state solution. Thus, the computational cost of evaluating the residual functions f_i is approximately the same as the cost of evaluating F. Because there are N residual functions, where N is greater than NDV, the adjoint variable formulation is quite computationally expensive in comparison to the direct formulation, because the adjoint variable formulation must invert a matrix equation for each function. The relative computational costs of these methods to estimate the derivatives for F and f_i are summarized in Table C.8.1. NDV is the number of design variables and N is the number of residual functions, where N is typically much larger than NDV.

Table C.8.1. Comparison of Gradient-Estimation Methods.

Method	$F(\vec{x})$	$\sum_{i=1}^{N} f_i(\vec{x})$
Finite Differences, CTSE	NDV Steady-State Solutions	NDV Steady-State Solutions
Adjoint Variable Formulation	1 Matrix Inversion	N Matrix Inversions
Quasi-Analytic Formulation	NDV Inversions	NDV Inversions

C.9. Levenberg-Marquardt Method

Because the Gauss-Newton method typically does not use line searches and thus has no mechanism to prevent unreasonably large changes in the design variables, the issue of stability of the successive iterates must be considered for the Gauss-Newton method. One characteristic of the Hessian matrix near a local minimum is that it is symmetric positive definite. Thus, all of the eigenvalues are positive. But if some of the eigenvalues of the approximate Hessian matrix are close to zero, then the matrix is ill-conditioned, and the solution $\vec{p}^n$ to the matrix equation can vary dramatically with small changes to the right-hand-side. Thus, round-off errors and other numerical errors can cause the search direction to be quite inaccurate and can guide the search to sets of design variables where the design is impossible or causes the numerical solver to fail to converge.

To overcome this difficulty and maintain good convergence properties, the Levenberg-Marquardt method is often employed. In this method, a constant is added to the diagonal of the matrix $(\nabla R^n)^T \nabla R^n$ which increases the eigenvalues by that constant and improves the condition number of the matrix. In other words, the Levenberg-

Marquardt method defines the search direction via

$$\left(\nabla^2 F + \nu^n I\right) \vec{p}^n = -\nabla F \tag{C.8.1}$$

If $\nu^n = 0$, then the search direction $\vec{p}^n$ is the Gauss-Newton search direction. As $\nu^n \to \infty$, $\vec{p}^n$ becomes parallel to the steepest descent direction. Thus, by adding a constant to the diagonal, the resulting search direction is a combination of the steepest descent direction and the Gauss-Newton search direction. In a sense, this method is similar to Trust Region Models because it determines the search direction as a combination of the Gauss-Newton direction and the steepest descent direction.

The Levenberg-Marquardt method can also be expressed as a restricted step method which solves

$$\min_{\vec{p}} \left(F(\vec{x}^n) + \nabla F(\vec{x}^n)^T \vec{p} + \frac{1}{2} \vec{p}^T \nabla^2 F(\vec{x}^n) \vec{p} \right) \tag{C.8.2}$$

subject to $\vec{p}^T \vec{p} \leq C$ for some constant C. Introducing a Lagrange Multiplier ν,

$$\mathcal{L}(\vec{p}, \nu) = F(\vec{x}^n) + \nabla F(\vec{x}^n)^T \vec{p} + \frac{1}{2} \vec{p}^T \nabla^2 F(\vec{x}^n) \vec{p} + \frac{1}{2} \nu (\vec{p}^T \vec{p} - C) \tag{C.8.3}$$

A necessary condition for solution of this problem is that $\nabla_{\vec{p}} \mathcal{L} = 0$, or

$$\nabla_{\vec{p}} \mathcal{L} = \nabla F(\vec{x}^n) + \nabla^2 F(\vec{x}^n) \vec{p} + \nu \vec{p} = 0 \tag{C.8.4}$$

or

$$\left(\nabla^2 F(\vec{x}^n) + \nu I\right) \vec{p} = -\nabla F(\vec{x}^n) \tag{C.8.5}$$

For the research in this thesis, the Levenberg-Marquardt constant was defined as the magnitude of the largest design space derivative for a particular design iteration, or

$$\nu = \max_k \left\| \frac{\partial F}{\partial \beta_k} \right\| \tag{C.8.6}$$

Thus, as the optimization process converged, the magnitude of the constant decreased as well.

The theoretical reasons for the success of this choice for the constant have not been established; however, the cause for the ill-conditioned nature of the approximate Hessian matrix probably lies with the use of Bezier and B-spline curves, in connection with the design variables.

C.10. The Use of Additional Function Evaluations

Often, an optimization algorithm requires or can be improved by the use of additional function evaluations in the search direction. Using the function values and derivative information, quadratic or cubic approximating functions are formed, and the distance to the minimum for this approximating function is used as the step size for the optimization algorithm. For this research, the function value and the gradient at the current design iteration are available and are used to form the approximating function.

Given the step size $\Delta\vec{x} = \alpha\vec{p}^n$ to be used for the additional function evaluation, an approximating quadratic function is created of the form

$$f(\lambda) = F\left(\vec{x} - \lambda\Delta\vec{x}\right) \tag{C.9.1}$$

where F is the objective function. The coefficients of the quadratic function are determined by using $f(0) = F(\vec{x})$, $f(1) = F(\vec{x} - \Delta\vec{x})$ and

$$f'(0) = \nabla F(\vec{x}) \cdot (-\Delta\vec{x}) \tag{C.9.2}$$

The minimum of this quadratic function occurs at

$$\lambda_{min} = \frac{\nabla F(\vec{x}) \cdot \Delta\vec{x}}{2(F(\vec{x} - \Delta\vec{x}) - F(\vec{x}) - \nabla F(\vec{x}) \cdot \Delta\vec{x})} \tag{C.9.3}$$

A special case must be considered - the case when $\lambda_{min} < 0$. Since $-\Delta\vec{x}$ is a descent direction, the numerator will always be positive. Thus, if $\lambda_{min} < 0$, then the denominator must be negative, or

$$F(\vec{x} - \Delta\vec{x}) < F(\vec{x}) + \nabla F(\vec{x}) \cdot \Delta\vec{x} \tag{C.9.4}$$

In this case, the actual function value at $\vec{x} - \Delta\vec{x}$ is less than the value predicted by the slope at $\vec{x}$. Thus, the new location $\vec{x} - \Delta\vec{x}$ is quite satisfactory, λ is set to 1, and the new location is accepted.

Once λ is determined, the new step size is $\Delta\vec{x}_{new} = \lambda_{min}\Delta\vec{x}$. If the function value at $F(\Delta\vec{x}_{new})$ is not acceptable, then more searches can be performed.