CHAPTER II

OVERVIEW OF OPTIMIZATION METHODS AND APPLICATIONS

2.1. Approaches to Optimization

Optimization problems are typically formulated to minimize an objective function $F(\vec{x})$ over the space of acceptable inputs Ω_{DV} with equality constraints $G(\vec{x}) = 0$ and inequality constraints $H(\vec{x}) \geq 0$. The choice of solution method for a particular optimization problem is based on many criteria, including problem size (i.e., the number of variables), cost of evaluating the objective function and the complexity, nonlinearity and multi-modal nature of the objective function. For optimization problems dealing with numerical simulation, such as parameter identification and design optimization, the most common solution methods have been inverse methods, genetic algorithms and gradient-based methods.

For high-fidelity design optimization, the cost of evaluating the objective function is on the order of solving a system of partial differential equations, which can be quite computationally expensive. For many applications that rely on computational simulation, the objective function can be constructed to be continuously dependent on the design variables for the region of the design space that is of interest. Furthermore, to avoid the issue of local versus global minima, one can assume that the objective function has few local minima. In practical applications, the goal is design improvement

rather than design optimization, so finding a local minimum is probably acceptable. Finally, there is typically only one objective function and a few constraint functions, a relatively small number of design variables (10's) and much larger number of flow variables (1000's). Knowing the properties of the objective function, one can now determine the best methods to use to search for the optimal solution.

2.1.1. Inverse Methods

Technically, an inverse method is not an optimization method; rather it inverts the relationship between dependent and independent variables. In particular, for a given set of variables $\vec{\beta}$, the numerical simulation tool gives flow field information $\vec{Q}$ via a functional relationship $\mathcal{F}$ so that

$$\vec{Q} = \mathcal{F}(\vec{\beta}) \tag{2.1.1}$$

An inverse method seeks to find the inverse functional $\mathcal{G}$ where

$$\vec{\beta} = \mathcal{G}(\vec{Q}) \tag{2.1.2}$$

so that the user can specify the desired flow field information and the functional relationship provides the design variables that correspond to this flow field. Unfortunately, since the functional $\mathcal{F}$ is not onto the space spanned by all possible flow fields, the inverse for a particular flow field may not exist, and the inverse method will fail, although an approximate inverse may be found. When the flow field information $\vec{Q}$ is based on analytic expressions, the inverse functional relationship may be estimated via inverting linear approximations to these expressions. However, if the flow field in-

formation is determined from a steady-state solution of a discretized system of partial differential equations, inverse methods are much more complicated to develop.

2.1.2. Probabilistic Methods

Genetic algorithms (GA) may be the most widely used class of probabilistic algorithms which also include neural networks, evolutionary algorithms and simulated annealing. Holland [**3**], in the 1960's, along with Goldberg [**4**], wanted to model the natural process of "survival of the fittest" to develop an algorithm that could be applied to a wide variety of problems. The resulting algorithm evolves a population of viable members. The genes of a particular member determine the location in the search space for that member. The fitness of this member is determined by a fitness function, and the probability that this member's genetic material will be transferred to the next generation is based on its fitness value. To generate the next generation, two members of the current generation are chosen randomly based on their fitness. Their genetic material is combined via a crossover technique and possibly mutated to generate two new children. The choice of genetic structure, the fitness function, the parental selection algorithm, the crossover and mutation techniques and the strategy for replacement into the population, along with the parameters that control the probability of the occurrence of certain events, must be determined based on the particular application.

Some of the advantages of the genetic algorithm are its ability to use discontinuous fitness functions and discrete input variables and its ability to overcome and avoid local minima. Because derivatives are not used, there are no requirements concerning

the continuity of the variables or the function. Thus, the genetic algorithm can be applied to a wide range of problems including scheduling and networking. Because even a poorly fit member has a chance to reproduce and because the genetic material can be mutated, the population can maintain genetic diversity. Thus, even if many members of the population begin to settle near a local minimum, the genetic algorithm has the ability to continue to search in other areas of the design space and find the global minimum. Another advantage of the genetic algorithm is that it can be applied to highly complex problems and find "good" solutions within a reasonable amount of time. These solutions may not be optimal, but for many applications, they are often better than the results of deterministic algorithms given the same amount of computational effort. Finally, the genetic algorithm is highly parallellizable due to the need to evaluate each member of the population at each generation, which can be done independently and hence in parallel.

The primary disadvantage of the genetic algorithm is that a large population must be evaluated over many generations. Hence, the computational cost can become prohibitive if the cost of a single evaluation is large. For many applications, this cost is quite small, so evaluating a population of one hundred members for 10,000 generations is quite reasonable. However, in computational fluid dynamics, the cost of analyzing a design can be quite expensive, especially for three-dimensional designs or for high-fidelity simulations. Thus, genetic algorithms are usually only used in computational fluid dynamics in conjunction with less computationally expensive, lower fidelity models.

2.1.3. Gradient-Based Methods

Gradient-based numerical design optimization methods have received much attention in recent years because they are conceptually simple, they are deterministic and many optimization algorithms have been developed which use the gradient. On a multi-dimensional surface, defined by $F(\vec{x})$, the gradient at a location $\vec{x}$ is a vector of partial derivatives $(\partial F/\partial x_1, \cdot, \cdot, \cdot, \partial F/\partial x_n)$ and points in the direction of the largest slope. In the context of design optimization, this gradient is termed the design space gradient, and the individual derivatives are called the design space derivatives, or sensitivity derivatives. Locally, the gradient direction is the best direction to travel to increase the function value, and the opposite direction is best to decrease its value. But, since the surface is likely to curve and bend, this direction may not be the best direction for larger step sizes. Curvature is a second derivative effect, and thus better directions can be found by using both first derivative (gradient) information and second derivative (Hessian) information. Unfortunately, obtaining this information can be quite computationally expensive, and the information may be inaccurate.

The design space derivatives can be obtained via a finite difference formula such as

$$\frac{\partial F(\vec{x})}{\partial x_i} \approx \frac{F(\vec{x} + e_i \Delta x) - F(\vec{x})}{\Delta x} \tag{2.1.3}$$

This formula is easy to incorporate within an analysis code, but for high-fidelity analysis codes, which are typically quite computationally expensive to run, the finite difference method is computationally costly since an additional function evaluation is

necessary for each design variable. Furthermore, the choice of Δx must be determined for each design problem in order to generate derivatives with acceptable accuracy. If Δx is too large, then the nonlinear effects obscure the results; however, if Δx is too small, then round-off or subtraction error can destroy the accuracy of the results.

To address the issue of accuracy of the design space derivatives, other techniques have been investigated including automatic differentiation and the complex Taylor's series expansion (CTSE) method. These methods do not rely on the finite difference quotient and thus avoid the error which results from subtracting two numbers that are quite close to each other. ADIFOR (Automatic Differentiation for FORTRAN) uses automatic differentiation to convert a code that yields functional information into a code that provides derivative information, by repeated use of the chain rule. ADIFOR was developed in the early 90's by Bischof and Carle[**5**] and has been used within aerospace applications to calculate design space derivatives and water resource applications to estimate parameter values for groundwater models. In automatic differentiation, the relationships between the function or dependent variable and the independent variables are established, and for each step in this relationship, the analytic derivative for the line of code is determined and included within the code. ADIFOR is easy to learn and to implement because it determines these relationships and generates the derivative codes automatically, based on the dependent and independent variables specified by the user. Also, unlike the other derivative estimation methods, ADIFOR can be used to obtain exact second derivatives, so that quadratic optimization algorithms, such as Newton's methods, can be employed.

A variety of researchers have used ADIFOR and ADIC (Automatic Differentiation for C) to generate derivative codes and report that the derivatives are highly accurate. However, the main drawback of automatic differentiation is that the method is quite computationally expensive. In general, the cost of the derivative code is on the order of the cost of evaluating the function times the number of design variables. More specifically, the cost of the derivative code depends on the cost of evaluating the function's derivatives. If the relationships between the function and the independent variables consist of many nonlinear relationships, the cost of evaluating the derivative can be substantially more than the cost of evaluating the function. The cost of using ADIFOR is quite similar to the cost associated with finite differences, although the accuracy of the result is much better. Furthermore, for parallel codes or for vector codes, the resulting derivative codes are not as computationally efficient as the original parallel or vectorized codes, and much re-coding must be performed in order to obtain similar computational benefits from these high-performance machines. Until the computational costs associated with the ADIFOR generated codes are substantially reduced, it is doubtful that automatic differentiation will be useful for realistic, high-fidelity design optimization.

Another method that yields exact derivatives is the complex Taylor's series expansion method. This method was first presented by Lyness and Moler [**6**] in 1967 but was overlooked until Squire and Trapp [**7**] applied this method to some simple functions. This method uses complex variables to avoid the errors associated with the subtraction step in the finite difference approximation by analyzing the Taylor's

series expansion of

$$f(x+i\Delta x) = f(x) + i\frac{df(x)}{dx}\Delta x - \frac{d^2f(x)}{dx^2}(\Delta x)^2 - i\frac{d^3f(x)}{dx^3}(\Delta x)^3 + O(\Delta x)^4$$
$$\left(f(x) - \frac{d^2f(x)}{dx^2}(\Delta x)^2 + O(\Delta x)^4, \frac{df(x)}{dx}\Delta x - \frac{d^3f(x)}{dx^3}(\Delta x)^3 + O(\Delta x)^5\right) \quad (2.1.4)$$

where $f(x)$ is a real-valued function. By considering the imaginary part and solving for $\frac{df}{dx}$, one obtains

$$\frac{df(x)}{dx} = \frac{Im\left[f(x+i\Delta x)\right]}{\Delta x} + O(\Delta x)^2 \quad (2.1.5)$$

By choosing Δx such that the higher order terms are smaller than machine zero, the gradient can be estimated to machine accuracy, because there is no subtraction error and hence no lower bound on the size of Δx. This method readily extends to functions of many variables.

The complex Taylor's series expansion method is quite easy to implement. The only modifications to the code are that the real variables must be changed to complex variables, which is quite easy in FORTRAN. Intrinsic functions that have special meanings for complex numbers, such as the absolute value, must be modified to be based only on the real portion, and any *if* statements using real numbers must be modified to compare the real parts of the variables. These changes, as well as changes to the input and output files, are minor, and a derivative code that generates the exact derivatives can be produced quite quickly. Unfortunately, this method is quite computationally expensive because a complex function evaluation must be performed for each design space derivative.

2.1.4. Sensitivity Analysis

The extremely large computational expense of estimating the design space gradient can be mitigated by the use of sensitivity analysis. This technique uses information obtained from differentiating the system of governing equations to estimate the design space derivatives without the need for further function evaluations. Hence, the computational cost of multiple steady-state solutions can be avoided. There are two basic approaches within sensitivity analysis - the continuous approach and the discrete approach. With the continuous approach, the continuous, governing system of partial differential equations are differentiated to yield a system of adjoint equations, which are solved in addition to the governing equations, to provide the necessary information. With the discrete approach, the discretized system of equations resulting from the governing partial differential equations are differentiated at each node in the computational domain, and a linear matrix equation is solved for the necessary information. Hence, in the continuous approach, the governing equations are differentiated and then discretized; whereas, in the discrete approach, the governing equations are discretized and then differentiated.

2.1.4.1. The Continuous Approach

Variational Calculus is often employed to derive the adjoint equations in the continuous approach, which is also referred to as the variational approach or the continuous adjoint approach. (The continuous approach also includes the derivation and use of the direct equations, which is analogous to the direct formulation of discrete sensitivity analysis; however, due to the computational cost associated

with the continuous direct method, which is on the same order of magnitude as finite differences, the direct equations in the continuous approach are rarely used.) The objective function F is a function over a specified region Ω_a of the computational domain and depends on the flow variables Q, the computational domain or grid χ and the design variables $\vec{\beta}$, or $F = \int_{\Omega_a} f(Q(\vec{\beta}), \chi(\vec{\beta}), \vec{\beta}) d\Omega$. For steady-state problems, the governing partial differential equations, which usually arise from conservation laws, are also functions of the flow variables, the grid and the design variables and can be expressed as $P_1(Q(\vec{\beta}), \chi(\vec{\beta}), \vec{\beta}) = 0$, $P_2(Q(\vec{\beta}), \chi(\vec{\beta}), \vec{\beta}) = 0$ up to $P_{Ne}(Q(\vec{\beta}), \chi(\vec{\beta}), \vec{\beta}) = 0$, where Ne is the number of equations. Boundary conditions are used to specify the flow variables at the boundaries and can be expressed as $B_1(Q(\vec{\beta}), \chi(\vec{\beta}), \vec{\beta}) = 0$, $B_2(Q(\vec{\beta}), \chi(\vec{\beta}), \vec{\beta}) = 0$ up to $B_{Nb}(Q(\vec{\beta}), \chi(\vec{\beta}), \vec{\beta}) = 0$, where Nb is the number of boundary conditions. A Lagrangian is formed from the objective function, the governing partial differential equations and the boundary conditions, as

$$\mathcal{L}(Q, \chi, \vec{\beta}, \lambda_j, \Gamma_k) = \int_{\Omega_a} f d\Omega + \sum_{j=1}^{Ne} \left(\int_{\Omega} \lambda_j P_j d\Omega \right) + \sum_{k=1}^{Nb} \left(\int_{S_k} \Gamma_k B_k dS \right) \tag{2.1.6}$$

The variables λ_j and Γ_k are often referred to as the co-state variables, as opposed to the flow variables Q which are referred to as the state variables. The number of adjoint variables λ_j is equal to the number of partial differential equations. When the partial differential equations and boundary conditions are satisfied, the value of the Lagrangian is simply the value of the objective function.

The variation of $\mathcal{L}$ is

$$\delta\mathcal{L} = \frac{\partial\mathcal{L}}{\partial Q}\delta Q + \frac{\partial\mathcal{L}}{\partial\chi}\delta\chi + \sum_i \frac{\partial\mathcal{L}}{\partial\beta_i}\delta\beta_i + \sum_j \frac{\partial\mathcal{L}}{\partial\lambda_j}\delta\lambda_j + \sum_k \frac{\partial\mathcal{L}}{\partial\Gamma_k}\delta\Gamma_k \tag{2.1.7}$$

When the governing equations are satisfied,

$$\frac{\partial\mathcal{L}}{\partial\lambda_j} = \int_\Omega P_j d\Omega = 0 \tag{2.1.8}$$

and

$$\frac{\partial\mathcal{L}}{\partial\Gamma_k} = \int_{S_k} B_k dS = 0 \tag{2.1.9}$$

Before determining $\frac{\partial\mathcal{L}}{\partial Q}$, the integrals in the Lagrangian that contain the adjoint variables λ_j are integrated by parts to move the derivatives to the adjoint variables. Also, $\frac{\partial\mathcal{L}}{\partial Q}$ is a vector that consists of the partial derivatives of the Lagrangian with respect to the conservative variables. Hence, by setting each term in this vector to zero, the system of adjoint equations are derived, as in the example to follow. After these substitutions and the assumptions, the Lagrangian becomes

$$\delta\mathcal{L} = \frac{\partial\mathcal{L}}{\partial\chi}\delta\chi + \sum_i \frac{\partial\mathcal{L}}{\partial\beta_i}\delta\beta_i \tag{2.1.10}$$

or

$$\begin{aligned}\frac{\delta\mathcal{L}}{\delta\beta_i} &= \frac{\partial\mathcal{L}}{\partial\chi}\frac{\delta\chi}{\delta\beta_i} + \frac{\partial\mathcal{L}}{\partial\beta_i} = \left.\frac{d\mathcal{L}}{d\beta_i}\right|_{\text{Q fixed}} \\ &\approx \frac{\mathcal{L}(Q(\vec{\beta}), \chi(\vec{\beta} + e_i\Delta\beta_i), \vec{\beta} + e_i\Delta\beta_i, \lambda_j, \Gamma_k) - \mathcal{L}(Q(\vec{\beta}), \chi(\vec{\beta}), \vec{\beta}, \lambda_j, \Gamma_k)}{2\Delta\beta_i}\end{aligned} \tag{2.1.11}$$

Once the adjoint variables λ_j and the boundary values Γ_k are determined from setting

$\frac{\partial \mathcal{L}}{\partial Q}$ to zero, the variation of the Lagrangian with respect to the design variables $\vec{\beta}$ can be determined in a variety of ways including hand-differentiation and finite differences, without the need for a steady-state solution. Furthermore, since the partial differential equations and boundary conditions are also satisfied, the variation of the Lagrangian with respect to the design variables is also the design space derivatives or

$$\frac{dF}{d\beta_i} = \left.\frac{d\mathcal{L}}{d\beta_i}\right|_{\text{Q fixed}} \tag{2.1.12}$$

For time-dependent objective functions, the initial conditions must be included within the Lagrangian and the terms in the Lagrangian must also be integrated with respect to time. Other considerations may also be necessary due to the time-dependence of the Lagrangian.

To demonstrate the derivation of the adjoint equations, consider the steady shallow water equations expressed as

$$\frac{\partial}{\partial x}\begin{pmatrix} F_1 \\ F_2 \\ F_3 \end{pmatrix} + \frac{\partial}{\partial y}\begin{pmatrix} G_1 \\ G_2 \\ G_3 \end{pmatrix} + \begin{pmatrix} H_1 \\ H_2 \\ H_3 \end{pmatrix} = 0 \tag{2.1.13}$$

where F_1, F_2, F_3, G_1, G_2, G_3, H_1, H_2 and H_3 are the appropriate terms which are defined in Appendix F. The three conservative variables are depth h, x-discharge p and y-discharge q, so there are three adjoint variables λ_1, λ_2 and λ_3. The objective function f is $(\bar{h} - h)^2$, where $\bar{h}$ is the average depth over Ω_a. Thus the Lagrangian is

$$\begin{aligned}\mathcal{L} = &\int_{\Omega_a} (\bar{h} - h)^2 d\Omega + \int_{\Omega} \lambda_1 \left(\frac{\partial F_1}{\partial x} + \frac{\partial G_1}{\partial y} + H_1 \right) d\Omega \\ &+ \int_{\Omega} \lambda_2 \left(\frac{\partial F_2}{\partial x} + \frac{\partial G_2}{\partial y} + H_2 \right) d\Omega + \int_{\Omega} \lambda_3 \left(\frac{\partial F_3}{\partial x} + \frac{\partial G_3}{\partial y} + H_3 \right) d\Omega \\ &+ \text{Boundary Integrals}\end{aligned} \tag{2.1.14}$$

Integrating by parts yields

$$\begin{aligned}\mathcal{L} = &\int_{\Omega_a} (\bar{h} - h)^2 d\Omega + \int_{\Omega} \left(\lambda_1 H_1 - \frac{\partial \lambda_1}{\partial x} F_1 - \frac{\partial \lambda_1}{\partial y} G_1 \right) d\Omega \\ &+ \int_{\Omega} \left(\lambda_2 H_2 - \frac{\partial \lambda_2}{\partial x} F_2 - \frac{\partial \lambda_2}{\partial y} G_2 \right) d\Omega + \int_{\Omega} \left(\lambda_3 H_3 - \frac{\partial \lambda_3}{\partial x} F_3 - \frac{\partial \lambda_3}{\partial y} G_3 \right) d\Omega \\ &+ \text{Boundary Integrals}\end{aligned} \tag{2.1.15}$$

So,

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial h} = &\int_{\Omega_a} 2(h - \bar{h}) d\Omega + \int_{\Omega} \left(\lambda_1 \frac{\partial H_1}{\partial h} - \frac{\partial \lambda_1}{\partial x} \frac{\partial F_1}{\partial h} - \frac{\partial \lambda_1}{\partial y} \frac{\partial G_1}{\partial h} \right) d\Omega \\ &+ \int_{\Omega} \left(\lambda_2 \frac{\partial H_2}{\partial h} - \frac{\partial \lambda_2}{\partial x} \frac{\partial F_2}{\partial h} - \frac{\partial \lambda_2}{\partial y} \frac{\partial G_2}{\partial h} \right) d\Omega \\ &+ \int_{\Omega} \left(\lambda_3 \frac{\partial H_3}{\partial h} - \frac{\partial \lambda_3}{\partial x} \frac{\partial F_3}{\partial h} - \frac{\partial \lambda_3}{\partial y} \frac{\partial G_3}{\partial h} \right) d\Omega \\ &+ \text{Boundary Integrals}\end{aligned} \tag{2.1.16}$$

The other partial derivatives $\frac{\partial \mathcal{L}}{\partial p}$ and $\frac{\partial \mathcal{L}}{\partial q}$ are similar to equation (2.1.16) with the exceptions that the partial derivatives are with respect to p and q rather than h and that the partial derivative of the objective function is zero because it does not depend on the x- or y-discharges. The adjoint equations, which are linear in the adjoint variables, are obtained by setting the integrands to zero. The boundary conditions can be determined by setting the integrands in the boundary integrals to zero.

The continuous approach generates a system of adjoint equations that must be solved, in addition to the system of governing partial differential equations. Since the system of governing partial differential equations are usually solved iteratively, as if they were time-dependent problems, the adjoint equations are often solved iteratively, following the same method that was used to solve the partial differential equations. However, other solution methods or grids can be used to solve the adjoint equations.

The primary difficulties of the continuous approach are the need to derive the adjoint equations, the need to rewrite the analysis code to solve a different set of partial differential equations and the computational cost of solving the adjoint equations, which is reported to be on the order of one steady-state solution of the governing equations. As the above presentation would suggest, derivation of the adjoint equations is not trivial. Furthermore, when changes to the mathematical model are made, the adjoint equations must be re-derived, and the result must be re-coded, in order to be consistent with the new set of governing equations.

2.1.4.2. The Discrete Approach

Discrete sensitivity analysis, also known as direct differentiation, is applied to the discretized set of equations, rather than the governing partial differential equations. Once the computational domain is discretized, the objective function can be dependent on the flow variables Q, the grid χ and the design variables $\vec{\beta}$, or $F(Q(\vec{\beta}), \chi(\vec{\beta}), \vec{\beta})$, and its design space derivative is

$$\frac{dF}{d\beta_i} = \frac{\partial F}{\partial Q}^T \frac{\partial Q}{\partial \beta_i} + \frac{\partial F}{\partial \chi}^T \frac{\partial \chi}{\partial \beta_i} + \frac{\partial F}{\partial \beta_i} \tag{2.1.17}$$

where $\frac{dF}{d\beta_i}$ represents the total variation of F with respect to the design variable β_i, which includes both the implicit and explicit dependencies. Since F is explicitly dependent on Q, χ and $\vec{\beta}$, the terms $\frac{\partial F}{\partial Q}$, $\frac{\partial F}{\partial \chi}$ and $\frac{\partial F}{\partial \beta_i}$ can be calculated easily by hand. The term $\frac{\partial \chi}{\partial \beta_i}$ can be estimated via finite differencing the results of the grid generation code or by differentiating explicit dependencies between the grid and the design variables via hand-differentiation, CTSE or ADIFOR. Unfortunately, the vector $\frac{\partial Q}{\partial \beta_i}$ can not be estimated via finite differences without an additional steady-state simulation. But, this vector also appears in the equation for the derivative of the discretized system of governing equations $W(Q(\vec{\beta}), \chi(\vec{\beta}), \vec{\beta}) = 0$ or

$$0 = \frac{dW}{d\beta_i} = \frac{\partial W}{\partial Q}\frac{\partial Q}{\partial \beta_i} + \frac{\partial W}{\partial \chi}\frac{\partial \chi}{\partial \beta_i} + \frac{\partial W}{\partial \beta_i} \tag{2.1.18}$$

Since W is an implicitly defined function, the total derivative of W with respect to any variable is zero. Again, each term except for $\frac{\partial Q}{\partial \beta_i}$ can be calculated, either by hand-differentiation or by finite differences, without the need for a steady-state simulation. Using equation (2.1.18) in conjunction with equation (2.1.17), the design space derivative $dF/d\beta_i$ can be estimated efficiently without the need for additional steady-state simulations.

In the adjoint variable formulation of discrete sensitivity analysis, equation (2.1.18) is multiplied by an adjoint variable λ and added to equation (2.1.17) to yield

$$\frac{dF}{d\beta_i} = \left(\frac{\partial F}{\partial Q}^T + \lambda^T \frac{\partial W}{\partial Q}\right)\frac{\partial Q}{\partial \beta_i} + \left(\frac{\partial F}{\partial \chi}^T + \lambda^T \frac{\partial W}{\partial \chi}\right)\frac{\partial \chi}{\partial \beta_i} + \frac{\partial F}{\partial \beta_i} + \lambda^T \frac{\partial W}{\partial \beta_i} \tag{2.1.19}$$

The need to calculate $\frac{\partial Q}{\partial \beta_i}$ is removed by setting the terms associated with $\frac{\partial Q}{\partial \beta_i}$ equal

to zero, or

$$\begin{aligned} \frac{\partial F}{\partial Q}^T + \lambda^T \frac{\partial W}{\partial Q} = 0 \\ \frac{\partial W}{\partial Q}^T \lambda = -\frac{\partial F}{\partial Q} \end{aligned} \tag{2.1.20}$$

For the adjoint variable formulation, λ is independent of the design variables and must only be calculated once for each objective function. For most single discipline optimization problems, there is only one objective function and a few constraints; whereas, typically there are many design variables. Since the adjoint variable formulation scales with the number of functions and constraints, it is more computationally efficient for this class of problems than the direct formulation which scales with the number of design variables. Thus, for this reason, many researchers have used the adjoint variable formulation in their optimization work.

In the direct formulation of discrete sensitivity analysis, or quasi-analytic method, the vector $\frac{\partial Q}{\partial \beta_i}$ is obtained by solving equation (2.1.18) directly or

$$\frac{\partial W}{\partial Q}\frac{\partial Q}{\partial \beta_i} = -\frac{\partial W}{\partial \chi}\frac{\partial \chi}{\partial \beta_i} - \frac{\partial W}{\partial \beta_i} = -\left.\frac{dW}{d\beta_i}\right|_{\text{Q fixed}} \tag{2.1.21}$$

This result is used to estimate the design space derivative. Since $\frac{\partial Q}{\partial \beta_i}$ is dependent on the design variable and not on the objective function, the vector $\frac{\partial Q}{\partial \beta_i}$ must be calculated for each design variable. Thus, the direct formulation is not as computationally efficient as the adjoint variable formulation when the number of design variables is greater than the number of objective functions and constraints. However, if one uses the Gauss-Newton optimization algorithm, which views each term in the objective function as a separate residual function, the effective number of objective functions

to be differentiated is much larger than the number of design variables, making the quasi-analytic method more cost effective. Because the Gauss-Newton algorithm yields super-linear convergence, the overall computational cost of this type of design optimization strategy may be less than any of the strategies available through the use of the adjoint variable formulation.

For both formulations of discrete sensitivity analysis, the flow variables are driven to steady-state, and the sensitivity analysis subroutines are invoked. For implicit implementations of the flow solver, the Jacobian matrix $\frac{\partial W}{\partial Q}$, as well as the matrix inversion subroutines, is available through the flow solver. Hence, all the necessary tools are available from the flow solver, and few modifications to the code are necessary. In addition, the computational cost for discrete sensitivity analysis is smaller than for the continuous approach of sensitivity analysis due to the need to perform the pertinent calculations only at steady-state.

The primary difficulty with discrete sensitivity analysis is the need to generate highly accurate Jacobian matrices. The discretized equations $W = 0$ will usually contain complicated terms resulting from friction and turbulence models and from the discretization scheme and upwinding methods. To hand-differentiate these terms correctly can be quite laborious and error-prone. Also, when spatially higher-order schemes are used, the connectivity stencil for the scheme usually includes more nodes than a first-order scheme. Thus, there are more non-zero entries in the Jacobian for higher-order schemes than for first-order schemes, which implies a greater computational expense to generate the Jacobians and to solve the matrix equation. As a

result, most implicit flow solvers only approximate the Jacobian matrix, by neglecting the friction and turbulence models, by making assumptions about the upwinding methods and by using first-order schemes. By iteratively solving the flow solver's matrix equation, the discretized equations are driven to zero. As long as the approximate Jacobian matrix succeeds in driving the system to zero, the approximation is accurate enough for the flow solver. However, since the matrix equation used in discrete sensitivity analysis is linear, iterative methods can not be used to solve equation (2.1.20) or (2.1.21) directly, and the accuracy of the design space derivatives depend greatly on the accuracy of the Jacobian. Iterative schemes can be used to avoid the computational cost of solving matrix equations that use higher order Jacobians as described in Appendix A; nevertheless, highly accurate Jacobians are still needed to generate the most accurate design space derivatives.

Furthermore, for explicit solution methods, the Jacobian matrix is typically not used and is unavailable for discrete sensitivity analysis. To remedy this problem, Shubin and Frank [**8**] suggested that finite differences be used to estimate the Jacobian matrix from the discretized equations. Instead of using finite differences, which have accuracy problems, the complex Taylor's series expansion (CTSE) method is used in this research to generate highly accurate Jacobian from the residual vector W. This method can also be used to generate the Jacobian matrix for explicit codes once the flow variables are determined. (See Appendix D for details.) By using the complex Taylor's series expansion method, the need to hand-differentiate the discretized equations correctly, which is the main difficulty of discrete sensitivity

analysis, is removed. This technique can be applied to both explicit and implicit codes and results in highly accurate design space derivatives.

2.1.4.3. Accuracy Issues

Within the discrete approach, error can be introduced into the calculations of the design space derivatives via several terms. In both the adjoint variable formulation and direct formulation, the same terms are calculated, although they are used in different ways. In particular, the partial derivatives of the objective function F, the Jacobian matrix $\frac{\partial W}{\partial Q}$ and the vector $\left.\frac{dW}{d\beta_i}\right|_{\text{Q fixed}}$ must be determined. In addition, the discrete approach assumes that the residual vector $W(Q)$ has been successfully driven to zero and that the matrix inversion operation is highly accurate, which may not be the case if iterative solvers are prematurely terminated. For the work in this dissertation, the partial derivatives of the objective function F are calculated exactly via hand-differentiation, and the Jacobian matrix is calculated via the complex Taylor's series expansion method, which is quite accurate. The finite difference approximation to the vector $\left.\frac{dW}{d\beta_i}\right|_{\text{Q fixed}}$ introduces error into the design space gradient, which can be removed if more accurate methods are used to calculate this term. Careful use of the flow solver and matrix inversion subroutines can greatly reduce the error that these two elements introduce into the calculations. Once these sources of error have been removed, highly accurate and even numerically exact design space derivatives can be generated.

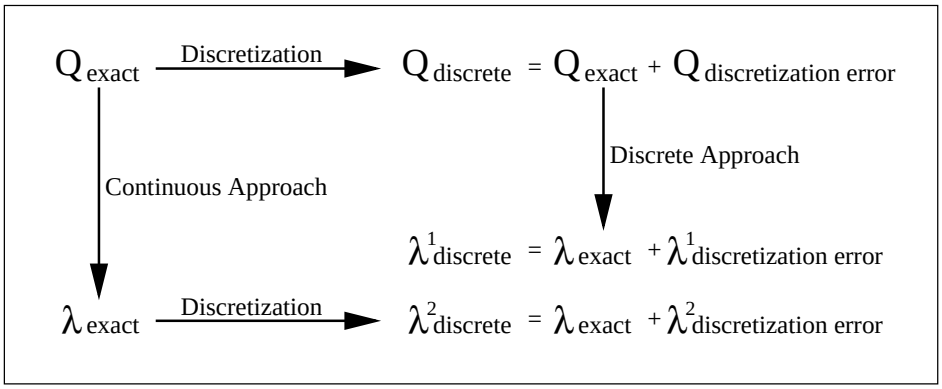

Figure 2.1.4.3.1. Discretization Error Comparison.

Within the continuous adjoint approach, the choice of grid and solution method will affect the accuracy of the adjoint variable λ and hence the accuracy of the design space derivatives. As shown in Figure 2.1.4.3.1, the discretization error produced by the continuous approach is not necessarily the same as the discretization error produced by the discrete approach. The discretization error produced by the continuous approach differs because the equations being discretized are different and because the grid and solution method can be different. However, the discretization error contained within the discrete approach is consistent with the discretization error produced by discretized the governing equations. Thus, the design space derivatives produced by the discrete approach will more closely model the exact derivatives of the objective function as modeled by the discretized governing equations. Furthermore, the other derivative estimation methods, such as ADIFOR and CTSE, also use the discretized variables $Q_{discrete}$ to estimate the gradient. As the discretization error goes to zero, the design space derivatives generated by both the continuous adjoint approach and the discrete approach will converge to the exact, continuous design space derivatives.

2.1.5. Conclusions about Optimization Methods

The advantages and disadvantages of the optimization methods presented in this chapter, as well as the type of optimization problem for which each method is best suited, is presented in Table 2.1.5.1. Because the open-channel design problem uses a high-fidelity simulation code, sensitivity analysis is used to generate the design space derivatives, since this method provides the design space derivatives within a reasonable amount of time. Furthermore, since it is less computationally expensive, easier to implement for implicit codes and provides consistent design space derivatives, discrete sensitivity analysis is used rather than continuous sensitivity analysis.

Table 2.1.5.1. Comparison of Optimization Methods.

Optimization Method	Type of Problem	Advantages	Disadvantages
Inverse Methods	Analytic Formula	Highly Efficient	Not Generally Applicable
Genetic Algorithm (Probabilistic Methods)	Discontinuous, Discrete, Cheap Simulations, Multi-Modal	Avoids Local Minima, No Gradient Needed	Many Function Evaluations
Finite Difference	Any	Easiest To Use	Large Comp. Cost, Accuracy
ADIFOR, CTSE	Any	Highly Accurate Derivative, Easy to Use	Large Computational Cost
Continuous Sensitivity Analysis	Explicit High-Fidelity	Computationally Efficient	Derive and Solve Adjoint Eqns.
Discrete Sensitivity Analysis	Implicit High-Fidelity	Accurate Derivatives, Efficient	Jacobian Matrix Needed

Table 2.1.5.2 compares the computational cost, the accuracy of the design space derivatives and the ease of implementation for the various gradient-estimation methods presented in this chapter. The use of the term "exact" indicates that the derivatives are accurate to machine precision. Since the simulation code solves the shallow water equations using an implicit method, discrete sensitivity analysis is an appropriate choice for this code, because of the relative ease of implementation and the promise of highly accurate design space derivatives.

Table 2.1.5.2. Costs and Benefits of Various Methods to Generate Derivatives.

Derivative Method	Computational Cost	Accuracy of Derivative	Ease of Implementation
Finite Difference	Expensive	Moderate	Very Easy
Automatic Differentiation	Expensive	Exact	Easy
Complex Taylor's Series Expansion Method	Expensive	Exact	Easy
Continuous Approach of Sensitivity Analysis	Cheap	Moderate	Very Difficult
Discrete Sensitivity Analysis for Implicit Codes	Cheap	Highly	Moderate
Discrete Sensitivity Analysis for Explicit Codes	Cheap	Moderate	Difficult

Finally, the distinctions between steady-state and time-dependent objective functions need to be addressed. For steady-state problems, such as the design optimization problems studied in this dissertation, the steady-state flow for a particular design is obtained by solving the partial differential equations. For time-dependent problems, the time-dependent partial differential equations are solved, and the objective function is evaluated based on the flow variables at each time level. Finite differences and the complex Taylor's series expansion method only consider the objective func-

tion, without any consideration of the evaluation method. Automatic differentiation uses the evaluation method to generate the derivative code, but it works the same for steady-state or time-dependent problems. Hence, these three methods can be applied in the same way to steady-state or time-dependent problems. (Much computational savings can be achieved by using previous steady-state information within these methods, so that the steady-state solution for the perturbed set of design variables is found in fewer iterations.) However, sensitivity analysis manipulates the derivatives of the governing equations. Since the time-dependent partial differential equations are different from the steady-state equations, sensitivity analysis as applied to the time-dependent equations is different from the sensitivity analysis that is presented in this section. In this section, only steady-state sensitivity analysis is studied. The equations for time-dependent discrete sensitivity analysis are derived in Appendix A.

2.2. Optimization in Aerospace Applications

Optimization problems within the aerospace branch of computational fluid dynamics generally deal with design improvement as opposed to parameter identification, which is more common in water resource applications. In particular, much research has focused on methods to optimize the shape of a two-dimensional airfoil for a variety of flight conditions, including Mach number and angle-of-attack. As computational resources have increased, more interest in three-dimensional flows, such as wing-body and wing-nacelle interactions, have developed. Outside of aircraft design, CFD has been used to simulate chemically-reactant flow from rocket engines, flow

through turbine engines, flow around submarines and their propellers, and flow past surface vessels, to name a few. Since there is interest in modeling these phenomenon accurately, there are design objectives that are driving these research thrusts. For instance, some of the objectives for rocket nozzle design could be to give better thrust outputs, to provide better maneuvering control and to prevent damage to the rocket.

Since inverse methods are generally restricted to analytic models, their use in high-fidelity CFD is limited. Dulikravich [**9**] gives an overview of inverse design methods that are currently being used in two and three-dimensional design of aircraft. Many of the methods are not applicable to viscous flows or become too computationally burdensome for three-dimensional models. In one example, Shinkyu, etal [**10**], used the linear small perturbation equation to formulate an inverse method for matching a target pressure distribution.

Because there are many, low-fidelity, computationally efficient methods for estimating the flow around airfoils, genetic algorithms (GA) have been used by many researchers in design optimization. Quagliarella and Della Cioppa [**11**] used a genetic algorithm in conjunction with a full potential flow solver to design a dual point transonic airfoil. King, etal [**12**], used a genetic algorithm to design the shape of an engine nozzle in order to generate the maximum thrust vectoring. Yamaguchi and Nakamura [**13**] used linear theory and slender body theory within a GA to design a supersonic transport that minimized the sonic boom. Anderson and Gebert [**14**] used a GA to design a subsonic wing, where the design parameters controlled the planform, twist and thickness distribution of a three-dimensional wing to produce a high lift-to-drag

ratio. The flow solutions were performed using finite wing theory, and the authors concluded that "applying more sophisticated aerodynamic simulation methods (Euler or Navier-Stokes based solutions) would make applying simple genetic algorithms impractical for such large optimization spaces given current computer hardware."[1] Nevertheless, Oyama, etal [**15**], used the three-dimensional Navier-Stokes equations within a genetic algorithm to optimize the shape of a transonic wing. To overcome the computational costs, the authors used a multigrid implementation on a parallel vector machine with 166 processors to evaluate the population. They also limited the size of the design space to 34 design variables. As a result, each generation of 50 members could be evaluated in approximately 80 minutes, and the entire design process took about 2 days. Thus, parallel computation may hold the answer to reduce the computational constraints of combining genetic algorithms and high-fidelity simulations.

Within gradient-based methods, finite differences, automatic differentiation, the complex Taylor's series expansion method and sensitivity analysis have been used to estimate the gradient. Huddleston [**16**] used finite differences to estimate the derivatives for Euler and Navier-Stokes codes applied to nozzles, airfoils and forebody simulators. His work focused on the use of the Gauss-Newton optimization algorithm and quasi-Newton algorithms to use the gradient information effectively. Huddleston determined that the finite difference derivatives were not accurate enough for use

[1] p. 370, Anderson, M. B. and Gebert, G. A., "Using Pareto Genetic Algorithms for Preliminary Subsonic Wing Design", 6th AIAA/NADA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA 96-38738, pp. 363-371, 1996.

within the BFGS update method to achieve convergence; however, the Gauss-Newton method was able to identify excellent designs in 4-6 design iterations.

Carle, etal [**17**], used ADIFOR to transform a three-dimensional, thin-layer, Navier-Stokes, multi-grid flow solver into a design optimization code. This code included several complicated turbulence models that were deemed to be too difficult for hand-differentiation, but ADIFOR was able to correctly differentiate the code and generate accurate results. However, the computational expense was quite large. The authors concluded that "one can obtain the derivative residual . . . at a cost roughly equal to that of evaluating (the function) multiplied by the number of design parameters."[2] They also stated that automatic differentiation provides reliable derivative information but "significant advances are necessary for the efficiency of ADIFOR-generated derivative code to become truly competitive with hand-differentiated code."[3] Taylor and Oloso [**18**] reviewed the application of ADIFOR to a commercial aerodynamic analysis code called CFL3D and concluded that the derivatives produced by ADIFOR were more accurate that those produced by finite differences. In their paper, the authors described some of the difficulties in applying ADIFOR to vector and parallel codes. They also discussed the computational and memory requirements of ADIFOR-generated derivative codes and concluded that "the well-known adjoint variable formulations will always have an insurmountable inherent advantage with respect to overall computational resources required (i.e., computer

[2]p. 5, Carle, A., Green, L. L., Bischof, C. H., and Newman, P. A., "Applications of Automatic Differentiation in CFD", 25th AIAA Fluid Dynamics Conf., AIAA 04-2197, 1994

[3]p. 1, Ibid

memory and CPU time)."[4] Methods are being investigated to improve the computational efficiency of the derivative code, including the use of ADIFOR within discrete sensitivity analysis to generate the Jacobian matrix, as suggested by Taylor and Oloso. Unger and Hall [**19**] used automatic differentiation to generate the derivative codes for a multi-disciplinary design problem. Some of the constraints in this problem were take-off gross weight, maximum fuel and maximum approach speed, and the primary goal was to maximize the range of the aircraft. From these papers, it is clear that ADIFOR is able to generate highly accurate derivative codes, but the computational cost of the codes is severely limiting.

The complex Taylor's series expansion method is a relatively new technique and has not received widespread use. However, Newman, etal [**20**], have used this technique to generate design space derivatives for aerodynamic and structural codes. The aerodynamic code was a finite-volume, unstructured, three-dimensional Euler/Navier-Stokes flow solver, while the structural analysis code was a finite element code, demonstrating that this method can be applied to both finite volume and finite element codes. However, the computational cost of this method is quite large, requiring a complex variable flow solution for each design variable, which is more than is needed for finite difference approximations. Whitfield and Taylor [**21**] used the complex Taylor's series expansion method to generate the flux Jacobian, which is the matrix $\partial W/\partial Q$, for a three-dimensional Navier-Stokes code. This application of the complex

[4] p. 5, Taylor, A. C., III, and Oloso, A., "Aerodynamic Design Sensitivities by Automatic Differentiation", 29th AIAA Fluid Dynamics Conf., AIAA 98-2536, 1998

Taylor's series expansion method was the same method used herein to generate exact Jacobians of the residual vector and is discussed in Appendix D.

The continuous approach of sensitivity analysis has been applied to potential flow solvers around airfoils, to Euler codes and more recently to the viscous Navier-Stokes equations. Jameson and Reuther [**22, 23**] applied control theory, which is another term for the continuous approach, to a potential flow solver and then to the inviscid Euler equations, for designing an airfoil. In his dissertation, Reuther [**24**] derives the adjoint equations for both types of flows and compares the computational expense of using the adjoint equations to generate the design space derivatives with using finite differences. The design space derivatives provided in his research show an agreement with the finite difference derivatives to within a few percent. Reuther concluded that the adjoint based gradient calculation was on the order of twice the computational cost of the flow analysis and that the adjoint based method was over 7 times more efficient than the finite difference method for a problem with 50 design variables. Furthermore, as the number of design variables increases, the computational savings becomes more pronounced. Due to the changes in the computational expense of generating the design space derivatives, Reuther states that "the development of a method to obtain cheap gradients (computationally) is forcing a reexamination of what design algorithm is appropriate."[5] In his research, he used a quasi-Newton (BFGS) method with several linear searches in the gradient direction.

[5]p. 81, Reuther, J. J., "Aerodynamic Shape Optimization Using Control Theory", Dissertation, Univ. Calif. Davis, 1996

Anderson and Venkatakrishnan [**25**] applied the continuous adjoint formulation to an inviscid, unstructured Euler code. Soermarwoto [**26**] successfully used the variational method to generate the adjoint equations for a viscous, two-dimensional, Navier-Stokes code. Extending this method to three-dimensions, Jameson [**27**] developed a design optimization code for the viscous Navier-Stokes equations, which has been used by McDonnell Douglas "to support design studies for the MDXX project"[6]. A brief glance at these papers reveals the complexity of the derivation of the adjoint equations; however, by successfully applying the continuous approach to a three-dimensional, high-fidelity, viscous code, a useful tool for design optimization within compressible fluid flow has been developed. Unfortunately, when a different turbulence model is developed, the adjoint equations must be re-derived. Furthermore, from the results quoted in these papers, it appears that the design space derivatives generated by these codes are only accurate to within a few percent. This level of error may prevent convergence to the optimal design, although significant design improvement is possible.

In a related method, Kuruvila, Ta'asan, Salas and Arian [**28, 29**] have investigated a method that generates an optimal design while computing only one flow analysis. This "one-shot" method takes advantages of multi-grid methods as applied to the adjoint equations, so that the entire design process takes about three times the computational cost of the flow analysis. In the paper by Kuruvila [**28**], the

[6]p. 45, Jameson, A., "Essential Elements of Computational Algorithms for Aerodynamic Analysis and Design", ICASE Report No. 97-68, 1997

"one-shot" method was applied to the potential flow equations; and in the paper by Arian and Ta'asan [**29**], this method was applied to the two-dimensional, Poisson equations with either Dirichlet or Neumann boundary conditions. Iollo, etal [**30**], developed a similar scheme for the two-dimensional Euler equations as applied to the airfoil design. In their method, they obtained significant computational savings by assuming that small changes in the shape of the airfoil would only cause significant changes to the flow field near the surface of the airfoil and hence performed most of the computations only in a small sub-domain of the entire flow field. Because of this assumption, this method is not readily extendible to other optimization problems.

In regards to discrete sensitivity analysis, Shubin and Frank [**8,31,32**] applied the discrete approach to a one-dimensional steady-state inviscid problem governed by the Euler equations and demonstrated that the discrete approach and the variational or continuous approach were mathematically identical, under certain circumstances. From their research, they concluded "that the gradients computed by the variational method can sometimes be sufficiently inaccurate to cause the optimization to fail"[7]

Baysal and Eleshaky [**33**] applied discrete sensitivity analysis to the design of nozzle-afterbody configurations using a two-dimensional Euler code. Baysal, etal [**34**], later extended this work to a third-order discretization of the Euler equations, which required a larger bandwidth in the Jacobian matrix. Eleshaky and Baysal [**35**] extended the use of discrete sensitivity analysis to viscous, two-dimensional, Navier-

[7]p. 67, Shubin, G. R., and Frank, P. D., "A Comparison of Two Closely-Related Approaches to Aerodynamic Design Optimization", 3rd Int. Conf. on Inverse Design Concepts and Optimization in Eng. Sci. (ICIDES-III), Oct. 19991.

Stokes flows over an airfoil. Burgreen and Baysal [**36**] applied discrete sensitivity analysis to three-dimensional wing design using the Euler equations. The primary goal of their research focused on methods to decrease the computational cost of each component of the design optimization process. In particular, the primary costs arise from the flow analysis, the gradient computation and the use of the gradient information, via linear searches. In many of these papers, they used Bezier curves to define the surface, so that there would be fewer design variables and hence fewer design space derivatives. To reduce the computational costs of the flow analyses, they used alternating direction implicit methods [**37**], preconditioned conjugate gradient methods [**38**] and domain decomposition of three-dimensional problems [**39**]. By using discrete sensitivity analysis, the cost of estimating the gradient was greatly reduced, and in several papers, they used the flow prediction component of discrete sensitivity analysis to reduce the computational cost of the linear searches.

Another group of researchers, Taylor, Hou and Korivi [**40,41,42,43**], have applied discrete sensitivity analysis to viscous, two-dimensional, thin-layer Navier-Stokes equations for flow through subsonic and supersonic nozzles, through double throated nozzles and around airfoils. In [**40**], they use the lift and drag coefficients for airfoils as the objective functions. In [**43**], they present detailed descriptions of the differences between the adjoint variable formulation and the direct differentiation approach, which is called the direct formulation in this research, and they analyze the need to handle the boundary conditions within the derivations consistently. The resulting design space derivatives are compared with finite differences and in many cases agree

to four significant digits. For the double-throated nozzle [**42**], which had 10 design variables and four objective functions, the CPU time for the direct differentiation method is 65 seconds, whereas the cost for finite differences is 1800 seconds.

Several recent articles survey the recent developments within design optimization as applied to aerodynamics. Sobieszczanski-Sobieski and Haftka [**44**] survey recent developments in multidisciplinary design. Jameson [**27**] gives an overview of three-dimensional, viscous analysis and design using multi-grid methods and control theory. Newman, etal [**45**] , gives a wide perspective of design efforts for structured and unstructured, two- and three-dimensional problems, with an emphasis on multi-disciplinary design.

From this survey of design optimization applications within the aerospace branch of CFD, one can conclude that the various gradient-based approaches, including automatic differentiation, the continuous approach of sensitivity analysis and the discrete approach of sensitivity analysis, have been successfully applied to a variety of high-fidelity simulation codes. The difficulties associated with viscous effects and turbulence modeling, with three-dimensional modeling and with the use of structured and unstructured grids have been overcome, to a large extent. Furthermore, these techniques have been applied to a wide range of problems within the design of aircraft, including wing-body design, wing-nacelle design, nozzle design, and nozzle-afterbody design. Future research will probably include the validation of the designs generated via numerical optimization, application of these techniques to more complex and realistic mathematical models, investigation of efficient use of the gradient information,

and a re-evaluation of the efficiency of these methods when applied to unsteady, to multi-point and to multi-disciplinary design optimization.

2.3. Analysis and Optimization in Water Resources

Within water resources applications, the primary areas of application can be divided into two groups - surface water modeling and groundwater modeling. Surface water modeling includes open-channel flow, contaminant transport and salinity migration in rivers and estuaries, interactions between sewage systems and bodies of water, surface runoff after precipitation events and the associated flood modeling, and evaporation and transpiration. Groundwater modeling includes modeling of hydraulic heads in confined and unconfined aquifers, contaminant transport and absorption of contaminants in aquifers, and the transfer of moisture through the region between the soil surface and the aquifer, better known as the vadose zone. Each group interacts with the other, as surface water can be absorbed by the ground and enter the aquifers, aquifers replenish streams and rivers, rainwater falls on the ground and can evaporate back into the atmosphere, so complete models of the water cycle can be extremely complex.

Hampering the modeling process is the lack of specific knowledge of the model parameters. For instance, in surface water modeling, a knowledge of soil types, vegetation types and elevation are needed, as well as knowledge of the precipitation concentrations over the river basin. Since this information often changes dramatically over the space of a few feet, accurate, detailed knowledge of these parameters is almost impossible. Furthermore, the available information will often be quite noisy,

containing spurious and erroneous information. For instance, in groundwater modeling, the only available information about the parameters are those obtained at the pumping and observation wells. Due to the age and possible lack of maintenance of the pumping mechanism, the measured pumping rates and hydraulic heads at the well may not be accurate.

As a result of the uncertainty of the model parameters, specialized methods, such as stochastic methods, Monte Carlo methods and fuzzy logic, are often used to analyze these systems in order to make predictions that are highly probable. Parameter identification is of chief importance in the proper use of the numerical and physical models, and most of the papers discussed in this section deal with parameter identification methods. Also, due to the temporal nature of many of these problems, the objective functions are often least squares functions that measure the difference between the actual, recorded data and the numerically predicted data, as a function of time. This temporal dependence will affect the choice of method to estimate the design space gradient. However, because the research in this dissertation deals with steady-state flow in open-channels, the objective function is not a function of time, and discrete sensitivity analysis is a good choice because this method only uses the steady-state information in the estimation of the design space gradient.

In surface water modeling, neural networks, genetic algorithms and continuous and discrete sensitivity analysis have been applied to a variety of problems. Sanchez, etal[**46**], used neural networks to analyze the design of coastal sewage systems by using the available information from previous rainfall and overflow events with the goal

of preventing sewage from detrimentally affecting tourist beaches. Mohan [**47**] used a genetic algorithm to identify three parameters for nonlinear Muskingum models for one-dimensional flood routing. Hsu, etal[**48**], used two-dimensional hydrodynamic models to estimate the friction coefficients and the turbulent diffusion and dispersion coefficients for salinity migration in a river system in Taiwan. Soulis and Psoni[**49**] used an iterative design method to match target wall depths in two-dimensional channel expansions and contractions.

Sensitivity analysis has recently been applied to a variety of open-channel flow and river analysis problems. Piasecki and Katopodes [**50**] used the continuous approach to determine design space derivatives of contaminant concentrations in critical areas with respect to times and rates of contaminant releases from industrial facilities. They used a two-dimensional, depth averaged model to calculate either the steady-state or periodic flow in the river system, including tidal effects, and derived the adjoint equation for the transport equation using the steady-state or periodic flow variables. Khatibi, etal [**51**], used the continuous approach as applied to the one-dimensional St. Venant equations to estimate friction parameters. They paid particular attention to the choice of objective function and effects of measurement errors on the parameter identification. Atanov, etal[**52**], used the continuous approach as applied to the one-dimensional St. Venant equations to determine the temporal control function for the upstream gate on a channel with the goal of minimizing depth variations in the channel, after the downstream gate has been opened. In a similar vein, Sanders and Katopodes [**53**] used the continuous approach as applied to the one-dimensional

shallow water equations to determine upstream and downstream gate positions to maintain a constant flow depth in the middle reach of the channel. Finally, Atanov, etal [**54**], used continuous sensitivity analysis to estimate the roughness or friction parameter for trapezoidal channels, as governed by the St. Venant equations. For a variety of reasons, including the temporal nature of these problems, these researchers deemed the continuous approach of sensitivity analysis to be the appropriate method for their work.

Within groundwater modeling, the main issues are the estimation of the parameters in the groundwater and transport equations. These parameters include the thickness of the aquifer, the hydraulic conductivity of the soil, the storativity of the soil and the dispersion coefficients. Unfortunately, these parameters can not be directly measured for each location in the domain. A large sampling of data could be obtained by digging many wells within the region to be studied, but this method would be quite expensive and impractical for realistic problems and would still rely on other methods to estimate these parameters for regions that were not sampled. Inverse methods can be used to determine these parameters based on the water levels in the wells, with the goal of generating the parameter values that produce the water levels given the pumping rates in the wells. Finally, stochastic methods have been used to give a probabilistic interpretation of the chances of the flow behaving in a certain fashion. Furthermore, error and uncertainty in the data plays a primary role in the current research efforts.

Eheart and Valocchi [**55**] used Monte Carlo methods to determine the optimal location of monitoring wells to detect groundwater contaminants from a landfill. In their method, they randomly determined the hydraulic conductivities based on a Gaussian distribution and randomly selected a contaminant leak location. Running their numerical model based on this information, they determined the effectiveness of the current placement of monitoring wells in detecting the leak. By generating enough random samples and determining the optimal placement that detects the leak for the entire set of samples, they could be reasonably confident that this placement of monitoring wells would be able to detect a leak for the actual landfill. These researchers used simulated annealing to determine the number and best placement of the monitoring wells, since the number of wells was not fixed.

Many gradient-based methods have been proposed for solving the parameter estimation and contaminant transport problems. Hantush and Marino [**56**] used the maximum likelihood method and Kalman filtering to solve the inverse problem with the goal of identifying the log transmissivities and storativities. Since they used an analytic function (negative log-likelihood), the gradient could be estimated via hand-differentiation. Whiffen, etal [**57**], used automatic differentiation to determine the derivatives required for an optimal control analysis for two-dimensional groundwater and transport models. In 1985, Townley and Wilson [**58**] used a form of discrete sensitivity analysis to calculate the derivatives of a particular functional with respect to all the parameters of the groundwater model. Sun and Yeh [**59**] derived the adjoint equations for a stochastic partial differential equation for two-dimensional

groundwater flow. By observing the dates on these papers, ranging from 1985 to current research, one can conclude that there are many methods to solve inverse problem of parameter estimation for the groundwater and contaminant transport equations and that this area of research is still quite active, even for two-dimensional models. Furthermore, Indelman, etal [**60**], addressed the issue of multiple leaky aquifers, while Swain [**61**] discussed the need to include aquifer recharge effects into the simulation, showing that many other issues remain to be studied.

Computational fluid dynamics (CFD) has been used in water resources to solve the flow of water in open-channels, spillways, rivers, estuaries and the ocean. As higher-fidelity, three-dimensional models are developed for these problems, the need for efficient parameter identification and design optimization will increase, especially as the complexity and size of the optimization problem grows. Examples of codes that have recently been developed to address these problems are the trapezoidal open-channel flow version of HIVEL2D developed by Stockstill, etal [**2**], a two-dimensional spillway flow model developed by Unami, etal [**62**], and three-dimensional open-channel flow models developed by Ye and McCorquodale [**63**] and by Casulli and Stelling [**64**]. Within groundwater modeling, three-dimensional groundwater and contaminant transport codes need to be converted into efficient parameter estimation codes. The issues of leaky aquifers, of aquifer recharge from lakes and rivers and through the vadose zone, and of transport of contaminants that do not mix with water remain to be addressed. Continuous and discrete sensitivity analysis have already been applied to many of these optimization problems, as well as genetic

algorithms, simulated annealing and automatic differentiation, and these methods will continue to be relevant to these areas of optimization in the future.