# Automatic Detection and Control of Hazardous Plumes in Wall-Bounded Flow Systems

by

April M. Warnock

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Civil Engineering)
in the University of Michigan
2013

For my son, Gavin.

# ACKNOWLEDGEMENTS

First and foremost, I would like to express my deep gratitude to my advisor, Professor Nikolaos Katopodes, for his continued support, guidance, and faith in me. Additionally, I am grateful for all the wonderful professors and fellow graduate students whose encouragement and knowledge I have benefited from during my tenure as a graduate student. A special thanks to Professor Kevin Maki for his help in getting me up to speed with OpenFOAM, as well as to Dr. Klaus-Peter Beier and Steffen Heise for their generous assistance in regards to developing the mesh for the McNamara Terminal.

I am grateful for the fellowships I have received from Rackham Graduate School and for the research grants provided by The National Science Foundation, both of which have made this research and my pursuit of a doctoral degree possible.

My family, who have provided me with encouragement and support in all aspects of my life, are a blessing for which I am always thankful. I cannot begin to enumerate the ways in which they have helped me achieve my academic goals, but without my sister's continuous and selfless friendship, my parents' perpetual concern and love, and my son's steadfast tolerance of every obstacle we have faced along the way, this achievement undoubtedly would not have been possible. Finally, a huge thank you to Dan, for being there.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| $\alpha$ | optimization step size |
| $\Delta t$ | time-step |
| $\delta$ | maximum acceptable value for the objective function $J$ |
| $\dot{q}$ | mass loading rate (continuous point source) |
| $\Gamma_\phi$ | generic diffusivity |
| $\Gamma_e$ | effective diffusivity |
| $\hat{C}$ | modeled contaminant concentration |
| $\mathbf{ds}$ | vector connecting two neighboring cell centroids |
| $\mathbf{f_T}$ | true optimization parameters |
| $\mathbf{f}$ | vector of optimization parameters |
| $\mathbf{f}_0$ | vector of initial optimization parameter guesses |
| $\mathbf{H}$ | PISO operator |
| $\mathbf{S}$ | outward pointing surface area vector |
| $\mathbf{U}$ | velocity vector |
| $\mathbf{U}_f$ | face-normal velocity |

| | |
|---|---|
| $\nu$ | kinematic viscosity |
| $\Omega$ | spatial domain |
| $\overline{\mathbf{U}}$ | Reynolds averaged velocity |
| $\phi$ | generic scalar field |
| $\phi_f$ | face interpolated quantity |
| $\sigma$ | regularization parameter |
| $\zeta$ | optimization convergence error |
| $a_E$ | matrix coefficient corresponding to cell neighboring centroid E |
| $a_P$ | matrix coefficient corresponding to cell with centroid $P$ |
| $B_k$ | optimization matrix |
| $C$ | contaminant concentration |
| $C$ | contaminant concentration |
| $D$ | diameter |
| $D_L$ | longitudinal diffusion coefficient |
| $D_T$ | transverse diffusion coefficient |
| $F$ | mass flux through a computational cell face |
| $f$ | generic source function |
| $F_b$ | flux across boundary face |
| $g_b$ | fixed gradient (for boundary conditions) |
| $h$ | finite differencing step-size |

|  |  |
|---|---|
| $i$ | general indexing subscript |
| $J$ | objective function; functional |
| $L$ | length; characteristic length scale |
| $N_c$ | number of computational cells in domain |
| $N_p$ | number of protected points for boundary control |
| $N_s$ | number of sensors |
| $N_{max}$ | maximum number of optimization iterations |
| $P$ | cell centroid |
| $p$ | kinematic pressure |
| $p_k$ | optimization search direction |
| $q$ | mass load (instantaneous point source) |
| $q_s$ | initial contaminant source strength |
| $R$ | radius, regulation term |
| $r$ | radial coordinate, adjoint source term |
| $T$ | finite prediction horizon |
| $t$ | time |
| $T_a$ | MPC advancement period |
| $T_f$ | termination time for MPC |
| $V$ | volume |
| CFL | Courant-Friedrichs-Lewy condition |

|  |  |
|---|---|
| GB | gradient-based optimization |
| Pe | Péclet number |

# ABSTRACT

Automatic Detection and Mitigation of Contaminant Plumes in Wall-Bounded Flows

by

April M. Warnock

Co-chairs: Nikolaos D. Katopodes and Valeriy Ivanov

Recent advances in technology and computational power have made the once theoretical concept of a real-time detection and control system, for the purpose of reducing risk from the deliberate or unintentional release of a hazardous plume, a practical reality. Such a system utilizes strategically placed sensor arrays and actuators in order to first detect the release of a hazardous chemical, and subsequently to determine the actions required by the actuators to effectively and expediently mitigate the plume. This basic framework is applicable to a number of real-world scenarios that can be described as wall-bounded, fluid-based systems, such as airport terminals, aqueducts, tall buildings and passenger tunnels.

The present research develops the theoretical and numerical framework for the automatic detection and control algorithm, which requires two main steps: a source inversion phase in order to trace the history of the plume and determine its original properties, followed by a boundary control phase in which the recovered source information is used to predict the propagation of the plume in time and space and thereby determine a control strategy to be performed in order to effectively mitigate it. Both the source inversion and boundary control phases can be formulated in terms of numerical optimization, and hence in this work a coupled CFD-optimization model has been developed using open source software. The CFD model implemented in this research is a RANS (Reynolds-Averaged Navier-Stokes) based finite-volume model developed using the OpenFOAM CFD library. The CFD model has been linked to an external optimization software suite (DAKOTA). The resulting model is used to simulate the source inversion and boundary control components of the automatic detection and control algorithm and to investigate the effect of various parameters on their accuracy, reliability and expediency. The results indicate that gradient-based optimization methods are successful for the boundary control phase; however the source inversion problem is complicated by issues arising from the discretization of the optimization parameters and ill-posedness. Hybrid optimization approaches offer some benefits with regards to the former problem, yet ill-posedness remains a significant challenge with respect to the source inversion process.

- **KEYWORDS: Model predictive control, hazardous release, computational fluid dynamics, contaminant source inversion, boundary control.**

# CHAPTER I

# Introduction

## 1.1 Problem Overview and Motivation

The goal of this thesis is to contribute to the development of the theoretical and numerical framework for an automatic contaminant plume detection and control system for wall-bounded flows. In the occurrence that a contaminant is accidentally or deliberately released into an ambient fluid, a need exists to identify the presence of the contaminant and subsequently take action to mitigate potential harm by either deflecting the contaminant or removing it from the system as rapidly as possible. The physical domain for this scenario could be any conduit carrying water or air, i.e. a channel, aqueduct, tunnel, tall building or passenger terminal. The contaminant plume could be comprised of any chemical that is capable of being detected by sensor arrays installed either along the boundaries or in the midst of the ambient fluid flow.

While the specific flow properties of each of these scenarios is variable, they can all be modeled as wall-bounded flow systems with the contaminant transport governed by the processes of advection and diffusion. Further, all of the aforementioned domains are capable of having ports along the boundaries that could act as actuators, providing blowing and/or suction capacity in order to control the fate of the contaminant plume. Figure 1.1 shows a schematic of an open channel flow with such a system of ports and sensors, whereby the sensors are depicted as being located at the boundary of

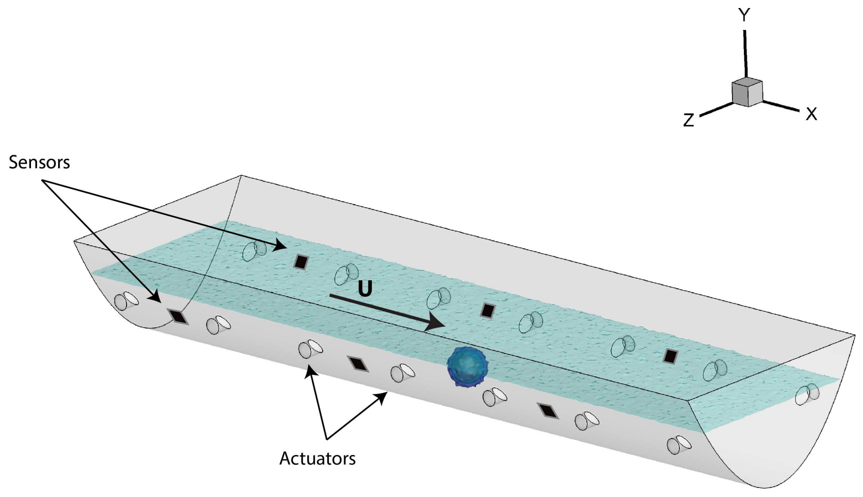


Figure 1.1: Schematic of a sensor-actuator system in an open channel flow.

.

the domain.

In the event of contaminant detection, a series of steps is initiated that aims to reduce the concentration of the contaminant by means of blowing and/or suction of the actuators. The process can be thought of as consisting of two phases. In the first phase, a *source inversion* is carried out to reconstruct the history of the contaminant release given the data obtained from the sensors. This step is necessary in order to provide the initial conditions, such as the time, location and source strength of the contaminant release, as input for the second phase, the *boundary control* model. In the boundary control step, the initial conditions from the source inversion are used to run a forward model which predicts the trajectory of the plume and solves an optimization problem in order to determine the optimal actuator(s) to apply the control action(s) that will best achieve the goal of reducing the contaminant at specified points in the domain. The effectiveness of the control actions are accessed using the updated flow field data and the boundary control step repeated in a closed feedback loop.

As will be shown in this chapter, the underlying methodology for carrying out both the source inversion and boundary control phases of the automatic detection and control algorithm are similar, as they both can be framed in terms of *numerical* *optimization* problems. However, the optimization goals and necessary groundwork for the two problems are quite distinct, and thus for the subsequent thesis these two phases will be addressed separately and in turn.

## 1.2 Contaminant Source Inversion

An inverse problem is one in which a series of observational values are used to ascertain the parameters characterizing a given system. This is in contrast to the 'forward' problem, in which the parameters are known and used to predict the desired observed values of the system. Inverse problems are ubiquitous in the fields of science and engineering, with applications ranging from determining the origin of an earthquake to reconstructing radiographic images.

Contaminant source inversion is a specific type of inverse problem in which observational data, typically in the form of point measurements obtained by sensors in the domain, are used to reconstruct the conditions which led to the observations. This is a scenario that occurs frequently in the case of an accidental or deliberate chemical release, either into the groundwater, atmosphere or a body of surface water. In such a scenario, the motivation to reconstruct the original plume is two-fold: 1) knowing the original parameters of the release (time of release, source strength, location, etc.) allows for the prediction of its future fate and transport, thus creating the possibility to prevent further damage, and 2) determining the origin of the release may allow for the responsible party to be held accountable.

While the applications are numerous, source inversion problems and inverse problems in general are challenging to solve for two fundamental reasons: firstly, depending on the number of parameters and the scale of the problem, there may exist a very large parameter space in which to search for the solution, leading to significant computational challenges when using numerical methods to solve them. Secondly, and perhaps more prohibitively, is the lack of well-posedness that often accompanies inverse problems. The inherent ill-posedness of inverse problems has a number of causes. For example, there may exist several combinations of parameters that lead to results that agree with the observations, leading to the problem of non-uniqueness. Additionally, the data points that provide the observations may be sparse or contain uncertainties. The lack of a guarantee of well-posedness in regards to the source inversion problem is an issue that should be kept in mind, and is readdressed throughout several of the chapters in this thesis.

Quite a bit of variability exists in the source inversion problem formulation, as is illustrated in the following literature review. The majority of the research conducted in the area of contaminant source inversion has been in the field of groundwater contaminant transport. More recently, source inversion has been applied to the problems of indoor air quality in buildings and to large-scale atmospheric pollutant transport. While the source inversion literature review herein includes studies carried out in all environments (groundwater, air or water-based), depending on the methodology used, the differences and additional complications for fluid-based inversion problems vs. groundwater keep these approaches from being necessarily transferable from one environment to another. In many ways, the groundwater inversion problem is an easier one to model and solve. Contaminant transport through porous media for example is adequately modeled in one- or two-dimensions, does not require turbulence modeling, and occurs in a very slow moving stationary velocity field. Due to these properties and the accompanying low Reynold's number of groundwater flows, the governing equations for groundwater flow are typically based on Darcy's equation. By contrast, modeling of fluid flows comprised of air or water may occur at high Reynold's numbers and almost always is in the turbulent regime for natural environments. The governing equations are the Navier-Stokes equations, a set of nonlinear partial differential equations which require numerical methods to solve fully. Depending on the scenario, buoyancy effects may be necessary to consider, in addition to a transient, three-dimensional flow field. Frequently, simplifications are made to these equations that allow them to be solved with much simpler methods, such as the advection-diffusion equation and the Gaussian plume transport model, but the assumptions behind these simplified models limit their general applicability to a narrow range of contaminant release scenarios.

The basic underlying approaches taken to perform source inversion can be divided into three main categories (*Liu and Zhai* (2007)): optimization methods, probability-based methods, and backward methods. This review of contaminant source inversion methods divides the body of literature roughly into these categories for clarity. However, some methods do not fit neatly into a single category and are discussed separately in the 'Additional Approaches' subcategory (Section 1.6). A compilation of significant research in the area of source inversion along with their pertinent characteristics is included in Table 1.1.

## 1.3 Optimization Approaches

Optimization methods take a deterministic approach to the source inversion problem, in which the problem is formulated in terms of an *objective function* or *functional*. This expression quantifies the error between the observations and the output of a forward model which aims to replicate these observations using estimated input parameters. The optimization approach then requires a series of strategic guesses to be made for the input parameters, and the results from the forward model are used in conjunction with the objective function to provide a measure of agreement between the guessed solution and the actual solution. By iteratively adjusting the input parameters with the goal of minimizing the objective function, a set of parameters can be identified that adequately reproduces the conditions that led to the observed values. Mathematically, this optimization procedure is formulated in general terms as

Table 1.1: Selected papers on contaminant source inversion

| **Main Inversion Approach** | **Reference** | **Dimensions** | **Parameters** | **Intended domain** |
|---|---|---|---|---|
| **Optimization** | *Gorelick et al.* (1983) | 2-D | $\mathbf{x}_s$, $q_s$ | Groundwater |
| | *Skaggs and Kabala* (1994) | 1-D | $C_0(t)$ | Groundwater |
| | *Mahar and Datta* (1997) | 2-D | $\mathbf{x}_s$, $C_0(t)$ | Groundwater |
| | *Skaggs and Kabala* (1998) | 1-D | $C_0(t)$ | Groundwater |
| | *Liu and Ball* (1999) | 1-D | $C_0(t)$ | Groundwater |
| | *Alapati and Kabala* (2000) | 1-D | $C_0(t)$ | Groundwater |
| | *Mahar and Datta* (2000) | 2-D | $\mathbf{x}_s$, $C_0(t)$ | Groundwater |
| | *Aral et al.* (2001) | 2-D | $\mathbf{x}_s$, $C_0(t)$ | Groundwater |
| | *Mahar and Datta* (2001) | 2-D | $\mathbf{x}_s$, $C_0(t)$, GW parameters | Groundwater |
| | *Akcelik et al.* (2003) | 2-D | $\mathbf{x}_s$, $q_s$ | Air/water |
| | *Bagtzoglou and Baun* (2005) | 2-D | $\mathbf{x}_s$, $C_0(t)$ | Atmosphere |
| | *Boggs et al.* (2006) | 2-D | $\mathbf{x}_s$, $q_s$ | Wall-bounded domain |
| | *Mahinthakumar and Sayeed* (2006) | 3-D | $\mathbf{x}_s$, $C_0(t)$ | Groundwater |
| | *Singh and Datta* (2006) | 2-D | $\mathbf{x}_s$, $C_0(t)$ | Groundwater |
| | *Allen et al.* (2007) | 2-D | $\mathbf{x}_s$, $q_s$, $\theta$ | Atmosphere |
| | *Sun* (2007) | 2-D | $C_0(t)$ | Groundwater |
| | *Yeh et al.* (2007) | 3-D | $\mathbf{x}_s$, $q_s$, $t_s$ | Groundwater |
| | *Datta et al.* (2009) | 2-D | $\mathbf{x}_s$, $C_0(t)$, GW parameters | Groundwater |
| | *Ayvaz* (2010) | 2-D | $\mathbf{x}_s$, $C_0(t)$ | Groundwater |
| | *Addepalli et al.* (2011) | 3-D | $\mathbf{x}_s$, $q_s$, $U$, $\theta$ | Atmosphere |
| | *Jha and Datta* (2013) | 3-D | $C_0(t)$ | Groundwater |
| **Probability-based** | *Wagner* (1992) | 2-D | $\mathbf{x}_s$, $C_0(t)$, GW parameters | Groundwater |
| | *Woodbury and Ulrych* (1996) | 1-D | $C_0(t)$ | Groundwater |
| | *Snodgrass and Kitanidis* (1997) | 1-D | $C_0(t)$ | Groundwater |
| | *Woodbury et al.* (1998) | 3-D | $C_0(t)$ | Groundwater |
| | *Neupauer et al.* (2000) | 1-D | $C_0(t)$ | Groundwater |
| | *Michalak and Kitanidis* (2004) | 3-D | $C_0(t)$ | Groundwater |
| | *Singh et al.* (2004) | 2-D | $\mathbf{x}_s$, $q_s$ | Groundwater |
| | *Khemka et al.* (2006) | 3-D | $\mathbf{x}_s$, $t_s$ | Atmosphere |
| | *Singh and Datta* (2007) | 2-D | $\mathbf{x}_s$, $q_s$ | Groundwater |
| | *Chow et al.* (2008) | 3-D | $\mathbf{x}_s$, $q_s$ | Atmosphere |
| | *Liu and Zhai* (2008), *Liu and Zhai* (2009) | 3-D | $\mathbf{x}_s$ | Interior/Air |
| | *Butera et al.* (2013) | 2-D | $\mathbf{x}_s$, $C_0(t)$ | Groundwater |
| **Backward** | *Skaggs and Kabala* (1995) | 1-D | $C_0(t)$ | Groundwater |
| | *Atmadja and Bagtzoglou* (2000) | 1-D | $C(x,t)$ | Groundwater |
| | *Bagtzoglou and Atmadja* (2003) | 1-D | $C(x)$ | Groundwater |
| | *Zhang and Chen* (2007) | 3-D | $\mathbf{x}_s$, $q_s$ | Interior/Air |
| | *Neupauer and Wilson* (2001) | 2-D | $C_0(x)$, $\mathbf{x}_s$, $t_s$, | Groundwater |
| | *Neupauer and Wilson* (1999) | 1-D | $\mathbf{x}_s$, $C_0(t)$ | Groundwater |
| | *Neupauer and Lin* (2006) | 2-D | $C_0(t)$ or $t_s$ | Groundwater |

$\mathbf{x}_s$ = source location, $q_s$ = source strength, $t_s$ = release time, $U$ =ambient velocity, $\theta$ =wind direction, $C_0(t)$ = release history, $C(x)$ = spatial distribution.

$$\min_{\mathbf{f} \in R^n} J(\mathbf{f}) \quad \text{subject to} \quad \mathbf{f} \in \Omega, \tag{1.1}$$

where $J$ is the objective function and $\Omega$ is a subset of the vector space $R^n$. Additional constraints in the bounds of $\mathbf{f}$ and inequalities may be included depending on the problem. While $J$ is by definition a measure of the difference between the observed and modeled data, its specific formulation is subjective. For source inversion problems it is usually a variation on the least squares formula. The definition of the source inversion problem as an optimization problem is covered in depth in Chapter III.

Optimization methods have the downside of being computationally demanding, as they typically require the exploration of a large parameter space, and thus may require many iterations in order to converge. Further complications arise in ensuring that the optimization problem is well-posed and amenable to minimization, as discontinuities and/or multiple local minima in the functional topography can create significant impediments to the optimization routine. For example, gradient-based optimization requires that the functional be convex, continuous and differentiable in order to guarantee convergence to a global minimum. However the parameter space formed by the objective function frequently does not meet these requirements, as will be illustrated in this work. Another weakness of the optimization approach is that a separate analyses must be conducted in order to assess the performance of the model with respect to uncertainty in the data. These uncertainty analysis are typically carried out by running a series of trials with perturbed data for example or performing Monte Carlo simulations.

Despite their drawbacks, optimization source inversion methods are appealing for several reasons. For one, they allow for a great deal of flexibility in the formulation of the source inversion problem. For example, the objective function and its associated constraints can be used to incorporate known information about the problem and to accommodate an arbitrary number of unknown parameters. The constraints can be used to narrow the parameter space and exclude values that are not expected. Further, as will be shown in this work, once the objective function and parameter space are defined, the optimization problem can be solved in a number of ways by implementing different optimization algorithms. Given the challenges in solving source inversion problems, having a greater number of tools available to broach a problem is highly desirable.

One of the earliest applications of the optimization technique to source inversion was carried out by Gorelick et al. (1983) , whereby the researchers recovered the source location and strength of up to two sources in a two-dimensional aquifer using a linear programming model based on a normalized least-squares objective function. The required inputs of their model were contaminant concentrations at several discrete points and up to 9 'potential' source locations.

Skaggs and Kabala (1994; 1998) also took a forward optimization approach and minimized an objective function based on Tikhonov regularization to recover the initial source strength and flux history of a single groundwater contaminant plume governed by the one-dimensional advection-diffusion equation. Liu and Ball (1999) also used this approach to investigate a known contaminant release at Dover Air Field Base in Delaware. Alapati and Kabala (2000) used the one-dimensional groundwater transport equation to solve for the source release history of a contaminant plume using a nonlinear least squares method as an alternative to the Tikhonov approach for a single known source location. Note that these approaches assume the source location is known.

The nonlinear least squares approach was also taken by Mahar and Datta (1997; 2000) to recover the locations and source release histories corresponding to up to 2 out of 3 potential sources locations, and later expanded their model to simultaneously solve for several aquifer parameters (*Mahar and Datta* (2001)). Datta et al. (2009) also incorporated the inversion of unknown aquifer parameters (porosity, hydraulic conductivities, dispersion coefficients) in addition to finding the source locations of up to 9 potential two-dimensional groundwater plumes. Sun (2007) used a constrained robust least squares approach to recover the source release history of a two-dimensional groundwater plume incorporating information on a number of potential source locations.

More recently, optimization approaches have been carried out using stochastic optimization methods such as genetic algorithms (GA). Aral et al. (2001) used a modified approach they call a progressive genetic algorithm to determine the two-dimensional coordinates and release history of a single groundwater plume given known aquifer parameters. Unlike the previous studies which considered the location optimization as a discrete problem with few possible solutions, they defined the possible source location in terms of 'regions' instead of specific points. They found that the starting parameter guesses for their model had little impact on the overall results.

Singh and Datta (2006) used a GA optimization approach to locate up to several unknown sources and flux histories in a two-dimensional aquifer. They considered scenarios of up to two actual sources out of multiple possible sources. They found that the errors increased significantly with the addition of more than one source. An additional drawback to the GA method is the lengthy run-time these algorithms require. For their more simple simulations the minimum run-time was 245 minutes; clearly far from 'real-time' modeling capabilities.

Long et al. (2006) and Allen et al. (2007) used a GA approach for the source inversion problem. In contrast to the previously described groundwater contaminant research, their work was intended for source inversion of a single atmospheric plume. Also notable is that their models, unlike the previously mentioned works in which source location was a recovered parameter, did not assume previously known potential source locations.

Mahinthakumar and Sayeed (2006) took a hybrid optimization approach in which they used a GA optimization method in sequence with a local search optimization for recovering the locations or release histories of up to several multiple groundwater sources in a three-dimensional domain. They found that recovering the source release histories from known source locations was an easier problem to solve than finding the unknown locations.

Addepalli et al. (2011) took a hybrid global/local optimization approach to the atmospheric source inversion problem. They cited the need for beginning the local, gradient-based optimization at a starting point located in the region of the functional that meets the convergence criteria of being approximately convex, continuous and differentiable. Thus they used a global GA optimization routine to locate an acceptable starting point before beginning the gradient-based optimization. The recovered parameters using this hybrid approach in conjunction with a three-dimensional Gaussian plume model included the source location, strength, ambient wind speed and direction. This approach also did not require initial potential source locations to be stipulated.

Another relatively recent optimization approach that has been taken is based on the method of simulated annealing (SA). Yeh et al. (2007) were able to obtain good results using a SA approach in conjunction with a tabu search method. They tried several starting locations with up to 7 monitoring points in a three-dimensional groundwater domain to recover the initial time of release and strength of a single contaminant plume. They used a relatively coarse (40 m) grid size, and assumed the source had a constant release rate. Their model requires the delineation of a 'suspected' source area, and they investigated several scenarios varying the size of this area.

Jha and Datta (2011, 2013) used a variation on the SA approach to find the unknown source flux magnitude, duration and timing of a groundwater contamination plume in two- and three-dimensions. They compared their 'adaptive simulated annealing' method to the genetic algorithm approach and found their method to be more efficient in recovering the source histories. However, in general SA methods, like genetic algorithm methods, tend to be computational intensive.

Ayvaz (2010) used an optimization approach called a heuristic harmony search algorithm to recover a two-dimensional groundwater contaminant plume. These results were compared to the results from two different approaches employed by Singh and Datta (2004, 2006) and deemed to be in good agreement.

## 1.4 Probability-based Methods

In contrast to the above described deterministic optimization methods, probability-based methods take a stochastic approach to solving the inverse problem. The outcome of these methods is typically in terms of probabilities of a plume having originated at a given location, time and given strength or the probability that the plume will be in a given location at a certain time.

Wagner (1992) used a maximum likelihood model to inversely locate the two-dimensional coordinates and release history of sources in addition to aquifer parameters in a two-dimensional domain, given two potential source locations. Woodbury and Ulrych (1996) used a probability-based approach known as Minimum Relative Entropy (MRE) inversion to reproduce the one-dimensional source release history results previously found by Skaggs and Kabala. They later expanded this method to include three dimensions and up to several possible source locations (*Woodbury et al.* (1998)). The results in one-dimensional applications were excellent given noise-free data, while the later three-dimensional simulations resulted in a large degree of variability in the possible source locations. Neupauer et al. (2000) compared the MRE approach to the Tikhonov regularization method for recovering the source release history of one-dimensional groundwater plumes and found that with error-free data the results were similarly accurate.

A probability-based application to atmospheric source inversion was developed by Chow et al. (2008) in which they made use of Bayesian inference methods that rely on Marcov Chain Monte Carlo sampling. Bayesian inference involves framing the inverse problem as the solution to a sampling of an ensemble of predicted simulations, by which a statistical comparison between the predictions and observations are used to guide the solution towards convergence. This work is particularly notable in that it represents one of the few studies that have used a full computational fluid dynamics (CFD) based code as the forward model. Their CFD model was a finite element code based on the Reynolds Averaged Navier-Stokes equations (RANS).

Artificial Neural Networks (ANN) are a probability-based method that involves 'training' a model to recognize patterns in data and respond appropriately. Singh et al. (2004) and Singh and Datta (2007) used this approach to find the initial source strength and location of sources in a two-dimensional aquifer provided known potential source locations.

Geostatistical methods are another probability-based inversion method that define the source properties in terms of random functions that can be characterized by their statistical properties. Snodgrass and Kitanidis (1997) used geostatistical inversion to recover the release history of a single one-dimensional groundwater plume given potential source locations. Michalak and Kitanidis (2004) also used a geostatistical approach in conjunction with an adjoint version of the advection-diffusion equation (ADE) to recover three-dimensional concentration histories of groundwater plumes, validated with real data. More recently, Butera et al. (2013) used the geostatistical method to recover a single plume location and source history given observational data and delineations of 'suspect areas' for a two-dimensional aquifer.

## 1.5 Backward Methods

Backwards methods aim to solve the representative equations in the reverse-time direction, starting with the final state. A notable example of this approach is given by Skaggs and Kabala whereas they defined a quasi-reversible (QR) function to represent the advection-diffusion of the contaminant. Another approach was taken by Atmadja and Bagtzoglou (*Atmadja and Bagtzoglou* (2000); *Atmadja and Bagtzoglou* (2001a); *Bagtzoglou and Atmadja* (2003)) using a method they developed based on the backward beam equation (BBE), which aims to solve a parabolic equation backwards with respect to time. Zhang and Chen (2007) used the QR approach with a full RANS-based CFD model to recover the source strength and location in a three-dimensional interior air quality model for several release points. They found that it was not possible to accurately recover the source strengths, due to the dispersive nature of thew QR equation.

Backward methods have the advantage of requiring only a single run and thus are more efficient than the iterative optimization methods. However strict backwards approaches have several significant limitations. Quasi-reversibility methods in particular have the downside that they only incorporate backwards advection, not diffusion, and thus tend to become very dispersive over time. Further, as backwards methods are generally only used to reconstruct the source flux history given a source location and the final state of the system, they are of limited relevance to the source inversion problem posed in this work.

## 1.6 Additional Approaches

Many previously studied source inversion methods are combinations of other methods and cannot be easily categorized. Most frequently, these methods combine aspects of one approach with an optimization scheme. Bagtzoglou and Baun (2005) for example applied their BBE technique in conjunction with a quasi-Newton gradient-based optimization technique to recover the location and time of release of a two-dimensional atmospheric plume given measurements from either a single moving sensor or several fixed location sensors.

Adjoint methods are a backwards type of inversion approach that may use optimization and probability techniques. In the adjoint approach, the forward equation is replaced with an *adjoint* equation which is the backwards-in-time equation of the forward-in-time model. Neupauer and Wilson (1999, 2001) and Neupauer and Lin (2006) combined an adjoint, probabilistic and backward approach to determine the predicted travel times and source location probabilities in several scenarios for one- and two-dimensional groundwater flows. Their approach involved developing probability density functions with the variables representing the random location or release time of an instantaneous point source based on one or more sampling locations or times. Cheng and Jia (2010) used this approach to calculate backwards probabilities related to flows in rivers. They used the shallow water equations for their forward model.

Akcelik et al. (2003) used a Newton optimization technique in conjunction with an adjoint version of the advection-diffusion equation to determine the location and release strength of a source in a generic flow system. Though they used a local search approach they did not investigate the effects of the initial guess on the outcome of the inversion. Under some conditions, adjoint methods can be an efficient method of formulating optimization problems, and are further discussed in Chapter IX.

Liu and Zhai (2009) applied a probabilistic-adjoint approach similar to Neupauer and Wilson towards indoor air quality modeling, in which they recovered the source locations of indoor contaminants given a known release time. Their computational model was based on a multi-zone approach, in which the various ambient fluid properties (temperature, velocity, pressure) are lumped into uniform values that are representative of zones, such as rooms in a building, for example. A similar study was carried out by Liu and Zhai (2008) whereby they utilized a full CFD model based on the Reynolds Averaged Navier-Stokes equations (RANS) as their forward model. They applied this model to two- and three-dimensional domains including an indoor building environment and an airplane cabin.

## 1.7 Source Inversion Discussion

As evidenced by the above literature review, the last few decades have resulted in many studies on the subject of contaminant source inversion. Some general conclusions and observations regarding these studies can be made.

Firstly, as previously noted, the majority of these studies have been carried out in the groundwater realm. As discussed above, there are fundamental differences in the applicable assumptions that can be made in a groundwater flow field in contrast to an air- or water-based one. Among the comparatively few air-based studies, several rely on a simplified Gaussian Plume Model (GPM) approach (*Khemka et al.* (2006), *Allen et al.* (2007), *Addepalli et al.* (2011)). Gaussian Plume Models are relatively simple to implement, however they cannot provide detailed information on the flow or incorporate turbulence modeling. Also, realistically, the offending source is unlikely to be a continuous plume as the GPM assumes, but rather an instantaneous release. By contrast, Computational Fluid Dynamics (CFD) models, in which the flow field is represented by the full (usually Reynolds Averaged) Navier-Stokes equations, are able to represent the source term in a variety of forms to simulate continuous release of constant or variable magnitude in addition to instantaneous source releases. Further, they are able to provide detailed flow information including the full pressure, velocity spatial distribution of the flow field. They are also able to model complex domains and in general offer a higher degree of flexibility and accuracy. On the downside, CFD methods tend to be computationally intensive and can be complicated to integrate with the specific inversion method employed. For these reasons, very few of the previous cited works (*Chow et al.* (2008), *Liu and Zhai* (2008), *Zhang and Chen* (2007)) have attempted the CFD approach to the source inversion problem. However, advances in computing power and the use of parallel methods are providing avenues to reduce the run-time of the inversion process and thus the added detail and capabilities they afford are very appealing for the source inversion problem.

Secondly, the parameters inversely solved for are one of the most important factors in comparing the previous works on contaminant source inversion. Namely, recovering the source location in addition to the strength and time of release is a very difficult task from both a computational and theoretical standpoint, especially in a three-dimensional flow field. Many of the above reviewed source inversion problems have avoided the issues of ill-posedness and intractable computational times associated with recovering the full set of characterizing source inversion parameters by stipulating a discrete number of potential source points, or delineating a relatively small region of the domain as the possible source area when inversely solving for the source location. Many other works do not attempt to recover the source location at all, instead recovering the release history and/or strength assuming that the source location is known. In some scenarios this is a valid assumption, however for many other problems, including the scenario described in this work, this information is necessary and cannot be assumed to be known a priori.

Thirdly, and with respect to the optimization approaches specifically, one of the most significant limitations of much of the above reviewed literature is in regards to the lack of a thorough analysis on the robustness of the source inversion model. While some researchers (*Aral et al.* (2001), *Yeh et al.* (2007)) have examined results from several starting optimization points, many others do not explicitly address the initial guesses for the optimization routines. As will be shown in this work, it is possible to obtain excellent inversion results for a specific starting point and have the optimization completely fail from another, and hence from a practical, applicability point of view, this is important information necessary to fully characterize the source inversion robustness and reliability. Further, applying an inversion technique to a few specific problems does little to illustrate its capabilities in terms of robustness and transferability.

Fourthly, in most of these studies, the run-time is not addressed as expediency is not a main concern of the inversion model. Few studies (*Boggs et al.* (2006), *Liu and Zhai* (2008)) have explicitly stressed the applicability of their source inversion methodology towards a real-time or near-real-time scenario. Additionally, global optimization methods that have been shown to have potential for locating the release point of contaminant sources such as GA and SA have also been shown to require lengthy convergence times, and thus have severe limitations to their applicability in a scenario where mitigation and environmental protection is sought.

Finally, another limitation of many of the above described past works in this area is in regards to the issue of ill-posedness. As mentioned above, ill-posedness most frequently arises from the problem of non-uniqueness, from errors in the data and from data sparseness. Some researchers have explicitly addressed the ill-posedness issue (*Atmadja and Bagtzoglou* (2001b), *Alapati and Kabala* (2000), *Woodbury and Ulrych* (1996)), while others have largely ignored the issue altogether. This is perhaps justifiable in a scenario where there are few potential source locations and thus fewer possible combinations of unknown parameters. In a more general scenario however, ill-posedness cannot be ignored, as it places limitations on the tractability of the source inversion problem.

Regularization, which aims re-cast an ill-posed problem as an approximate, well-posed problem, has its limitations as Skaggs and Kabala discussed in their 1998 paper. They discussed the inability of regularization to guarantee an accurate solution in many cases, especially with increased diffusion of the original source. Also, while regularization mitigates the problems of numerical instability and errors in the data, non-uniqueness cannot be eliminated via its implementation.

### 1.7.1 Present Research Contributions

The present work takes a gradient-based optimization, CFD approach to the source inversion problem. The CFD model employed is a RANS-based finite volume model. This model is interfaced with an external optimization toolkit which provides a range of optimization algorithms, several of which are compared and contrasted as a part of this work.

This work also aims to shed light on the general capabilities and limitations of the optimization-based CFD source inversion model and to clearly illustrate the challenges pertaining to this approach in regards to the discretization of the problem and issues of ill-posedness. An in-depth analysis of the effectiveness of the source inversion algorithm as a function of the various parameters of the problem is undertaken, and the robustness and accuracy of the source inversion model as a function of the starting guess given to the optimizer is explicitly addressed.

As with the research summarized in Table 1.1, in order to make this very complex problem tractable, several underlying assumptions must be made. These assumptions are outlined below. Further description of the contaminant source inversion problem will be discussed in detail in Chapter III.

- It is assumed there is only a single source.
- The ambient flow velocity properties and the diffusion properties are assumed to be known.
- The flow field is assumed to be constant and uni-directional.
- The source is assumed to be non-reactive, conservative and neutrally buoyant with respect to the ambient fluid and to be governed by the advection-diffusion

equation.

- The sensors and observations are assumed to be perfect. While a sensitivity analysis is a key component to assessing the realistic performance of a source inversion model, this step is beyond the scope of the current work, which instead focuses on the underlying theory and mechanisms.

## 1.8 Boundary Control for Fluid Systems

A great deal of research has been undertaken on the subject of boundary control of fluid systems over the past few decades, as there is significant motivation to develop robust methods of controlling the evolution and behavior of turbulent properties in a fluid system. In industrial applications for example, increased turbulence equates to increased mixing, which may be desirable in a manufacturing application. Conversely, reducing drag is beneficial from an efficiency standpoint in many other applications. The body of literature on the topic of fluid control is vast and much of it is related to the goal of controlling turbulence, drag, or boundary layer separation. As these topics and the associated methodology in carrying out the flow control are not all directly applicable to the problem of controlling a hazardous plume, this work does not purport to include an exhaustive literature review of boundary control in general. Rather, the basic methods underlying boundary control of fluids are outlined below along with relevant examples of their applications.

Methods of boundary fluid control fall into the categories of either passive or active methods. Passive methods are suited for steady-state systems, and typically include altering the properties of the surfaces or installing vortex generators along the boundaries in order to affect the size of the turbulent eddies that evolve in the flow, and hence to reduce drag or boundary layer separation. Passive methods are unable to respond to changes in the flow and as a result are of limited value in dynamic systems.

The vast majority of research done in regards to boundary control has been carried out in reference in active control methods, and specifically using an *active feedback control* approach. Active feedback flow control is an interdisciplinary field incorporating knowledge of systems control theory, fluid dynamics and numerical modeling. There are several approaches that can be taken in achieving active feedback control of fluid systems. These methods will be briefly outline here for thoroughness, although as will become clear, they are not all suitable for the contaminant plume boundary control problem. Following Moin and Bewley (1994) approaches fall into one of four categories based on the relationship of the control scheme to the governing equations: adaptive, intuitive-based, dynamical-systems based schemes and optimal control.

### 1.8.1 Adaptive Methods

Adaptive methods take an empirical approach to flow control in which the control algorithm is trained through iterative methods to produce the desired output based on the measurements. A significant advantage to this approach is that no prior-based knowledge of the flow field is required. One of the more sophisticated examples of an adaptive scheme is the Artificial Neural Network (ANN) method. This approach involves expressing the relationship between the actuator input and the sensor output as a nonlinear function with variable coefficients. The coefficients are 'trained' and then updated using current data, similar to the ANN methodology used in the source inversion problem cited above. Artificial neural networks have been used to reduce drag up to 20% channel flow at low Reynold's numbers (*Babcock et al.* (1996); *Lee et al.* (1997)). Adaptive schemes however are limited in their applicability to complex flow regimes.

### 1.8.2 Intuitive-based Methods

Intuitive-based methods rely on a solid understanding of the underlying physics of the flow allowing for actuator controls which directly mitigate the turbulence such as through active cancellation or opposition schemes. They have been used to achieve skin friction reduction (*Choi et al.* (1994)), drag reduction (*Hammond et al.* (1998)), and control of near-wall turbulence (*Rebbeck and Choi* (2001)). The main drawback to intuitive-based schemes is the need to have an adequate understanding of the flow dynamics a priori, which is often not possible. Like adaptive methods, this approach has limited validity to complex, dynamic flow regimes.

### 1.8.3 Dynamical-systems Approaches

Dynamical-systems based schemes extrapolate from linear systems theory and aim to decompose the turbulence into a finite number of representative modes. This process maps the turbulent dynamics to a lower dimensional phase state, hence these methods can also be referred to as reduced order models. The goal of the control problem is stabilizing the attractors of the chaotic system. Two well studied methods in this category are the OGY method, named for its inventors Ott, Grebogi and Yorke (*Ott et al.* (1990)) and Proper Orthogonal Decomposition, or POD (*Berkooz et al.* (1993); *Berggren* (1998); *Ravindran* (2000); *Bergmann and Cordier* (2008); *Semeraro et al.* (2011)).

### 1.8.4 Optimal Control

Flow control based on optimal control theory aims to minimize an objective function as applied to the Navier-Stokes equations. This is the same basic approach as is taken in solving the source inversion problem via optimization, where the objective function in this case describes the physical properties which are desired to be controlled. The goal of the optimization problem is again to find the parameters which result in the minimization of the objective function as subject to the constraints defined by the problem over a specified control horizon. Further details on the optimal control method as applied to boundary control are given in Chapter VII.

The main advantage of optimal control over the aforementioned methods is that optimal control is applied directly to the underlying (Navier-Stokes) equations and hence is the most theoretically correct. Also, the lack of assumptions in the derivation translates to a lack of restrictions over the applications of optimal control theory. The functional can be defined as drag, turbulent kinetic energy, or any other property of the flow. However, due to the intense computations and data resolution required, optimal control theory is also the most computationally intensive of the active feedback control methods. To make optimal control feasible for practical applications, some attempts have resorted to 'suboptimal' methods, in which the control horizon is reduced to a very short period. Choi et al. (1993) first successfully applied a suboptimal control method to the Burgers equation. Bewley and Moin (1994) applied suboptimal control to turbulent flows and were able to reduce drag by around 17% . Lee et al. (1998) were able to apply a suboptimal procedure requiring only pressure or shear-stress data at the wall and were able to reduce drag up to 22% by choosing a cost functional directly related to the streamwise vorticity. It is clear however, that the effectiveness of the controller will be reduced as the control horizon over which the functional is optimized is shortened. Bewley et al. (2001) demonstrated this by comparing the results of their three-dimensional DNS experiments with the suboptimal method results and concluding that suboptimal methods were severely limited in their effectiveness over the benchmark DNS results.

Variations on the optimal control approach that use simplified versions of the Navier-Stokes equations for larger-scale applications include the works by Sanders and Katopodes (1999, 2000). They applied gradient-based optimal control theory and the adjoint sensitivity method toward adaptive flow control in channels, using a model based on the two-dimensional, depth-averaged shallow water equations. Piasecki and Sanders (2002) used a similar approach with a one-dimensional model in which the goal was to prevent salt-water intrusion above a specified threshold in an estuary environment.

### 1.8.5 Contaminant Control

The body of literature in regards to the specific problem of contaminant control in fluid systems is extremely limited. Notable works include the studies by Piasecki and Katopodes (1997a, 1997b). Their work focused on the real-time control of hazardous releases in rivers and estuaries. More recently, Alvarez-Vàzquez et al. (2009) used optimal control theory to demonstrate the potential to mitigate a pollutant in a section of a river by injecting clean water at an optimal rate. However to this author's knowledge, the present work is unique in its application of an optimization routine with a CFD model to apply optimal control to the Navier-Stokes equations in a wall-bounded flow system. As is the case for the source inversion problem, the model is based on the RANS equations and utilizes a two-equation k-$\epsilon$ eddy viscosity model.

The goal of the optimization problem for the boundary control phase of the automatic detection and control algorithm is to reduce the strength of the contaminant at specified points or entire regions of the domain as much as possible, based on the constraints of the optimization problem. These constraints include the underlying equations describing the ambient flow and the fate and transport of the contaminant, in addition to the constraints on the control parameters themselves. In this case, the control parameters take the form of the velocities at the boundary ports, and therefore they have bounds defined by the physical capabilities of the ports. The methodology of the boundary control is explained in more detail in Chapter VII.

## 1.9 Summary

In summary, this thesis provides the basis for an automatic detection and control system, by which a contaminant plume release is detected and mitigated using an optimization approach. The model is based on the Reynolds Averaged Navier-Stokes equations, but has the capability to be amended to employ a turbulence model of greater accuracy such as Large Eddy Simulation. The main contributions of this work are summarized as follows:

- A CFD, RANS-based source inversion model is developed using the scalar transport equation for modeling the contaminant plume. The CFD-optimization model is composed of two distinct software packages: OpenFOAM and the DAKOTA Optimization Toolkit.

- An analysis on the effect of the flow parameters, sensor locations and starting point is carried out and results are compared for the source inversion problem in terms of 2-4 unknown parameters.

- Several optimization approaches are investigated, including a conjugate-gradient solver, a quasi-Newton solver, a hybrid approach incorporating a global-based algorithm in conjunction with a gradient-based solver, and a model predictive control approach that repetitively corrects the optimization trajectory based on current information. The results of the different approaches are qualitatively and quantitatively compared.

- A CFD, RANS-based model is developed for the optimization of contaminant control via boundary control.

- The effect of the bounds on the parameters is explored and quantified.

- The effect of obstacles in the flow on the boundary control algorithm is investigated.

- Conclusions are drawn and recommendations are made for the boundary control problem.

### 1.9.1 Thesis Organization

The following chapter describes the theory and governing equations related to the computation fluid dynamics modeling used in this work. The turbulence model, CFD software and basic meshing principles are also covered. Chapter III presents the source inversion theory and its representation as an optimization problem. Optimization approaches are discussed and the theory behind several gradient-based optimization methods is presented. Although this optimization problem is presented in regards to the source inversion problem, the optimization theory introduced in this section also applies to the boundary control problem discussed in Chapter VII. The numerical issues that are encountered due to the problem discretization are introduced in this chapter. The following chapter, Chapter IV shows the optimization results of two gradient-based optimization algorithms tested on two analytical test functions, and the discretization problems are further illustrated with these examples.

Chapter V extends the optimization algorithm to the source inversion problem with respect to two unknown spatial parameters. Several variations of the optimization approaches are tested with scenarios in which the flow properties and sensors differ in order to compare the relative capabilities and robustness of the different approaches. The ensuing chapter, Chapter VI, extends the source inversion algorithm to problems with three or more unknown parameters.

Chapter VII deals with the second phase of the detection and control algorithm, i.e. boundary control. Results of taking an optimization approach to this part of the algorithm are shown and the effect of the observation and control horizon are presented. Chapter IX gives background on a possible method for increasing the efficiency of the detection and control algorithm: adjoint equations. Finally, Chapter X summarizes the above and discusses the conclusions drawn from this work.

# CHAPTER II

# Governing Equations and Numerical Methodology

This chapter defines the governing equations for the fluid dynamics model. Methods of numerical modeling are briefly discussed and the discretization practices implemented in this work are developed. The CFD model employed in this work is described and basic validation is provided.

## 2.1 Governing Equations

Viscous fluid flow is governed by a system of partial differential equations known as the Navier-Stokes equations. This section will define these equations and the meaning of each term. First, it is stated that all simulated fluid flows in this work are assumed to be incompressible and isothermal. Incompressibility, which is defined as a flow in which the density remains constant in a parcel of fluid, is a valid assumption in flows with a very low Mach number, $M = U/a$, where $U$ is the characteristic velocity and $a$ is the speed of sound in the fluid. At the velocities simulated herein the Mach numbers are always well below the compressible/incompressible threshold of $\approx 0.25$ and therefore the incompressible assumption holds regardless of the ambient fluid being modeled. Hence the governing equations for this fluid flow are the incompressible Navier-Stokes equations, which are a set of equations describing conservation of mass and linear momentum of a Newtonian fluid, respectively:

$$\nabla \cdot \mathbf{U} = 0, \tag{2.1}$$

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{U}) = \mathbf{g} - \nabla p + \nabla \cdot (\nu \nabla \mathbf{U}). \tag{2.2}$$

where $p$ is the pressure, $\nu$ is the kinematic viscosity, $\mathbf{g}$ is the gravitational constant and $\mathbf{U}$ is the velocity vector $[U_x\ U_y\ U_z]$. The terms of Eqn. (2.2) from right to left represent the rate of change, inertia, gravitational body force, pressure gradient and dissipation terms.

An additional equation is needed for the contaminant transport. As stated in Section 1.2, it is assumed that the contaminant is non-reactive and neutrally buoyant. The contaminant transport is then modeled with the scalar transport equation:

$$\frac{\partial C}{\partial t} + \nabla \cdot (\mathbf{U}C) - \nabla^2(\Gamma_e C) - f = 0 \tag{2.3}$$

where $C$ is the transported scalar and $\Gamma_e$ represents the effective diffusivity of $C$. The term $f$ represents a source which in this work takes the form of a Delta-Dirac function in both time and space for the case of an instantaneous release

$$f = q\delta(\mathbf{x} - \mathbf{x_s})\delta(t - t_s) \tag{2.4}$$

or a Delta-Dirac function in space only for a continuous point release

$$f = \dot{q}\delta(\mathbf{x} - \mathbf{x_s}) \tag{2.5}$$

where $\mathbf{x} = [\mathrm{x\ y\ z}]^T$, $\mathbf{x_s}$ represents the coordinates of the release, $t_s$ is the time of the release, $\dot{q}$ is the mass loading rate and $q$ is a mass load.

### 2.1.1 Turbulence Modeling

The word turbulent comes from the Latin turbulentus ("disorder, tumult") and describes flow which is characterized by apparently erratic motions and instability. Turbulent flows are inherently three-dimensional and time-dependent, as the vorticity generating mechanism necessary to sustain a turbulent flow cannot be achieved otherwise. Turbulence remains one of the least understood natural phenomena in fluid mechanics. One of the greatest contributions to this area of research was made by Komolgorov (1941) who accurately described turbulent flows as being comprised of vortices that encompass a wide range of scales, from the largest (integral) length scale to the smallest (Kolmogorov) length scale. He defined the concept of the energy cascade, whereby energy from the larger eddies is transferred to the smaller eddies and eventually dissipated through viscosity at the smallest scale. Kolmogorov's work did much to advance the scientific understanding of turbulence and to further the field of turbulence modeling. Chief to the development of energy cascade theory and to the characterization of turbulent flows in general is the concept of the Reynolds number ($Re$), a non-dimensional number used to give a measure of the ratio of a flow's inertial to viscous forces:

$$Re = \frac{UL}{\nu} \tag{2.6}$$

where $U$ is the characteristic flow velocity and $L$ is the characteristic length scale. This number can be used to classify a specific type of flow as falling within the turbulent, transition or laminar regime. At very low Reynolds numbers, viscous forces dominate and serve to restore equilibrium to the flow, damping the instabilities that would otherwise lead to turbulent eddies and resulting in smooth, sheetlike motion termed *laminar* flow. The higher the Reynolds number, the more likely the flow is to develop turbulent characteristics, and above a certain threshold known as the *critical Re*

($Re_{cr}$), the flow can be assumed to be in the turbulent regime. Estimates for $Re_{cr}$ have been theoretically and/or experimentally calculated for several flow types, a sampling of which are shown in Table 2.1.

Table 2.1: Approximate critical Reynolds numbers for various flows

| Flow geometry | $Re_{cr}$ |
|---|---|
| pipe | 2300 |
| smooth sphere | $3\text{x}10^5$ |
| golf ball | $4\text{x}10^4$ |
| free atmosphere | $3.85\text{x}10^5$ |
| open channel | 600 |

Most natural fluid flows occur in the turbulent regime, and as such, it is necessary to adequately represent the turbulent effects for accurate fluid flow simulations. Turbulent motions are inherently included in the full Navier-Stokes equations, and thus can be resolved directly by solving Eqns. (2.1) and (2.2) in their entirety. This approach, termed Direct Numerical Simulation (DNS) is the gold standard for turbulent CFD simulations but comes at a high computational cost. Great care must be taken to create a mesh that is finely enough discretized to fully capture the entire range of turbulent eddies in the flow and the large number of computational points leads to long simulation times and large computer requirements. Due to these restrictions, DNS simulations are prohibitive for the majority of applications except those that occur on a very small scale. Fortunately, detailed turbulence information is not required for most CFD applications, and approximate models can be employed that greatly reduce the computational demands of DNS. There are several proposed turbulence models from which to choose, depending on the required level of accuracy and available computational resources. These models include, in order of increasing accuracy, Reynolds-averaged Navier-Stokes (RANS) models, in which the governing equations are ensemble-averaged; large eddy simulation (LES) models, in which the large eddies are resolved in full and smaller scale eddies are parameterized; and detached eddy simulation (DES) models, which employ a hybrid method that applies RANS modeling to the near-wall regions and resolves the interior flow with LES methods.

Higher accuracy approaches such as DES and LES are attractive for the increased level of detail they provide, however like DNS they require careful meshing and solving strategies in order to be accurately applied (*de Villiers* (2006)). For example, if boundary layers (discussed in Section 2.7.5) are not properly treated the higher level detail from these methods is lost. In light of these caveats and the fact that turbulence is not the focus of this work and a high level of individual eddy realization is not required, the comparatively less burdensome RANS modeling approach is used for the simulations in this work.

### 2.1.2 RANS Turbulence Modeling

RANS models, also known as eddy-viscosity models, parameterize the turbulence as time-averaged 'stresses' derived using Reynolds decomposition. The basis of this approach is to represent the bulk flow as the sum of an average component and a component that represents the perturbations about the mean value

$$\mathbf{U} = \overline{\mathbf{U}} + \mathbf{U}' \tag{2.7}$$

where the mean component $\overline{\mathbf{U}}$ is formally defined as

$$\overline{\mathbf{U}} = \lim_{T\to\infty} \frac{1}{T} \int_0^T \mathbf{U}(x,t)dt \tag{2.8}$$

This decomposed velocity can be substituted into Eqns. (2.1) and (2.2). Averaging these equations over time results in the unsteady Reynolds Averaged Navier-Stokes equations:

$$\nabla \cdot \overline{\mathbf{U}} = 0, \tag{2.9}$$

$$\frac{\partial \overline{\mathbf{U}}}{\partial t} + \nabla \cdot (\overline{\mathbf{U}}\overline{\mathbf{U}}) = \mathbf{g} - \nabla \overline{p} + \nabla \cdot (\nu \nabla \overline{\mathbf{U}}) + \overline{\mathbf{U}'\mathbf{U}'}. \tag{2.10}$$

This system requires an additional equation for closure, as the process of introducing the perturbed velocity component renders it under-determined. Specifically we must find a way to estimate the final term on the right of Eqn. (2.10), referred to as the Reynolds stress tensor. The accuracy and level of computational burden for RANS models varies depending on the specific method that is chosen to represent the *Reynolds stress tensor*. For linear eddy-viscosity models the methods include algebraic (0-equation), one-equation and two-equation closure models. Algebraic models simply use the Boussinesq approximation to calculate the Reynolds stress tensor. One-equation models use a single transport equation to characterize the turbulence. Typically this quantity is the turbulent kinetic energy, $k$, although other variables 'viscosity-like' variables may be defined such as in the well-known Spalart-Allmaras model.

Two-equation models such as the $k - \epsilon$ approach used herein are the most commonly used closure models. Usually one of the employed transport equations is $k$, while the other is usually a measure of the scale of the turbulence. In the case of the $k - \epsilon$ model the transport quantities are $k$ and the turbulent energy dissipation rate per unit mass, $\epsilon$. The turbulent kinetic energy is formally defined in Cartesian coordinates as

$$k = \frac{1}{2}\overline{\mathbf{U}' \cdot \mathbf{U}'} = \frac{1}{2}(U_x'^2 + U_y'^2 + U_z'^2), \tag{2.11}$$

where for CFD purposes it is conventional to estimate $k$ using the relation

$$k = \frac{3}{2}\left(\mathbf{U}_{ref}\mathbf{I}\right)^2, \tag{2.12}$$

where $\mathbf{U}_{ref}$ is a problem-defined reference velocity, typically taken to be the average velocity of the flow at the boundary.

The turbulent energy dissipation rate per unit mass is defined as

$$\epsilon = \frac{C_\mu^{3/4} k^{3/2}}{l}. \tag{2.13}$$

where $l$ is a quantity defined to be the *mixing length*, frequently taken to be $0.07 * L$ for inlet boundaries (where $L$ is the characteristic length of the inlet) and $C_\mu$ is a model constant. The initial turbulent intensity $\mathbf{I}$ is typically taken to 5% of the inlet velocity for fully developed turbulent flow. With the above definitions, the transport equations for $k$ and $\epsilon$ are fully defined for incompressible flow as

$$\frac{\partial k}{\partial t} + \overline{U}_j \frac{\partial k}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + \left[\nu_t\left(\frac{\partial \overline{U}_i}{\partial x_j} + \frac{\partial \overline{U}_j}{\partial x_i}\right) - \frac{2}{3}k\delta_{ij}\right]\frac{\partial \overline{U}_i}{\partial x_j} - \epsilon \tag{2.14}$$

$$\frac{\partial \epsilon}{\partial t} + \overline{U}_j \frac{\partial \epsilon}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\frac{\nu_t}{\sigma_\epsilon}\frac{\partial \epsilon}{\partial x_j}\right) + C_{\epsilon 1}\frac{\epsilon}{k}\nu_t\left(\frac{\partial \overline{U}_i}{\partial x_j} + \frac{\partial \overline{U}_j}{\partial x_i}\right)\frac{\partial \overline{U}_i}{\partial x_j} - C_{\epsilon 2}\frac{\epsilon^2}{k} \tag{2.15}$$

where the turbulent viscosity $\nu_t$ is derived from $k$ and $\epsilon$ as

$$\nu_t = C_\mu k^2/\epsilon \tag{2.16}$$

.

The $k - \epsilon$ constants used in this work are summarized in Table 2.2.

Table 2.2: Empirical constants used in the $k - \epsilon$ model

| $C_\mu$ | $C_{\epsilon 1}$ | $C_{\epsilon 2}$ | $\sigma_k$ | $\sigma_\epsilon$ |
|---|---|---|---|---|
| 0.09 | 1.44 | 1.92 | 1.0 | 1.3 |

## 2.2 Modeling Approaches

In order to numerically simulate fluid flow, the governing equations must be converted into a form that is amenable to being solved algebraically, a process referred to as discretization. There are several methods used to achieve this. The majority of methods fall into one of three categories based on how the spatial discretization is handled: finite difference (FD), finite element (FE) or finite volume (FV). The finite difference method is the oldest and most straight-forward of these. Essentially the domain is discretized by a regular distribution of grid points. Backwards, central or forward differences, derived by the Taylor expansion, are then used to calculate both the spatial and temporal derivatives. Finite difference methods are relatively easy to code but are ill-suited for domains that have complicated geometries with irregular boundaries. Further issues can arise when applying FD codes to realistic fluid simulations, in which the state variables are not necessarily conserved.

The Finite Element method was originally invented for structural analysis modeling and later extended for use in fluid dynamics. Finite element methods define piece-wise linear functions across each ‘element’ using basis functions. Spectral methods also fall within this category, as they employ basis functions of a different variety. While FE methods perform relatively well for coarse grids and have a great deal of flexibility, they are not well suited for modeling turbulence and also may lack of local conservation of the state variables.

The Finite Volume method discretizes the domain into polyhedral volumes defined by a cell center and a set of faces. These control volumes by definition fill the entire domain and do not overlap. The unknown variables are defined at the control volume’s center, which is the geometric centroid of the polyhedral. The cell faces are classified as internal if they are adjacent to another cell face, or external if they coincide with the domain boundary. The vector **S** is defined as to be normal to the cell face and to be equal in magnitude to the area of the face. A diagram of a typical control volume, $P$ and a neighboring cell, $E$, is shown in Figure 2.1. This notation will be employed throughout this chapter.

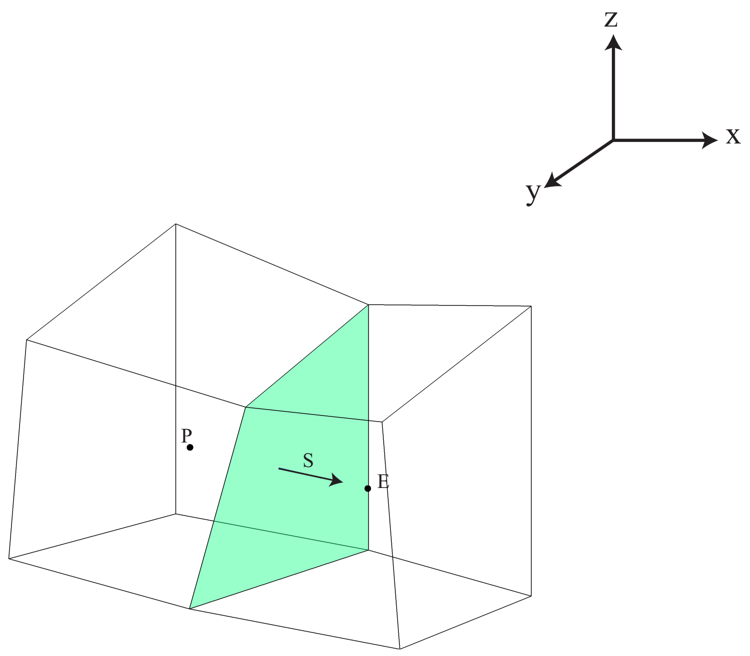


Figure 2.1: Example control volume.

One of the advantages to this discretization approach is that the control volumes may take any shape and be spaced irregularly, allowing for easy implementation in complex geometries. As will become apparent shortly, the method also has the benefit of being inherently conservative. The majority of computational fluid dynamics models are based on Finite Volume methods including the model used in this work, and therefore the remainder of this chapter will be dedicated to its discretization conventions and implementation.

## 2.3 Navier-Stokes Discretization

The theoretical basis for the Finite Volume method is Gauss' Divergence Theorem, which states that the rate of change of a property in a control volume is equivalent to the net flux of that properties across its boundaries. In its general form this is given mathematically as

$$\int_V \nabla \star \phi dV = \int d\mathbf{S} \star \phi \tag{2.17}$$

where $V$ is the control volume, $\phi$ is a general scalar field and the $\star$ symbol can be substituted with the inner, outer or cross tensor product and the divergence, gradient or curl associated with those products, respectively. Using this relationship between the volume and surface fluxes, the system of non-linear equations given above is transformed into a set of algebraic relations which is capable of being solved computationally. We shall examine this procedure for each of the terms in the governing equations. The process is facilitated by first presenting the equation for generic transport of a transported quantity $\phi$, which will be used to illustrate the discretization process in general terms

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\mathbf{U}\phi) - \nabla \cdot (\Gamma_\phi \nabla \phi) = S_\phi(\phi) \tag{2.18}$$

where $\Gamma_\phi$ represents diffusivity. The left hand side contains the transient acceleration, advection and diffusion terms. On the right hand side is the source term. The Finite Volume method requires that the integral form of Eqn. (2.18) be satisfied for each control volume around the cell centroid, $P$ :

$$\int_{t}^{t+\Delta t}\left[\frac{\partial}{\partial t}\int_{V_P}\phi dV+\int_{V_P}\nabla\cdot(\mathbf{U}\phi)dV-\int_{V_P}\nabla\cdot(\Gamma_\phi\nabla\phi)dV\right]dt=\int_{t}^{t+\Delta t}\left(\int_{V_P}S_\phi(\phi)dV\right)dt, \tag{2.19}$$

where $V_P$ is the control volume. Due to second derivative in the diffusion term this is a second order equation and thus the discretization in time and space must be at least second order. This requires the assumption that $\phi$ has a linear variation about the point $P$ and time $t$ , so that

$$\phi(\mathbf{x})=\phi_P+(\mathbf{x}-\mathbf{x_P})\cdot(\nabla_\phi)_P, \tag{2.20}$$

and

$$\phi(t+\Delta t)=\phi^t+\Delta t\left(\frac{\partial\phi}{\partial t}\right). \tag{2.21}$$

The incompressible Navier-Stokes equations in (2.1) and (2.2) are comprised of the continuity equation, which serves as a constraint, and the 3 momentum equations in the x- y- and z- component directions. There are two main difficulties in solving this system of equations: The first is that the inertial term in Eqn. (2.2) is nonlinear. The second is that absent the density variations of compressible flow, the equation of state cannot be used as an explicit equation for the pressure $p$. That renders a system of 4 unknowns, $u$, $v$, $w$ and $p$ , where $p$ only appears in the 3 momentum equations. In other words, we need $p$ to find the velocity components, and we need the velocity components in order to find $p$. This interdependence can be dealt with either by a simultaneous approach or through a sequential process by which the equations are solved in turn, then the fields iterated until all fields are deemed sufficiently accurate. As simultaneous approaches are typically too computationally expensive to use albeit for very small simulations, segregated methods are more frequently used. There are several options on how to carry how this velocity-pressure coupling. This work makes use of Issa's proposed PISO (Pressure Implicit with Splitting of Operators) algorithm, in which the pressure is coupled to the velocity by the principle of flux conservation *Issa* (1986). This process will be described in Section 2.3.6. First, we shall develop the discretization for the other terms which do not require special treatment.

### 2.3.1 Spatial Discretization

In the Finite Volume method, the first step to discretizing the governing equations is to use Eqn. (2.17) to transform each of the terms from a volume integral to a surface integral. Taking into account the variation of $\phi$ given by Eqn. (2.20) leads to

$$\int\limits_{V_P} \phi(x) dV = \phi_P V_P \tag{2.22}$$

Similarly, the discretization of the divergence operator can be represented as

$$\int\limits_{V_P} \nabla \cdot \phi dV = \int\limits_{\nabla V_P} d\mathbf{S} \cdot \phi = \sum_f \left( \int\limits_f d\mathbf{S} \cdot \phi \right) = \sum_f \mathbf{S} \cdot \phi_f \tag{2.23}$$

where the subscript $f$ indicates the centered face indices.

### 2.3.2 Advection Term

This term is discretized using Gauss' theorem as follows:

$$\int\limits_{V_P} = \nabla \cdot (\mathbf{U}\phi) dV = \sum_f \mathbf{S} \cdot (\mathbf{U}\phi)_f = \sum_f [\mathbf{S} \cdot \mathbf{U}_f]\, \phi_f = \sum_f F \phi_f \tag{2.24}$$

where $F$ is the mass flux through the face . The question remains of how to calculate the value of $\phi$ at the face to provide $\phi_f$. This is addressed using an advective differencing scheme. A commonly used choice is the upwind advection scheme. Upwind schemes choose values for $\phi_f$ that give more weighting to the cell-centered value of $\phi$

in the upwind direction, hence the name. The first-order upwind scheme, originally developed by Courant, Isaacson and Rees (1952), is schematically depicted in Fig. 2.2. In this formulation, $\phi_f$ is assigned to be the same as the value at the cell center at the upstream face:

$$\phi_f = \begin{cases} \phi_P & \text{if } F \geq 0 \\ \phi_E & \text{if } F < 0 \end{cases} \tag{2.25}$$

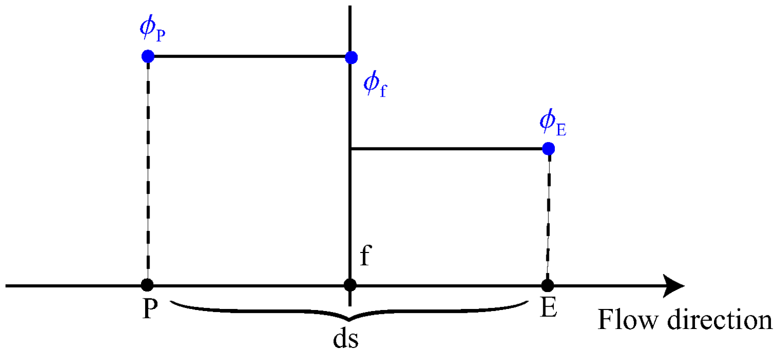


Figure 2.2: First-order upwind interpolation between two control volumes onto the shared face.

The first-order upwind scheme is relatively stable but is diffusive, in particular for multi-dimensional flows. An alternate approach is to interpolate the values of $\phi$ from the cell-centered values of the cells on either side of the face. This methodology is illustrated in Fig. 2.3. The interpolation is calculated as

$$\phi_f = f_x \phi_P + (1 - f_x)\phi_E \tag{2.26}$$

where the interpolation factor $f_x$ is defined as the ratio of the distances between points $f$ and $E$ and $P$ and $E$:

$$f_x = \frac{\overline{fE}}{\overline{PE}} \tag{2.27}$$

This scheme is termed Central Differencing (CD). While formally second-order accurate, CD schemes have the disadvantage of being prone to numerical oscillations and instability, particularly for transient conditions.

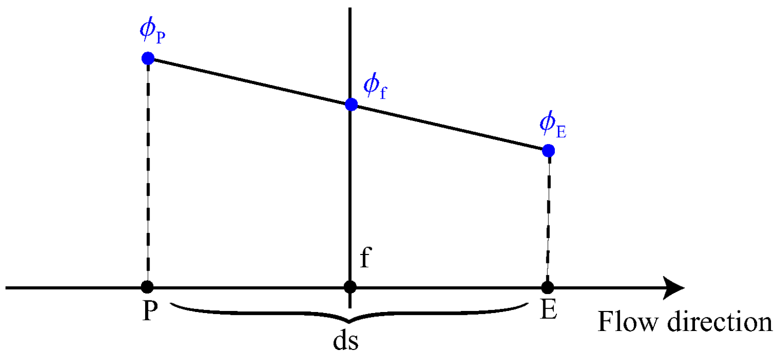


Figure 2.3: Central differencing interpolation between two control volumes onto the shared face.

A bevy of additional second or higher order interpolation schemes have also been developed, including second-order upwind schemes, which extrapolate the face value using two upstream values, and third-order accurate quadratic schemes (i.e. QUICK, Leonard 1979). Most third-order schemes have as their basis some variation of the upwind bias methodology. In general higher order schemes are more accurate but also more prone to instability and oscillatory behavior. The use of flux limiters can mitigate some of these effects. Most of the advective differencing in this work is carried out using CD or first-order upwind differencing.

### 2.3.3 Diffusion Term

Applying the appropriate form of Gauss' Theorem, the discretization for the diffusion term takes the general form of:

$$\int_{V_P} \nabla \cdot (\Gamma_\phi \nabla \phi)\, dV = \sum_f \mathbf{S} \cdot (\Gamma_\phi \nabla \phi)_f = \sum_f (\Gamma_\phi)_f\, \mathbf{S} \cdot (\nabla \phi)_f \tag{2.28}$$

where the subscript $f$ denotes a face interpolated quantity. In an orthogonal cell, determining the right hand side of Eqn. (2.28) is straightforward. The vector $\mathbf{S}$ is parallel to $\mathbf{ds}$ in this case and the flux can be calculated by

$$\mathbf{S} \cdot (\nabla\phi)_f = |\mathbf{S}| \frac{\phi_E - \phi_P}{|\mathbf{ds}|} \tag{2.29}$$

However, on an unstructured grid the possibility exists that the adjoining cell faces are slanted leading to *non-orthogonality* between the vectors **S** and **ds**. In this case we need an alternative method of handling the term $\mathbf{S} \cdot (\nabla\phi)_f$ in order to correct for the non-orthogonal component of the dot product. The non-orthogonal corrector approach involves decomposing this term into an orthogonal and non-orthogonal component:

$$\mathbf{S} \cdot (\nabla\phi)_f = |\mathbf{S}_\Delta| \frac{\phi_E - \phi_P}{|\mathbf{ds}|} + \mathbf{S}_d \cdot (\nabla\phi)_f \, . \tag{2.30}$$

where $\mathbf{S}_\Delta$ is the vector component that is parallel to **ds** and the following equality between $\mathbf{S}_\Delta$, **S** and $\mathbf{S}_d$ must hold:

$$\mathbf{S}_d = \mathbf{S} - \mathbf{S}_\Delta. \tag{2.31}$$

This configuration is depicted in Fig. 2.4. There is more than one option for constructing the component vectors that satisfy Eqn. (2.31). Jasak (1996) found that the "over-relaxed" approach to be the most reliable. This approach involves defining the orthogonal vector component as

$$\mathbf{S}_\Delta = \frac{\mathbf{ds}}{\mathbf{ds} \cdot \mathbf{S}} |\mathbf{S}|^2 \tag{2.32}$$

A final concern regarding the non-orthogonality issue is boundedness. The non-orthogonal correction can lead to negative coefficients and create unboundedness and instability. Limiters can be used to mitigate this issue, although it is often preferable to use a mesh with a low degree of non-orthogonality. Meshing considerations are further discussed in Section 2.7.

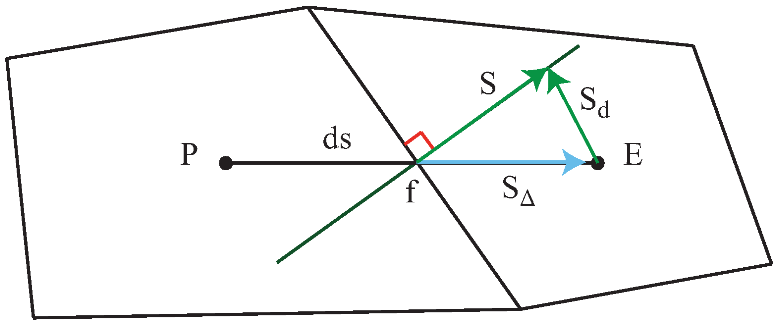


Figure 2.4: Non-orthogonal correction.

### 2.3.4 Source Terms

Source terms include all terms that do not fit the form of the other terms in Eqn. (2.2). The nature of the specific source and its associated bounds should be considered in devising a discretization scheme, however in general the process involves first linearizing the source term, $S_\phi(\phi)$, into two terms and then using the generic equation to convert the volume fluxes to face fluxes. The linearization takes the form

$$S_\phi = S_u + S_p\phi \tag{2.33}$$

where $S_u$ and $S_p$ are the constant and linear components of the source term, respectively, and may be functions of $\phi$. The volume integral can then be calculated using midpoint interpolation as

$$\int_{V_p} = S_\phi(\phi)dV = S_uV_p + S_pV_p\phi_p \tag{2.34}$$

### 2.3.5 Temporal Discretization

Using Eqns. (2.34), (2.24) and (2.30), Eqn. (2.18) can be rewritten in "semi-discrete" form as:

$$\int_{t}^{t+\Delta t}\left[\left(\frac{\partial \phi}{\partial t}\right)_P V_P + \sum_f F\phi_f - \sum(\Gamma_\phi)_f \mathbf{S}\cdot(\nabla\phi)_f\right] dt = \int_{t}^{\Delta t}(S_u V_P + S_p V_P \phi_P)dt. \tag{2.35}$$

There are several options of how to discretize the temporal derivative and integral terms, resulting in schemes of various degrees of order and stability. Three methods are presented here: the Explicit Method, the Euler Implicit Method and the Second-Order Backward Method. The latter two of these methods are implicit, meaning they solve for the face values by use of the updated time cell-centered values, as opposed to the explicit formulation, which rely on previous time-step values to update the fluxes. Implicit methods have the benefit of not being constrained by the Courant-Friedrichs-Lewy (CFL) condition, which for a FV scheme is defined as

$$CFL = \frac{\mathbf{U}_f \cdot \mathbf{ds}}{\Delta t} \tag{2.36}$$

where $\mathbf{U}_f$ denotes the velocity of the cell face and $\Delta t$ is the time-step . For stability, the CFL must remain below unity in the case of explicit time marching schemes. However, even with an implicit time scheme, the CFL condition is still a constraint due to the method of handling the pressure-velocity coupling, as will be discussed in Section 2.3.6.

Neglecting the variation of the face values of $\phi$ and $\Delta\phi$, Eqn. (2.35) can be written as:

$$\frac{\phi_P(t+\Delta t) - \phi_P(t)}{\Delta t} V_P + \sum_f F\phi_f - \sum_f (\Gamma_\phi)_f \mathbf{S}\cdot(\nabla\phi)_f = S_u V_P + S_P V_P \phi_P. \tag{2.37}$$

This formulation is only first-order accurate in time. In the **Explicit** method, the current time-step values are used for the evaluation of the face values of $\phi$ and $\Delta\phi$:

$$\phi_f = f_x \phi_P(t + \Delta t) + (1 - f_x)\phi_E(t), \tag{2.38}$$

$$\mathbf{S} \cdot (\nabla\phi)_f = |\Delta| \frac{\phi_E(t) - \phi_P(t)}{|\mathbf{ds}|} + \mathbf{S} \cdot (\nabla\phi(t))_f. \tag{2.39}$$

where $f_x$ is the interpolation factor as defined in Section 2.3.2. With this formulation, the next value of $\phi$ can be directly calculated without solving a system of linear equations:

$$\phi_P(t + \Delta t) = \phi_P(t) + \frac{\Delta t}{V_P}\left[\sum_f F\phi_f - \sum_f (\Gamma_\phi)_f \mathbf{S} \cdot (\nabla\phi)_f + S_u V_P + S_p V_P \phi_P(t)\right]. \tag{2.40}$$

By contrast, the **Euler Implicit Method** uses the predicted time-level values to determine $\phi$ and $\Delta\phi$:

$$\phi_f = f_x \phi_P(t + \Delta t) + (1 - f_x)\phi_N(t + \Delta t) \tag{2.41}$$

$$\mathbf{S} \cdot (\nabla\phi)_f = |\Delta| \frac{\phi_E(t + \Delta t) - \phi_P(t + \Delta t)}{|\mathbf{ds}|} + \mathbf{S} \cdot (\nabla\phi(t))_f. \tag{2.42}$$

The **Second-Order Backwards Differencing** is a method that increases the accuracy to second-order in time. This is achieved by expanding the terms in Eqn. (2.35) using a Taylor series about $\phi(t + \Delta t)$. This leads to additional expressions for $\phi(t)$ and $\phi(t - \Delta t)$:

$$\phi(t) = \phi t + \Delta t - \frac{\partial\phi}{\partial t}\Delta t + \frac{1}{2}\frac{\partial^2\phi}{\partial t^2}\Delta t^2 + O(\Delta t^3) \tag{2.43}$$

$$\phi(t - \Delta t) = \phi t + \Delta t - 2\frac{\partial \phi}{\partial t}\Delta t + 2\frac{\partial^2 \phi}{\partial t^2}\Delta t^2 + O(\Delta t^3) \tag{2.44}$$

The temporal derivative can now be constructed to second-order accuracy:

$$\frac{\partial \phi}{\partial t} = \frac{\frac{3}{2}\phi(t + \Delta) - 2\phi(t) + \frac{1}{2}\phi(t - \Delta t)}{\Delta t} \tag{2.45}$$

Putting this all together, the Backward Differencing scheme in discretized form is:

$$\frac{\frac{3}{2}\phi(t + \Delta t) - 2\phi(t) + \frac{1}{2}\phi(t - \Delta t)}{\Delta t} V_P + \sum_f F\phi_f(t + \Delta t) - \sum_f (\Gamma_\phi)_f \mathbf{S} \cdot (\nabla \phi(t + \Delta t))_f \tag{2.46}$$

### 2.3.6 PISO Algorithm

One of the issues with the incompressible Navier-Stokes equations is the lack of an explicit equation for the pressure evolution. In this work, this problem is addressed numerically by implementing the Issa's Pressure Implicit Splitting of Operators (PISO) algorithm (*Issa* (1986)). The basic theory behind this algorithm starts with the assumption that the pressure-velocity coupling in Eqn. (2.2) is stronger over small time-steps than the coupling between $\mathbf{U} - \mathbf{U}$ in the nonlinear term. This allows for an iterative process by which the pressure is corrected while the momentum equation is held 'frozen'. The idea is to first solve Eqn. (2.2) for the initial (conservative) velocity field, then apply the pressure corrector steps. One downside to this approach is that the assumption of the frozen momentum terms is only true at small CFL numbers. Hence even with a implicit time discretization, the scheme will still be bound by the CFL limitations, which are typically more stringent than the time-step maximum of unity (*de Villiers* (2006)). The PISO algorithm is also sensitive to mesh quality, a topic that is covered in Section 2.7.

We start the PISO derivation by developing an equation for the pressure. Since the incompressible Navier-Stokes equations lack an explicit equation for the pressure, one must be derived from Eqns. (2.1) and (2.2). First, the momentum equation is written in semi-discretized form as follows

$$a_P \mathbf{U}_P + \sum_E a_E \mathbf{U}_E = \frac{\mathbf{U_0}}{\Delta t} - \nabla p \tag{2.47}$$

where $\mathbf{U_0}$ is the latest available velocity field and $a_P$ is the matrix coefficient corresponding to the cell with centroid $P$, and likewise $a_E$ is the matrix coefficient corresponding to a neighboring cell with centroid $E$ . To facilitate the notation, an operator $\mathbf{H}$ is introduced as

$$\mathbf{H}(\mathbf{U}) = \frac{\mathbf{U_0}}{\Delta t} - \sum_E a_E \mathbf{U}_E \tag{2.48}$$

Substitution of the $\mathbf{H}(\mathbf{U})$ operator into Eqn. (2.47) allows us to express the velocity field as

$$\mathbf{U}_P = \frac{(\mathbf{H}(\mathbf{U}) - \nabla p)}{a_P} \tag{2.49}$$

The interpolated face velocities of Eqn. (2.49) are given by

$$\mathbf{U}_f = \frac{(\mathbf{H}(\mathbf{U}))_f - (\nabla p)_f}{(a_P)_f} \tag{2.50}$$

Next, we discretize the continuity equation

$$\nabla \cdot \mathbf{U} = \sum_f \mathbf{S}_f \cdot \mathbf{U}_f = \sum_f F = 0 \tag{2.51}$$

where $F$ is the flux through a cell face. Using Eqn. (2.50) and Eqn. (2.51), $F$ can be expressed as

$$F = \mathbf{S}_f \cdot \mathbf{U}_f = \mathbf{S}_f \cdot \left[ \frac{(\mathbf{H}(\mathbf{U}))_f - (\nabla p)_f}{(a_P)_f} \right] \tag{2.52}$$

By substituting Eqn. (2.50) into Eqn. (2.51) we obtain the pressure equation for incompressible flow

$$\nabla \cdot \left( \frac{\nabla p}{a_P} \right) = \nabla \cdot \left( \frac{\mathbf{H}(\mathbf{U})}{a_P} \right) = \sum_f \mathbf{S}_f \cdot \left( \frac{\mathbf{H}(\mathbf{U})}{a_P} \right)_f \tag{2.53}$$

We now have the Navier-Stokes equations in a discretized form amenable to solving the pressure-velocity coupling:

$$a_P \mathbf{U}_P = \mathbf{H}(\mathbf{U}) - \sum_f \mathbf{S}_f \cdot p_f \tag{2.54}$$

$$\sum_f \mathbf{S}_f \left( \frac{\nabla p}{a_P} \right)_f = \sum_f \mathbf{S}_f \cdot \left( \frac{\mathbf{H}(\mathbf{U})}{a_P} \right)_f \tag{2.55}$$

With these expressions, the PISO loop sequence is as such:

1. Solve the discretized momentum equation (Eqn. (2.47)) for the initial estimated velocity field using the previous time-step's pressure field. This is the *momentum predictor* step.

2. Construct Eqn. (2.48) for $\mathbf{H}(\mathbf{U})$ and solve Eqn. (2.53) for the new pressure field. This is the *pressure solution* step.

3. A corrector step is applied in order to enforce convergence on the non-orthogonal components of $p$. The number of non-orthogonal correctors is prespecified and is typically one or two.

4. The velocity field and fluxes are updated according to Eqns. (2.54) and (2.50) .

5. Steps 2-4 are repeated until the dependent variables converge.

## 2.4 OpenFOAM

Several Computational Fluid Dynamics codes exist for solving the Navier-Stokes equations. This research uses the OpenFOAM (Open Field Operation and Manipulation) CFD Toolbox, which is an open source CFD software code produced by OpenCFD Ltd. OpenFOAM has a large user base across most areas of engineering and science, from both commercial and academic organizations. Aside from the extensive variety of solvers and pre- and post-processing utilities it provides, a significant advantage to using this software is its extensibility. The open source licensing and ease of which new equations, boundary conditions and utilities can be added to the code make it an extremely flexible CFD tool. As an example of the simplicity afforded by the high level of data abstraction implemented in the code, the format for the scalar transport equation (Eqn. (2.3)) is shown here for comparison

```
fvScalarMatrix TEqn
(
   fvm::ddt(T)
   + fvm::div(phi, T)
   - fvm::laplacian(DT, T) ==
   f
);
```

The OpenFOAM software library includes solvers for many specific simulation types, including transient and steady state two-phase flows, reacting flows, and compressible flows. In this work the primary solver employed is pisoFOAM, which is a transient solver for incompressible and turbulent flows.

## 2.5 Boundary Conditions

Cells located along the boundaries must receive special treatment in order to properly enforce the physics of the simulated problem. While OpenFOAM includes a large number of precoded boundary conditions, only the basic types of boundary conditions relevant to this work will now be described. These are the fixed value and fixed gradient boundary conditions, also known as Dirichlet and Von Neumann boundary conditions. From these two primitive boundary conditions, many others can be derived.

Due to the flexible geometry of the finite volume method, non-orthogonality of the boundary faces must first be addressed. Figure 2.5 depicts a boundary cell, in which the vector $\mathbf{ds}$ is now the distance from $P$ to the center of the boundary face, $b$. In this case, the vector $\mathbf{ds}_n$ which is the vector normal to the boundary extending from $P$, does not coincide with $b$. Therefore the face area vector $\mathbf{S}$ can be decomposed into an orthogonal part ($\mathbf{S}_\Delta$) which is perpendicular to the boundary face, and a parallel part ($\mathbf{S}_d$). However, as the prescribed boundary conditions are assumed to be valid over the entire boundary face and $\mathbf{S}_\Delta$ is equal in magnitude and direction to $\mathbf{S}$, the perpendicular component alone is sufficient to calculate the boundary terms.

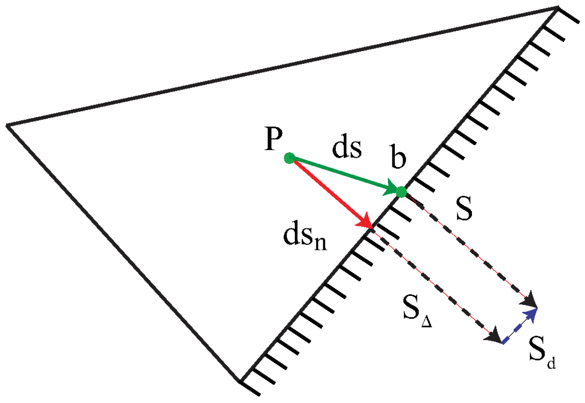


Figure 2.5: Control volume located on the boundary.

### 2.5.1 Fixed Value Boundaries

The fixed boundary condition enforces the value of a generic variable, $\phi$ at the boundary face $b$ as $\phi_b$. According to the advection term discretization given by Eqn. (2.24), the boundary term contribution to the sum on the right is

$$(F\phi_f)_{f=b} = F_b\phi_b \tag{2.56}$$

where $F_b$ is the flux across the boundary face.

For the diffusion term, using the discretization in Eqn. (2.28), the face gradient at $b$ is determined using the specified face value and the value at the cell center:

$$\mathbf{S} \cdot (\nabla\phi)_b = |\mathbf{S}|\frac{\phi_b - \phi_P}{|\mathbf{ds}_n|}. \tag{2.57}$$

$$\int\limits_{V_P} \nabla \cdot (\Gamma_\phi \nabla\phi)dV = \sum_f (\Gamma_\phi)_f \mathbf{S} \cdot (\nabla\phi)_f. \tag{2.58}$$

where again the value of the normal gradient of $\phi$ at the boundary needs to be found using a corrector step similar to the correction applied above:

$$\mathbf{S} \cdot (\nabla\phi)_b = |\mathbf{S}_d|\frac{\phi_b - \phi_P}{|\mathbf{ds}|} + \mathbf{S}_\Delta \cdot (\nabla\phi)_P. \tag{2.59}$$

### 2.5.2 Fixed Gradient Boundaries

The fixed gradient boundary condition is implemented by prescribing the dot-product of the gradient and the outward pointing unit normal at the boundary:

$$\left(\frac{\mathbf{S}}{|\mathbf{S}|} \cdot \nabla\phi\right)_b = g_b, \tag{2.60}$$

where $g_b$ represents the specified fixed gradient. The convection term is handled by calculating $\phi$ using the prescribed gradient and the cell centered value as:

$$\phi_b = \phi_P + \mathbf{ds_n} \cdot (\nabla\phi)_b = \phi_b + |\mathbf{ds_n}| g_b. \tag{2.61}$$

The diffusion term for the fixed gradient boundary condition is calculated using the dot product between $\mathbf{S}$ and $(\nabla\phi)_b$, which is $|\mathbf{S}|g_b$. The diffusion boundary term is then

$$(\Gamma_\phi)_b |\mathbf{S}| g_b \tag{2.62}$$

The general boundary conditions types used in this work are given in Table 2.3. Specific values are included in the simulation descriptions in the following chapters. Values for $k$ and $\epsilon$ are estimated according to Eqns. (2.12) and (2.13).

Table 2.3: General boundary conditions for CFD simulations.

| Boundary | $p$ (m$^2$/s$^2$) | $U$ (m/s) | k (m$^2$/s$^2$) | $\epsilon$ (m$^2$/s$^3$) | $C$ (units) |
|---|---|---|---|---|---|
| inlet | zero gradient | fixed value | fixed value | fixed value | fixed value |
| outlet | fixed value | zero gradient | zero gradient | zero gradient | zero gradient |
| walls | zero gradient | fixed value | wall function | wall function | zero gradient |
| ports | zero gradient | fixed gradient | zero gradient | zero gradient | zero gradient |

As a final note on the boundary conditions, the above formulations are only first order accurate due to the fact that $\mathbf{ds}_n$ may not point to the center of the boundary face. In order to increase the boundary calculation accuracy, corrections using the components $\mathbf{S_\Delta}$ and $\mathbf{S_d}$ can be implemented similar to the procedure described in Section 2.3.3.

## 2.6 Laminar Pipe Flow Validation

A simple pipe flow example is included as a basic validation case for the CFD model. Steady state flow with an inlet velocity of 0.001 m/s is modeled in a pipe with radius 0.0508 m and length 1.21 m. The mesh is composed of 94,843 hexagonal cells. The fully developed radial velocity profile for laminar flow in a pipe is parabolic, with zero velocity at the boundary as enforced by the no-slip wall boundary condition and the maximum velocity occurring at the center of pipe. The analytical solution for this flow profile is derived by integrating the incompressible Navier-Stokes equations in radial coordinates and is given by

$$U_r = -\frac{R^2}{16\mu}\left(\frac{\partial p}{\partial x}\right)\left[1-\left(\frac{r}{R}\right)^2\right] \tag{2.63}$$

where $D$ is the pipe diameter, $R$ is the radius and $r$ is the radial coordinate. Also note that the pressure, $p$, in this example is the total pressure and not the kinematic pressure. The dynamic viscosity, $\mu$, is $1 \times 10^{-3}(Ns)/m^2$ for water. The pressure gradient term is found using the Hagen-Poiseuille equation to calculate the drop in the pressure over the length $L$ of the pipe

$$\frac{\Delta p}{L} = \frac{128\mu L}{\pi D^4} Q \tag{2.64}$$

A comparison of the modeled velocity profile obtained from OpenFOAM and the analytical velocity profile calculated from equations (2.63) and (2.64) is shown in Fig. 2.6. The OpenFOAM results show excellent agreement with the analytical profile.

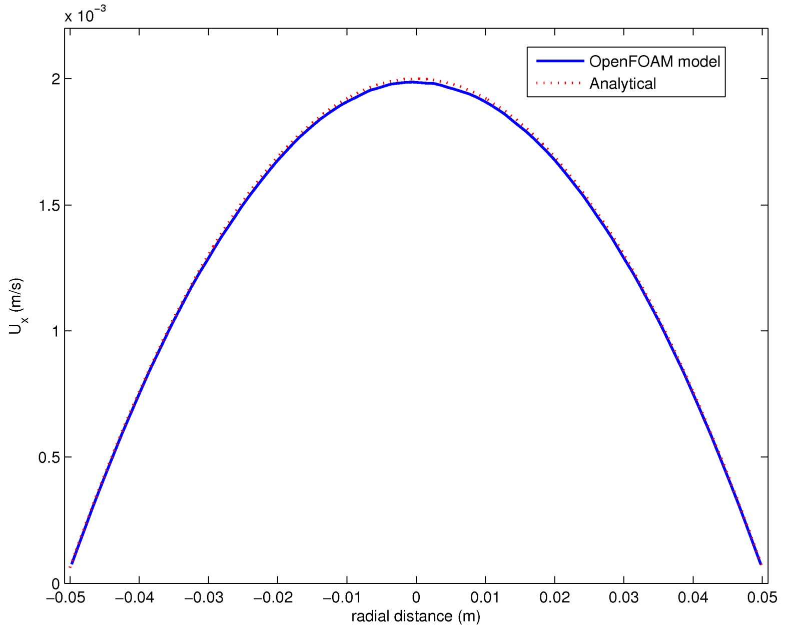


Figure 2.6: Laminar pipe flow validation.

## 2.7 Meshing Considerations

It is necessary at this point to discuss the importance of adequate meshing for getting accurate CFD results. While one benefit to finite volume methods is the flexibility to model complicate geometry and uneven boundaries, the fact remains that the success of the simulation will be greatly affected by the quality of the cells. Bad meshing can be improved by using schemes other than CD, or higher order approximations. Corrections can be implemented for all of the meshing problems, however at best the solution will be slower to converge. At worse, the accuracy will be severely compromised. Meshing is a complicated field of study in itself, but the main considerations can be summed up as follows:

### 2.7.1 Spatial Resolution

As a rule, finer spatial discretization will lead to more accurate CFD results. However, the CFL condition will greatly inhibit the feasibility of modeling a large scale problem as the cell minimum size becomes very small. As was previously discussed, this is a limitation that holds for the PISO solver in addition to explicit time marching schemes.

### 2.7.2 Aspect Ratio

The aspect ratio (AR) of the triangle or tetrahedron is the length of the longest edge divided by the length of the shortest altitude (Fig. 2.7). For a tetrahedral cell the aspect ratio is defined as $\frac{\sqrt{3}\min(h_i)}{\sqrt{2}\max(l_i)}$. For a hexagonal cell the aspect ratio is simply $\frac{\min(l_i)}{\max(l_i)}$, where $l_i$ is the face length and $h_i$ is the height. Limits on ideal acceptable aspect ratios are dependent on the region of the grid. For interior locations, it is recommended that $0.2 < AR < 5$ , while near the boundary the direction perpendicular to the flow can typically tolerate larger aspect ratios. Cells that have too large of aspect ratios will lead to high interpolation errors and degrade the validity of the simulation.

### 2.7.3 Skewness

Two types of cell skewness can be defined: Non-orthogonality, which was introduced in Section 2.3.3, defines skewness that arises when the face normal and the vector $\mathbf{ds}$ from one cell centroid to the next are not parallel. This leads to the angle $\theta$ between $\mathbf{ds}$ and $\mathbf{S_f}$ being greater than 0, such as is pictured in Fig. 2.7. As was described above, the errors introduced in the diffusion term via non-orthogonality can be reduced using the non-orthogonal correction.

A second type of skewness occurs when the arrangement of the cells results in the vector $\mathbf{ds}$ not passing through the center of the adjoining cell face, as illustrated in Fig. 2.7. This is a problem because typically it is assumed that the face value at the midpoint can be linearly interpolated from the cell centroids at either side of the face, therefore when $\mathbf{ds}$ does not pass through the face centroid interpolation errors occur. Highly skewed cells also should be avoided to improve convergence time.

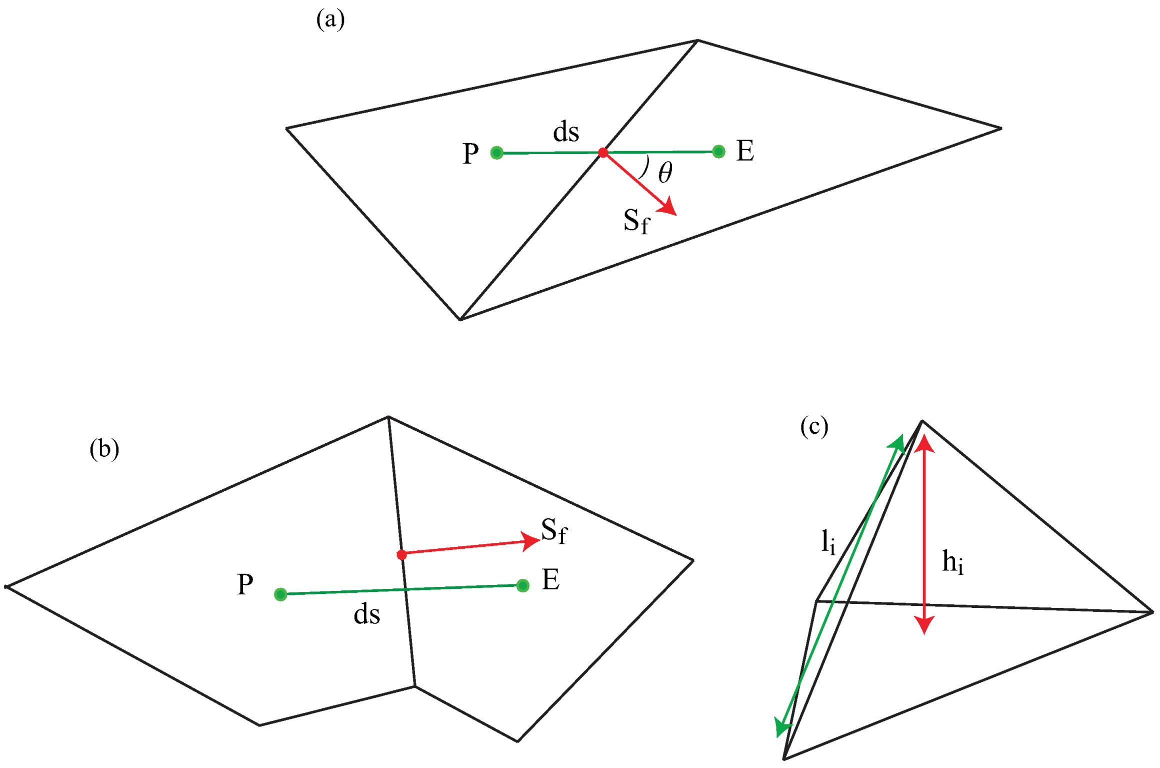


Figure 2.7: Skewness due to non-orthogonality (a), skewness due to **ds** not bisecting the face midpoint b), aspect ratio (c).

### 2.7.4 Tetrahedral vs Hexagonal Cells

The example shown in Fig. 2.9 illustrates the errors that can arise when using tetrahedral vs. hexagonal cells for the pipe flow validation presented in Section 2.6. Both meshes are composed of roughly 180,000 cells, however the mesh comprised of tetrahedrals has higher aspect ratio, skewness and non-orthogonality measures, although still within the acceptable limits. The PISO algorithm with 2 non-orthogonal corrector steps is again used to run the model until steady state is reached, and the resulting radial flow profiles compared. The simulation with the tetrahedral mesh leads to errors in the velocity towards the center of the pipe.

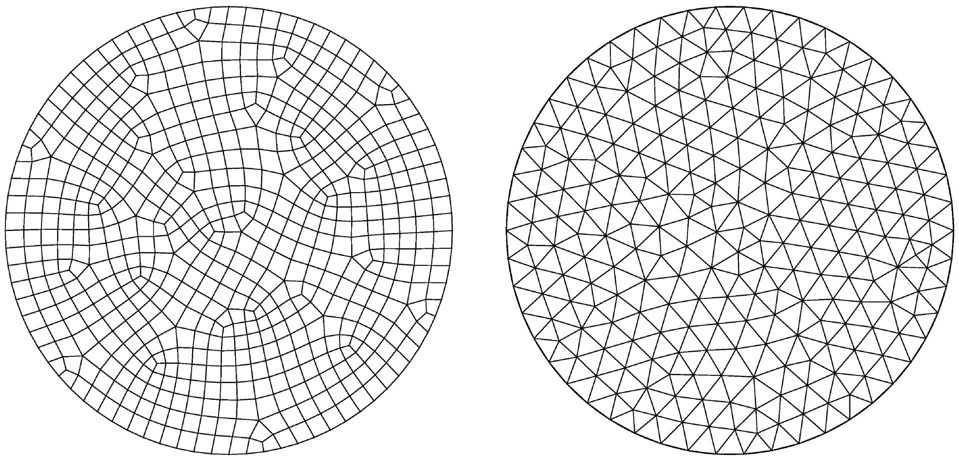


Figure 2.8: Hexagonal (left) vs. tetrahedral cells (right).

### 2.7.5 Wall Treatments

A final meshing consideration concerns boundary layers. For turbulent flows, the region near the boundary is a region of high shear and large gradients. Due to the discretization treatment of linearly interpolated he state variables between cells, additional constraints are placed on the spatial resolution in order to accurately model this region. In the bulk (outer) region of the flow meshing constraints are relaxed and larger cell volumes can usually be used. In order to maximize simulation efficiency, it is preferable to gradually increase the resolution with proximity to the boundary as

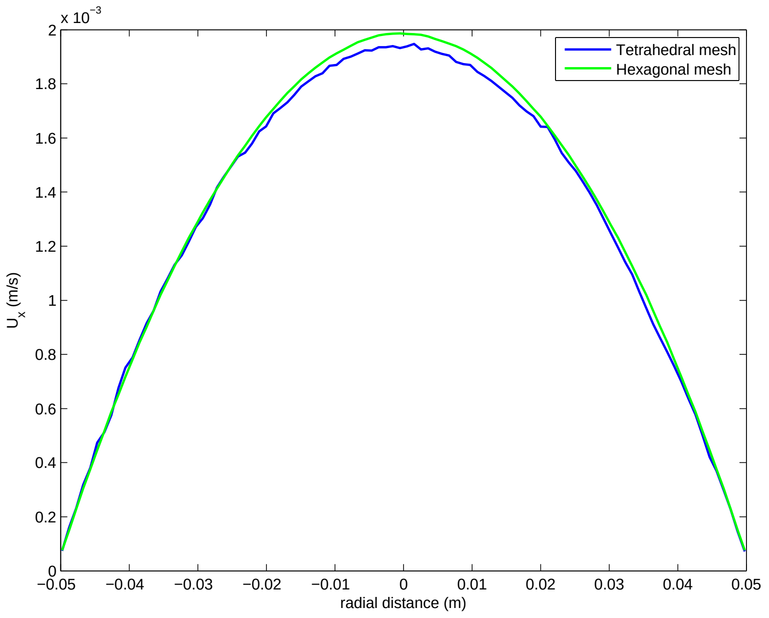


Figure 2.9: Radial laminar pipe flow velocity profile generated with tetrahedral cells and hexagonal cells.

opposed to using a uniformly fine grid throughout the domain, a technique referred to as *boundary layer* meshing. An example of this is shown in Fig. 2.10. In order to determine the optimal boundary layer resolution, the concept of the dimensionless wall distance, $y^+$ is introduced. This value is defined as

$$y^+ \equiv \frac{u_* y}{\nu} \tag{2.65}$$

where $u_*$ is the *shear velocity* at the wall and $y$ defines the distance to the wall.

Before going further into the boundary layer treatment, it is necessary to provide a brief explanation as to the behavior of the velocity profile in the near wall region. The mean shear layer velocity profile of fully developed flow is divided into three main regions. The viscous sub-layer, nearest the boundary, is directly affected by the no-slip condition which results in a linear velocity profile:

$$u^+ = y^+ \qquad 0 < y^+ < 5 \tag{2.66}$$

The outer layer, where Reynolds stresses dominate the flow, is called the logarithmic inertial layer and has the form:

$$u^+ = \frac{1}{\kappa}\ln y^+ + C \qquad y^+ > 30 \tag{2.67}$$

where $\kappa$ and $C$ are constants. The region between $5 < y^+ < 30$ is the *buffer region*, wherein neither Eqn. (2.66) or (2.67) hold because both viscous and Reynolds stresses affect the flow. A schematic of the boundary layer velocity profile is shown in Fig. 2.11.

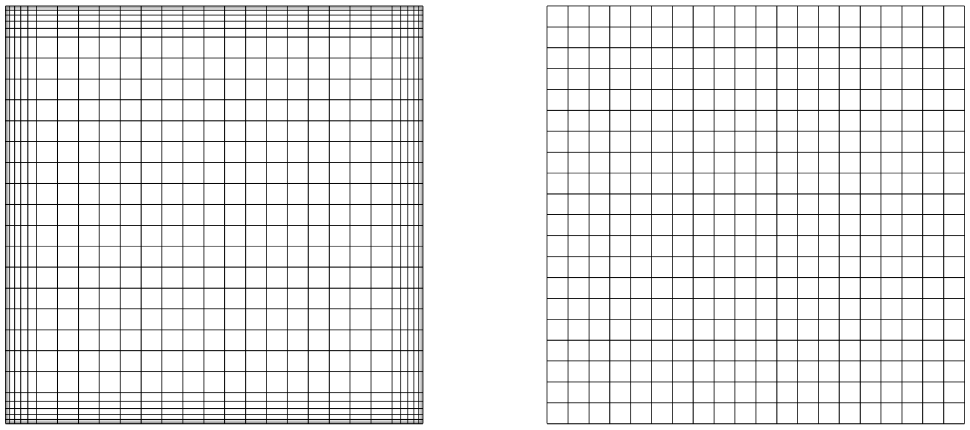


Figure 2.10: Example of a grid with (left) and without boundary layers.

In order to fully resolve the boundary layer it is necessary to ensure that the first grid point lies at a distance, $\Delta y$, that is (1) within the boundary layer and (2) at a distance above the viscous sublayer. This corresponds to a $y^+ \approx 1$. The $\Delta y$ needed to match this $y^+$ value and to fully resolve the boundary layer up to the wall is typically very small, and thus adds significantly to the computational burden of the simulations. A more economical alternative is to apply approximate models known as *wall-treatments* in this region when the fully resolved flow is not a necessary output of the model. Wall functions are semi-empirical relationships that

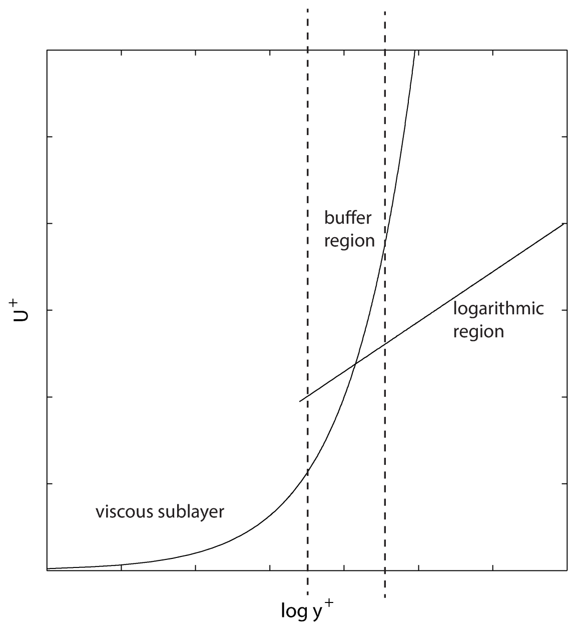


Figure 2.11: Non-dimensionalized shear layer velocity profile as a function of distance from the wall.

describe ideal flow in the logarithmic inertial layer, based on the log-law given in Eqn. (2.67). The use of wall-treatments allows for a coarser grid near the wall, as the $y^+$ value should be estimated to fall within the logarithmic layer. While wall functions have significant limitations, particularly in the cases of complex geometries, the computational savings they afford make them an acceptable alternative in many cases. In this work, standard wall functions are used as one goal of this research is to develop algorithms that can be implemented rapidly. OpenFOAM calculates the $y^+$ value at the distance $\Delta y$ specified by the first cell centroid nearest the boundary and applies the wall function in the case that $y^+ > 30$.

# CHAPTER III

# Source Inversion Theory

This chapter begins with the definition of the source inversion problem. Numerical optimization, which is the mathematical basis of solving the source inversion problem in this work, is then described. A discussion of the basic optimization classifications including gradient-based and derivative-free methods is included with a focus on the gradient-based approaches. The interfacing between the optimization software used in this research and the CFD model described in the previous chapter are outlined, along with the problems encountered when implementing the optimization scheme within a discretized framework. Finally, the concept of regularization with respect to the source inversion problem is introduced, and the Tikhonov regularization implemented herein is formulated.

## 3.1 Source Inversion

Source inversion is a specific application of an *inverse problem.* Inverse problems are those in which the goal is to retrieve unknown parameters of a system based on a given set of observations characterizing that system. This is in contrast to the traditional *forward modeling* approach, which describes the process of generating system observations as a function of known system parameters. Inverse problems arise frequently in a range of disciplines, as it is often the case that a sparse number of observations are the only information available to deduce the original properties of a system of interest. Inverse problems are typically more difficult to solve than forward problems due to issues related to the concept of ill-posedness, a topic that will be revisited subsequently.

Some common examples of inverse theory applications include the field of medical imaging, where inverse theory is used to construct images from CT scans and other imaging technology. Many further examples are related to the field of geophysics, where inverse methods are used to deduce the location and properties of tectonic plates, reconstruct the Earth's gravitational field based on sparse surface measurements, and to locate the origin of earthquakes via data records of seismic waves. Of the above examples, this last one illustrates the concept of the *source inversion* inverse problem: one in which the goal is to reconstruct the properties of an unknown source from finite data measurements. The 'properties' sought to be recovered typically include the source location in addition to any other unknown parameters necessary to characterize the system. These parameters are denoted as $\mathbf{f}$ in this text, which represents a vector of unknown parameters which define the output of the source inversion process.

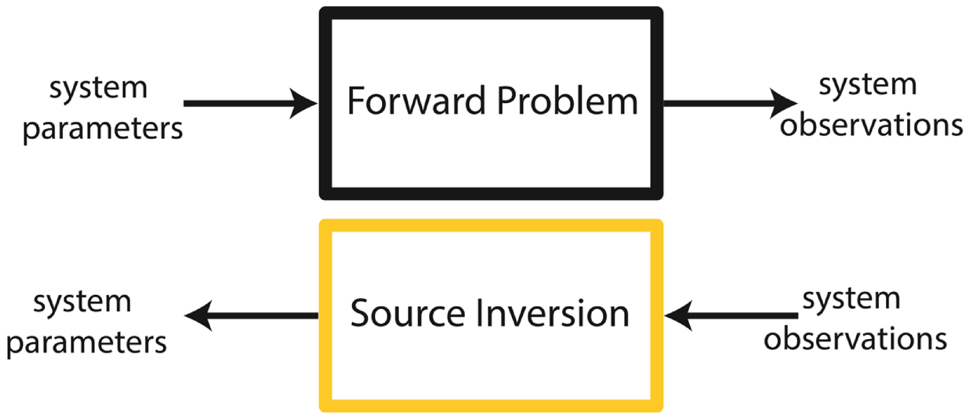


Figure 3.1: Difference between the forward and source inversion modeling approaches.

If computing time is not a limiting factor, the simplest method of source inversion is by an **exhaustive parameter search**. In this approach, every possible combination of the unknown parameters in the vector $\mathbf{f}$ is tested as a possible solution to the source inversion problem. At every permutation of the elements in $\mathbf{f}$, a forward model characterizing the system is run and a measure of the error between the observed and modeled values is computed. This measure of error is herein denoted as $J(\mathbf{f})$. The values of $\mathbf{f}$ that result in the minimum measure of error are determined to be the best-fit parameters of the problem. By this method, an exhaustive parameter search is guaranteed to find the best set of parameters, *given that the source inversion problem is well-posed.* Well-posedness defines a problem that exhibits existence, uniqueness and stability with respect to the solution. A problem that does not meet these requirements is hence classified as being ill-posed. The common occurrence of ill-posedness in inverse problems is a significant limitation to their tractability, as was alluded to above. Methods of dealing with this issue will be revisited in Section 3.7 and in Chapter VI.

To illustrate the concept of the exhaustive parameter search method, consider a two-dimensional channel in which a contaminant is released at an unknown location at $t = t_0$. Assuming that all other variables of the problem, such as the initial source strength $q_s$, time of release $t_s$, and ambient flow characteristics are all known, the parameters sought in this scenario are the source release coordinates $\mathbf{f} = (x_s, y_s)$. This scenario is depicted in Fig. 3.2. Note that the sensors in this case are located at the boundaries, however in general the location of the sensors is arbitrary and a further variable to investigate via the source inversion algorithm.

Every grid point in the discretized domain, indicated by the gray dots in Fig. 3.2, is a possible solution to this source inversion problem. The forward model in this case is a model simulating the ambient flow and the advection and dispersion of the contaminant. The observed data values are a time series of contaminant measurements, $C_m(t)$, where the subscript $m$ refers to the $m_{th}$ sensor. In this case, the output of the forward model is the predicted values of the contaminant concentration at the

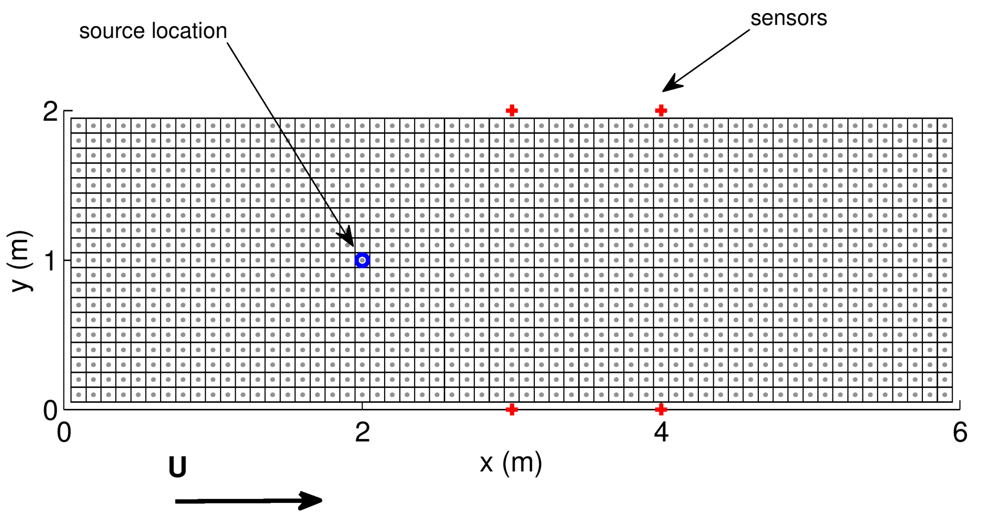


Figure 3.2: Grid for the 2 x 6 channel.

sensor locations, $\hat{C}_m(t)$ and $J(\mathbf{f})$ is a user-defined measure of the error between the modeled and observed concentration values, most frequently in the form of a least squares function:

$$J(\mathbf{f}) = \sum_{m=1}^{N_s} \left( C_m - \hat{C_m}(\mathbf{f}) \right)^2 \tag{3.1}$$

where $C_m$ is the observed contaminant concentration at sensor $m$, $\hat{C_m}$ is the modeled concentration given the current parameters at the location of sensor $m$, and $N_s$ is the total number of sensors.

To perform the exhaustive parameter search, each grid point is tested as the potential true source location $(x_s, y_s)$, the forward model run with these parameters, and Eqn. (3.1) evaluated. The minimum error between the $C$ and $\hat{C}$ values will occur at the evaluated grid point nearest to the actual source location, and in the case that the true source location $(x_s, y_s)$ falls directly on a cell center point and assuming perfect sensors (no error and no noise), $J = 0$ when $x_c = x_s$ and $y_c = y_s$, where the subscript $c$ denotes cell-centered grid points. This problem is well-posed, and thus by evaluating the model and Eqn. (3.1) a total of $N_c$ times, where $N_c$ is the number of cells in the computational domain, the closest point to $(x_s, y_s)$ is guaranteed to be found.

This simple example brings up two important points: The first is in regard to the impact of grid discretization on the accuracy and robustness of the solution. In the likely case that the true source location does not coincide directly with a cell center, there will be an error between the true source location and the best-fit parameters determined by the source inversion. The magnitude of the error will be a function of the density of grid points, as the fewer the number of grid points, the greater the possible discrepancy between the true source location and the values of $(x_s, y_s)$ that result in the smallest $J$. This discretization error is one of several issues that arise due to the discretization of the source inversion problem which are discussed in Section 3.6.1. One obvious method of reducing this discretization error is to test more points (i.e. increase the resolution of the grid). However doing this will lead to a longer source inversion process, and brings us to the second point.

While the above described method of inversely determining system properties has the benefit of being extremely robust, it is easy to see how the required number of computations for a system with a large parameter space quickly becomes unwieldy, as the number of possible permutations of $\mathbf{f}$ becomes very large and hence necessitates numerous repetitions of the forward model. Instead of testing every possible solution, it would be much more economical to evaluate $J$ at specific points and use information inferred at those points to make an intelligent guess as to what parameters to test next in the search for the optimal values of $\mathbf{f}$. This is the rationale behind framing the source inversion problem in terms of *numerical optimization.*

## 3.2 Numerical Optimization

Section 3.1 laid the groundwork for the source inversion problem and illustrated how it is computationally advantageous to seek a method of efficiently minimizing the objective function by taking an optimization approach. This section serves to briefly educate the reader on the subject of numerical optimization, including the basic classification of optimization algorithms and their corresponding mathematical bases.

Simply stated, optimization is the procedure of finding the minimum of a function given specified constraints on the solution. Problems that have no constraints or constraints that do not affect the solution are referred to as *unconstrained optimization* problems, while those problems subject to limitations on the parameters are *constrained optimization* problems. Constrained optimization problems may have simple boundary constraints, general linear constraints, complex nonlinear inequalities between the various variables, or any combination of these. If the objective function and all associated constraints are linear the optimization problem is further classified as a *linear optimization* problem. Conversely, if any of the elements in the optimization problem are nonlinear the problem is a *nonlinear optimization* problem. Most of the optimization problems relevant to the fields of science and engineering fall into this latter category.

In addition to the constrained/unconstrained and linear/nonlinear classifications, optimization problems vary in the scope of the sought after solution. *Global* optimization problems require a solution that holds for the entire domain, while a *local* optimization problem does not necessitate that the solution be the best solution for the entire parameter space, as long as it meets the optimization criteria within the defined subspace of the domain.

The significance of the above classifications is in regard to picking an appropriate optimization method to employ for a given problem, as the various approaches and underlying algorithms are suited to some problem types and not others (see Section 3.3).

The formal definition of an optimization problem, including any possible constraints, is:

$$\min_{\mathbf{f}\in R^n} J(\mathbf{f}) \quad \text{subject to} \quad \begin{array}{l} c_i(\mathbf{f}) = 0, \quad i \in \mathcal{E}, \\ c_i(\mathbf{f}) \geq 0, \quad i \in \mathcal{I}. \end{array} \tag{3.2}$$

where $J$ is the user-defined measure of error between the observed data points and the modeled values, referred to as the *objective function* or *functional*, and $c_i$ are the equality and inequality constraints which are referenced by the indices $\mathcal{E}$ and $\mathcal{I}$, respectively. For the two-dimensional source inversion problem described above, the constraints on the optimization problem are given by the physical bounds alone. That is, $x_{min} \leq x_s \leq x_{max}$ and $y_{min} \leq y_s \leq y_{max}$. The objective function which we wish to apply the optimization to can be formulated a number of ways. The example given in Eqn. (3.1) is one of the most simplistic formulations. However in order to make the functional surface more amenable to numerical optimization, variations on this formula are typically taken in source inversion applications. In this work the source inversion objective function is defined as

$$J(\mathbf{f}) = ln\left[\frac{1}{2}\sum_{m=1}^{N_s}\int_0^T\int_\Omega\left[\delta(\mathbf{x}-\mathbf{x}_m)\delta(t-t_s)(C-\hat{C})^2\right]d\Omega dt + \frac{1}{2}\sigma\mathcal{R}(\mathbf{f})\right] \tag{3.3}$$

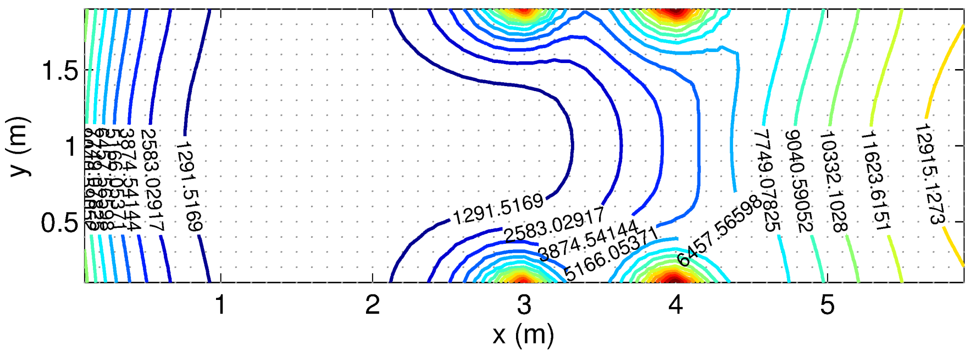
where $C$, $\hat{C}$, $N_s$ and $\mathbf{f}$ are as defined above, $\Omega$ is the spatial domain, and the last term on the right is a source regularization term that will be defined in Section 3.7. Eqn. (3.3) represents a surface which will have a minimum at the location nearest to the true source and local maxima at the location of the sensors. For a continuous point source, $t_s = t$ for every discretized time-step, whereas for an instantaneous point source $t_s$ is only defined at one value. This equation is a function of the contaminant concentration $C$, and thus the objective function is constrained by the system of equations defined in Chapter II, Eqn. (2.3), (2.1) and (2.2).

Taking the natural log of the functional is a method of making the functional topography more favorable for minimization. Without taking the natural log or log of the functional (the results are similar), the large value of the functional at the sensor locations dominates the topography of the surface, resulting in a functional surface that varies abruptly in some directions and gradually in others. This causes difficulties with gradient-based optimizers in particular, for reasons that will become more clear in the following sections. Taking the natural log of the functional has the effect of scaling the functional so that the gradients are closer in order of magnitude, allowing for more efficient and accurate gradient-based optimization.

An illustration of the logarithmic effect on the functional is shown in Fig. 3.3. These figures are generated by calculating Eqn. (3.3) at every grid point for the two-dimensional channel described above, with and without taking the natural log of the terms in the brackets. In this example the true source is located at $(2, 1)$ and 4 sensors are located at coordinates of $(3, 0)$, $(3, 2)$, $(4, 0)$ and $(4, 2)$ as shown in Fig. 3.2. The top subfigure in Fig. 3.3 shows the functional contours for the functional in Eqn. (3.3) excluding the natural log operator. The 4 maxima that occur at the locations of the sensors are in the range of $\approx 5000 - 7000$ while the minimum value of 0 occurs at the true source location. The region surrounding the true source location is relatively flat, with very mild gradients compared to the regions surrounding the sensors. The bottom subfigure shows the functional after taking its natural log. The surface in this case is much more favorable for optimization as the gradients near the maxima are now on the same order of magnitude as those surrounding the minimum.

## 3.3 Numerical Optimization Methods

Successful numerical optimization can be a challenging task, especially for more complicated optimization problems. Methods vary in their properties of robustness, efficiency and accuracy. Ideally, a method would have all three of these qualities, but in reality one property inevitably comes at the expense of another. Furthermore, there



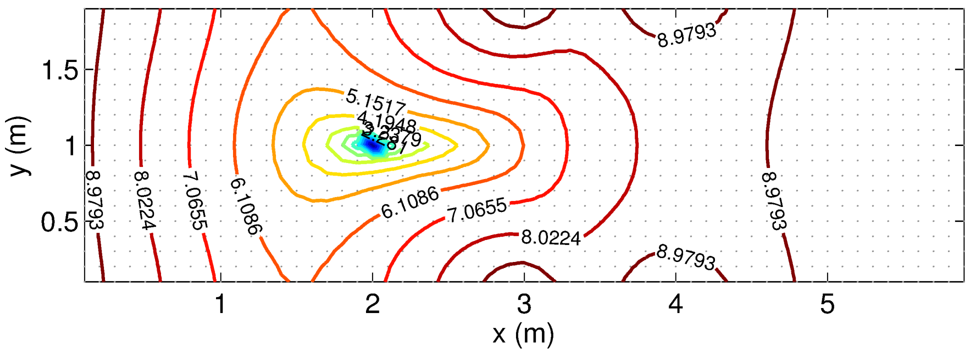


Figure 3.3: Effect of taking the natural log of the objective function. Top figure is without taking the natural log, bottom figure shows the functional contours after taking the natural log.

are no fool-proof optimization methods that guarantee a solution for all problems, let alone an efficient and robust one. However a plethora of methods have been developed that have proven to be reliable for specific problem types. The task is then to match the problem type to a solver that is best suited for the problem at hand. As source inversion optimization is a bound-constrained, global optimization problem, we seek an algorithm that is suited for this optimization type.

All optimization methods have in common that they are iterative. The general approach is to make an initial guess at the solution $\mathbf{f}_0$, assess the accuracy of the guess, and intelligently formulate a next guess for $\mathbf{f}$ that is likely to be closer to the true solution. This process repeats until the guessed solution meets the problem criteria to within a specified acceptable tolerance or until no further progress can be made and the iterations are terminated. All optimization approaches can be divided into two broad categories, with respect to their underlying method of determining the 'intelligent' way of choosing new iteration points based on the previous guess or guesses. These categories are the *Gradient-based* and *derivative-free* methods. One may think of these two types as being analogous to a rifle vs. splatter approach: gradient-based methods aim for their target carefully using one 'bulls-eye' at a time and zero-in on the solution by making a trajectory that approaches the true bulls-eyes in a step-by-step fashion at each iteration. Derivative-free algorithms by contrast aim broadly and directly evaluate the functional at many scattered points within a search region at each iteration, with the goal of using this wide-scale information to compare to past information gathered at other points and determine a strategy for the next optimal set of search points. Through this repetitive process, their goal is also to hone in on the true bulls-eye. Both of these approaches have their respective merits and disadvantages, as are discussed below.

### 3.3.1 Gradient-Based Optimization

*Gradient-based* methods, as the name suggests, use analytical or numerical gradients from the objective function in order to determine the next guess for $\mathbf{f}$. They thus evaluate a single point in the parameter space at a time, typically using this current iterations information only in deciding the parameter guess for the next iteration. Gradient-based methods are well-suited for problems in which there is a single, global minimum. They vary in their efficiency, but for a well-scaled functional in which there is a global minimum they can converge relatively quickly. However, due to their reliance on point-by-point information, gradient-based methods are finicky about the functional topography, and problems arise when the objective function is not 'well-behaved'. That is, a functional that does not exhibit the ideal qualities of being uni-modal, convex and continuous. At the very least, when the objective function does not conform to these ideals, gradient-based methods typically require a good starting guess in order to increase their odds of successfully converging to the true solution.

#### 3.3.1.1 Convexity

The term *convex* describes an important condition in numerical optimization, particularly in regards to gradient-based optimization, and thus deserves further discussion. A function is formally defined as convex when the following relationship holds:

$$J[\gamma x + (1-\gamma)y] \leq \gamma J(x) + (1-\gamma)J(y)], \tag{3.4}$$

for all $x, y \in \Omega$ and $\gamma \in [0,1]$. In other words, a convex function is one in which the value at the midpoint of every possible interval in the defined domain does not exceed the mean of the values at the endpoints of the interval. An example of this property for a simple one-dimensional function is shown in Fig. 3.4.

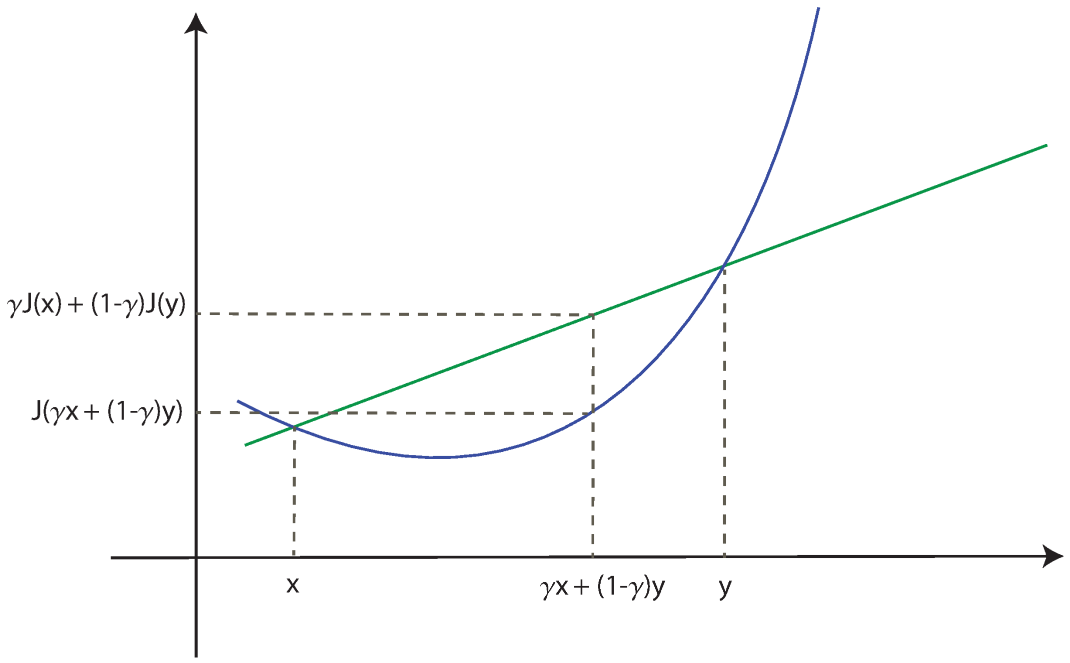


Figure 3.4: Illustration of convexity for a one-dimensional functional.

Convexity is a desirable quality in a functional because of its associated favorable properties for optimization purposes. First of all, a strictly convex function contains a single, global minimum. This greatly increases the odds of success for gradient-based algorithms in particular. Secondly, a strictly convex functional has a positive-definite Hessian. This describes the quality that for any non-zero vector **f**,

$$\mathbf{f}^T B\mathbf{f} > 0, \tag{3.5}$$

where $B$ is an $n$ x $n$ matrix. In physical terms, a positive-definite Hessian form corresponds to a functional that is shaped like a paraboloid-bowl. This is in contrast to functionals with semi-definite or indefinite Hessians, which in the former case contain multiple solutions, and in the latter, produce saddlepoints. Examples of these three functional types are shown in Fig. 3.5. It is clear by examination that the functionals corresponding to the semi-definite and indefinite matrix types will not be amenable to optimization methods. These examples are extremes, but serve to illustrate the ideal and non-ideal optimization surfaces from a gradient-based optimization standpoint.

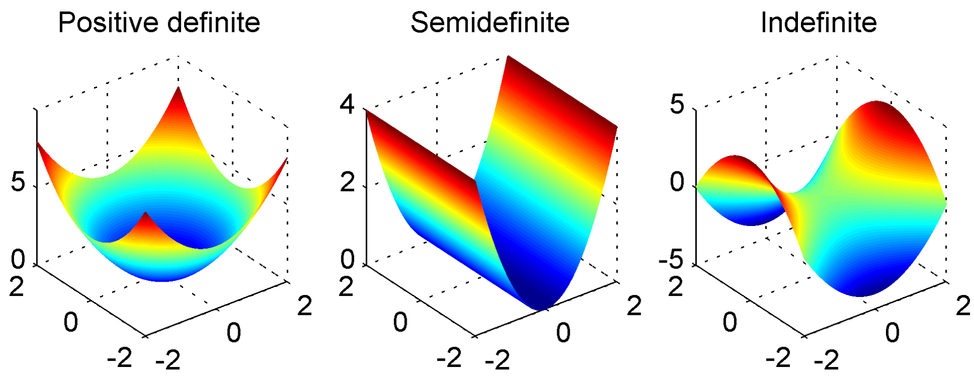


Figure 3.5: Examples of positive-definite, semi-definite, and indefinite functional surface types.

### 3.3.2 Derivative-Free Optimization

*Derivative-free* methods, also known as *direct* or *global* optimization methods, rely on means other than gradient information in order to assess the functional field. Typically this involves defining a population of points in the parameter space at every iteration and directly evaluating the functional at all of these points. A plethora of algorithms, some quite sophisticated, are employed in order to formulate a superior population of points to test at the next iteration based on this sampling of functional data and possibly data points from former iterations as well. Derivative-free methods also allow for methods of optimizing several optimization functionals simultaneously (multi-objective optimization), which can provide a significant benefit over gradient-based methods for some problems. Examples of derivative-free methods include evolutionary algorithms, which are stochastic methods that use elements of evolutionary and genetic theory to determine new search points at each iteration, and the Nelder-Mead method, a heuristic technique applicable for non-linear optimization problems.

Derivative-free optimizers have the advantage of needing only objective function values and are thus suitable for optimization problems in which gradients are unavailable analytically and/or are difficult to estimate. They also are more successful at finding global minima than gradient-based solvers, whose targeted aims may mistakenly converge to a local minimum. However the ‘splatter’ approach leads to a comparatively lengthy iterative process which in large-scale applications can become prohibitive. Additionally, the theory on derivative-free based solvers is less developed than that of gradient-based methods and as a consequence these methods are frequently less robust.

Evolutionary algorithms will be discussed further in Chapter V. The remainder of this chapter focuses on gradient-based optimization methods, which are explored first for the source inversion problem due to their comparative efficiency, a significant benefit from the standpoint of developing rapid method of source inversion for the automatic detection and control algorithm.

## 3.4 Gradient-Based Algorithms

The gradient-based algorithms can be further categorized as steepest descent type methods, Conjugate-Gradient type methods and Newton type methods. These classifications and their individual advantages and drawbacks will briefly be discussed in the following subsections, with a focus on the particular methods that are used in this research. For more in-depth background on gradient-based optimization methods in general, the reader is referred to the text by Nocedal and Wright (2006) .

As stated above, the goal of gradient-based algorithms is to use gradient information with regard to the functional and with respect to the unknown parameters in $\mathbf{f}$ in order to formulate the next iterative guess for the solution. Steepest descent, Newton and conjugate gradient approaches all are examples of *line search* algorithms and hence follow a similar procedure in order to achieve this. First, a search direction is defined which has the general form

$$p_k = -B_k^{-1} \nabla J_k \tag{3.6}$$

The matrix $B_k$ takes different forms depending on the algorithm used and will be further defined in the following subsections. The next value of $\mathbf{f}$ is chosen by moving in the search direction with a step size of $\alpha$ :

$$\mathbf{f}_{k+1} = \mathbf{f}_k + \alpha p_k \tag{3.7}$$

The question is then how to choose the value of $\alpha$. A step too large risks overshooting the minimum, while a step too small will decrease the efficiency of the algorithm. Line search methods approach this issue by formulating and solving a one-dimensional minimization problem for the optimal $\alpha$:

$$\min_{\alpha>0} J(\mathbf{f}_k + \alpha p_k) \tag{3.8}$$

In practice, Eqn. (3.8) is not solved exactly, as to do so would require too much computational time at each iteration. Instead, it is preferable to solve Eqn. (3.8) approximately by requiring only that $\alpha$ meet some predetermined conditions that result in a sufficient reduction in $J$ necessary to achieve convergence to the minimum of the functional. Typically the algorithm will test a series of possible $\alpha$ values and choose the first one of these that meets the stopping criteria. A thorough discussion of line search methods is beyond the scope of this work, and the reader is again referred to Nocedal and Wright (2006).

After determining an adequate $\alpha$, the solution is updated with Eqn. (3.7). The algorithm then checks for convergence, and if the convergence criteria has not been met it advances to the next iteration, $k = k+1$. The process repeats until the solution converges or fails to make further progress.

### 3.4.1 Steepest Descent Method

One of the most straight-forward methods, the steepest descent method chooses the search direction directly in the direction of the negative gradient. In this case the matrix $B_k$ in Eqn. (3.6) is simply the identity matrix and each new search direction is orthogonal to the last. However the step-size can quickly become very small using this approach, and while highly accurate the algorithm can be exceedingly slow to converge on complex problems. Steepest descent methods perform particularly poorly on functions that have a long, shallow descent to the minimum. In light of these deficiencies, more sophisticated methods of choosing $B_k$ are typically used in optimization applications.

### 3.4.2 Newton Type Methods

Newton type methods are similar to the steepest descent algorithm but significantly improve on the convergence rate by deriving the search direction from a second order Taylor series expansion about $J(\mathbf{f}_k + p)$:

$$J(\mathbf{f}_k + p) \approx J_k + p^T \nabla J_k + \frac{1}{2} p^T \nabla^2 J_k p \overset{def}{=} m_k(p) \tag{3.9}$$

Physically this corresponds to constructing a quadratic function around each $\mathbf{f}_k$ and minimizing this function. The optimal search direction will simply be the $p_k$ that minimizes the approximate function $m_k(p)$. By taking the derivative of $m_k(p)$ and setting it to zero this direction is given as

$$p_k = -(\nabla^2 J_k)^{-1} \nabla J_k \tag{3.10}$$

This equation is equivalent to the standard form used for steepest descent except for the matrix $B_k$ in Eqn. (3.6) is now the Hessian, $\nabla^2 J(\mathbf{f}_k)$.

The Newton method is second order accurate and converges much more quickly than the steepest descent method. However it also requires computation of the Hessian $\nabla^2 J(\mathbf{f})$ at each iteration. Since Hessian calculations are expensive to perform, a related algorithm has been developed that uses an approximated Hessian as $B_k$ that is updated at each iteration via a low-rank Hessian formula. This is known as the quasi-Newton method. Quasi-Newton methods are typically a good compromise between the much slower steepest descent algorithm and the more computationally intensive full Newton methods. To derive this approximation, we again start with a Taylor expansion about $J(\mathbf{f}) + p$:

$$\nabla J(\mathbf{f} + p) = \nabla J(\mathbf{f}) + \nabla^2 J(\mathbf{f}) p + O(\| p \|), \tag{3.11}$$

where it can be assumed that the error is on the order of the magnitude of $p$ for continuous functionals. Substituting $\mathbf{f}_k$ for $\mathbf{f}$ and $\mathbf{f}_{k+1} - \mathbf{f}_k$ for $p$, we arrive at

$$\nabla J_{k+1} = \nabla J_k + \nabla^2 J_k (\mathbf{f}_{k+1} - \mathbf{f}_k) + O(\| \mathbf{f}_{k+1} - \mathbf{f}_k \|), \tag{3.12}$$

It can be seen from Eqn. (3.12) that as long as $\mathbf{f}_k$ and $\mathbf{f}_{k+1}$ are in the region near the actual solution $\mathbf{f}$, the error term will be far less significant than the second order term, and therefore we can write

$$\nabla^2 J_k (\mathbf{f}_{k+1} - \mathbf{f}_k) \approx \nabla J_{k+1} - \nabla J_k, \tag{3.13}$$

This finding is the basis for what is known as the secant equation, in which Eqn. (3.13) is expressed in a form where it is clear we need to approximate $B$ in order to make the following relationship hold:

$$B_{k+1} (\mathbf{f}_{k+1} - \mathbf{f}_k) = \nabla J_{k+1} - \nabla J_k. \tag{3.14}$$

The secant equation provides a method to iteratively construct the approximate Hessian. There is more than one method of estimating and updating the approximate Hessian. This work makes use of the BFGS (*Shanno* (1970)) method, which defines the approximate Hessian as

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \tag{3.15}$$

where $s_k = \mathbf{f}_{k+1} - \mathbf{f}_k$ and $y_k = \nabla J_{k+1} - \nabla J_k$. Eqn. (3.15) can now be substituted into Eqn. (3.6). A quadratic model of the objective function at the current value of $p$ is formulated as:

$$m_k(p) = J_k + \nabla J_k^T p + \frac{1}{2} p^T B_k p \tag{3.16}$$

This quadratic model will be minimized by Eqn. (3.6), whereas Eqn. (3.15) is used as the approximation for $B_k$. The quasi-Newton BFGS algorithm is as follows:

**Algorithm 1:** Quasi-Newton BFGS algorithm

```
Initialization: Define starting point $\mathbf{f}_0$, convergence tolerance $\epsilon$ and inverse
Hessian approximation, $H_0$;
while $||\nabla J_k|| > \epsilon$ do
    Compute the search direction according to Eqn. (3.6);
    Perform approximate line search via Eqn. (3.8) and set $\mathbf{f}_{k+1} = \mathbf{f}_k + \alpha_k p_k$;
    Set $s_k = \mathbf{f}_{k+1} - \mathbf{f}_k$ and $y_k = \nabla J_{k+1} - \nabla J_k$;
    Determine the new approximate Hessian by Eqn. (3.15);
    $k \leftarrow k + 1$;
end
```

### 3.4.3 Conjugate Gradient Method

The final type of gradient-based solver that will be discussed is the conjugate gradient method. To derive this method, we start with the linear conjugate gradient method formulation, which is designed to solve a system of linear equations with the form

$$A\mathbf{f} = b, \tag{3.17}$$

where $A$ is a symmetric, positive definite matrix of size $n$ x $n$. Eqn. (3.17) can be re-formulated as an optimization problem:

$$\underset{\min}{J(\mathbf{f})} \overset{def}{=} \frac{1}{2}\mathbf{f}^T A\mathbf{f} - b^T\mathbf{f}, \tag{3.18}$$

where now the gradient of $J$ is simply the residual of the system given by (3.17):

$$\nabla J(\mathbf{f}) = A\mathbf{f} - b \overset{def}{=} r(\mathbf{f}), \tag{3.19}$$

The advantage of the conjugate gradient method is its reliance on choosing search directions based on vectors that have the property of *conjugacy*. A set of vectors given by $p_0, p_1, ..., p_N$ that is conjugate with respect to $A$ has the property

$$p_i^T A p_j = 0, \quad \text{for all } i \neq j. \tag{3.20}$$

By successively minimizing $J$ along one-dimensional search directions in a conjugate set, the conjugate gradient method convergences efficiently. Further, the algorithm only requires the previous vector $p_{k-1}$ in order to determine the current $p_k$, negating the need to store previous conjugate vectors. The linear conjugate gradient algorithm is as follows:

**Algorithm 2:** Linear Conjugate Gradient algorithm

```
Given $\mathbf{f}_0$, set $r_0 \leftarrow A\mathbf{f}_0 - b$, $p_0 \leftarrow -r_0$, $k \leftarrow 0$;
while $r_k \neq 0$ do
    $\alpha_k \leftarrow \frac{r_k^T r_k}{p_k^T A p_k}$;
    $\mathbf{f}_{k+1} \leftarrow \mathbf{f}_k + \alpha_k p_k$;
    $r_{k+1} \leftarrow r_k + \alpha_k A p_k$;
    $\beta_{k+1} \leftarrow \frac{r_{k+1} r_{k+1}}{r_k^T r_k}$;
    $p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k$;
    $k \leftarrow k + 1$;
end
```

Fletcher and Reeves extended this method to nonlinear functions by making two adjustments to the above algorithm. First, in order to estimate $\alpha$ with a nonlinear functional, a line search method must be implemented that finds an approximate minimum to $J$ along $p_k$. Additionally, the gradient that previously was given by the residual $r$ needs to be replaced with the gradient of the nonlinear objective function. With these amendments, the algorithm becomes:

**Algorithm 3:** Non-linear Fletcher-Reeves algorithm

```
Given $\mathbf{f}_0$: evaluate $J_0 = J(\mathbf{f}_0)$, $\nabla J_0 = \nabla J(\mathbf{f}_0)$;
Set $p_0 \leftarrow -\nabla J_0$, $k \leftarrow 0$;
while $\nabla J_k \neq 0$ do
    Compute $\alpha_k$ and set $\mathbf{f}_{k+1} = \mathbf{f}_k + \alpha_k p_k$;
    Evaluate $\nabla J_{k+1}$;
    $\beta_{k+1}^{FR} \leftarrow \frac{\nabla J_{k+1}^T \nabla J_{k+1}}{\nabla J_k^T \nabla J_k}$;
    $p_{k+1} \leftarrow -\nabla J_{k+1} + \beta_{k+1}^{FR} p_k$;
    $k \leftarrow k + 1$;
end
```

## 3.5 DAKOTA Optimization Toolkit

The DAKOTA (Design Analysis Kit for Optimization and Terascale Applications) toolkit developed by Sandia National Laboratories (*Adams et al.* (2009)) is used in this work to perform the numerical optimization for both the source inversion and boundary control (Chapter VII) problems. The DAKOTA toolkit has the benefit of offering numerous gradient-based and derivative-free optimization algorithms, in addition to providing ancillary software for sensitivity analysis, parameter estimation and uncertainty quantification. The wide range of optimization methods included in the DAKOTA toolkit is one of its chief merits, as it allows for the testing of several different methods with only a few changes to the input files, negating the need to learn a new input file system and command syntax in order to test a new solver. Another benefit to the use of DAKOTA is its ability to be interfaced with external models in a relatively simple, 'loosely-coupled' fashion, as is described below.

DAKOTA is able to provide a range of optimization methods and algorithms by virtue of having integrated several internally and externally developed software packages within its framework. A brief description of the packages employed in this research and their associated algorithms is given here.

- CONMIN

  The CONMIN (CONstrained MINimization) library (*Vanderplaats* (1973)) is freely distributed through NASA and includes gradient-based algorithms specifically designed for constrained and unconstrained minimization problems. The unconstrained method, which is based on the Fletcher-Reeves Conjugate Gradient (FRCG) method, is employed for several of the optimization problems in the following chapters. While it is termed 'unconstrained', this is somewhat misleading, as the method is applicable to problems with simple boundary constraints such as the source inversion problem. While the CONMIN library includes more than one algorithm, for the remainder of this text the term 'CONMIN' will refer specifically to the FRCG CONMIN solver.

- OPT++

  The OPT++ library (*Meza et al.* (2007)) was developed internally by Sandia National Laboratories. It includes several gradient-based methods in addition to an algorithm that employs a derivative-free (parallel direct search) method. The gradient-based methods include several variations of Newton type solvers as well as a conjugate gradient solver. In this work, the OPT++ library is used exclusively for its quasi-Newton solver, which is based on the BFGS method. The convention for the remainder of this work will be to refer to the quasi-Newton OPT++ solver as OPT++ for simplicity.

- SCOLIB

  The SCOLIB library (*Hart* (2007)), until recently referred to as COLINY, includes a variety of derivative-free algorithms that make use of the Common

Optimization Library INterface (COLIN). The SCOLIB library is accessible via a BSC license through the website ACRO (A Common Repository for Optimizers). This library is used herein exclusively for its evolutionary algorithm, COLINY-EA, which will be further described in Chapter V.

### 3.5.1 OpenFOAM-DAKOTA Interfacing

The optimizer interfacing with OpenFOAM is achieved via a loose-coupling between the two models, in which the required optimization information obtained from OpenFOAM is passed to DAKOTA's optimizer and the optimizer in turn passes new parameter guesses to OpenFOAM at each iteration. A combination of Bash and Python scripts are used to prepare the pertinent files in which to pass the information between DAKOTA and OpenFOAM. The basic algorithm for the OpenFOAM-DAKOTA gradient-based optimization routine begins with an initial input file which defines several parameters with regards to the optimization problem, including the method or methods of optimization to be applied, the parameters to be optimized for, their associated constraints and the initial guessed values, $\mathbf{f}_0$. The initial guess for the parameters is an important factor in determining the success of the gradient-based optimization routine, as will be addressed in the following chapters. The input parameters are substituted into the files that OpenFOAM reads at the beginning of each simulation, stipulating the initial conditions.

When using a gradient-based optimization method, OpenFOAM makes an initial run to obtain the objective function and then runs an additional $n$ times in order to obtain the values of $\nabla J(\mathbf{f})$, where $n$ is the length of $\mathbf{f}$. The format of the output file from the OpenFOAM runs consists of the objective function value and the numerical gradients with respect to each of the unknown parameters. This information is used by DAKOTA to test for convergence and to formulate a new guess for $\mathbf{f}$ in the case that the convergence criteria is not met. This procedure is visually depicted in the flowchart shown in Fig. 3.6.

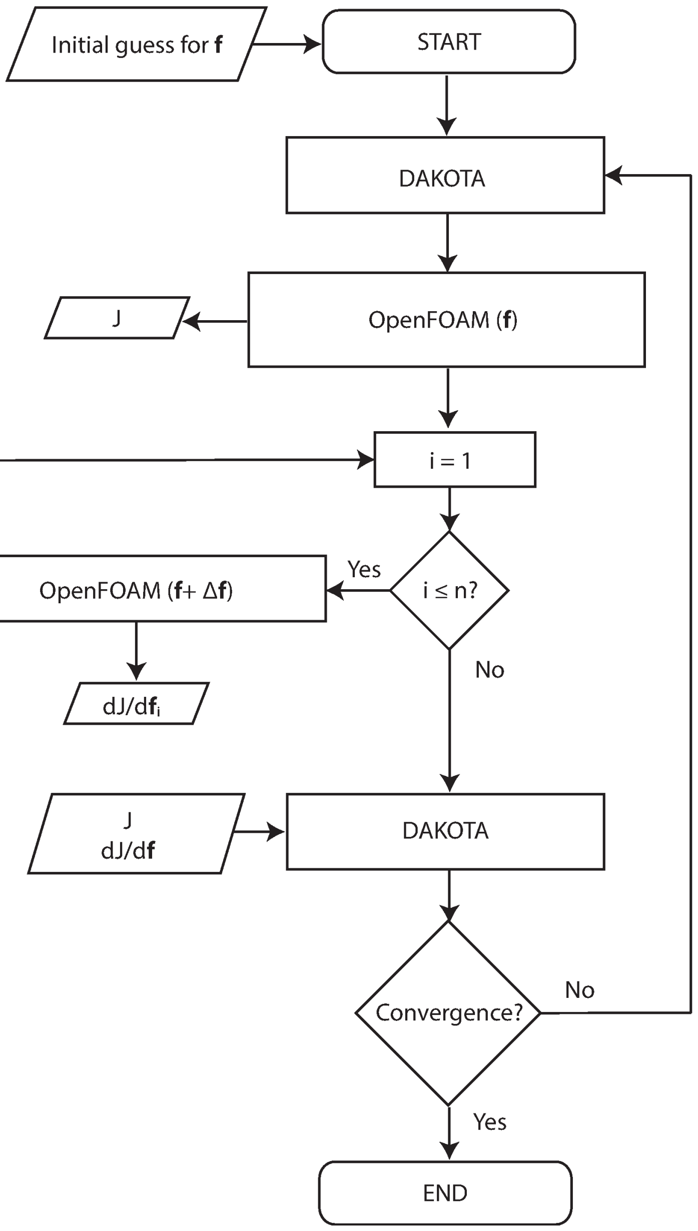


Figure 3.6: DAKOTA-OpenFOAM iterative loop.

The algorithm for interfacing a derivative-free optimization method in DAKOTA with OpenFOAM is similar to process delineated in Fig. 3.6, with the main difference being that OpenFOAM is only run once for each guessed $\mathbf{f}$, as no gradient information is required in this case.

### 3.5.2 Gradient Approximations

Gradient-based optimizers require gradient approximations at each iteration for the current $\mathbf{f}$. When the objective function is differentiable these gradients can be calculated exactly from the analytical function. However for most realistic optimization problems, including the source inversion problem, the analytical gradients are not available. In order to obtain gradients on a finite volume grid the algorithm must estimate the gradients using discrete cell-centered values. Two popular methods of numerically calculating the gradients are the forward and centered differencing formulas. These formulas are given as:

$$\nabla J(\mathbf{f}) \cong \frac{J(\mathbf{f} + h\mathbf{e}_i) - J(\mathbf{f})}{h} \tag{3.21}$$

$$\nabla J(\mathbf{f}) \cong \frac{J(\mathbf{f} + h\mathbf{e}_i) - J(\mathbf{f} - h\mathbf{e}_i)}{2h} \tag{3.22}$$

where $\mathbf{e}_i$ is the $i_{th}$ unit vector and the step size $h$ is chosen so that variables are kept within their respective bounds. In this work, the gradients are estimated in the spatial dimensions using a step size $h$ that is defined as the grid spacing in the respective spatial directions. That is, $h = \mathbf{ds}$, where $\mathbf{ds}$ is the distance $\Delta x$, $\Delta y$ or $\Delta z$ from one cell center to the adjacent cell center in the $x$, $y$ and $z$ dimensions, respectively. The step sizes used with respect to other possible source inversion parameters are defined in the examples provided in Chapter VI. The approximate Hessian, required for the quasi-Newton solver, is similarly estimated according to Eqn. (3.15).

Centered differencing is second-order accurate, an improvement over the first-order accuracy provided by forward differencing. However the centered method also requires an additional run of the forward model at each iteration as it requires the functional evaluation at three points versus two. Thus in this work forward differencing is employed because of its simplicity and relative expedience.

An additional complication of optimization applied to the finite volume grid is that the above gradient estimations assume a regular grid. In two dimensions, this means that points evaluated in Eqn. (3.21) in the x- and y-directions are assumed to be directly to the east and north of the current point, respectively. This is a limitation of the source inversion routine that could be improved in future versions by implementing a more sophisticated method of approximating the gradients that allows for non-regular cell distributions.

## 3.6 Limitations of Gradient-based Optimization for Source Inversion Problems

As was mentioned in Section 3.3, the performance of gradient-based optimization methods is highly dependent on the functional behavior. Functionals that lack the properties of convexity and continuity are challenging for gradient-based optimization methods at the very least, and can lead to complete optimization failure at worst. In particular, the presence of local minima, saddle-points and other complexities present challenges to obtaining an accurate and efficient solution. This makes gradient-based methods sensitive to noise in the objective function. In the case of source inversion, the degree of complexity the functional exhibits is determined by the number and location of sensors as well as the flow parameters. For example, the local maxima that exist due to the sensor locations and Eqn. (3.3) are one factor that affect the idealness of the objective function topography. An additional factor is due to the effect of the flow parameters and location of the source. This is because the shape of the functional is highly dependent on the flow velocity, diffusivity/turbulence characteristics, and the location of the source relative to the sensors. In some cases, the functional may not have a clearly defined, global minimum, as illustrated in the functional shown in Fig. 3.7. The parameters in this case are a continuous source with $\dot{q} = 20,000$

units/s, D=0.01 m$^2$/s and $U_{max} = 2$ m/s. In this example the objective function greatly deviates from the ideal convex shape and instead contains a narrow, flat region downstream of the sensors. A gradient-based optimizer that is initiated in this region or whose trajectory sends it to this area may get ‘stuck’ and be unable to proceed towards the global minimum. The issues will be demonstrated more fully in the following chapters.

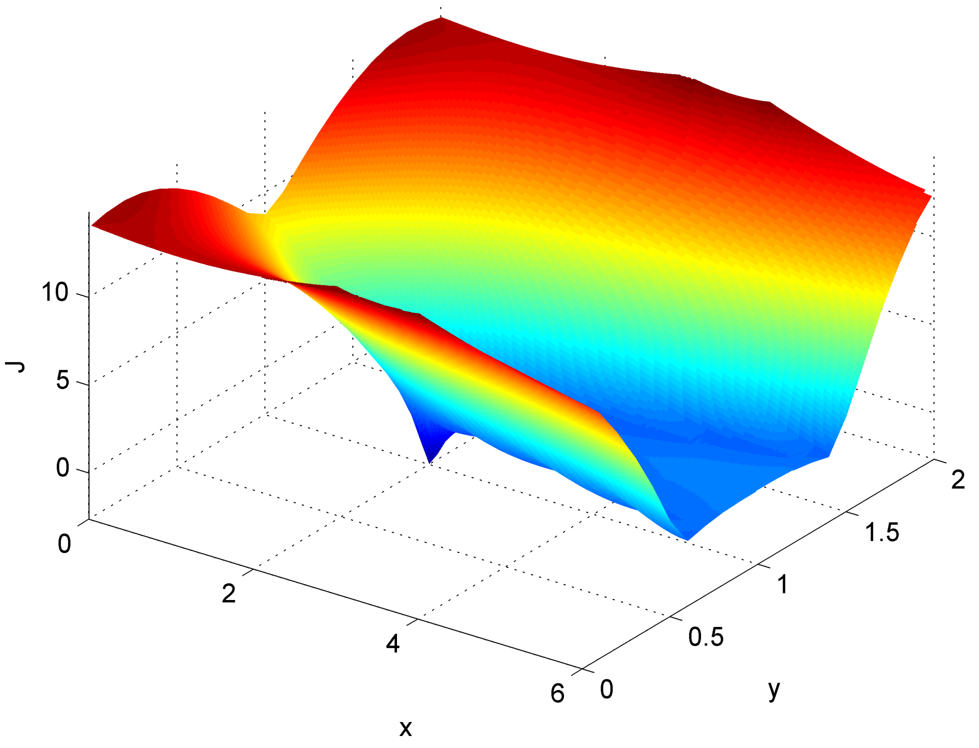


Figure 3.7: An example of a non-ideal functional surface.

### 3.6.1 Discretization Errors

A second limitation arises with the gradient-based methods dues to the discretization of the problem. Technically, the source inversion optimization is not entirely a *discrete optimization* problem, defined as an optimization problem with unknown variables that are restricted to the integer set of values. However, the possible values for the spatial and temporal variables in the CFD model are defined at discrete intervals. For example, the functional is defined spatially only at the centers of the finite volume cells, and therefore there are only a discrete number of possible solutions given by the cell centers. Similarly in the time dimension, the possible values for the release time $t_s$ are confined to the intervals defined by $\Delta t$. This situation is in contrast to a *continuous optimization* problem in which the unknown variables may take on any value within the bounds and/or constraints of the problem, such as the case for optimizing without gridded values or optimizing the source strength $q_s$, which can assume any value within the CFD model.

Discrete optimization problems are less tractable than continuous problems because the restriction of evaluating the functional at specific set of points only inevitably leads to a loss of information. Depending on the nature of the functional, the behavior at one point may not be an accurate estimate of its behavior at a nearby point. Thus, errors arise due to the incongruity between the CFD model test points and the test points requested by the optimizer. This is because the numerical optimizer treats the coordinates as continuous variables, and hence returns new guesses at the most appropriate points as determined by the optimization algorithm. These coordinates will likely fall somewhere between the cell centers, and therefore the nearest cell center must be chosen to represent the optimizer-specified point. As the optimizer expects the new evaluation to take place at exactly the location it has requested instead of the nearest cell center, a discrepancy occurs between the two models.

An illustration of this problem is shown in Fig. 3.8. Here, the initial point for the optimizer is given as $(1.8, 0.1)$. This location coincides with a cell center, and thus the two models, OpenFOAM and DAKOTA, are in agreement at the beginning of the optimization process. However at the second iteration a discrepancy between the location the optimizer requests functional evaluation (blue circles) and the actual location where the functional is being evaluated (red circles) by the CFD model arises, and continuous to various degrees at each iteration.

A related problem occurs when the optimizer takes a step size $\alpha$ that is too small

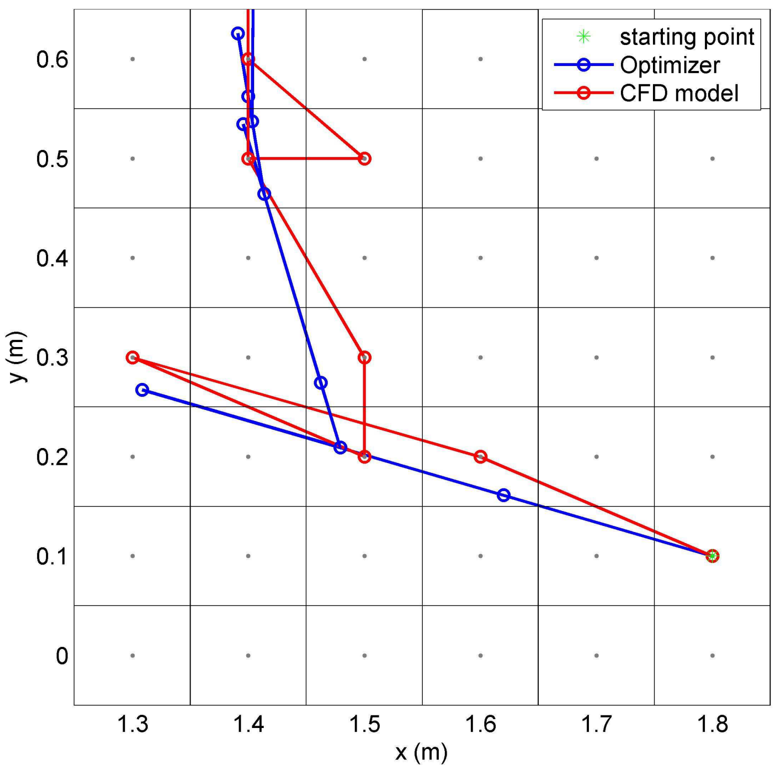


Figure 3.8: Discrepancy between the optimizer and CFD model trajectories.

to fall outside of the former guessed cell's area of convergence, meaning that effectively the new guess is the same as the last and the gradient-based optimizer will be tricked into evaluating the same cell several times in a row. In many cases the combination of the evaluation discrepancy and the step-size issue lead to a premature termination of the optimization routine, by which the minimum is not reached.

In spite of the limitations on the gradient-based optimization, the relative efficiency of this method over derivative-free methods makes it appealing for the real-time detection and control algorithm. To this end, 3 potential work-arounds for the discretization issues are considered:

- Increasing the resolution: This is the most obvious solution, as refining both the spatial and time-steps results in a smaller possible discrepancy between the optimizer requested values and the CFD model evaluation points. However, increasing the resolution to the point where the spatial and temporal intervals are on the same scale as that of the smallest $\alpha's$ would result in an unacceptable length of computational run time. Therefore, alternative solutions to the discretization problems are preferable.

- 'Nudging' of the solution in OpenFOAM: By this method, OpenFOAM checks to see if the solution is getting 'stuck' due to a small $\alpha$ from one iteration to the next. If it is, the test parameter that is repeated at the same finite value as the previous iteration is 'nudged' in the direction of the steepest gradient with respect to that variable and forced to assume the next possible discrete value. This solution is generally most successful when the parameters are very close to the true solution. Otherwise, the disconnect between the actual evaluation parameters in OpenFOAM and the requested evaluation values stipulated by the optimizer becomes larger by this method and causes the algorithm to prematurely terminate. More detail on the nudging procedure is provided in Chapter IV.

- Scaling the variables: DAKOTA allows for automatic or manual scaling of the parameters in the optimization phase in order to adjust two or more parameters to a common scale. Scaling the parameters has the benefit of affecting the step-sizes determined by the optimizer, and thus can be used to help prevent the case of optimization steps that are too small for the grid resolution. This can be a successful method of preventing the problem of the solution getting stuck, however it still does not account for the discrepancy between the model and optimizer tested values, and in many cases requires manually scaling vs. automatic in order to be successful. From the standpoint of developing a robust source inversion algorithm, this trial-and-error scaling process is not desirable, as in a realistic setting without a priori knowledge of the functional, manually setting optimization parameters on a case-by-case basis is not feasible. Scaling and its effects on the optimization process will be discussed further in Chapters IV and VI.

The source inversion and optimization concepts introduced above will be expanded on in the following chapters, and examples of the discretization problems will be presented for both analytical optimization and source inversion functionals.

## 3.7 Ill-Posedness and Source Regularization

As mentioned in Section 3.1, a regularization term is included in the source inversion functional given by Eqn. (3.3). This term is included as a means of dealing with the ill-posedness that is frequently encountered in source inversion problems. A problem is deemed 'ill-posed' on the basis of failing to meet the conditions of existence, uniqueness and stability. That is, a solution must exist, that solution must be the only possible solution given the inputs, and that solution must be repeatable even with small perturbations in the initial conditions. Regarding source inversion, the physical assumption made that there is a plume guarantees existence, however the latter of these two criteria are not assured. The sparse data points lead to the possibility of other solutions which are consistent with the observations. Additionally the solution may be sensitive to noise in the data, leading to instability. Although in this work the sensors are assumed to give perfect measurements free of any noise, round-off errors inevitably lead to the introduction of small errors in the data.

The purpose of regularization is to incorporate further information into the functional about the desired solution, in order to ensure a stable and accurate result. The regularization term itself can be considered a 'stabilizing' term added to the objective function in order to provide a well-posed solution. This term may take a variety of forms, depending on the goal of the regularization and the availability of a priori information. We shall consider regularization of the Tikhonov type (Tikhonov and

Arsenin, 1977). The general form of a Tikhonov regularized functional is as follows

$$\underset{min}{J} = \| \mathbf{s}(\mathbf{f}) - \mathbf{s}^* \| + \sigma \| \mathbf{Ls} \|^2 \tag{3.23}$$

where $\mathbf{s}$ is the solution which is a function of $\mathbf{f}$, $\mathbf{s}^*$ is the matrix of data measurements, $\mathbf{L}$ is known as the derivative operator, $\sigma$ is the regularization parameter and the $\| \cdot \|$ operator is the Euclidean norm. The first term on the right corresponds to the least squares functional as defined by the first term on the right of Eqn. (3.3). A frequently used choice for $\| \mathbf{Ls} \|^2$ is

$$\| \mathbf{Ls} \|^2 = \int_0^T \int_\Omega (\nabla^n \mathbf{s})^2 \, d\Omega dt \tag{3.24}$$

where $n$ represents the order of regularization. The choice of $n = 0$ minimizes the magnitude of $\mathbf{s}$, while $n = 1$ minimizes the magnitude of the spatial changes in $\mathbf{s}$ and $n = 2$ minimizes the oscillations in $\mathbf{s}$ and therefore serves to smooth the functional. Previous researchers (Weese 1992, Provencher 1982a, Neuman and de Marsily 1976) have found that $n = 2$ provides adequate smoothing of the objective function. Akcelik et al. (2003) tested several regularization terms for two-dimensional source inversion based on the advection-diffusion equation. They determined that Tikhonov regularization was adequate as long as the functional was smoothly varying, but that discontinuities in the source required the more sophisticated Total Variation (TV) regularization, at a much greater computational cost. In this Tikhonov regularization is used in order to increase the algorithm efficiency. The regularization term must be discretized in order to be amenable to numerical optimization. Applying the discretization to Eqn. (3.24) and substituting the relevant variables from Eqn. (3.3) we arrive at an expression for the regularization term

$$\mathcal{R}(\mathbf{f}) = \| L\hat{C} \|^2 = \sum_{k=1}^{T} \sum_{m=1}^{N_s} \nabla^2 \hat{C}_m^k(\mathbf{f}) \tag{3.25}$$

The final task before implementing the regularization term is to specify the parameter $\sigma$. This parameter determines the strength of the regularization and is very important in determining the accuracy of the optimization scheme. A value too small will render the solution vulnerable to small errors in the measurements, while a value too large will overpower the signal and yield a functional that is artificially smooth and lacks a well-defined minimum. Several methods of calculating $\sigma$ have been proposed, and the reader is referred to the text by Vogel (2002) for a thorough background. Some of these methods require detailed knowledge of error statistics and hence require a priori knowledge of the data. Another popular approach is the L-curve method, by which $\sigma$ is determined by plotting the log of the squared norm of the regularized solution versus the squared norm of the regularized residual over a range of $\sigma$ values. This plot should have an L-shape, and the optimal regularization parameter is taken as the $\sigma$ that corresponds to the point of highest curvature.

In this work, the source inversion simulations use a value of $\sigma = 10^{-5}$, which is experimentally found to result in an acceptable and beneficial amount of smoothing to the functionals without drowning out the defining characteristics. This constant value is used for the source inversion problems presented in the following chapters unless otherwise specified. The rationale for using a small, constant $\sigma$ is in regards to one of the overarching goals of this research, which is to determine the feasibility of a real-time automatic detection and feedback algorithm. To this end, tuning this parameter to each functional is not a realistic or desirable endeavor. Thus, while a carefully chosen $\sigma$ based on the specific source inversion problem will increase the overall accuracy of the optimization process, using a constant, conservatively small value of this parameter is a more pragmatic and feasible approach.

As a final note on regularization, it must be stated that regularization guarantees that a solution can be found to a given problem, however it does not say anything about the accuracy of that solution *Skaggs and Kabala* (1998). Indeed, this is a significant limitation to the source inversion process in general, particularly for inversion problems that have several unknown parameters that result in an ill-posed functional. In this case, Tikhonov regularization may not be beneficial, and without further information on the true solution the source inversion problem may not be tractable. This is an issue that will be further discussed in Chapter VI.

## 3.8 Summary

Source inversion is a method of recovering the original parameters of a contaminant release based on a sparse number of data measurements, while numerical optimization is a method of approaching source inversion problems by which an efficient algorithm is sought to perform the inversion process. Optimization methods range from the more focused, efficient gradient-based approaches to the comparatively comprehensive yet time-consuming derivative-free methods. Both approaches have their respective pros and cons, and within each category there are several further subcategories with their own respective strengths and weaknesses. Because the reliability of a particular algorithm can vary widely with reference to the problem it is applied to, it is important to investigate the most appropriate solver for a given problem. The optimization toolkit employed in this research, DAKOTA, allows for a variety of algorithms to be tested with relative ease and further, is able to be interfaced with the CFD model with a minimal amount of scripting, providing a powerful method of modeling the source inversion problem.

Several obstacles to the numerical optimization process have been presented in this chapter. These obstacles arise from the discretization of the forward problem, which restricts the available functional evaluation to a set of discrete points in the spatial and temporal dimensions, in addition to the necessity of numerically approximating the gradients. Possible solutions to some of these issues were discussed, with the conclusion that while there are no easy fixes for these issues, there are a couple of methods of finessing the respective optimization and CFD models in order to increase the accuracy of the optimization process. These topics will be revisited in the following chapters with examples given for both analytical test functions (Chapter IV) and source inversion functionals (Chapters V and VI).

Finally, the subject of regularization was introduced in this chapter and the form of regularization used in the source inversion functional in this work, Tikhonov regularization, was defined. The choice of the regularization parameter is an important one, as an inappropriate value can either be too large and distort the functional to a degree that it is no longer representative of the problem or be too small and fail to achieve its intended purpose. However the optimal value of the parameter $\sigma$ is dependent on the particular problem, and in order to discover this value typically statistical methods involving a priori knowledge of the data is required. This is disadvantage for the automatic detection and mitigation problem, in which we seek to develop a robust method of source inversion that does not require ‘tuning’ to fit a specific set of (unknown) parameters. Thus, a small value of 1 x $10^{-5}$ is chosen for this parameter, which is experimentally found to adequately smooth the majority of the tested functional surfaces without ‘drowning out’ the relevant functional features.

CHAPTER IV

# Analytical Optimization Test Problems

This chapter presents an analysis of two of DAKOTA's gradient-based optimization methods, the Conjugate-Gradient Fletcher-Reeves algorithm (CONMIN) and the quasi-Newton method (OPT++), applied towards two simple analytical functions. The analytical functions, the Sphere function and a simplified version of Ackley's function, approximate ideal 2-dimensional objective function surfaces. First, the method of handling the gradients and the nudging technique used to reduce the problem of 'stuck' trajectories when using discretized parameter values are further described. These methods are then applied to the two test problems, and the optimization convergence errors as a function of the starting guess are assessed for the cases of both continuous and discretized parameters. These examples provide an understanding of the behavior of the tested optimization algorithms when using numerical methods, and illustrate the difficulties encountered in the numerical optimization process even under circumstances where the functionals are ideal.

## 4.1 Evaluating Optimization Performance

As was previously discussed, one of the issues regarding choosing an appropriate optimization method for a given problem is due to the fact that certain optimization methods optimize some problems better than others. It is therefore advantageous to test the performance of more than one algorithm for the problem at hand. Even idealized test functionals such as the functions analyzed in this chapter are more successfully optimized using certain approaches than others. As will be shown in the following chapter, this selective tendency is one of the most significant obstacles to robust source inversion via optimization. This is because in a realistic source inversion problem, the functional characteristics will change in response to variations in the problem formulation. Therefore where one method works well for a given set of parameters and initial starting point, it may fail completely to optimize a functional created from slightly different parameters using the same initial point.

When assessing the performance of optimization algorithms there are several factors to bear in mind. For one, when optimization methods are initially developed, their superiority over similar algorithms is typically demonstrated by testing the optimization algorithm with a small number of test functions and using starting points near to the global minimum. This practice has been viewed with skepticism by some researchers (*More et al.* (1981)) who point out that evaluating optimizer performance based on a few select points near a known minimum, with respect to a few test functions, does not provide an accurate appraisal of an algorithm's effectiveness. In practical optimization problems, not only is the near-minimum region frequently unknown, but the functional shape may vary considerably from that of the test function's used to verify the optimization algorithm. In other words, an algorithm may be 'tuned' to certain test problems, which results in excellent optimization capabilities for a few select problems and initial points, but less accurate outcomes when variations from these ideal conditions are introduced.

One reason that several functionals and/or various starting points are typically not assessed when testing algorithms is due to the emphasis placed on convergence efficiency over the robustness and reliability of the method. While efficiency can be an important factor, drastic differences in run-times are generally encountered only when comparing vastly different algorithms (i.e. gradient-based vs. derivative-free methods). With modern computing power the differences between similar algorithm efficiencies are often less significant, and are likely less important than the algorithm's accuracy. Thus the focus on efficiency over robustness and reliability is not justified for many applications.

An additional complication in comparing methods stems from the inconsistent optimization parameters included with the respective algorithms. Optimization routines typically include several user-defined parameters which can be adjusted to better fit a given problem. For example, the OPT++ routine in DAKOTA includes controls for the gradient tolerance and maximum step size for gradient calculations. The DAKOTA CONMIN routine however, does not offer analogous controls. Therefore direct comparisons between two methods are not always possible. Also, adjusting the various user-defined controls to suit a given problem is a tedious endeavor, and while the results in this dissertation no doubt could be improved upon by investing in a thorough investigation of the optimal algorithm controls, this is not the goal of this research. Rather, the goal is to evaluate the feasibility of the detection and control algorithm in a realistic setting, in which it would not be possible to adjust the optimization parameters to fit the functional for each particular source inversion scenario. Therefore, the default parameters of the optimizers and uniform automatic scaling approaches are employed herein as much as possible.

With these factors in mind, this work attempts to avoid the pitfalls of many previous works by assessing the performance of the two gradient-based optimization methods with respect to less than ideal conditions and with a variety of functionals in order to more fully characterize their respective robustness, efficiency and limitations. To this end, the two test problems used to validate the DAKOTA-OpenFOAM interface are tested for convergence accuracy using a wide range of points within the domain boundaries as the initial starting guess. Further, the effect of the numerical gradient calculations and the discretized parameter values is assessed independently.

## 4.2 Gradient Calculations

DAKOTA accepts either externally provided or internally calculated gradients for its gradient-based algorithms. Externally supplied gradients can be obtained either analytically or numerically by the user and passed to DAKOTA through the loosely-coupled interface previous described in Chapter III. Alternatively, when internal gradients are specified, DAKOTA uses one of its available custom method of approximating the gradients, which involves automatically perturbing each parameters and re-running the forward model with the perturbed values in a similar process as to the externally numerical gradients are obtained by the user. The main difference between the externally and internally supplied numerical gradients is that while the internal method negates the need to extra coding in order to perturb the input parameters and re-run the forward model, using DAKOTA's internal finite differencing algorithm allows for less user-control in regards to how the finite difference step-size $h$ is determined. This is particularly problematic when there are explicit restrictions on the value $h$ can assume as is the case with the discretized parameters in this work. For example, in both OPT++ and CONMIN, the size of the perturbation is determined by a user-defined step size scaling parameter, $\delta_s$. This value is multiplied by the current parameter value in order to obtain the step size $h$ used in the gradient calculations. A minimum $h$ of $0.01{*}\delta_s$ is imposed in order to handle calculations where the parameter value is close to zero. This scaling method is an example of an optimization algorithm being 'tuned' to a specific type of problem: scaling the finite difference step-size by the parameter values works excellently for the case of a function that is minimized at the origin, such as with both the Sphere function and Ackley's function, and in fact leads to near perfect convergence for all initial starting points when assuming continuous variables. However, this method does not make any sense for the source inversion problem or many other realistic optimization problems. In the source inversion problem the $x$ and $y$ parameters could assume any value within the physical boundaries of the domain and no correlation exists between smaller values of $x$ and $y$ and the proximity to the sought source location $(x_s, y_s)$. Therefore, internal DAKOTA finite differencing is not a viable method in this work and is not used in the following optimization problems. Instead, the gradients must be calculated externally and passed into DAKOTA. To do this, an $h$ is defined for each parameter and OpenFOAM is re-run with each perturbed parameter value in order to calculate the gradients according to the forward-differencing scheme Eqn. (3.21). The choice of the spatial step-size is based on the distance between two adjacent cell centers, which for the uniform grids used for the following examples is $\Delta x = \Delta y = 0.1$. This method of estimating the gradients is not perfect, and problems arising from the finite difference gradient approximations will be discussed in the subsequent sections. Discretizing the parameters adds an additional level of difficulty to the optimization process, for the reasons discussed in Section 3.6.1. This problem is partially mitigated with the nudging method as outlined below.

## 4.3 'Nudging' the the Discretized Parameters

Chapter III introduced the method of 'nudging' the test parameters in OpenFOAM in order to help mitigate the problem of small $\alpha$'s leading to the optimization trajectory becoming stuck in a given cell. As was discussed, this is a problem in both the spatial and time dimensions. The nudging procedure that was introduced in Section3.6.1 is explicitly defined here for a general parameter $b$ at iteration $k$:

**Algorithm 4:** Nudging procedure

```
if $b_k = b_{k-1}$ and $J > \delta$ then
    Normalize the gradient component of $J$ with respect to $b$: $|\nabla J_b| = \frac{\nabla J_b}{|\nabla J_{\mathbf{f}}|}$;
    Define a scaling step-size $s$ based on the maximum distance to the nearest discrete value $a$;
    Redefine the test parameter value as: $b_k = b_k - s|\nabla J_b|$;
    Locate the nearest discrete value of $b$ in OpenFOAM and define the test parameter as this value;
```

The parameter $\delta$ is a small number that represents a maximum acceptable value for $J$, preventing the algorithm from nudging the parameters in the case that the minimum has been reached. All of the following results for the Sphere function and Ackley function include the nudging fix with the exception of an example of the results without this fix, presented for comparison in Section 4.5. The value of $\delta$ used in these examples is set at 0.001. The step size scaling parameter $s$ is designed to be big enough to 'push' the parameter value out of the current cell (or time-step), without overstepping and testing a point that is very far from the optimizer-requested test parameter. For the case of the spatial parameters $x$ and $y$ it is defined as half of the maximum distance between adjacent cells, which for the uniform grids used in these examples leads to $s = \frac{1}{2}\sqrt{0.1^2 + 0.1^2} = 0.071$.

## 4.4 Sphere Function

The Sphere function, also called the first De Jong's function, is a commonly used benchmark test function in optimization. It is uni-modal, smooth, symmetric, and convex, which makes it one of the most ideal optimization problems. In its simplest form as a function of two variables it is defined as

$$J = x^2 + y^2. \tag{4.1}$$

The global minimum of $J = 0$ is clearly located at the origin. For this example the domain is constrained to $-5 \leq x \leq 5$ and $-5 \leq y \leq 5$ and discretized into 10,000 finite volume cells with $\Delta x = \Delta y = 0.1$. For all optimization runs, forward differencing with a step size $h = 0.1$ is implemented. The convergence tolerance is set at $\epsilon = 1 \text{ x } 10^{-5}$.

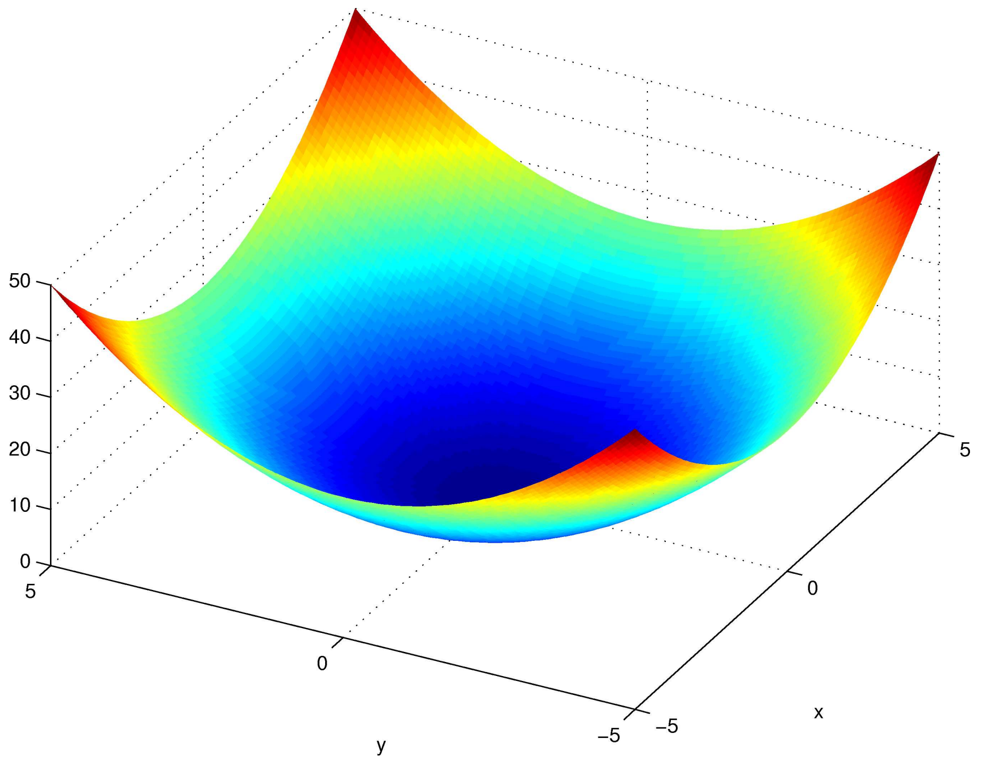


Figure 4.1: Two-dimensional Sphere function.

The performance of CONMIN and OPT++ is compared for Eqn. (4.1). The convergence error $\zeta$ was measured using all possible grid points within the region $-4.9 < x < 4.9$ and $-4.9 < y < 4.9$ as the initial starting point for the optimization trajectory. The cells bordering the domain were not tested, as the forward differencing scheme is undefined along the east and north boundaries of the domain. For symmetry, the south and west boundary cells are also not used as initial trajectory points.

For each of the test optimization methods, results were obtained for two cases:

[1] The parameters $(x, y)$ are continuous variables and may assume any value. In this case the function is evaluated at exactly the parameters guesses requested by the optimizer.

[2] The parameters $(x, y)$ are discretely defined at the finite volume grid points. In this case, the function may only be evaluated at the cell center nearest to the values requested by the optimizer.

For both cases, the optimization algorithms were tested assuming initial trajectory points at each of the grid points, regardless of whether continuous or discrete variables were allowed during the optimization process.

Results of the optimization convergence error as a function of the starting point are shown for Case [1] using OPT++ in Fig. 4.3 and for CONMIN in Fig. 4.4. Via OPT++ or CONMIN, the most accurate results occur when the optimizer is initiated in a region in which both gradients, $\partial J/\partial x$ and $\partial J/\partial y$, are positive as initially assessed by the forward differencing approximation. In the other regions, and particularly for starting points in the shallow 'bowl' region surrounding the minimum, the conjugate gradient method performs poorly compared to the quasi-Newton method.

With either algorithm, it is clear that there are limitations of the optimization process for these cases and necessary to understand why the optimizers struggle to locate the minimum more or less exactly. At this point, it should be reiterated that when minimizing the Sphere function using DAKOTA's forward finite differencing scheme and continuous variables the results are nearly perfect, regardless of the starting point for the trajectories. The reason for this is again attributed to DAKOTA's internal finite differencing method, which automatically reduces the finite difference step-size as the parameters approach zero, allowing the optimizer to successfully hone in on the true minimum over a series of increasingly small gradient evaluations. However, as previously stated, it is necessary to take a different approach in this work to allow for the optimization of functionals in which there is no correlation between the magnitude of the parameter values and the proximity to the minimum. Thus for the spatial gradients evaluated on a finite volume grid, an unscaled value of the finite differencing step-size must be defined.

A logical value for $h$ is the distance from one cell center to the next, as a value smaller than this will lead to an undefined gradient using cell-centered values ($\mathbf{ds} = 0$) and a step-size larger than this will lead to greater error in the gradient approximation. However, the fact that the gradients can only be evaluated at set points based on the fixed $h$ makes it difficult for many trajectories to converge to the true minimum. The mechanism behind these failures is illustrated in Fig.4.2, which shows a one-dimensional cross-section of the Sphere function in the vicinity of the minimum.

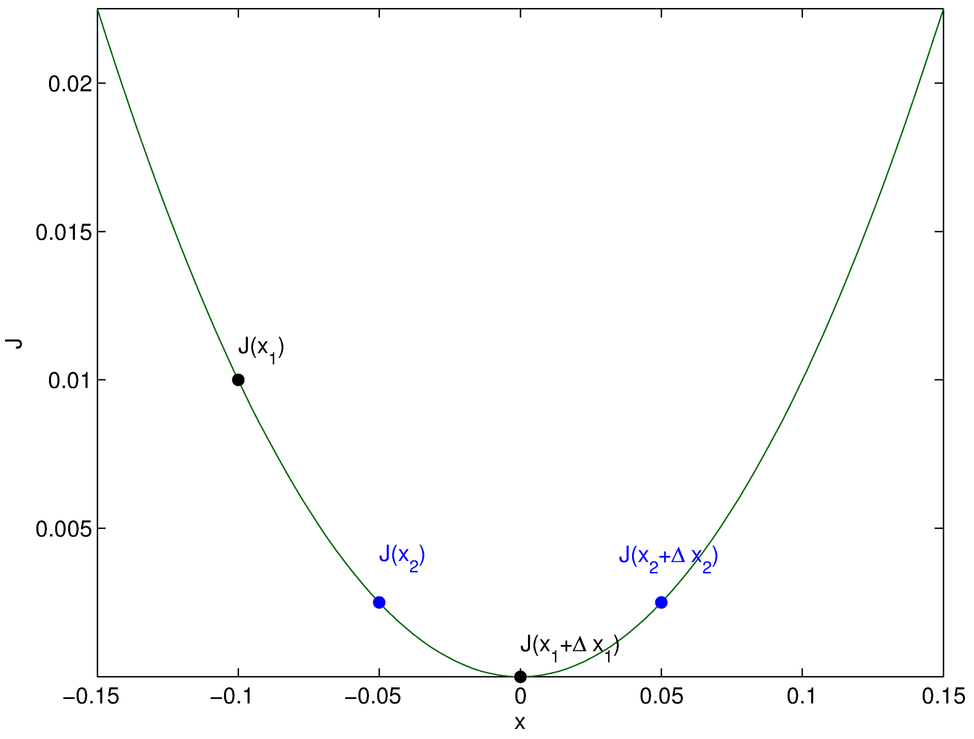


Figure 4.2: Example of errors using a fixed finite differencing step-size.

Because in this case $h$ is set at 0.1 but $x$ is a continuous variable, the current parameter value of $x$ could fall on any point of the x-axis. Let's suppose that at a given iteration the trajectory lands at a point where the current guess for $x$, which we will denote as $x_1$, is in the range $[-0.1, -0.05)$. The numerical gradient calculated via forward differencing for an $x_1$ in this range will have a negative value, since the function evaluated at a distance $h$ from any $x_1$ in this range will have a value less than $J(x_1)$. The negative gradient will send the optimization trajectory in the forward direction by a step-size $\alpha$. Let's suppose $\alpha$ is a relatively small number and consequently sends the trajectory to a point where the new guess for $x$, denoted by $x_2$, is in the range of $(-0.05, 0]$. The forward difference gradient will now be positive, sending the trajectory in the reverse direction. If the value of $\alpha$ at this iteration is similar to the value at the previous iteration, which is likely due to the similar relative magnitudes of the functional value and the gradients at $x_1$ and $x_2$, it is easy to see that the trajectory will end up back at a point that was in the initial range of $x_1$. This leads to a situation where the trajectory will effectively become trapped between these two points and unable to move closer to the true minimum.

Further, it can be seen that if $x_1 = -0.05$, the calculated gradient becomes

$$\frac{f(x+h) - f(x)}{h} = \frac{0.0025 - 0.0025}{0.1} = 0.$$

Therefore as the trajectory makes small steps towards the minimum from the left of $x = -0.05$, the numerical gradient will approach zero as the difference between $J(x_1)$ and $J(x_1 + \Delta x_1)$ gets smaller and smaller due to the symmetry of the function, eventually becoming zero at a point *prior* to the true minimum. Since the forward gradient becomes positive to the right of $x = -0.05$, from the perspective of the optimizer, the minimum occurs *at* $x = -0.05$. This analysis explains why the results for both optimizers are consistently correct when the trajectories begin in the upper right region of the domain, where the gradients are both positive, allowing for the trajectories to avoid this problem when approaching the true minimum. Additionally, it explains why the majority of the OPT++ trajectories converge just outside of the true minimum. The comparatively scattered and varied results of the CONMIN trajectories can be attributed to the differences between the way the algorithms approach the minimum as a function of the starting point.

The OPT++ and CONMIN convergence errors are shown for Case [2] in Figs. 4.5 and 4.6, respectively, where now the combined effects of the numerical gradient calculations and the gridded parameter values can be investigated. Interestingly, the OPT++ optimizer performs better with the discretized parameters, while the CONMIN results are more chaotic and erroneous than the Case [1] results. These findings can again be better understood by examining the behavior of the trajectories in the vicinity of the minimum as shown in Fig. 4.2. Now, $x_1$ can only assume values of $[-4.9 : 0.1 : 4.9]$. Therefore the gradient will frequently end up being assessed directly at the point $x = -0.05$ where it is zero by the forward differencing method with $h = 0.1$. However, due to the nudging technique that was previously described, OpenFOAM recognizes that the objective function is still above the specified minimum threshold $\delta_d$, and when the optimizer tests a slightly perturbed point and the value falls again into the finite volume cell containing cell center $x_c = -0.05$, the nudging procedure will be initiated. This procedure will force the parameter into another cell *if the gradient in the other dimension is also not zero.* That is, trajectories which end up in the cell where $x_c = -0.05$ and $y_c = -0.05$ will not be able to be nudged because their estimated forward gradients will both be 0. However, for trajectories that land in a cell where one gradient is non-zero, they parameters will be nudged in the direction of the negative gradient, which in these cases will help the trajectory converge to the true minimum.

This effect is clearly seen in Fig. 4.5, where there is a distinct pattern between the starting trajectories and the convergence errors. Yet again due to the differences between the algorithms and how they choose their subsequent parameter guesses, the results from CONMIN for the gridded variables are much more scattered and unpredictable and do not exhibit the pattern that is seen with the OPT++ results.

Another factor leading to the higher percentage of exact trajectories in the gridded variable case is simply due to the fact that the exact minimum coincides directly with a cell center. Because of this, the gridded optimization allows for a greater margin of error in $x$ and $y$, as any combination of $x$ and $y$ that falls within the boundaries of the cell containing the minimum will effectively be 'rounded' to the cell center, resulting in a final convergence error of 0.

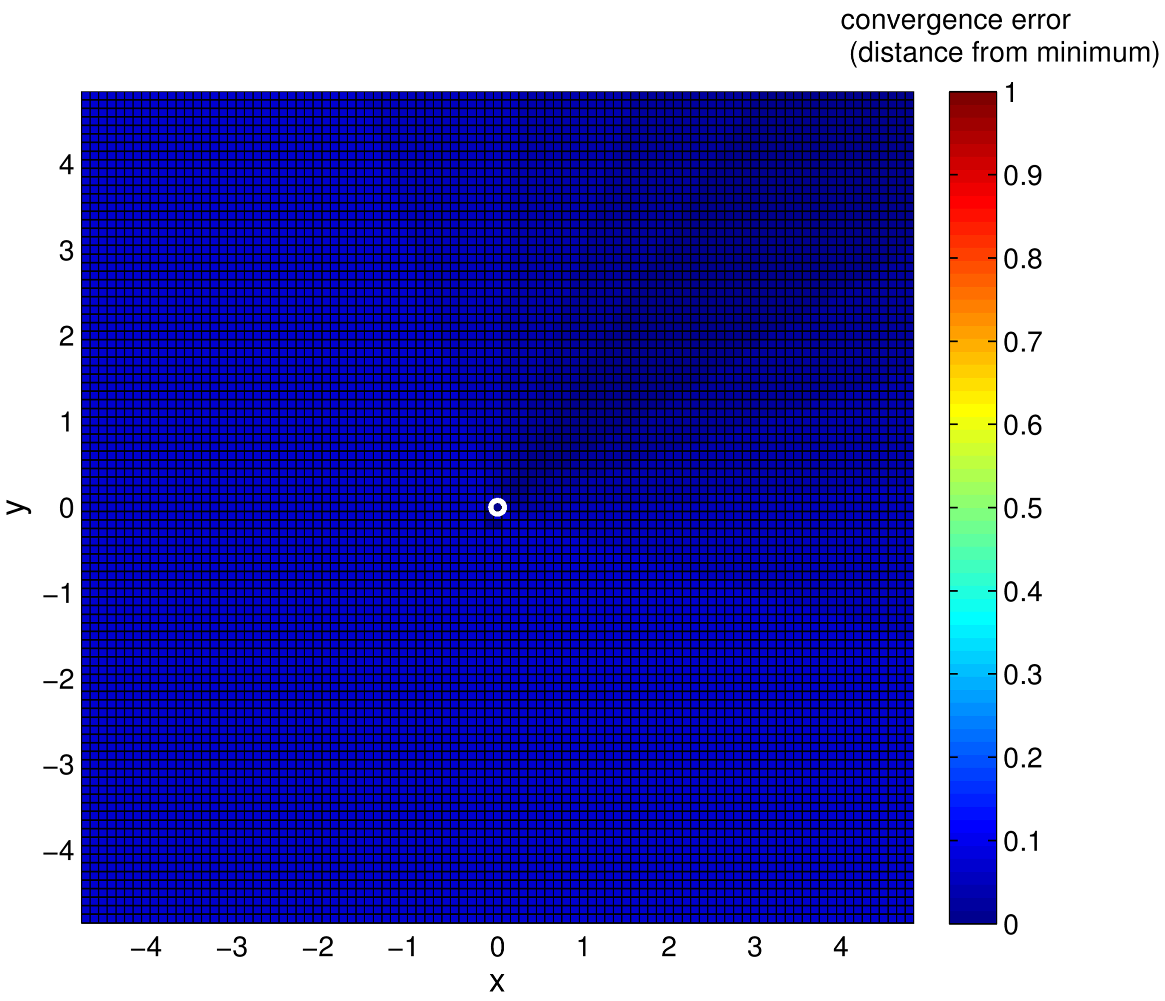


Figure 4.3: Sphere function optimization convergence error with OPT++ using continuous variables.

Summary statistics for the Sphere function tests are shown in Table 4.1. For both

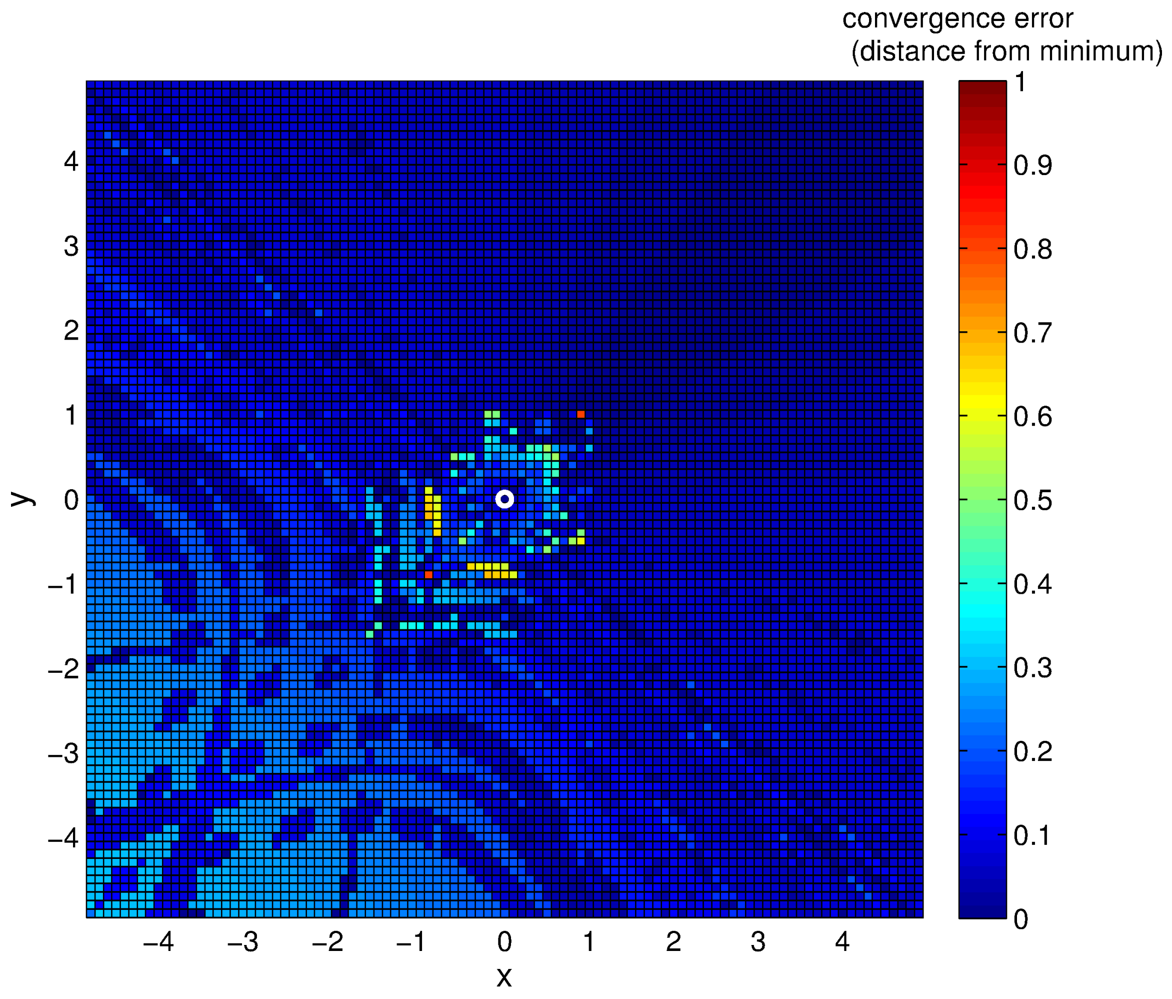


Figure 4.4: Sphere function optimization convergence error with CONMIN using continuous variables.

the continuous and discrete variables, the quasi-Newton algorithm implemented by OPT++ performs consistently more accurately than the conjugate gradient algorithm implemented in CONMIN. While the mean errors were similar for the continuous case, the differences in mean error were more significant in the gridded case. Overall, the percentage of trajectories that resulted in locating the minimum more or less exactly ($\zeta < 1 \times 10^{-5}$) is extremely small for both the Case [1] experiments, reinforcing the fact that the forward numerical gradient method has significant drawbacks even in the case of continuous parameter values.

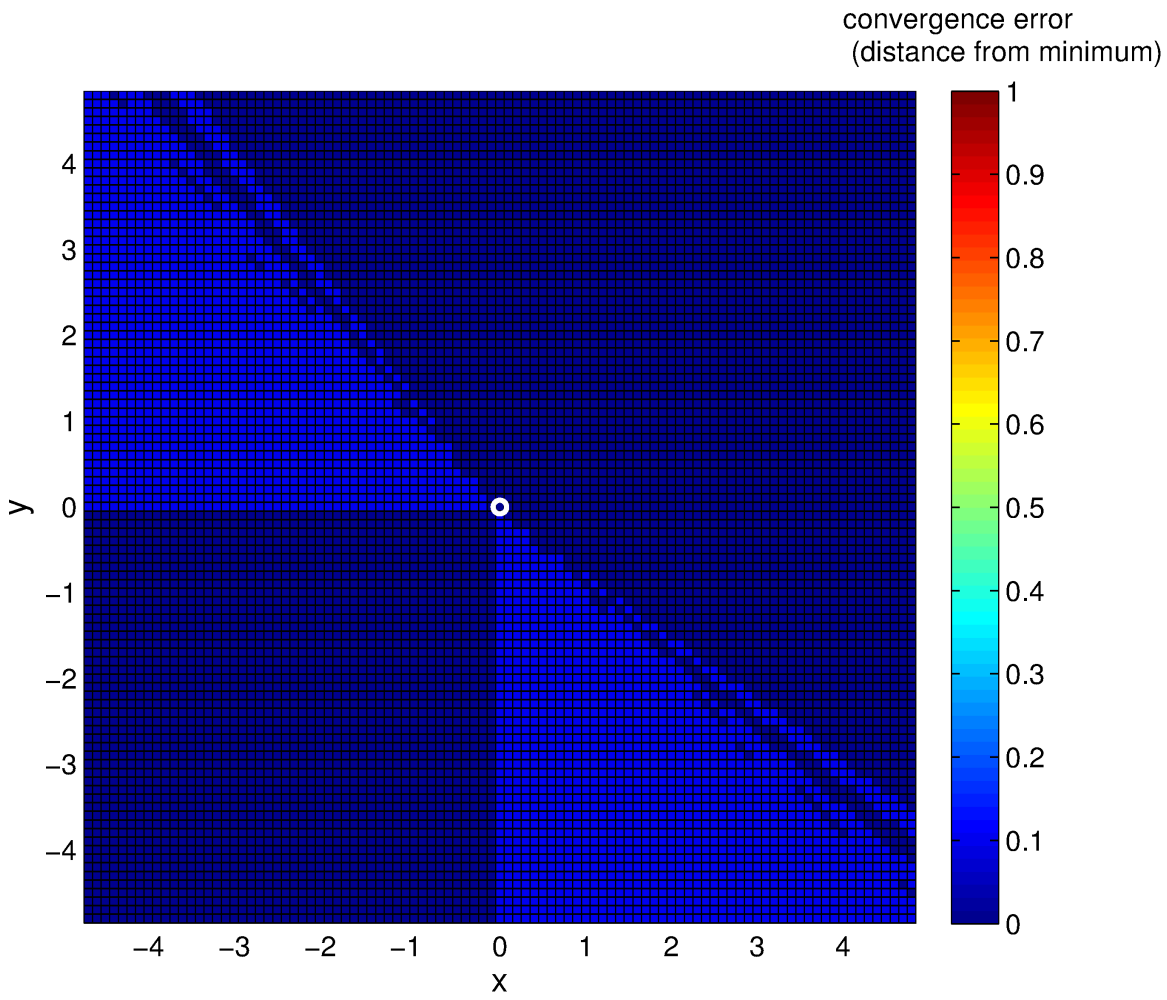


Figure 4.5: Sphere function optimization convergence error with OPT++ on a finite volume grid.

In regards to the computational times, the Case [1] OPT++ method was the fastest, taking only an average of 15 seconds per trajectory vs. 46 seconds for the continuous CONMIN case, using an Intel(R) Xeon(R) CPU 2.40 GHz processor. The mean convergence times for the gridded parameter simulations were roughly the same for either method, 51 and 54 seconds respectively. Note that these times should be used for relative comparisons only, as the overall convergence times could be greatly improved by using a faster processor.

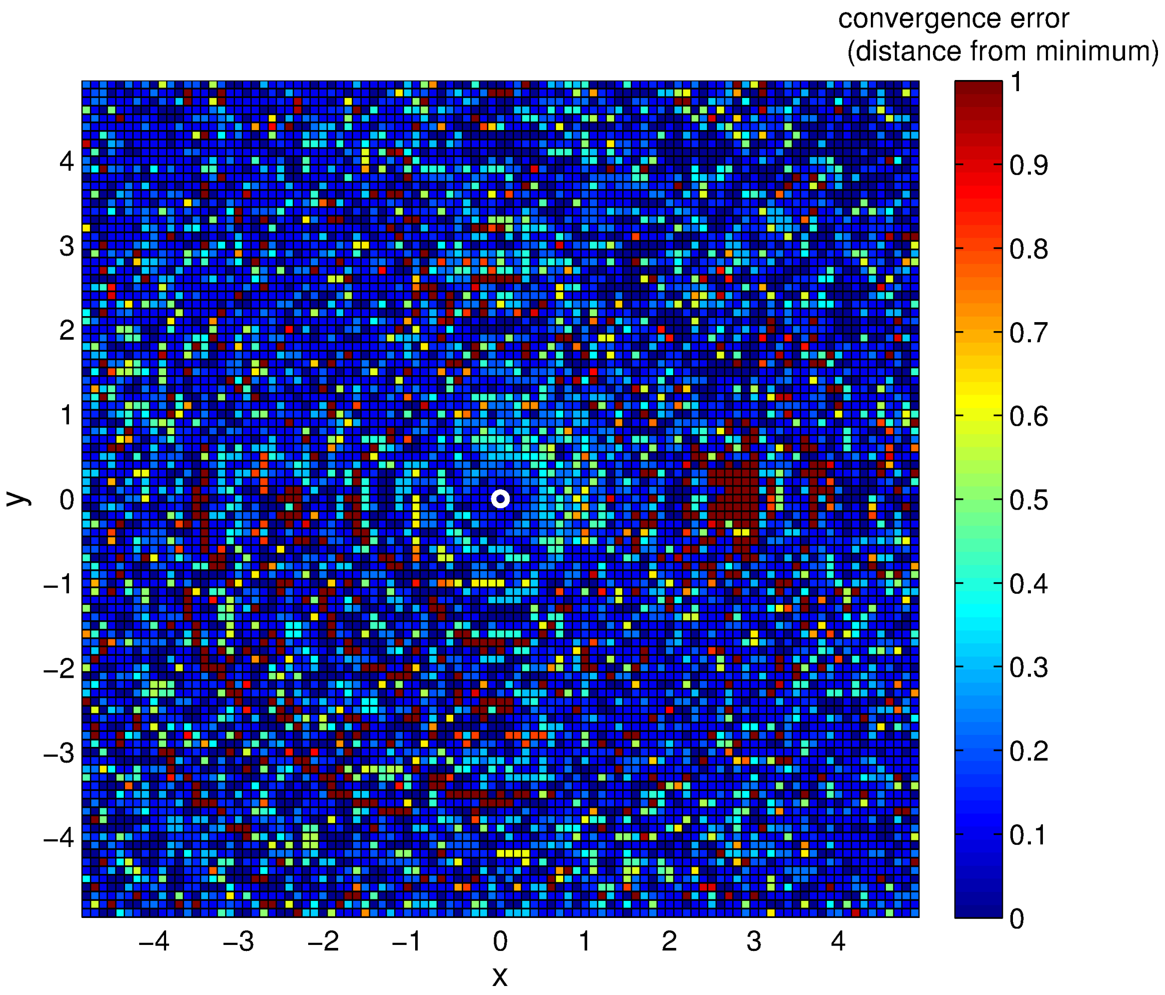


Figure 4.6: Sphere function optimization convergence error with CONMIN on a finite volume grid.

Table 4.1: Sphere function statistics

| Case | Method | $\zeta < 10^{-5}$ | $\zeta < 0.1$ | $\max(\zeta)$ | $\text{mean}(\zeta)$ | mean run-time (s) |
|---|---|---|---|---|---|---|
| Case [1] | OPT++ | 0.02% | 100% | 0.07 | 0.06 | 15 |
| | CONMIN | 0.45% | 74.9% | 0.81 | 0.09 | 46 |
| Case [2] | OPT++ | 70.5% | 100% | 0.1 | 0.03 | 51 |
| | CONMIN | 21.3% | 53.4% | 3.68 | 0.26 | 54 |

## 4.5 Ackley's Function

A second idealized test function, a simplified variation of Ackley's function, is tested in the same manner as the Sphere function. Although Ackley's function can be extended to an infinite number of variables, we shall consider the two-dimensional case:

$$J(x, y) = -a \exp\left[-b\sqrt{\frac{1}{2}(x^2 + y^2)}\right] + \exp\left[\cos(cx) + \cos(cy)\right] + a + \exp(1) \qquad (4.2)$$

This function also has a global minimum at $(0, 0)$, but the inclusion of the exponential term results in numerous local minima, making this a popular multi-modal test function. In this example however, the function is made uni-modal by adjusting the coefficients to smooth the oscillations, as the purpose of the test cases in this chapter is to evaluate the effects of the discretization practices on optimization performance with idealized functions. The coefficients used are $a = 1000$, $b = 0.5$ and $c = 2\pi$. The three-dimensional surface resulting from these coefficient values is shown in Fig. 4.7. The main differences between this function and the Sphere function are the greater overall range of values ($0 \leq J < 1000$ for Ackley's function vs. $0 \leq J < 50$ for the

Sphere function), leading to a greater range of gradients over the entire domain, in addition to the steeper and more concentrated region of descent to the minimum with Ackley's function compared to the more uniformly graded Sphere function.

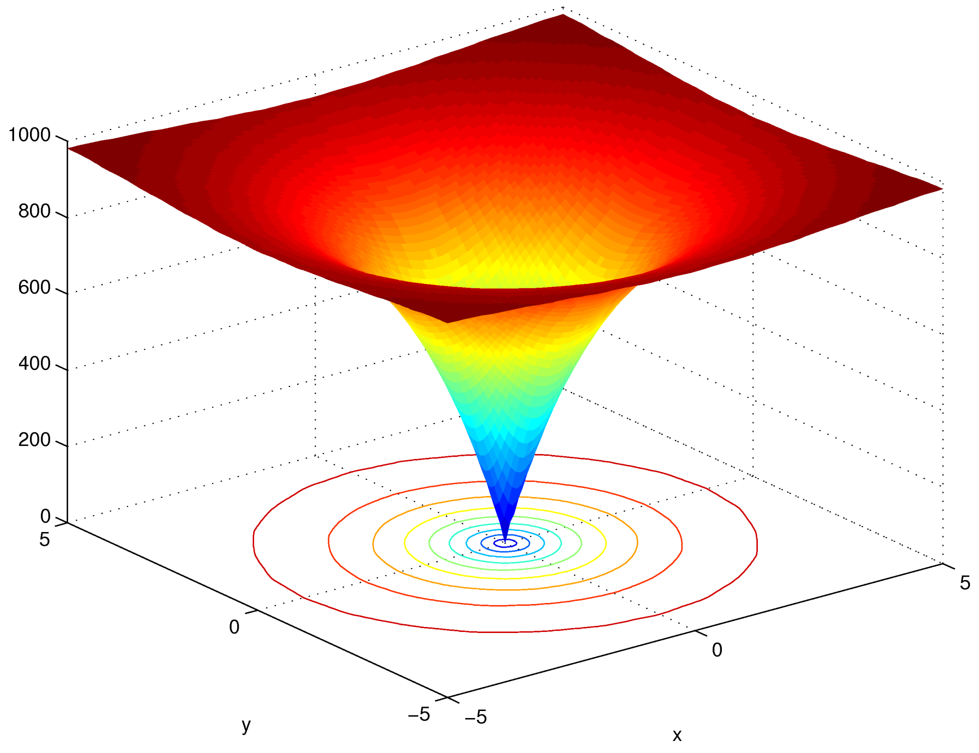


Figure 4.7: Two-dimensional Ackley function.

Again, the performance and robustness of CONMIN and OPT++ were evaluated for this function using both continuous and grid-defined values for $x$ and $y$. The Case [1] results are shown in Figs. 4.8 and 4.9 for the respective solvers. At the majority of starting points (86%), the OPT++ solver is able to converge to within $\zeta < 0.1$, however very few (0.07%) of trajectories find the true minimum. The reasoning for this is again due to the combination of the symmetry of the function, step-size restriction, and forward differencing implemented in the gradient calculations as explained above.

The main difference between these results and the Case [1] results for the Sphere function are in the convergence errors that occur with initial trajectory points in the outer, relatively flat region of the function (Fig. 4.8). In some of these cases the

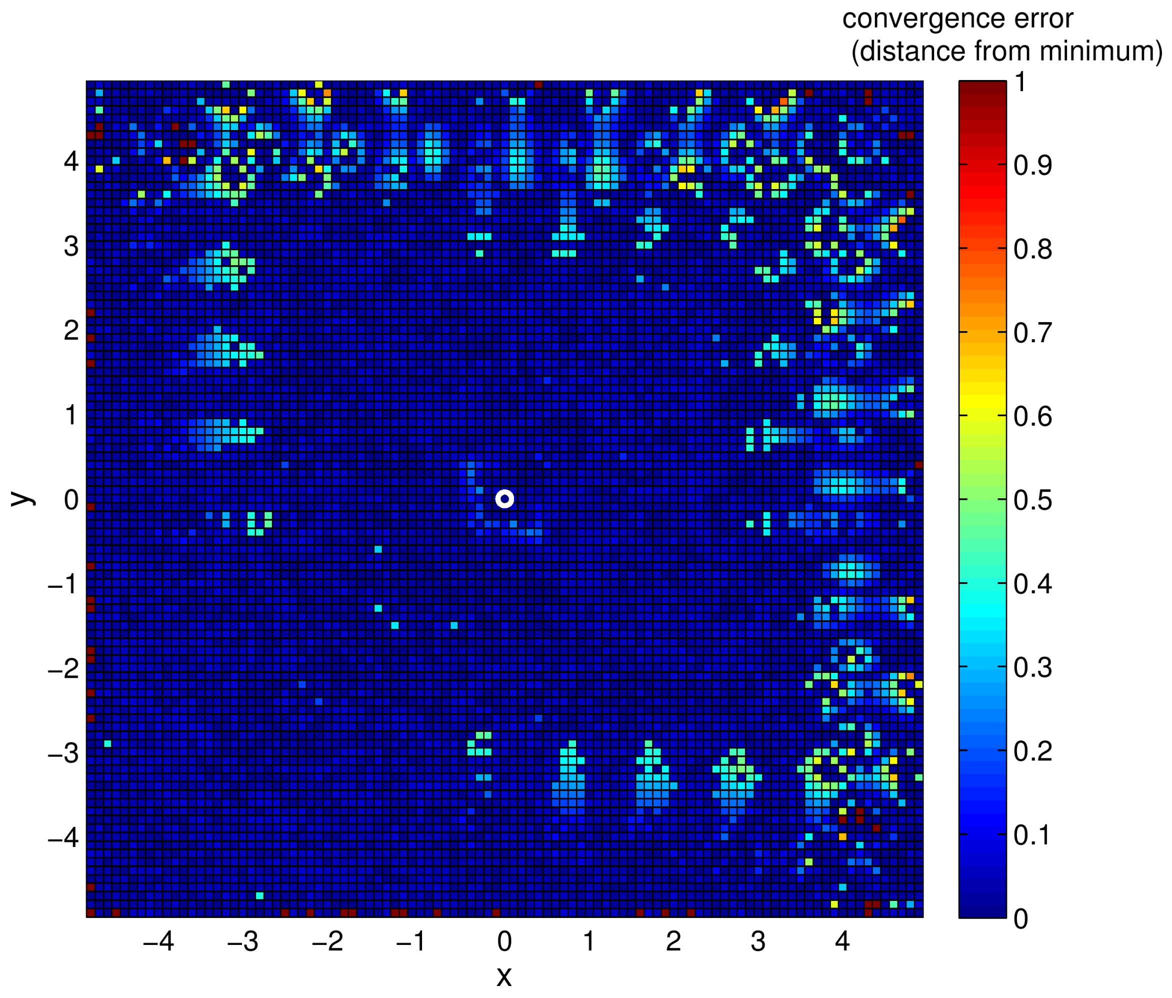


Figure 4.8: Ackley function optimization success with OPT++.

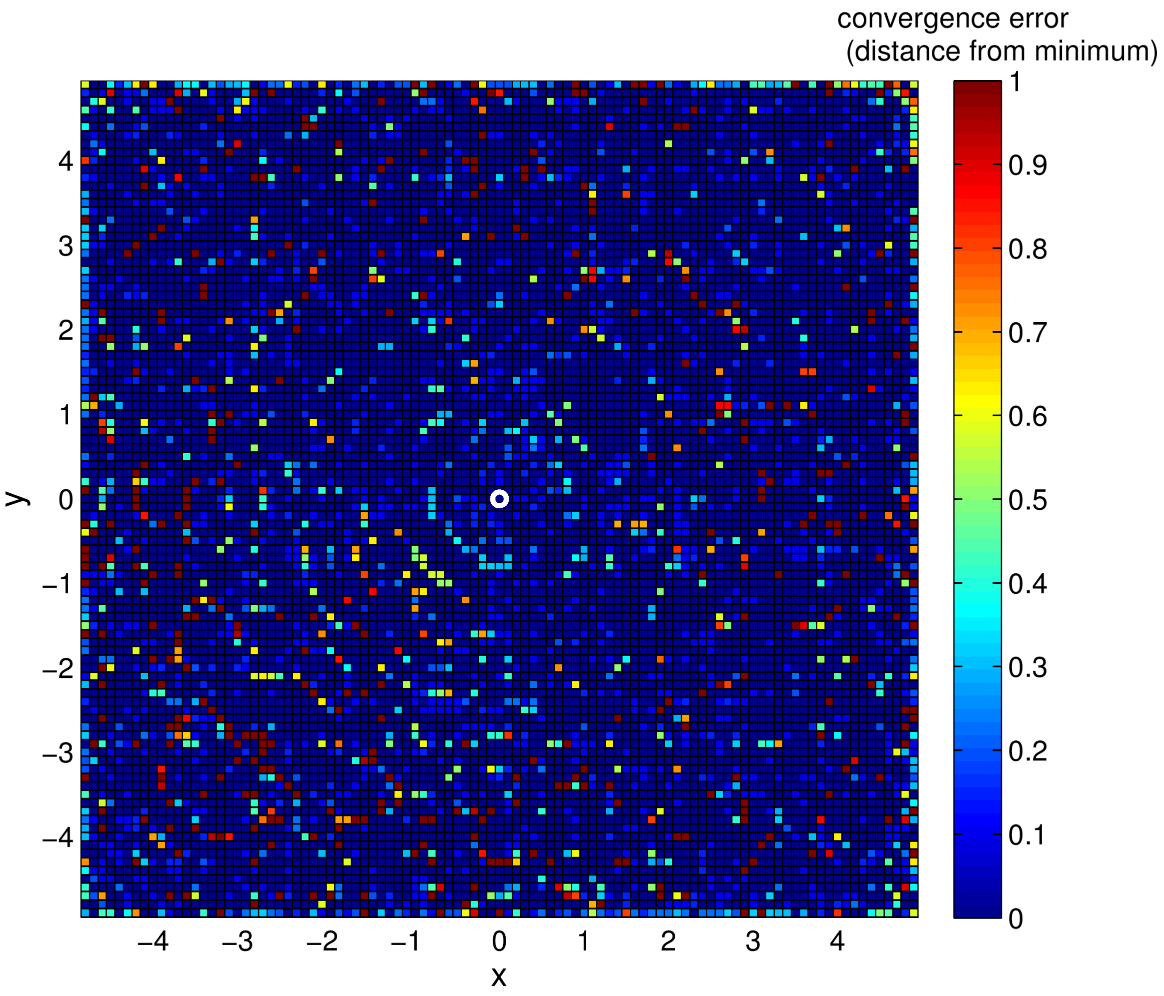


Figure 4.9: Ackley function optimization success with CONMIN.

convergence error is as high as 4.83. Clearly these errors are not occurring because of the gradient issues explained for the near-minimum region shown in Fig. 4.2, as these trajectories never make it that close to the minimum. Also, the initial trajectory points where these errors occur are not symmetric around the outer region of the domain, and in fact most of the trajectories that begin in the lower left quadrant are able to converge more or less exactly to the true minimum.

As it happens, the majority of these 'failed' trajectories tend to end up terminating in the same problematic function locations. This is shown in Fig. 4.10 where the initial and final points corresponding to all trajectories with optimization convergence errors greater that 0.3. As can be seen, many of the errors tend to get to within a radius of $\approx 0.6$ from the minimum before terminating. This intuitively does not make sense, as the gradients in this region, calculated with the fixed $h$, are large and should give the optimizer adequate information from which to formulate a new trajectory point closer to the minimum. However, what actually occurs is that the optimizer returns a new guess that is a very short distance from the current guess, leading to a new evaluation in which the objective function differs from the value of the last iterations by less than $\epsilon < 10^{-5}$ in some cases. The OPT++ line search algorithm fails is this region and the trajectories terminate prematurely. One possible reason for this is that because the function value in this region is changing very rapidly over the prescribed $h$, a very small $\alpha$ is being deemed sufficient for the step-size. The algorithm is designed this way so that in steep areas where the functional is changing drastically the optimizer does not want to risk overshooting and missing the minimum, thus a small $\alpha$ is prudent. However, the $\alpha$'s being calculated in this case are so small that the objective function at the new guess varies very little from the former, and after a few iterations in this fashion this the optimizer prematurely terminates. The problem tends to occur where both gradients are of the same order of magnitude and are large.

The results for the gridded (Case [2]) Ackley function tests are shown in Figs. 4.8

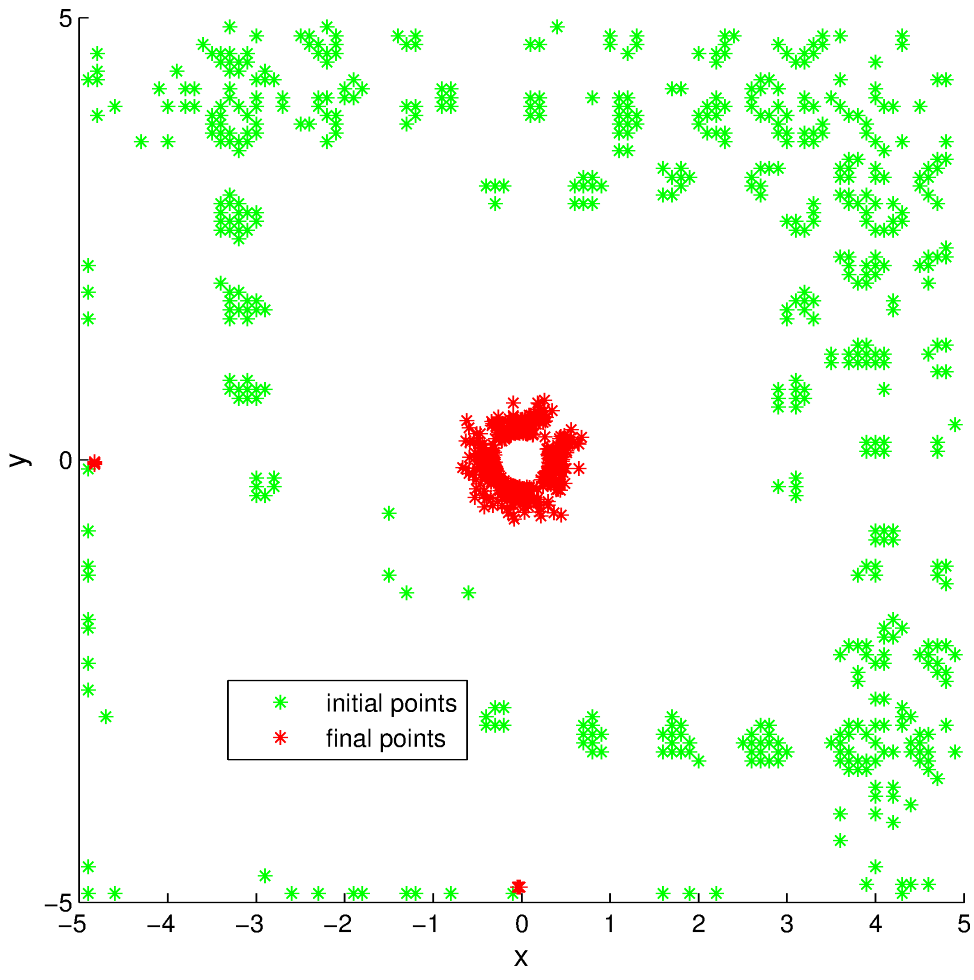


Figure 4.10: Scatter plot of the initial trajectory and final convergence points of all trajectories in which $\zeta > 0.3$ for the continuous Ackley function optimization.

through 4.12 for the OPT++ and CONMIN optimizer, respectively. Unlike the Sphere function, the results in this case are worse for the OPT++ gridded case (Fig. 4.11) than the continuous case (Fig. 4.8), particularly towards the boundaries of the domain, again due to the tendency of these trajectories to end up in the problematic regions of the functional. Another difference between the optimizer performance with this function vs. the Sphere function is that it is CONMIN that performs better with the gridded values (Fig. 4.12) than the continuous (Fig. 4.9) in this case.

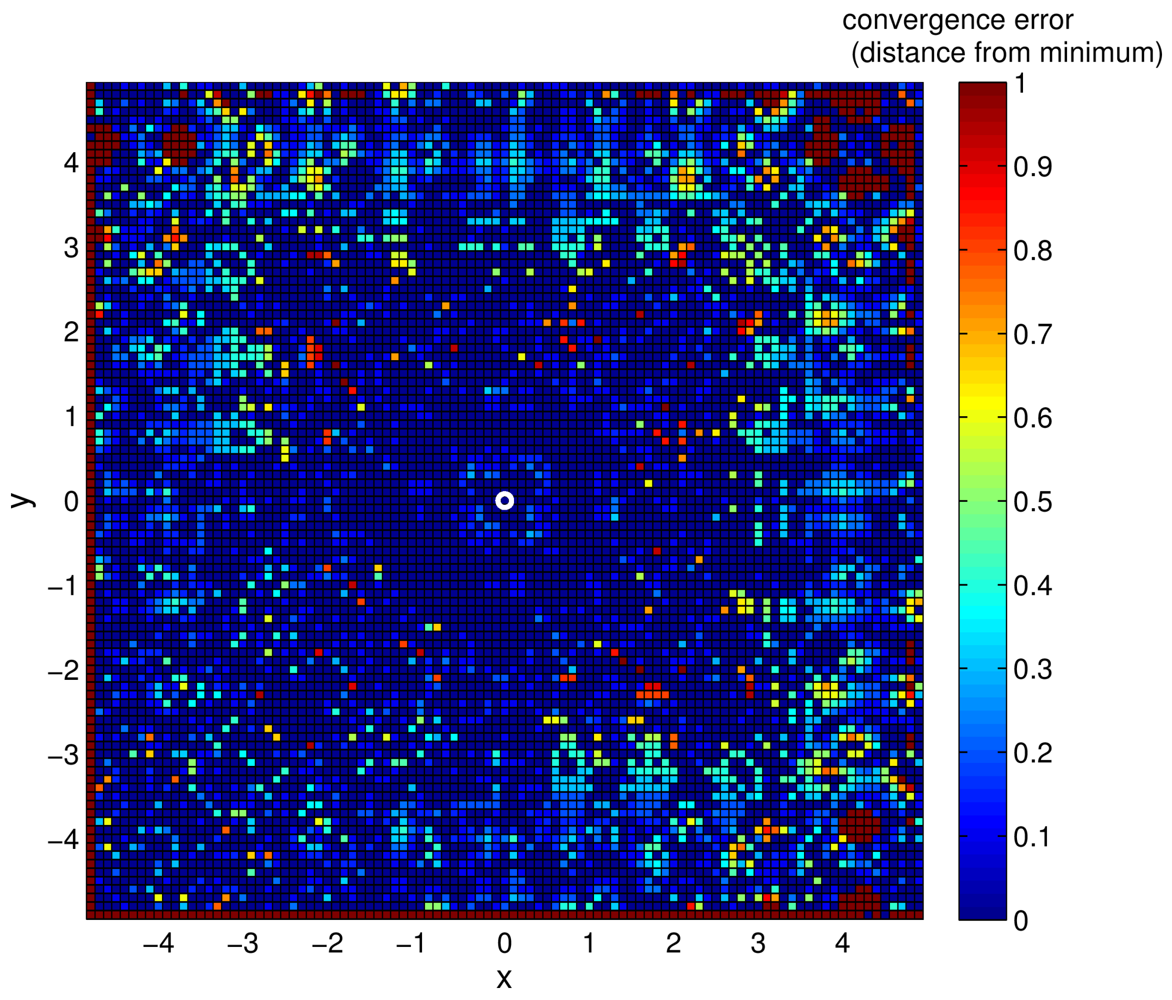


Figure 4.11: Ackley function optimization convergence error with OPT++ on a finite volume grid.

Finally, the CONMIN results of the Ackley function without the nudging algorithm implemented are shown in Fig. 4.13. These results should be compared to

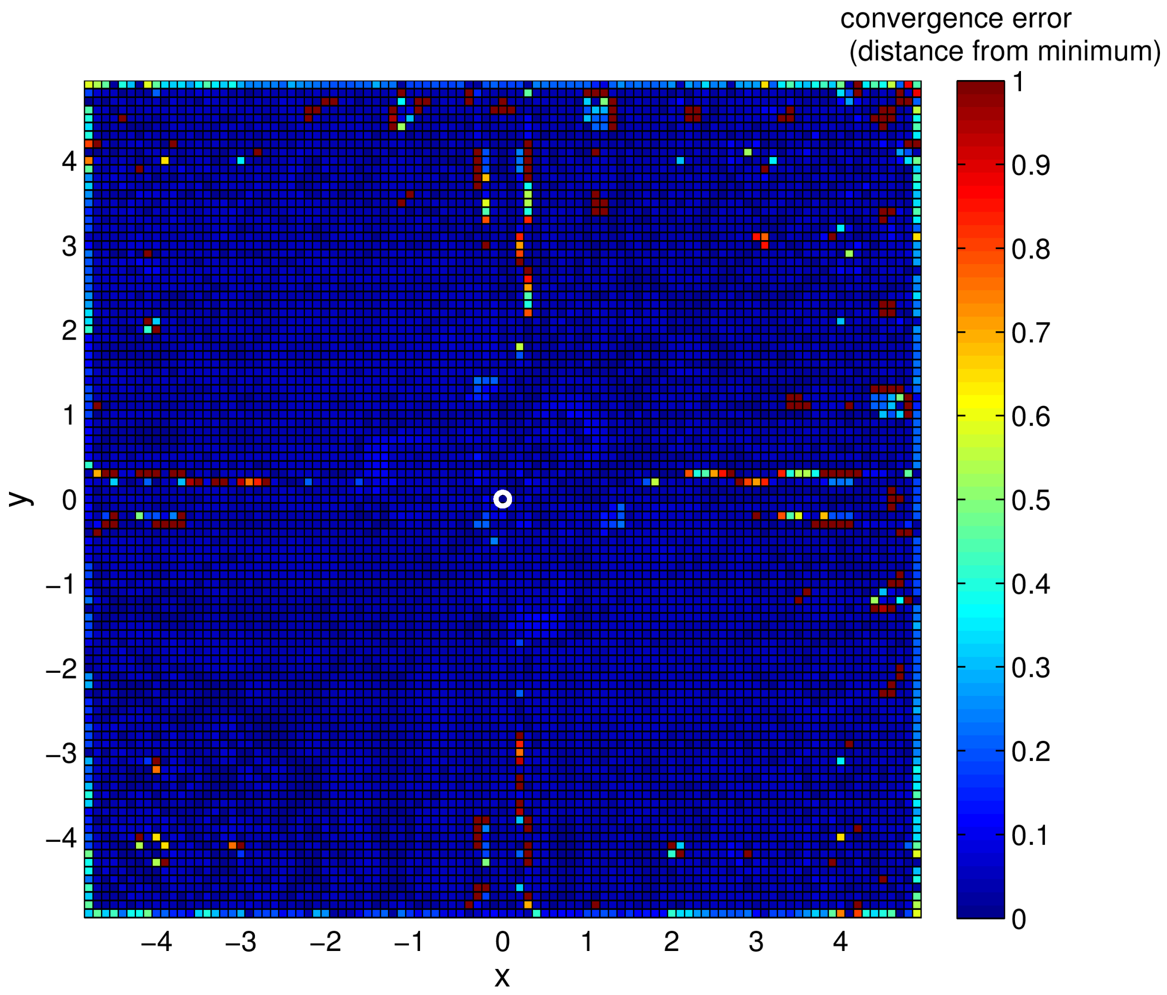


Figure 4.12: Ackley function optimization convergence error with CONMIN on a finite volume grid.

the CONMIN results for Case [2] (Fig. 4.12). The improvement due to the nudging is most significant for trajectories initiated in the region surrounding the minimum, where the problem of the trajectories getting ‘stuck’ in the cells due to the small $\alpha$’s stipulated by the conjugate gradient solver are the most troublesome. However, the nudging procedure has its limitations due to the discrepancy between the optimizer requested testing parameters and the actual values tested in OpenFOAM, as mentioned in Chapter III. Because of this, although the overall algorithm is more robust with the additional nudging fix, there are some initial trajectory locations that actually converge with a greater error with the nudging implemented.

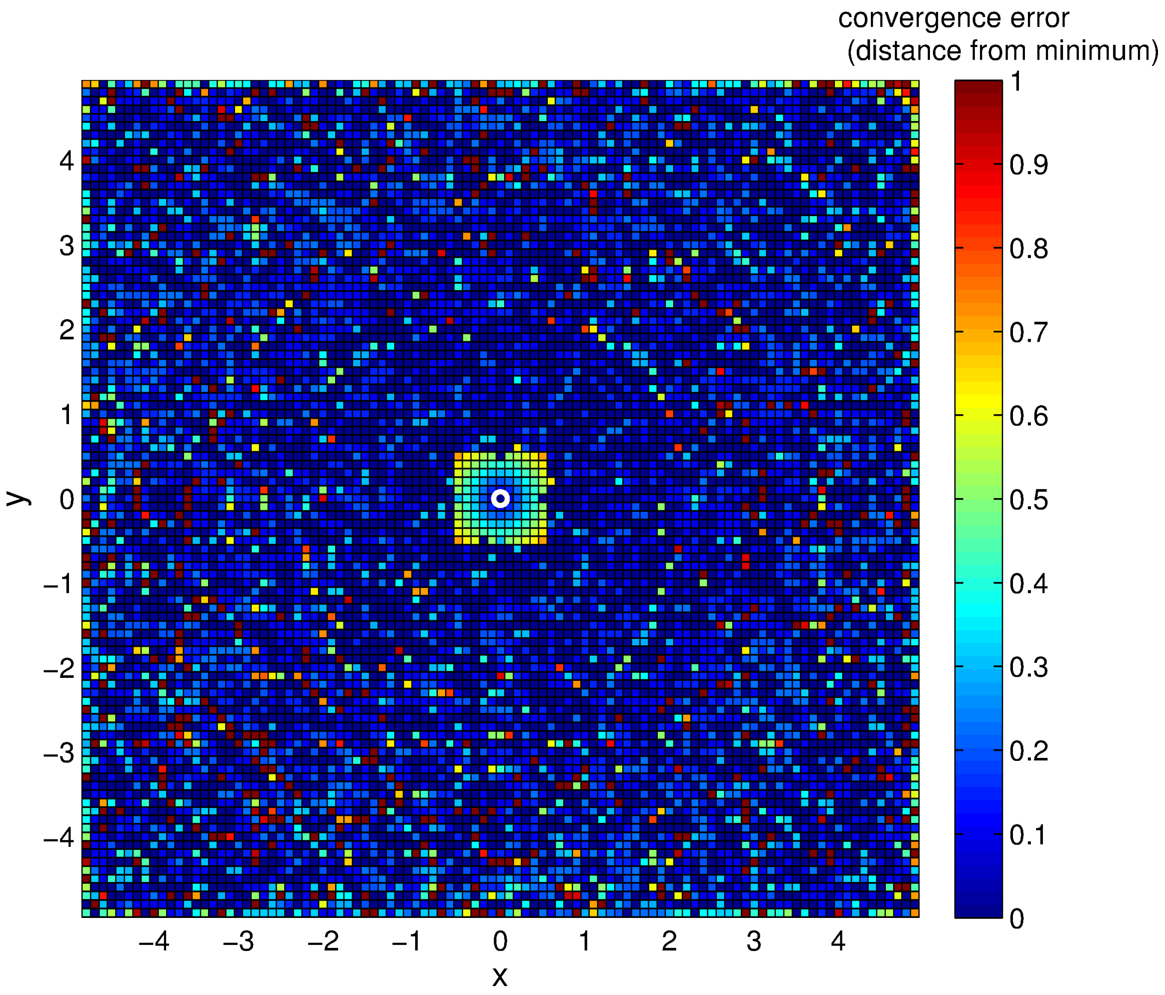


Figure 4.13: Ackley function convergence error with CONMIN and no numerical nudging in OpenFOAM.

Summary statistics for Ackley's function are presented in Table 4.2. Overall, very few trajectories in the Case [1] experiments resulted in $\zeta < 1 \times 10^{-5}$; less than 0.1% for either algorithm. However both optimizers were able to locate the minimum to within a distance of 0.1 in a large percentage of trials; 86.5% for OPT++ and 92.7% for CONMIN. The gridded results in the Case [2] experiments were more dichotomous in nature for the OPT++ algorithm, with over half of the trajectories converging successfully and the rest resulting in errors greater than 0.1. Further, the trajectories started from some of the locations in the flatter outer region often ended up further from the minimum than from where they initially started.

As with the Sphere function, there were a much greater percentage of completely accurate ($\zeta < 1 \times 10^{-5}$) trajectories for the gridded parameter experiments vs. continuous. Again, this is due to the 'forgiving' nature of the cell discretization in conjunction with the nudging technique implemented in the discretized optimization routine.

Table 4.2: Ackley function statistics

| Case | Method | $\zeta < 10^{-5}$ | $\zeta < 0.1$ | max($\zeta$) | mean($\zeta$) | mean run-time (s) |
|---|---|---|---|---|---|---|
| [1] | OPT++ | 0.07% | 86.5% | 4.83 | 0.09 | 63 |
|  | CONMIN | 0.06% | 92.7% | 3.15 | 0.09 | 60 |
| [2] | OPT++ | 59.7% | 59.7% | 4.87 | 0.27 | 92 |
|  | CONMIN | 73.7% | 85.6% | 4 | 0.11 | 43 |
|  | CONMIN, no nudging | 45.0% | 45.0% | 3.24 | 0.18 | 28 |

The mean run-time was again shorter for the OPT++ trajectories compared to CONMIN for Case [1], but for the gridded variables CONMIN was the more efficient algorithm. Also, the inclusion of the 'nudging' fix generally leads to longer trajectory times, as the optimizer is prevented from terminating prematurely at a 'stuck' location. While in this case the nudges accounts for an average of 15 more seconds run-time per trajectory, this is a negligible increase in time in light of the increase accuracy and robustness of the solution.

## 4.6 Scaling

A final topic explored in this section is the effects of the automatic scaling algorithm implemented in DAKOTA on the optimization results. As was mentioned in Chapter III, scaling is a method by which the efficiency and/or accuracy of the optimization problem is increased by adjusting specific quantities in the problem to a common scale. These quantities could be comprised of the parameters, objective functions for multi-objective function analysis, or the optimization constraints. For the source inversion problem, the goal of the scaling is to make the parameters similar in magnitude so that their respective gradients do not differ by several orders of magnitude. This affects the optimization process in two main ways: first, when the order of magnitude of two or more variables differs greatly, it is difficult to define adequate convergence criteria in order to terminate the optimization. The bigger problem with regards to the problem at hand however is in regards to the optimization step-size, $\alpha$, that is determined at each iteration. When the magnitudes of the parameters vary greatly, their respective gradients as assessed with a uniform $h$ vary greatly as well. Therefore the choice of magnitude and direction of the trajectory step will be 'dominated' by the parameter with the larger gradient, possibly impeding the optimization in respect to the other parameter(s) of the problem.

DAKOTA offers several options for implementing parameter scaling. The scaling criteria can be explicitly defined by the user, in which a scaling value is provided for each parameter and the parameters adjusted by dividing the parameter by its scale factor. Any bounds placed on the parameter are adjusted as well. Alternatively, automatic scaling can be enabled that adjusts the parameters so that they are within the interval [0,1]. Finally, logarithmic scaling can be specified and combined with user-defined values. With this method, the variables are scaled on a log base 10 scale after first applying any user-defined parameter scale values.

For reasons that were discussed in Chapter III, it is preferable to apply the ‘one size fits all’ automatic scaling in the DAKOTA optimization instead of using user-supplied scaling values. This is because manually adjusting the scale factors on a case-by-case basis is another method of ‘tuning’ the optimization to a specific problem, which should be avoided in order to preserve generality. In addition, the use of automatic scaling is much easier to implement.

The results of some of the optimization test cases above are repeated here with the inclusion of the DAKOTA automatic scaling in order to assess the effects of using the scaling option. Firstly, the Case [1] OPT++ results for the Sphere function are shown in Fig. 4.14, where the figure on the left shows the previously presented unscaled results and the figure on the right shows the results with the automatic scaling implemented in DAKOTA. In this case, the previous problem of the trajectories identifying the minimum just outside of the true minimum due to the forward differencing scheme is eliminated in the lower left region of the domain, indicating that the alteration of the trajectories due to the scaled optimization step-sizes are causing trajectories initiated in this region to overshoot the minimum and then converge from the side in which the forward differencing gradients are positive with respect to both variables.

Next, we consider the effects of the automatic scaling on the Case [2] gridded CONMIN function, which was one of the cases that had highly unreliable results. The unscaled vs. scaled convergence errors in this case are shown in Fig. 4.15. Overall there is little change to the behavior of the optimizer with the addition of the scaling, however there is a slight *decrease* in the percentage of accurate trajectories with the inclusion of the scaling option: 19.8% vs. 21.2% . Additionally, the percentage of trajectories that result in an error of less than 0.1 decreases as well, going from 53.4%

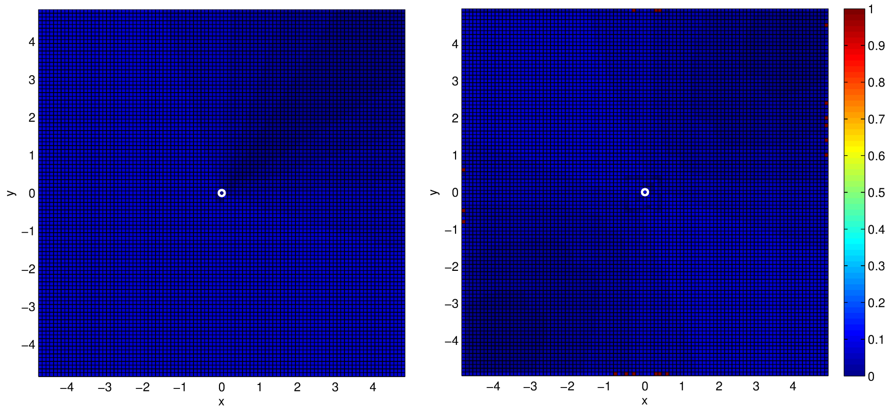


Figure 4.14: Comparison of the convergence errors for the Case [1] Sphere function OPT++ results obtained without scaling (left) and with automatic scaling (right).

without scaling to remaining at 19.8% with scaling, indicating that the scaling option leads to not only a greater number of failed trajectories, but to trajectories with a greater magnitude of error.

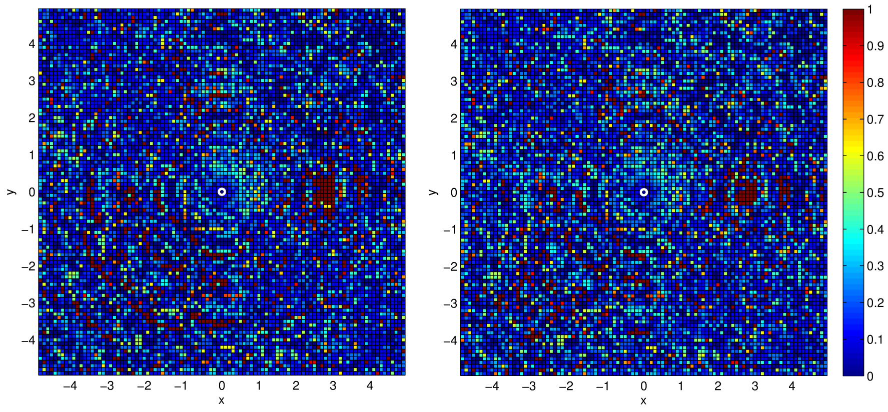


Figure 4.15: Comparison of the convergence errors for the Case [2] Sphere function CONMIN results obtained without scaling (left) and with automatic scaling (right).

The Case [2] CONMIN results for the Ackley function are shown in Fig. 4.16 with and without the scaling invoked. While points initiated in the 'crater' region surrounding the minimum convergence reliably with the addition of the scaling, trajectories initiated in the outer, flatter regions do much worse. In particular, the upper right quadrant trajectories tend to consist of larger steps when the scaling is initiated, and in many cases the trajectories get stuck after overshooting the minimum.

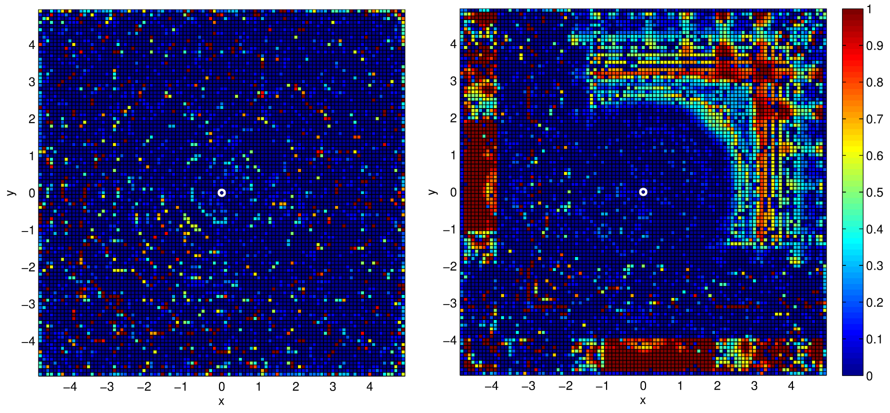


Figure 4.16: Comparison of the convergence errors for the Case [2] Ackley function CONMIN results obtained without scaling (left) and with automatic scaling (right).

Finally, Fig. 4.17 shows the unscaled vs. scaled results for the Case [2] Ackley function solved using OPT++. The scaled results for this case are also worse than the unscaled, lowering the percentage of accurate trajectories from nearly 60% without scaling to 41% with scaling.

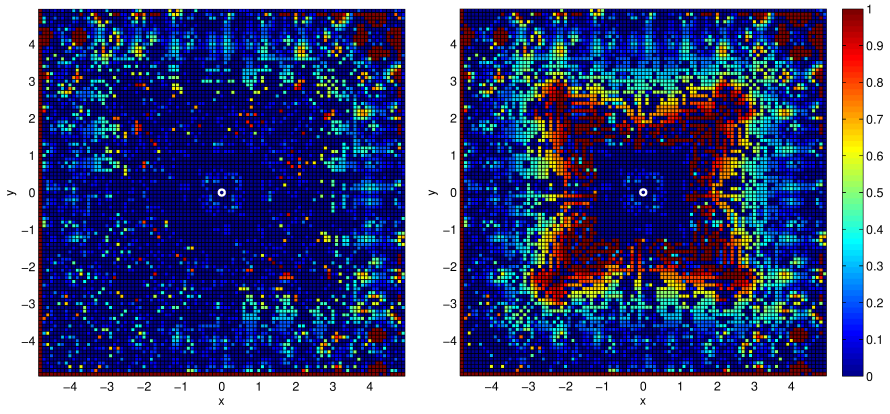


Figure 4.17: Comparison of the convergence errors for the Case [2] Ackley function OPT++ results obtained without scaling (left) and with automatic scaling (right).

The above results are somewhat disconcerting, as it would not be expected that scaling would adversely affect the optimization process. One possible explanation for the less reliable results in some of these cases could be that neither of these analytical functions exhibit parameters that vary greatly in magnitude from the other, and therefore, it is possible that attempting scaling on problems that are already well-scaled creates problems with the optimization algorithms. This is evidenced by the results in Fig 4.16 in particular, whereby many of the trajectories that fare worse with the scaling are those which tend to make a diagonal path towards the minimum and hence are guided by evaluated gradients with respect to x and y that are equal in magnitude. What is clear is that the results of all the above tested methods, and in particular those in which gridded parameters are used in conjunction with the forward differencing, are highly sensitive to small changes in the trajectories.

Due to the inconsistent results of these scaling test cases and the lack of a consistent and drastic benefit, scaling is not implemented in the following chapter's two-dimensional source inversion problems. However, this topic is revisited in Chapter VI for the source inversion in higher dimensions, by which it will become apparent that in some cases scaling is necessary.

## 4.7 Summary and Discussion

The analysis in this chapter reinforces the previous discussion in Chapter III in regards to the difficulty matching an appropriate solver to a particular problem. For the shallow, uniformly graded Sphere function, the quasi-Newton method is superior, particularly for the gridded case where the problems due to the fixed step-size are mitigated to some degree but the nudging algorithm. However in the case of the Ackley function, which more closely approximates the two-dimensional source inversion objective functions, the conjugate gradient method performs significantly better. The respective run-times for each method varied widely for each case, generally taking longer for initial trajectory points further from the minimum, as would be expected.

This chapter also presents examples of the issues encountered in optimization discretized problems that were introduced in Chapter refchap:chap3. These issues are the related problems of (1) numerical gradients and (2) discretized parameters. The use of the fixed finite-difference step $h$ in particular is a detriment to the numerical gradient approximations. This is made evident by comparing the robust and reliable results of the algorithms when using the internal DAKOTA differencing algorithm, which scales the finite-difference step-size by the current parameter values, to the less accurate results obtained using a fixed value of $h = 0.1$ for the gradient calculations. While this choice makes sense on a discretized domain, when using continuous values this fixed $h$ leads to under- and over-shooting in the vicinity of the minimum. The results would no doubt be improved by using a smaller uniform $h$ in this case, but would also likely lead to much longer convergence times. And more importantly, the use of a smaller $h$ is not possible for the case of the discretized parameters. That is, when moving from the continuous case to the discretized grid, there is a restriction on the minimum value of $h$ that can be used in calculating the gradient, where an $h < \mathbf{ds}$ will not be defined. Thus, from a practical standpoint, it makes sense to define a local $h$ based on the next nearest east or north located cell. A value smaller than this is impossible, and a value larger than this will lead to a greater error between the actual and calculated gradient field.

While increasing the resolution would allow for a smaller value of $h$ and thus would lead to a more accurate assessment of the gradients, the problem of the discretized parameters and the minimum defined $h$ still exists for a refined grid, just on a smaller scale. Further, the optimizer will struggle even more in flat regions of the optimizer, being confined to the adjacent cells in which to obtain gradient information. When these adjacent cells are even closer to the current cell, the local region will look flatter to the optimizer, sometimes resulting in estimated gradient components that are zero and hence causing the optimizer to erroneously terminate in the cell in which the trajectory was initiated.

Clearly a more dynamic method of adjusting $h$ based on the current parameters would be ideal for improving the discretized optimization solution. However, this would be difficult to implement without knowing the nature of the functional a priori. The method used by the internal DAKOTA finite differencing is able to adjust $h$ in a manner that is highly efficient for these two test problems because it is designed with the knowledge that the minimum is located at the origin. Such an algorithm however is not applicable to functionals in which this is not the case.

Another possibility for increasing the robustness of the discretized optimization problem is to apply central differencing for the gradient calculations. This alternative is not used in this work due to the facts that (1) central differencing approximations will lengthen the overall run-time of the source inversion optimization and (2) alternative differencing schemes are still limited by restriction of the minimum $h$ that can be used and thus will run into the same problems due to the discretization of the domain. Based on the above analysis and for lack of a viable alternative, a fixed $h$ based on the distance from one cell to the next, $\mathbf{ds}$, is used in the source inversion problems in the following chapters regarding the source inversion problem.

Lastly, the effects of the automatic scaling option in DAKOTA were assessed for several of the analytical optimization problems. The results with the addition of the scaling varied based on the solver and function assessed. The Sphere function tended to be improved by the automatic scaling, whereby the Ackley function in many cases fared worse with the scaling option.

# CHAPTER V

# Two-Dimensional Source Inversion Applications

In this chapter the gradient-based optimization methods that were tested in Chapter III on analytical functionals are applied towards the two-dimensional, spatial source inversion problem. The effectiveness of the gradient-based approach for this simple source inversion problem is assessed as a function of the flow parameters, source type (instantaneous vs. continuous), and location and number of sensors. In light of the limitations of the gradient-based optimization methods, an optimization technique known as hybrid optimization is explored as an alternative source inversion approach. This method exploits the benefits of both derivative-free and gradient-based optimization methods. Additionally, the use of a Model Predictive Control approach to mitigate the discretization errors is also explored. The two-dimensional simplified domains used in this chapter's examples facilitate a solid understanding and visualization of the source inversion process and the influence of the various input parameters on the optimization outcome.

## 5.1 Modeling Specifics

In the subsequent source inversion examples, the goal is to locate the two-dimensional coordinates $(x_s, y_s)$ corresponding to a source release in a channel, using contaminant concentration data as measured at the sensor locations. The ambient flow velocity and diffusivity properties are assumed to be known, as well as the initial source strength $q_s$ and time of release $t_s$. The sensor data in each case are obtained by running the forward model with the true source location and recording the contaminant concentration at the points in the domain that are defined as 'sensors'.

The physical domain used in the source inversion simulations in this chapter is the two-dimensional channel that was first defined in Chapter III. This channel is 6 m long and 2 m wide and discretized into 1281 uniform cells of dimensions 0.1 x 0.1 meters each. As with the analytical test cases, the possible parameter values are confined to the interior cells to avoid problems with the forward finite differencing scheme. Therefore only cell centers in the range of $0.1 \leq x \leq 5.9$ and $0.1 \leq y \leq 1.9$ are included in the parameter space defined for $\mathbf{f} = (x_s, y_s)$. The simulations in this chapter assume laminar flow with the boundary conditions given in Table 5.1.

Table 5.1: Boundary conditions for source inversion of parameters $(x_s, y_s)$

| Boundary | $p$ (m$^2$/s$^2$) | $U$ (m/s) | $C$ (units) |
|---|---|---|---|
| inlet | zero gradient | parabolic, $U_{max} = 2$ | fixed value, $C = 0$ |
| outlet | fixed value, $p = 0$ | zero gradient | zero gradient |
| walls | zero gradient | fixed value, $U = 0$ | zero gradient |

The contaminant in all cases is a point source (continuous or instantaneous) released at coordinates $(x_s, y_s) = (2, 1)$ starting at $t_s = t_0 = 0$ seconds. To simulate this, the passive scalar transport equation, Eqn. (2.3) is used with the source function $f$ provided by either Eqn. (2.5) for a continuous release or Eqn. (2.4) for an instant release. The optimization is carried out using Eqn. (3.3) as the objective function and a regularization parameter of $\sigma = 1 \times 10^{-5}$. As with the analytical test problems, forward finite differencing is applied for the gradient-based optimization using the distance $\mathbf{ds}$ to the adjacent east and north cells of the current cell as the step-size $h$. With the regularly spaced grid implemented herein, this corresponds to $h = 0.1$ in both the $x-$ and $y-$directions. The time-step $\Delta t$ used in all of these simulations is

0.04 seconds, which maintains the Courant number at a value well below 1.

## 5.2 Influence of Flow Characteristics on Optimization Performance

As was shown in Chapter IV, the relative performance of the Fletcher-Reeves Conjugate Gradient solver CONMIN and the quasi-Newton solver OPT++ varies with the specific functional being optimized. In that chapter it was shown that the Conjugate Gradient Fletcher-Reeves algorithm implemented in CONMIN performed more accurately with the simplified Ackley function, which contained a wider range of gradient magnitudes and a more concentrated and steep descent to the minimum, whilst the shallower and uniformly varying Sphere function was better optimized with the quasi-Newton method applied in OPT++. With either solver, the optimization may be completely accurate if a ‘good’ starting guess $\mathbf{f}_0 = (x_0, y_0)$ is chosen. The problem of course is that a good starting guess is not always obvious or predictable. Again, this was shown in Chapter IV in regards to the analytical test optimization problems, in which the numerical optimization failed for a number of randomly dispersed initial starting points, especially when the possible parameter values were restricted to the discretized grid cell centers.

The requirement of a good starting point can be even more stringent for the source inversion functionals. This is because the shape of the functional, which directly influences the performance of the gradient-based optimization, is highly dependent on the characteristics of the source inversion problem. These characteristics include the ambient flow velocity, diffusivity, and dimensions of the channel. The location and number of sensors in the domain are additional factors influencing the functional shape, which will be addressed in Section 5.4. As previously discussed, gradient-based optimization works best with continuous and convex functionals, in which a smooth descent to a global minimum can be delineated by the optimization scheme. Many combinations of the influencing source inversion characteristics will result in a functional topography that highly deviates from this ideal, as will be shown.

The influential flow parameters can be symbolized by the Péclet number, a non-dimensional number that represents the relative influence of advection and diffusion in a given flow:

$$Pe = \frac{UL}{D}, \tag{5.1}$$

where $L$ is the characteristic length scale and $D$ is the diffusion coefficient in units of [m$^2$/s]. A high Péclet number indicates a flow that is dominated by advective processes, while a low Péclet number indicates that diffusion is the dominating phenomenon. The shape of the objective function governed by Eqn. (3.3) is highly sensitive to this parameter and therefore the success of the inverse optimization problem is directly dependent on its selection. If the Péclet number is too low, diffusion will quickly dilute the contaminant over the domain and the objective function may not have an adequately defined minimum or strong enough gradients for the optimizer to succeed. If the Péclet number is too high the functional will not be ideal either, as the weak signal strength obtained from the sensors will lead to a loss of information and a functional surface that is not appropriately convex. In this case the effective search area in which the minimum can accurately be converged upon will be constrained to a small region of the domain, rendering a randomly guessed starting point less likely to lead to successful trajectory to the global minimum. Also note that increasing the Péclet number is analogous to widening the channel in the case of boundary sensors, as the signal is similarly reduced. Therefore examining the effects due to larger Péclet numbers can be extrapolated to the case of a wider channel.

Examples of two-dimensional functional contours obtained at three different Péclet numbers are shown in Fig. 5.1. Here, the Péclet number is determined by taking the length of the channel to be $L$ and the characteristic $U$ to be $\frac{1}{2}U_{max}$. All of these cases assume isotropic diffusivities, in which the transverse diffusion coefficient $D_T$ is equal to the longitudinal diffusion coefficient $D_L$. Four sensors are located at coordinates $(3, 0)$, $(3, 2)$, $(4, 0)$ and $(4, 2)$. Figure 5.1(a) shows the functional contours for a relatively small Péclet number of 0.6. In this case the high diffusivity creates a functional with a 'bowl' containing the minimum that is wide in the transverse direction and narrow in the longitudinal direction, amidst a much flatter surrounding region. As a result, the ideal, convex region of the domain is confined to a narrow region around the minimum. Similarly, the strong advection in the case of a Péclet number of 600 (Fig. 5.1(c)) results in a region downstream of the sensors that is nearly completely flat, also an impediment for a gradient-based optimizer. In addition, while the sides of the functional upstream of the sensors are highly sloped, the gradients in the direct vicinity of the minimum are relatively mild, which cause the optimizer to have a harder time zeroing in on the exact minimum. A more ideal functional is given with a Péclet number of 6 (Fig. 5.1(b)), where in this case the region surrounding the minimum contains gradients that are more uniformly and symmetrically distributed around the minimum, and the convex region extends over a greater area of the domain.

A further consideration is anisotropy of the diffusion coefficients. In modeling a realistic channel, the diffusion coefficients would be replaced by dispersion coefficients in order to account for transverse and longitudinal mixing, whereas the longitudinal dispersion would be expected to be larger than the transverse dispersion. This anisotropy leads to changes in the functional as well. Specifically, a relatively larger longitudinal diffusion/dispersion coefficient will create a more narrow, trough-shaped objective function than in the isotropic case, again creating an less than ideal functional from the standpoint of gradient-based optimization.

In summary, the optimization sensitivity to the Péclet number is a compounding factor that causes additional difficulties with the gradient-based optimization meth-

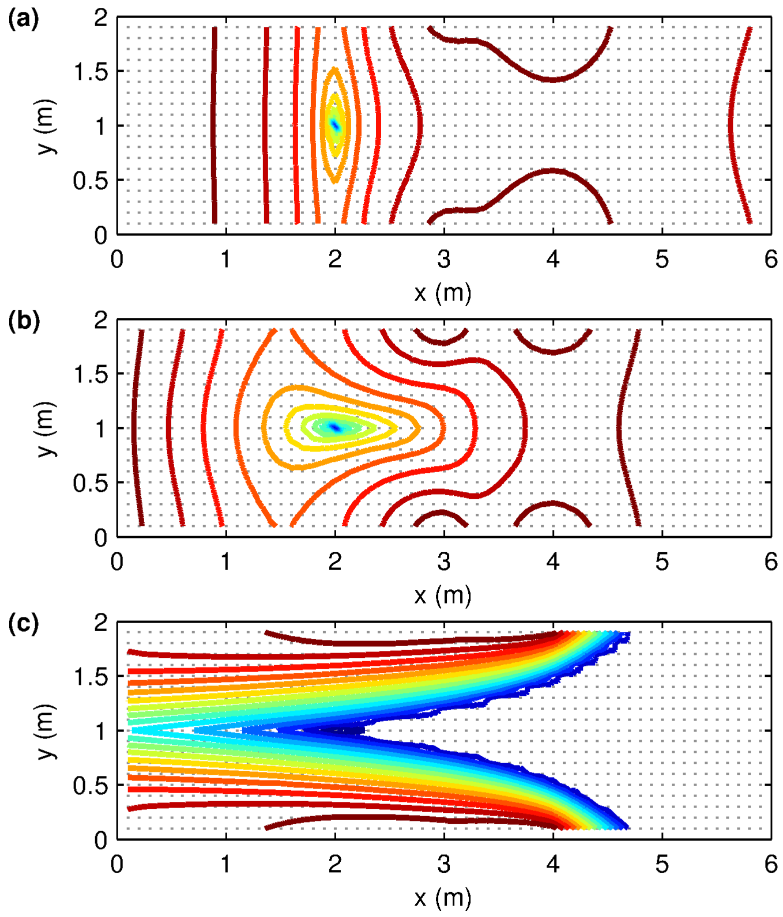


Figure 5.1: Comparison of functional contours with a Péclet number of 0.6 (a), 6 (b) and 600 (c).

ods, along with the previously discussed issues regarding the discretization of the parameters and the numerical gradients. These factors lead to significant limitations regarding the gradient-based optimization approach in some source inversion cases, as will be shown in the results below.

## 5.3 Results for the Two-Dimensional Case

In this section, four main cases are tested with both CONMIN and OPT++:

[1] Continuous release of $\dot{q} = 2000$ units/s with isotropic diffusivity of $D_T = D_L = 1$ m$^2$/s.

[2] Continuous release $\dot{q} = 2000$ units/s with anisotropic diffusivities of $D_T = 1$ m$^2$/s and $D_L = 0.1$ m$^2$/s.

[3] Instantaneous release of $q = 80$ units with isotropic diffusivity of $D_T = D_L = 1$ m$^2$/s.

[4] Instantaneous release $q = 80$ units with anisotropic diffusivities of $D_T = 1$ m$^2$/s and $D_L = 0.1$ m$^2$/s.

In all of these cases four sensors are located at the coordinates $(3, 0)$, $(3, 2)$, $(4, 0)$ and $(4, 2)$ and the observation time is 10 seconds. The sensor observations corresponding to these scenarios are shown in Fig. 5.2, where subfigures (a)-(d) correspond to Cases [1]-[4] described above. Due to the location of the source, which is equidistant from the pairs of sensors located on either side of the channel, the sensors at $x = 3$ m and $x = 4$ m measure the same concentration values, resulting in only 2 visible lines in Fig. 5.2(a)-(d).

Firstly, the results for Case [1] are compared for the CONMIN and OPT++ solvers in Fig. 5.3. This figure depicts the final optimization convergence error $\zeta$ in terms of the distance between the true source location $(x_s, y_s)$ and the final optimization

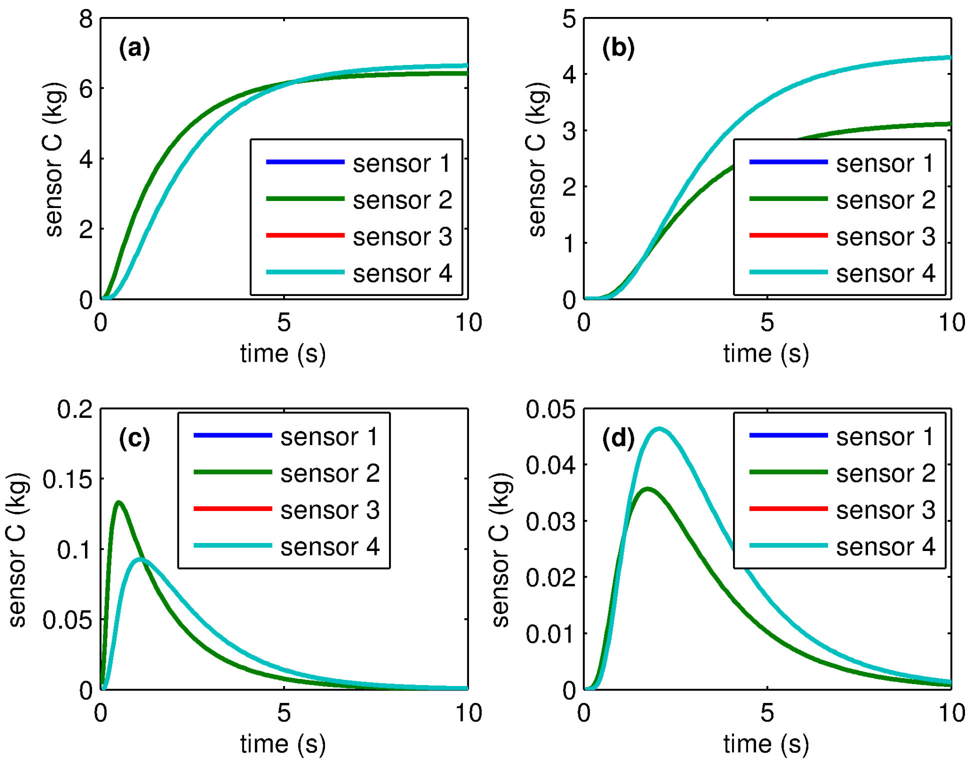
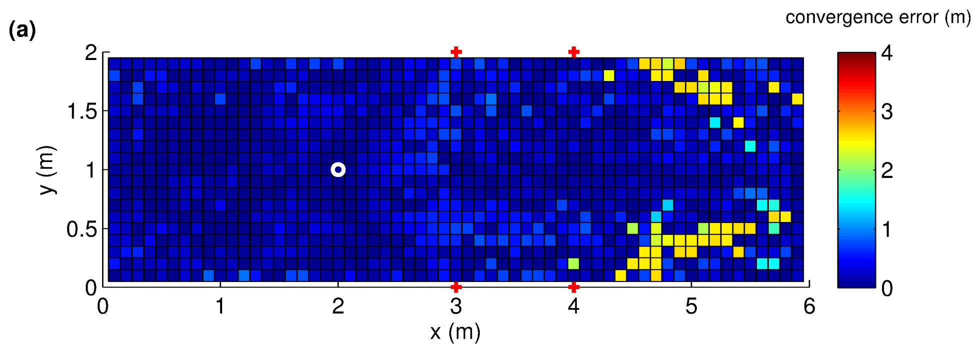


Figure 5.2: Contaminant concentration measured by the 4 boundary sensors. Cases [1]-[4] correspond to subfigures (a)-(d).

parameters $(x_f, y_f)$, as a function of initial guess $\mathbf{f}_0$. Note that the true source location is marked by the white circle and the red +'s indicate the sensor locations.

The source inversion algorithm in this case is completely successful at 21% of the tested starting guesses with CONMIN optimization and 35% of the OPT++ optimization tests. For 61% of the CONMIN starting points and 63% of the OPT++ starting points, the solution converges to less than or equal to 0.2 m from the true source. Practically speaking however, in a uni-directional flow it can be reasonably assumed that the source originates upstream of the sensor data record that contains the peak value of $C$. With the sensor data given in Fig. 5.2 this would restrict the possible initial points to the region where $x \leq 3$, however for generality the analysis herein considers the sub-domain of 'upstream' points to be those within the range $0 \leq x < 4$ m. Restricting the the values of $x_0$ and $y_0$ to this subdomain results in an error of less than or equal to 0.2 m in 66% and 67% of the starting locations for the

CONMIN and OPT++ optimizers, respectively.



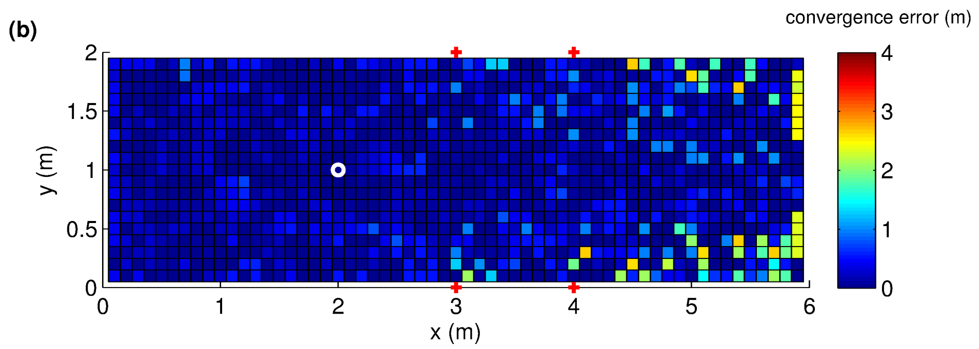


Figure 5.3: Convergence errors for a continuous point source and $D_T = D_L = 1$ (Case [1]) with CONMIN (a) and OPT++ (b) optimization.

While the convergence accuracy of CONMIN and OPT++ are similar for this case, the run-times are typically longer with OPT++, at an average convergence time of 6.3 minutes vs. 5.2 minutes with CONMIN. As noted previously, these times should be used for relative comparisons only, as the overall convergence time is a factor of the computing platform used and the number of processes run on the computer at a given time, the latter of which was not strongly controlled for in these experiments.

The results of Case [2] are presented in Fig. 5.4, where again subfigures (a) and (b) correspond to the CONMIN and OPT++ convergence errors, respectively. As predicted, the anisotropic diffusivities in these cases lead a much smaller radius of convergence for both methods. The number of successful trajectories using CONMIN is now only 2.6% of all the tested points, and only increases to 3.4% when considering points upstream of 4 m alone. Defining an acceptable error of $\zeta \leq 0.2$ does not lead to many more successful trajectories, as the errors exceed this minimum value in 91.2% of cases considering all points and 87.6% of the cases for points upstream of 4 m.

OPT++ fares better over a wider range of starting points than CONMIN in this case, with a total of 15% and 17% of completely accurate trajectories out of all the test points and upstream test points respectively, and 23.4% and 26.7% of trajectories meeting the $\zeta \leq 0.2$ criteria for all and upstream test points, respectively.

The mean run-time is again less for CONMIN, 2.7 vs. 3.9 minutes for Opt++. However it is clear that there are many more 'bad' trajectories with the CONMIN solver than the OPT++ solver, and thus the possibility exists that shorter CONMIN run-times may be correlated with less accurate trajectories. In order to investigate this, the mean times were also calculated and compared for the successful ($\zeta = 0$) vs. 'failed' ($\zeta > 2$) trajectories and indeed found to be typically longer for the successful trajectories. This topic is discussed in Section 5.3.1, and the mean convergence times calculated for the various cases, error categories and solvers is presented in Table 5.3.

Results for an instantaneous, isotropic release (Case [3]) are shown in Fig. 5.5. Interestingly, the percentage of completely successful trajectories considering all possible starting points is higher than the results for Case [1]: 49.2% for CONMIN and 58.8% for OPT++. These numbers increase to 57.6% and 66.9% when considering initial points in the upstream region $x < 4$ m. Increasing the error tolerance to $\zeta \leq 0.2$ m, the percentage of all successful starting points increases to 64.9% and 77.5% for CONMIN and OPT++, respectively, and to 75.4% and 87.6% for points $x < 4$ m. The better performance of both optimizers in this case is attributed to the altered shape of the functional as a result of the instantaneous release, whereas the instantaneous source creates a more ideal, convex functional than the continuous source. The OPT++ solver again takes longer to converge for this case: an average of 6.3 minutes vs. 5.2 minutes per trajectory for CONMIN.

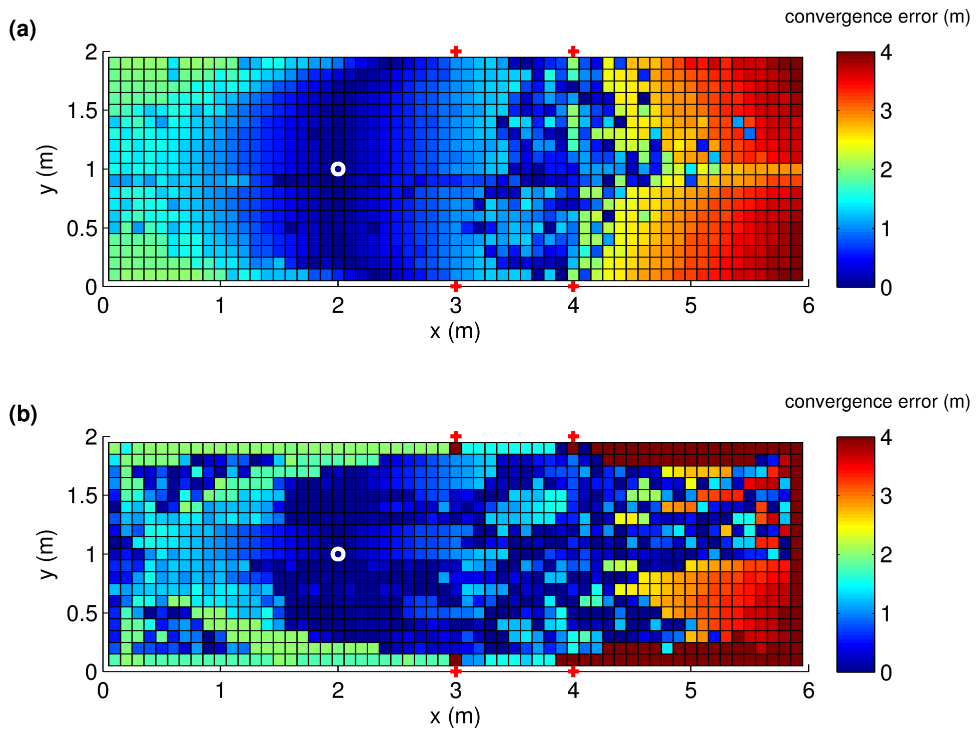


Figure 5.4: Convergence errors for a continuous point source and $D_T = 1, D_L = 0.1$ (Case [2]) with CONMIN (a) and OPT++ (b) optimization.

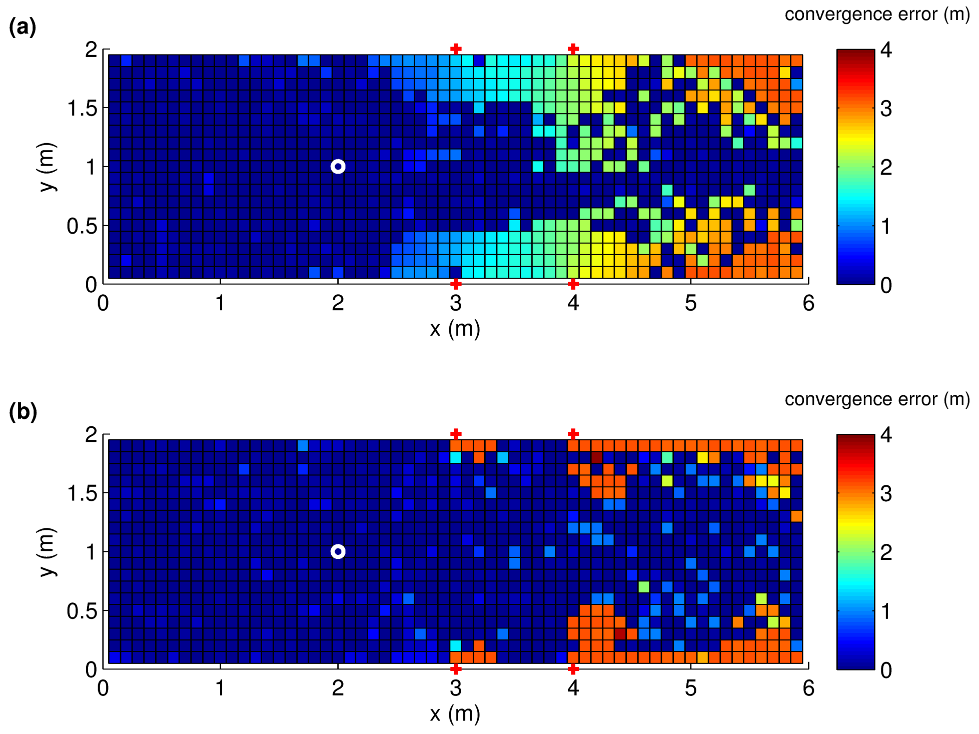


Figure 5.5: Convergence errors for a instantaneous point source and $D_T = D_L = 1$ (Case [3]) with CONMIN (a) and OPT++ (b) optimization.

Comparing the results of the two optimization methods in Figs. 5.4 and 5.5, it is clear that CONMIN suffers more from the problem of getting 'stuck' in the initial cell, particularly along the boundaries downstream of the first sensor set. The better performance of the OPT++ algorithm is due to its tendency to take larger jumps from one iteration to the next. This 'skipping around' frequently enables the trajectory to escape the problematic downstream regions of the functional in order to end up in region more amenable to optimization. However, this tendency to take large, random steps can backfire, as many of the trajectories initiated in the problematic regions end up farther from the minimum than they started due to 'jumping' in a direction that sends it into an even less favorable region of the domain at a distance even further from the minimum. This phenomenon typically does not occur in CONMIN. Rather, when the gradient information is not sufficient to move the trajectory towards the minimum, the trajectory frequently ends up terminating near to the location in which it started.

The Case [4] results are shown in Fig. 5.6. The CONMIN optimization results are the least robust of all the tested cases for this scenario. A mere 2.1% and 1.9% of trajectories are completely successful with respect to all and upstream points, respectively. Most of these successful starting points are confined to the cells in the immediate vicinity of the true source, although a number of downstream points in the center of the channel are also able to successfully converge. Interestingly, this is the only tested case where the percentage of successful trajectories is lowered, albeit slightly, by considering upstream points only. When considering errors in the range $\zeta \leq 0.2$, the CONMIN convergence success increases slightly to 6.1% and 7.2% for all and upstream points, respectively.

OPT++ on the other hand actually performs better with the lower transverse diffusivity than in the isotropic instantaneous release case shown in Fig. 5.4(b). In this case converging with 100% accuracy in 20% and 21% of all and upstream points.

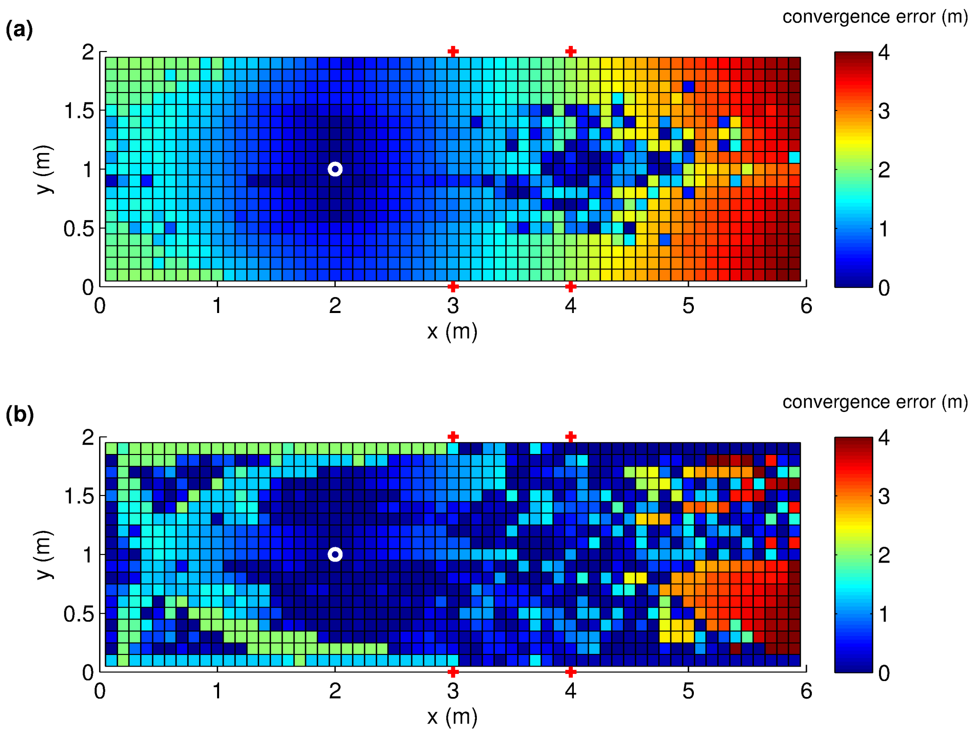


Figure 5.6: Convergence errors for a instantaneous point source and $D_T = 1, D_L = 0.1$ (Case [4]) with CONMIN (a) and OPT++ (b) optimization.

For $\zeta \leq 0.2$, the success increases in 35.9 and 35.1% of all and upstream points, respectively. And as in the previous cases, the mean run-time with Opt++ is longer (4.3 vs. 2.8 minutes).

The difference between the two methods is most pronounced for this final case, with Opt++ clearly outperforming CONMIN. Although the difference in mean run-time is also most drastic, the quicker run-time with CONMIN is irrelevant given its high rate of errors.

### 5.3.1 Comparison and Discussion of Case Results

Compilations of the convergence errors for Cases [1]-[4] are given in Table 5.2 and comparisons of the run-times are shown in Table 5.3. In order to determine if the mean times were correlated to the accuracy of the trajectory (i.e. more accurate trajectories taking longer to converge), the mean times are also compared based on their corresponding optimization convergence errors. For example, the times of all the trajectories that were completely accurate ($\zeta = 0$) are compared in column 2 of Table 5.3. The mean CONMIN run-times were longer for the accurate trajectories in all cases, by an average of 12-48 seconds per trajectory, indicating that the shorter mean convergence times for the CONMIN solver are in part due to the fact that the trajectories tend to terminate prematurely compared to the OPT++ solver.

There is less of a relationship between the mean convergence times and the success of the trajectories in the OPT++ case. While there is a slight increase in mean run-time in Cases [3] and [4] of 30 and 24 seconds, respectively, the successful trajectories on average converged slightly faster in Case [1] and there was no difference in the mean times for the accurate and most erroneous categories in Case [2].

Of course it should be noted that the sample sizes for the accurate trajectories in Cases [2] and [4] are much smaller than that of the 'failed' trajectories using either method. Hence it should be reiterated that in light of the many factors influencing the mean trajectory convergence times, the numbers in Table 5.3 are included for rough comparison purposes and should be viewed as broad indicators only.

Table 5.2: Percentages of trajectories by convergence error

| | | All points | | $x < 4$ m | |
|---|---|---|---|---|---|
| | | $\zeta = 0$ | $\zeta \leq 0.2$ | $\zeta = 0$ | $\zeta \leq 0.2$ |
| Case [1] | CONMIN | 20.9% | 61.4% | 23.5% | 65.6% |
| | OPT++ | 34.6% | 63.1% | 36.7% | 66.8% |
| Case [2] | CONMIN | 2.6% | 8.8% | 3.4% | 12.4% |
| | OPT++ | 15.0% | 23.4% | 17.0% | 26.7% |
| Case [3] | CONMIN | 49.2% | 64.9% | 57.6% | 75.4% |
| | OPT++ | 58.8% | 77.5% | 66.9% | 87.6% |
| Case [4] | CONMIN | 2.1% | 6.1% | 1.9% | 7.20% |
| | OPT++ | 20.0% | 35.9% | 21.1% | 35.1% |

Comparing the percentage of accurate trajectories for all the cases (Table 5.2) it is clear that OPT++ is more reliable and accurate than CONMIN for all of the tested scenarios. The superior performance of OPT++ is most obvious for the relatively problematic functionals in Cases [2] and [4]. As discussed above, this difference can be attributed to the quasi-Newton algorithm's tendency to make 'jumps' in the trajectory that help the trajectory escape the most intractable regions of the functional surface. CONMIN by comparison tends to get stuck quickly in the difficult parts of the domain and as a result ends up terminating the trajectories near the initial test point.

Table 5.3: Two-dimensional source inversion mean convergence times (minutes).

| | | All $\zeta$ | $\zeta = 0$ | $\zeta > 2$ |
|---|---|---|---|---|
| Case [1] | CONMIN | 5.2 | 4.8 | 4.0 |
| | Opt++ | 6.3 | 6.5 | 6.7 |
| Case [2] | CONMIN | 2.7 | 2.6 | 2.4 |
| | Opt++ | 3.9 | 4.0 | 4.0 |
| Case [3] | CONMIN | 5.0 | 5.7 | 4.2 |
| | Opt++ | 6.2 | 6.3 | 5.8 |
| Case [4] | CONMIN | 2.8 | 2.9 | 2.4 |
| | Opt++ | 4.3 | 4.6 | 4.2 |

From the above analysis it is also clear that the functionals in Cases [1] and [3] are more reliably optimized by either of the tested gradient-based methods than the anisotropic cases [2] and [4]. These results can be better understood by examining and comparing the respective functional topographies for all of the test cases. The functionals corresponding to Cases [1]-[4] are shown as surface plots in Fig. 5.7(a)-(d) and as contour plots in Fig. 5.8(a)-(d).

As discussed above, the effect of the anisotropic diffusivities on the functional is to create a long, trough-like area with mild gradients in the region surrounding the minimum, which represents the true source location. While the mild gradients are challenging for the optimizer on their own, an additional encumbrance is the presence of a local minimum in Case [2] (Fig. 5.8(b) and Fig. 5.7(b)) which leads to poor results for both the CONMIN and OPT++ optimizers. To a lesser degree, the corresponding region in Case [4] is also problematic, as it is not quite a local minimum, yet still contains relatively mild gradients in the longitudinal direction

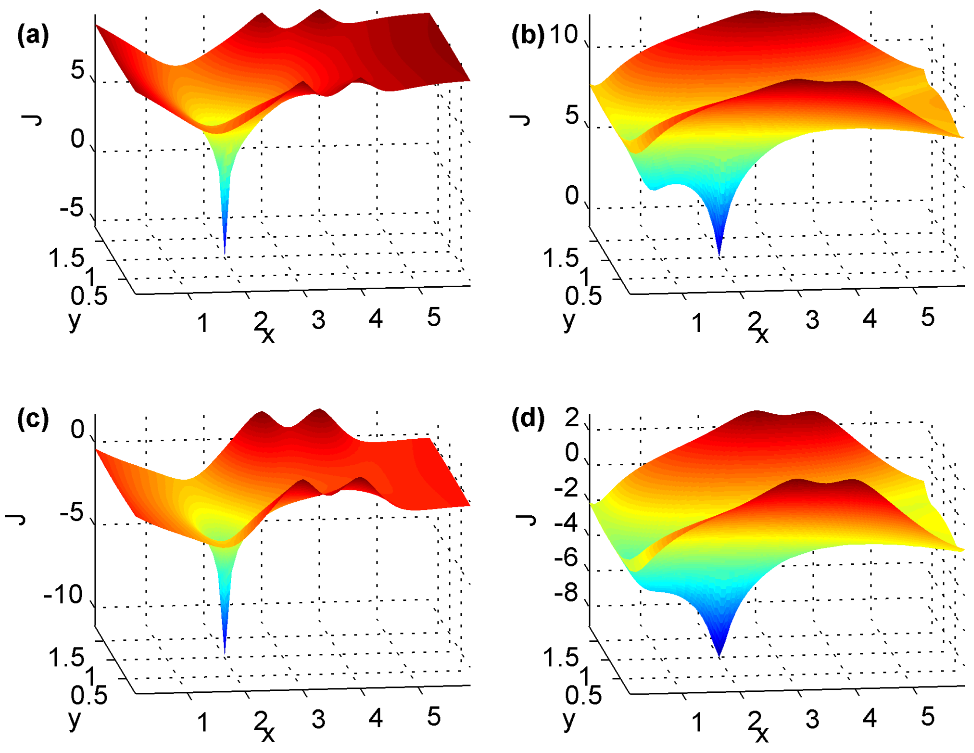


Figure 5.7: Functional surface plots. Subfigures (a)-(d) correspond to Cases [1]-[4].

compared to the transverse direction. The poor performance of the optimizers in these 2 cases, particularly the CONMIN optimizer, may be improved by additional manually scaling of the parameters in DAKOTA as discussed in Chapter III to deal with the strongly asymmetric gradient fields.

## 5.4 Effect of Sensor Numbers and Location

As has already been shown, the performance of the gradient-based optimization is influenced by the selection of the Péclet number in addition to the difficulties in adequately representing the gradient field and the issues related to the discretized parameters. This section discusses another complicating factor in determining the success of the algorithm: the number of sensors and their locations within the domain.

In order to demonstrate the effect that the sensor numbers and locations have on the functional, the functional surfaces corresponding to several possible sensor

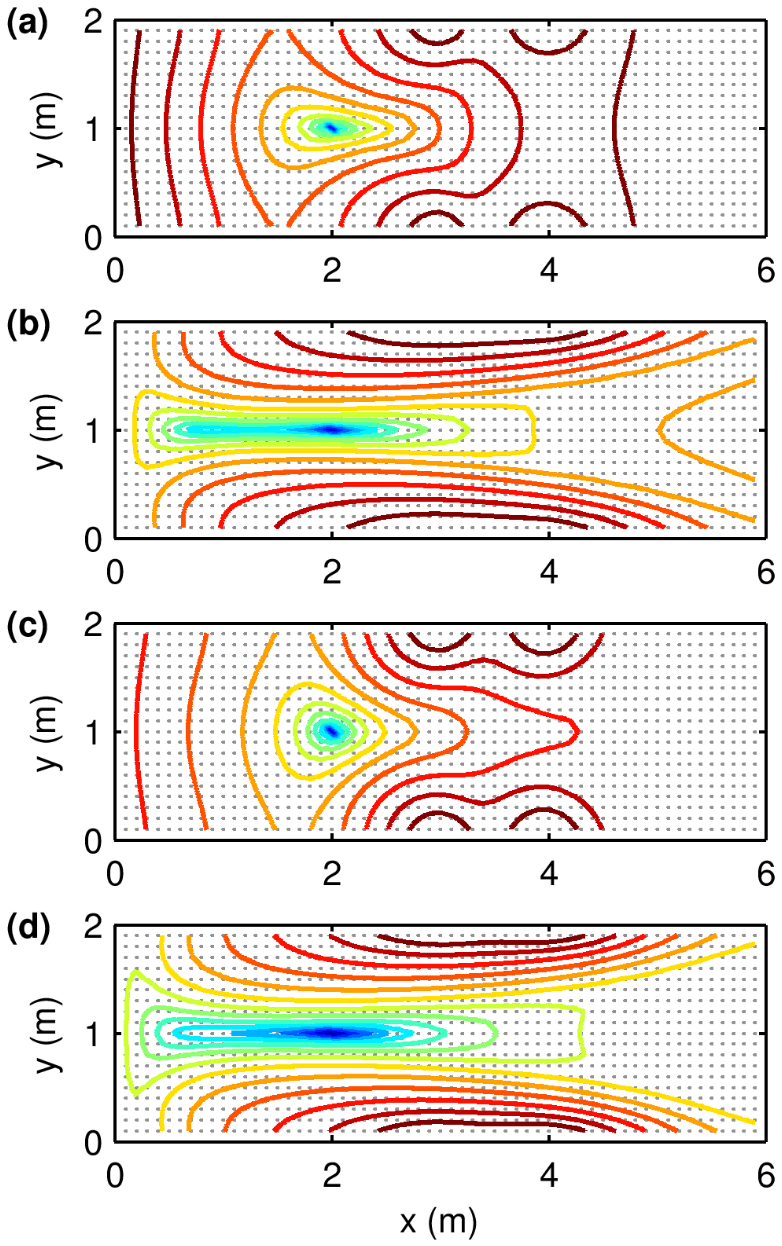


Figure 5.8: Functional contours. Subfigures (a)-(d) correspond to Cases [1]-[4].

configurations are plotted, holding all other factors constant. In each of these cases, isotropic diffusivities of $D_T = D_L = 1$ m$^2$/s are used and the point source is continuous and located at coordinates of $(2, 1)$ as in the previous cases.

Figure 5.9(a) shows the functional surface corresponding to Case [1] above. As was already shown, this functional is relatively easy to optimize compared to the other cases, and the results using gradient-based techniques were fairly accurate regardless of the initial starting point $\mathbf{f}_0$. However the optimization is still not 100% accurate, and small errors result in many of the trajectories. By contrast, Fig. 5.9(b) is a functional surface corresponding to the situation where every discretized point in the domain is defined as a sensor. The complete information field in this case results in an ideal function, with a large convex region that is highly optimizable via gradient-based methods regardless of the initial guess.

At the other extreme, Fig. 5.9(c) shows the functional surface for a scenario in which there is a single sensor located at the boundary at coordinates of $(3, 0)$. In this case the functional surface becomes highly distorted and the region surrounding the true minimum contains small pockets of local minima. The contours corresponding to Fig. 5.9 are also shown for comparison in Fig. 5.10, where there relative gradient fields are easier to discern.

Given the difficulties encountered with the lower diffusivities and/or wider channels, it is reasonable to assume that moving the sensors towards the center of the channel may serve to increase the performance of the gradient-based optimizer. An example of a functional with midstream sensors in shown in Fig. 5.9(d), where the 4 sensors are located at coordinates $(3, 0.75)$, $(3, 1.25)$, $(4, 0.75)$ and $(4, 1.25)$. In general, sensors that are located midstream will create more complex functional surfaces due to the creation of local maxima at the points where they occur, and therefore one of the benefits of placing the sensors at the boundaries is that the functional is more likely to approximate a convex surface. In this example, the position of the

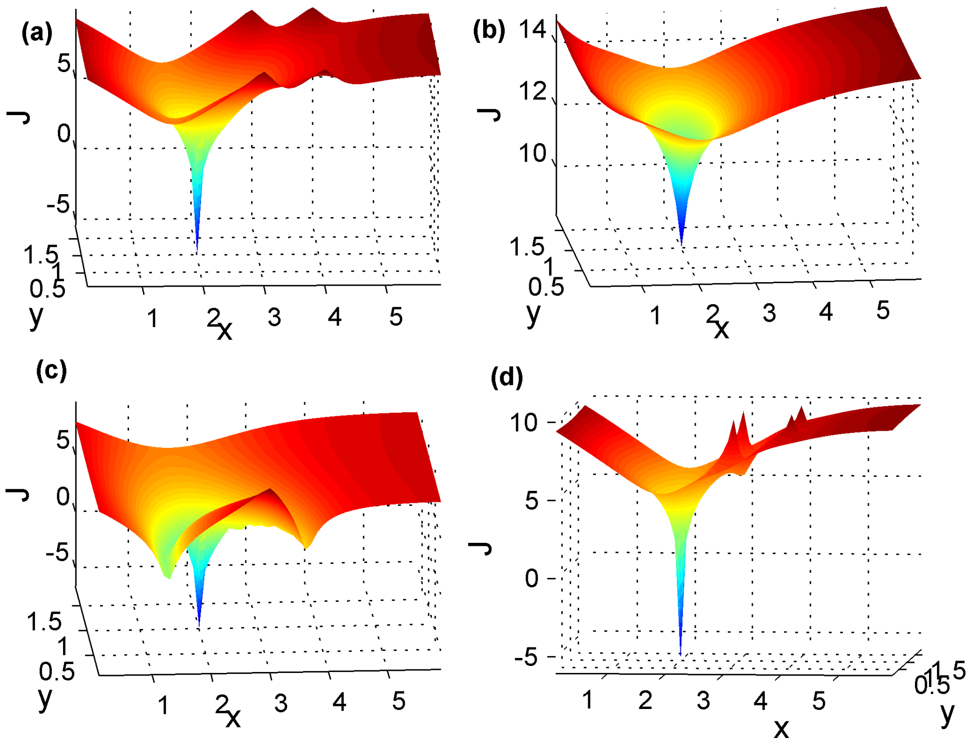


Figure 5.9: Functional surfaces corresponding to 4 boundary sensors (a), sensors located at every point (b), 1 boundary sensor (c) and 4 midstream sensors (d).

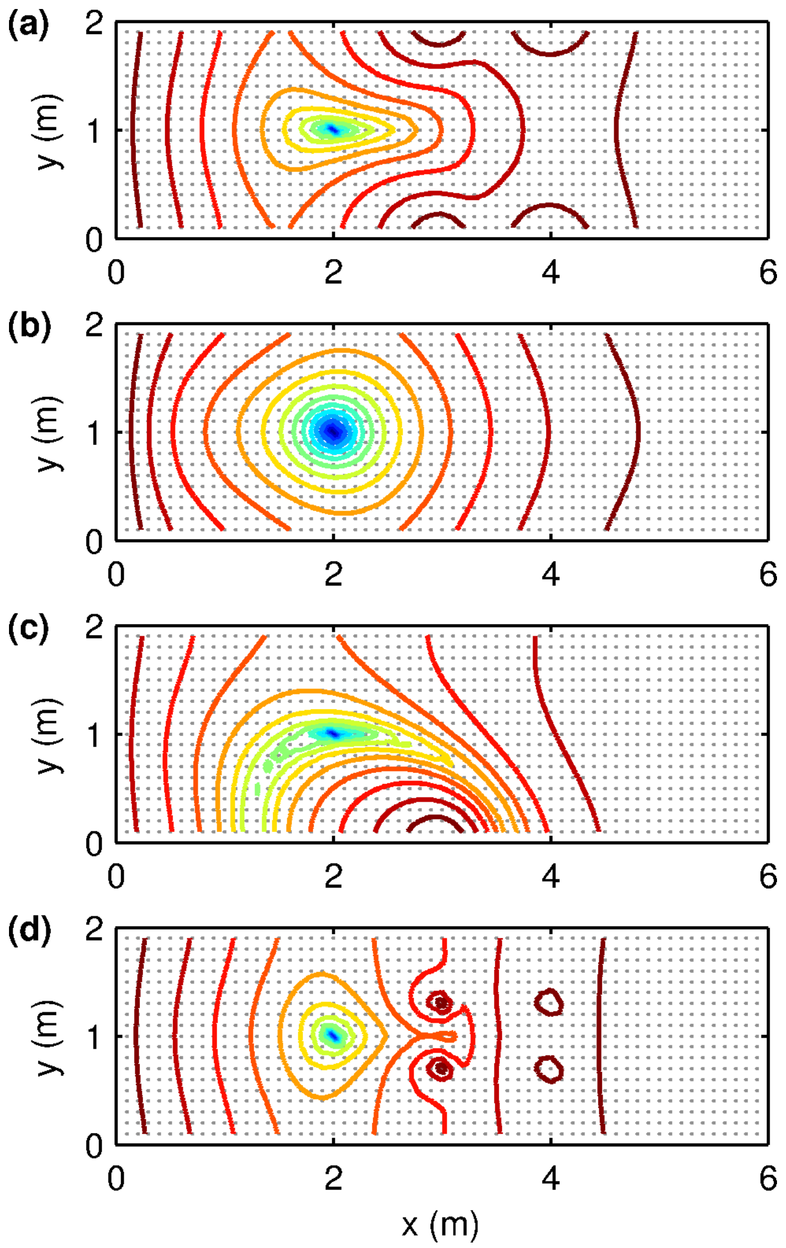


Figure 5.10: Comparison of functional contours with for the case of 4 sensors (a), sensors at every point (b), a single boundary sensor (c) and 4 midstream sensors (d).

midstream boundary sensors results in the development of a local minimum between the two most upstream sensors. Even with the complication of the local minimum however, the overall results are improved for the midstream sensors in this case due to the stronger contaminant concentration signal received at these locations, with the caveat that the starting location should not be in the region surrounding or between the sensors. Using points upstream of the sensors only, the optimization corresponding to Case [1] with midstream sensors and CONMIN optimization is completely accurate in 54.4% of the starting points and accurate to within 0.2 m in 80.6% of the starting points. These results are shown in Fig. 5.11 and should be compared to the Case [1] CONMIN results in Fig. 5.3(a).

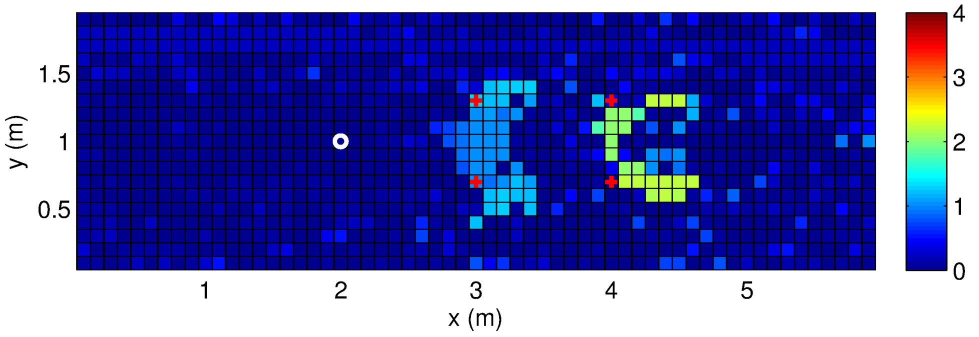


Figure 5.11: Case [1] convergence errors using midstream sensors and CONMIN optimization.

## 5.5 Hybrid Optimization Source Inversion

It is clear at this point that given the myriad obstacles involving the functional sensitivity to input parameters in addition to the discretization and gradient issues, gradient-based methods are not a robust method for the source inversion problem in CFD. Yet while the derivative-free methods introduced in Chapter III offer alternative methods of optimization that circumvent these issues, they come with the disadvantage of typically requiring much longer convergence times. In light of the respective limitations of the gradient-based and derivative-free optimization approaches, it is desirable to combine the benefits of both methods into one, in what is termed a *hybrid minimization* approach. In hybrid optimization, two or more optimization algorithms are applied in succession to a single problem. In this case, first a global, derivative-free minimization routine is used to explore the parameter space and locate an ideal starting point for the gradient-based optimizer. The gradient-based optimizer then begins its iterations with the improved starting guess.

A hybrid approach similar to the one tested in this work was used for large-scale atmospheric source inversion by Addepalli et al. (2001), as was mentioned in the introduction. They first applied a stochastic derivative-free optimizer to find an ideal starting location for the gradient-based optimization, in what they termed the convex-continuous-differentiable region of the functional space. They redefined the objective function for the initial derivative-free search in order to account for the large-scale domain and presence of zero-value measurements at some sensors. Their forward model was a Gaussian Plume model and the gradient-based method they employed was a Newton optimizer. Other researchers (*Mahinthakumar and Sayeed* (2006), *Yeh et al.* (2007)) have applied hybrid optimization approaches to the groundwater contaminant plume problem.

In this work the derivative-free solver used is an Evolutionary Algorithm included in the DAKOTA toolkit: the Common Optimization Library INterface Evolutionary Algorithm (COLINY-EA). A brief overview of Evolutionary Algorithms is given below. More detailed background on EA methodology is provided in the text by Sivanandam (2008).

### 5.5.1 Evolutionary Algorithms

Evolutionary Algorithms (EA) are based on the Darwinian concept of survival of the fittest. The process involves defining an initial *population* which is determined by random seed values. The functional is evaluated at all of these points in the parameter space, and the most optimal, or ‘fit’ parameters sets are retained and allowed to ‘breed’, spawning a new generation of points at which to test. This process continues in an iterative fashion until the convergence criteria is reached. The general EA process is shown in the flowchart in Fig. 5.12.

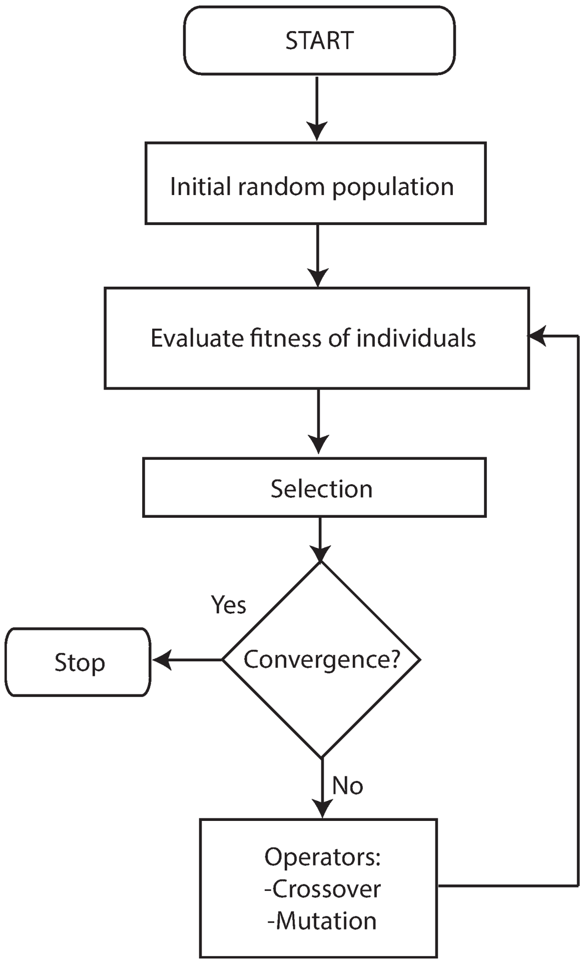


Figure 5.12: Flowchart of a basic Evolutionary Algorithm.

Evolutionary algorithms are stochastic in nature, vs. the deterministic approach adopted by gradient-based optimization. Since they are population-based algorithms, they always store a number of solutions and draw from these previous values in order to come up with new and improved parameters. The main operators used to do this are *crossover* and *mutation*. Crossover is an operator that recombines one or more parent solutions in order to generate ‘children’. Mutation is an operator which perturbs a solution, resulting in a new solution to be tested. The main difference between these two operators is that the crossover operator is applied to several solutions while mutation deals with only a single solution.

While there are several variables that influence the EA process, such as the crossover rate, mutation rate, fitness type, crossover type, replacement type, initialization type and mutation type, one of the most basic user-defined parameters affecting the outcome of the EA is the population size. If the specified population size is too small, the EA output may not be a sufficient initial parameter combination for the gradient-based optimizer, leading to the same issues that were encountered with the straight gradient-based optimization methods implemented above. However, setting these values too large will result in an unacceptably long run-time.

A final important distinction between gradient-based and derivative-free optimization is in reference to the initial starting point. Where the result of the gradient-based optimization is highly dependent on the initial starting guess, the Evolutionary algorithm's method of 'seeding' the domain with a specified number of test points generated at random locations leads to an outcome of the EA optimization procedure that is independent of the initial location chosen. This is a significant benefit over the straight gradient-based approach. Additionally, non-ideal functionals with complicated topography are less problematic, given that the initial population is large enough to adequately explore the parameter space.

### 5.5.2 Hybrid Results for the Two-Dimensional Cases

The 4 cases that were presented in Section 5.3 are repeated in this section using the hybrid approach. Each case is again assessed using CONMIN and OPT++ as the gradient-based algorithm in the hybrid process. With the goal of reducing the overall convergence times, the hybrid method was initially tested using a single iteration of the EA and a conservative population size of $P_0 = 10$. Due to the stochastic nature of the EA and independence of the hybrid results with respect to the starting location, the accuracy and robustness of the hybrid method must be assessed in a different manner than for the gradient-based only simulations. Therefore to investigate the accuracy and reliability of the hybrid method each source inversion case was repeated for a total of 100 trials. The statistics regarding the convergence accuracy and computational time based on these trial results are shown in Table 5.4. The convergence errors for each of the cases are also depicted in terms of histograms in order to visualize the accuracy and reliability of the hybrid method.

The final convergence errors for Case [1] are shown in Fig. 5.13, were subfigures (a) and (b) again correspond to CONMIN and OPT++ respectively. Overall the mean and maximum convergence errors were similar for CONMIN and OPT++, but a higher percentage of the inversion runs resulted in $\zeta \leq 0.2$ in the OPT++ case, at 80% vs. 69% for CONMIN. The mean convergence times were very similar as well, both taking a little over 3 minutes per inversion.

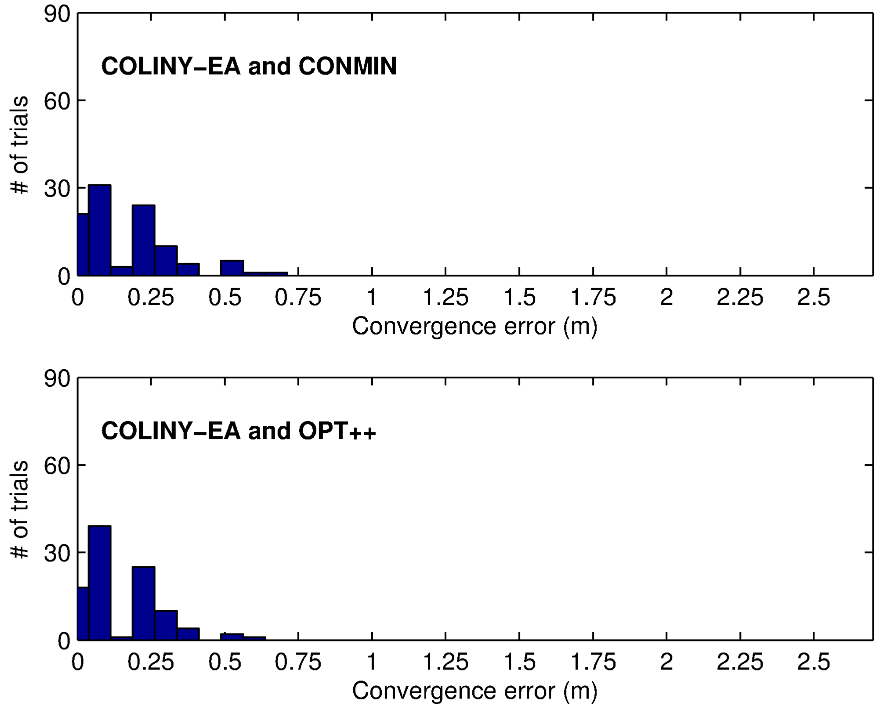


Figure 5.13: Case [1] histograms of the convergence error using the hybrid optimization approach.

The hybrid results corresponding to Case [2] are shown in Fig. 5.14. As for the straight gradient-based optimization results, this source inversion was less successful for this case than for the first. Overall there is little difference between the two gradient-based optimization methods, with both leading to roughly a quarter of the 100 trials terminating with $\zeta \leq 0.2$ in approximately 3 minutes. The similar results CONMIN and OPT++ for this case and the others is not surprising. Since the main benefit of OPT++ is in regards to successfully navigating the optimization trajectory out of less favorable regions of the functional surface, it is expected that this benefit will not be as important in the hybrid approach, where ideally the gradient-based optimizer does not 'take over' the optimization process until the EA has located a point in the general convex region of the functional for it to start its trajectory.

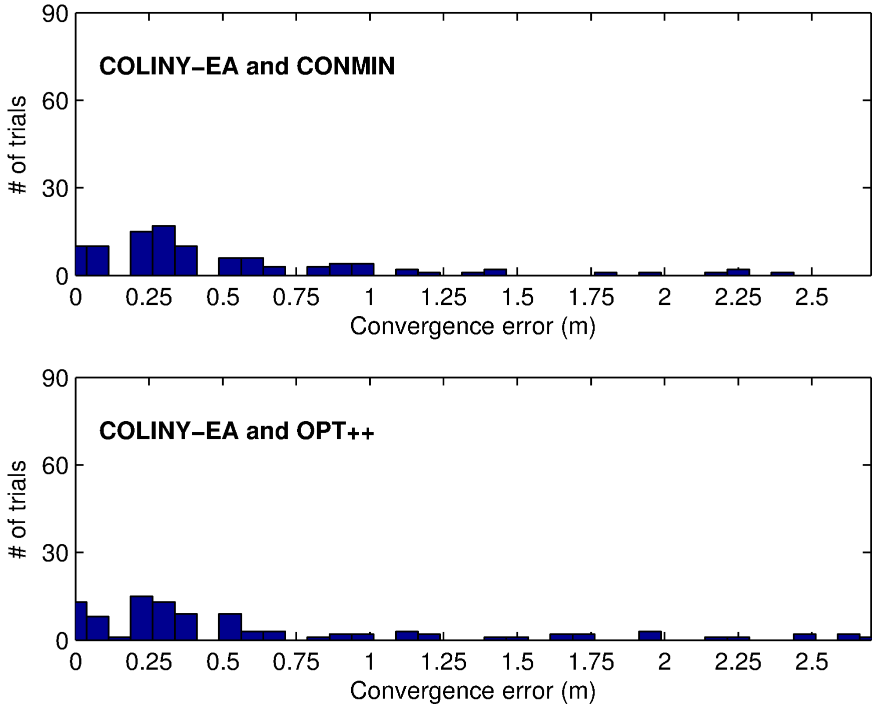


Figure 5.14: Case [2] histograms of the convergence error using the hybrid optimization approach.

It also can be seen from Fig. 5.14 that there is quite a bit more spread in the convergence errors as compared to the results of Case [1]. While in Fig. 5.13 the errors were all confined to $\zeta < 0.75$, there were several hybrid optimization runs that resulted in errors up to 2.75 for Case [2]. From a reliability standpoint, these findings are not much better than the gradient-based only optimization results, and in fact, due to the random outcome of the hybrid optimization process, can be considered worse. That is, given one optimization run, there is a greater level of predictability and control in the gradient-based only approach compared to the hybrid approach. In the former case, if additional information about the flow, sensor locations and contaminant plume can be used to reduce the entire domain to a smaller region that is likely to contain the source, the gradient-based method is more likely to be successful than in the latter case, where the initial guess is not needed but where there is a risk that the hybrid optimization will randomly converge with a large error.

Table 5.4: Statistics generated from 100 hybrid source inversion test runs.

| Case | G-B solver, notes | mean($\zeta$) | max($\zeta$) | $\zeta \leq 0.2$ (% of trials) | mean time (min) |
|---|---|---|---|---|---|
| [1] | CONMIN | 0.17 | 0.70 | 69 | 3.27 |
| | Opt++ | 0.15 | 0.60 | 80 | 3.35 |
| [2] | CONMIN | 0.52 | 2.38 | 24 | 2.85 |
| | Opt++ | 0.63 | 2.75 | 25 | 3.21 |
| [3] | CONMIN | 0.02 | 0.30 | 98 | 5.81 |
| | Opt++ | 0.04 | 0.41 | 92 | 3.95 |
| [4] | CONMIN | 0.70 | 1.90 | 19 | 3.17 |
| | Opt++ | 0.71 | 1.70 | 23 | 3.27 |
| | Opt++, 3 iters | 0.54 | 1.40 | 33 | 4.01 |
| | Opt++, 3 iters., 2x pop. | 0.41 | 1.50 | 38 | 5.57 |
| | Opt++, 3 iters., 3x pop. | 0.44 | 1.50 | 33 | 5.46 |

The Case [3] hybrid results are shown in Fig. 5.15. The outcome of the 100 test trials was highly successful with this scenario, with both the gradient-based optimization methods converging to $\zeta \leq 0.2$ in over 90% of the trials. This example is also the sole case where CONMIN outperformed OPT++, although this finding is subject to the randomness of the trial outcomes in general. CONMIN also had the longest of the hybrid mean run-times in this case: 5.81 minutes vs. 3.95 minutes for OPT++. In both cases, the maximum errors were confined to $\zeta < 0.41$. It is difficult to draw direct comparisons between the gradient-based only trajectory results and the hybrid results due to the fact that the odds of success in the former case are to a large degree a function of the starting location, while in the latter case they are more random. However, given that the maximum error in the hybrid results for this case is still relatively low, it can be concluded that the hybrid optimization is more reliable in this case given that a 'good' starting guess is not known for the gradient-based optimizer.

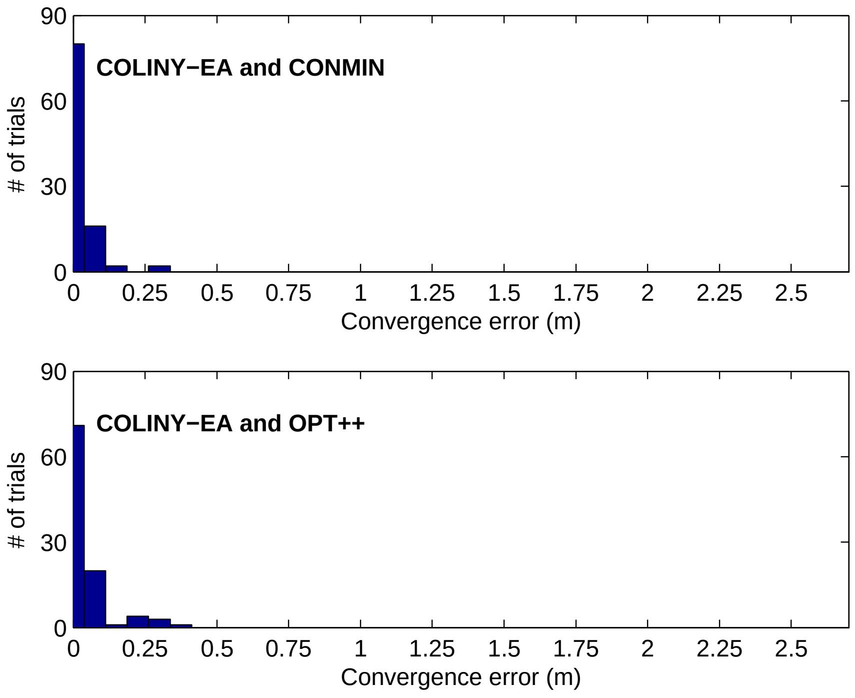


Figure 5.15: Case [3] histograms of the convergence error using the hybrid optimization approach.

The hybrid results for Case [4] are shown in Fig. 5.16. The results for this case are also very similar between CONMIN and OPT++, and overall quite similar to the results in Fig. 5.14, although the maximum errors in this case are slightly less (1.9 with CONMIN vs. 2.75 via OPT++ in Case [2]). Again there is quite a bit of spread in the optimization errors and the odds of the hybrid optimization converging to the correct source location are not much higher than the odds of the method converging to a location nearly 1.5 meters away. Again, when drawing conclusions between the reliability of the hybrid methods and the gradient-based only methods, the availability of a good starting guess must be taken into consideration. Fig. 5.6

shows that for using the OPT++ solver, there is a radius of convergence of roughly 0.6 meters, in which the trajectory will converge to either the exact source location within a very small error range if the gradient-based optimizer is initiated in that region. Additionally, the OPT++ method is able to successfully converge for several points throughout the rest of the domain. When comparing these findings to the hybrid results using OPT++ or CONMIN, it can be concluded that the results of the hybrid method are much less predictable, as in many cases the final output of the EA is not in a region close enough to the radius of convergence to allow for the gradient-based optimizer to successfully converge.

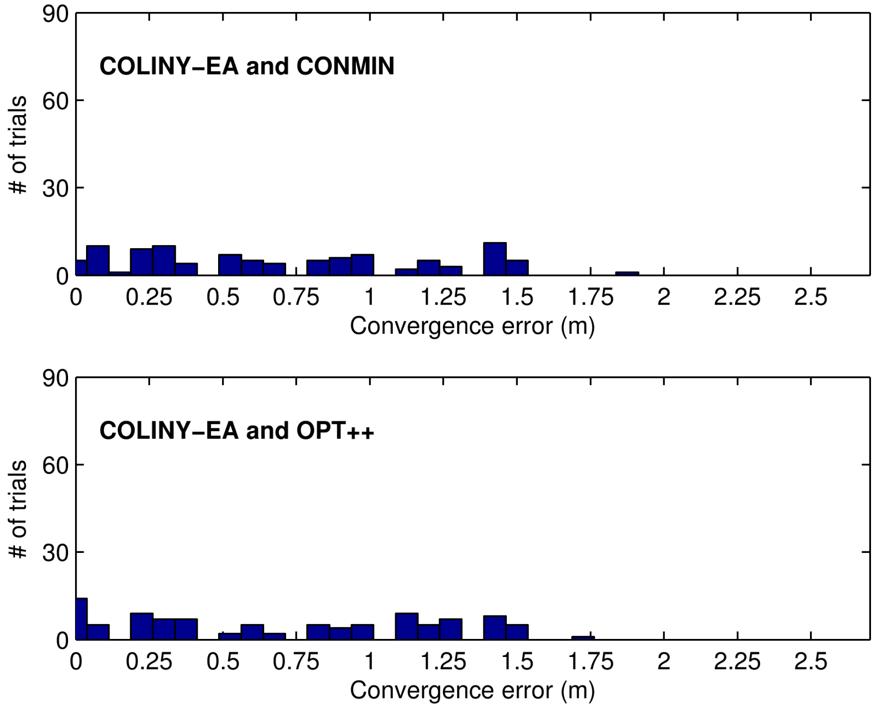


Figure 5.16: Case [4] histograms of the convergence error using the hybrid optimization approach.

In order to determine if increasing the number of iterations or the population size could increase the accuracy of the hybrid approach, 3 more variations of the OPT++ Case [4] simulations were conducted. In the first, the number of iterations was increased from 1 to 3. In the 2nd variation, the increase in iterations was combined with doubling the initial population from 10 to 20. Finally, the 3rd variation included

3 iterations and an initial population of 30. The results of these trial runs are also included in Table 5.4.

While increasing the number of iterations led to a slight decrease in the mean and maximum errors, adding to the initial population had little effect other than increasing the run-time. It can be concluded that the number of iterations is a more important factor than the initial population size in this case. The possibility of further increasing the number of iterations to increase the reliability of the hybrid method exists, although it will coincide with longer times to convergence.

## 5.6 Model Predictive Control Approach

A final variation of the gradient-based source inversion is presented in this section, based on the concepts applied in the field of Model Predictive Control (MPC). Model Predictive Control describes the process by which a predictive (forward) model is used to optimize a trajectory over a defined simulation period, known as the *finite prediction horizon*. However, the output of the optimization is not the optimal conditions at the end of this prediction horizon. Instead, the output of the optimization trajectory at the first step is used to update the optimization problem with new information and reiterate the procedure. Thus, MPC optimizes over a designated period ahead of the current time period, but regularly corrects the optimization trajectory to account for the current conditions as well as the future predicted behavior of the system given by the forward model, allowing for the continuous incorporation of new information into the optimization problem. The theory of MPC is discussed further in the Boundary Control section of this work (Chapter VII).

Applied to the source inversion problem, the MPC approach tackles the issue first introduced in Chapter III regarding the discrepancies between the optimizer and CFD optimization trajectories. As was discussed in Section 3.6.1, this problem arises due the discrete spatial and temporal parameters and the black-box coupling between the optimizer and CFD model. These discrepancies were accepted as part of the discretization errors in the above simulations. However, these errors can be mitigated via the MPC approach, by which the optimizer makes a single trajectory step, the CFD model finds the nearest cell-centered point to the endpoint of that trajectory, and the optimizer is re-started from the cell-centered grid point, thus eliminating the discrepancy between the location the optimizer believes it is obtaining information and the actual location being evaluated in OpenFOAM.

This process is illustrated in Fig. 5.17, where the green dots denote the optimization initial trajectory points and the red dots indicate the final trajectory points after one iteration of the optimizer. The solid blue lines show the path that the trajectory takes over one iteration of the optimizer, while the dashed blue lines show the correction that is made from the optimization-determined trajectory point to the nearest cell center on the grid.

This approach was applied to two cases of the above two-dimensional source inversion simulations: Case [1] and Case [4]. In both cases, OPT++ was used as the optimization method.

The results of the MPC Case [1] simulations are shown in Fig. 5.18. Two interesting things are notable in these results compared to Fig. 5.3(b). First, a significantly higher percentage of all the trajectories converge with no error with this approach: 44% vs. 35% for the continuous trajectory approach implemented above. When considering the points upstream of 4 m only, the percentage of successful trajectories becomes 52% for no error and 73% for $\zeta \leq 0.2$, compared to 37% and 67% without the MPC approach. Second, there is much more of a dichotomous relationship between the values of the errors. The downstream, boundary points all tend to converge within an error of approximately 2.7 meters, with the remainder of the trajectories converging to within 0.6 m of the minimum. There are no error values in between these two error brackets, which indicates that the MPC approach tends to lead to

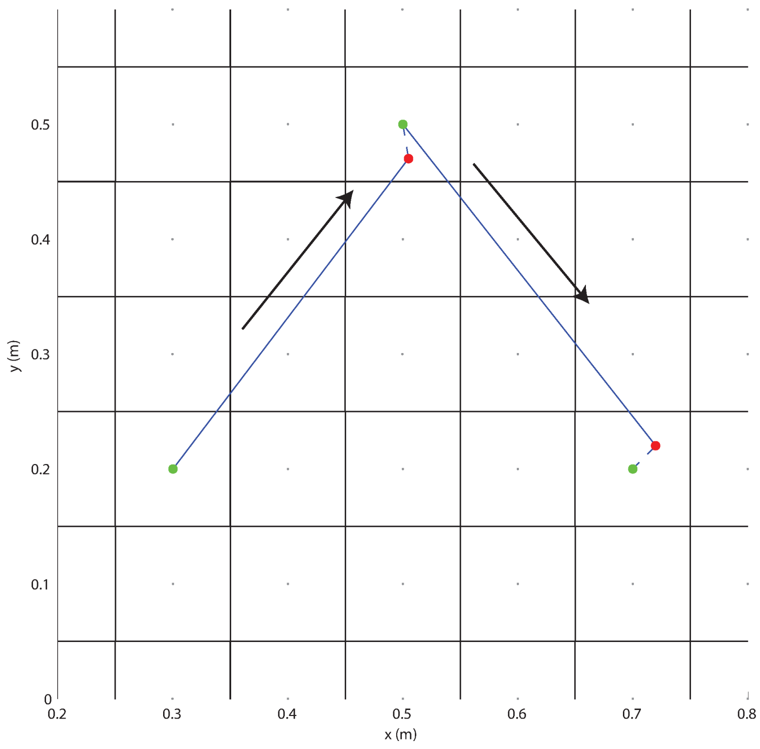


Figure 5.17: Illustration of the process of stopping and restarting the trajectory after each iteration in order to correct for the discrepancy between the optimization-requested test points and the grid-discretized test points.

more of a ‘hit or miss’ outcome.

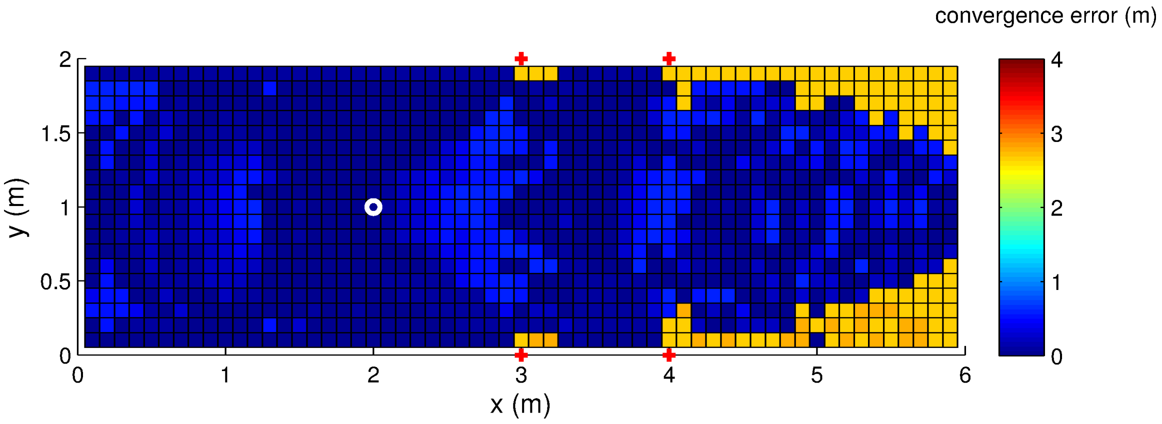


Figure 5.18: Case [1], OPT++ results using the MPC approach of restarting the trajectory after each optimization iteration to correct for the discrete vs. continuous parameter discrepancy.

Overall the MPC method improves the accuracy of the optimization for this case, however the convergence times are also longer. The run-times with the stopping and restarting of the optimizer average 9.1 minutes for all trajectories, vs. 6.3 using the continuous trajectory approach. The completely successful trajectories averaged 9.0 minutes for convergence, while the unsuccessful trajectories took even longer at an average of 14.9 minutes for $\zeta > 2$. This long run-time can be attributed to the method of implementing the MPC procedure, by which the process is not terminated until the change in the output remains the same for 3 iterations. This is a problem when the trajectory oscillates back and forth between two or more cells for a series of steps, which is a frequent occurrence with problematic trajectories. A maximum number of iterations is specified to kill the optimization in these cases, and thus the lengthy run-times are the result of the optimization process continuing until the termination criteria is reached.

Figure 5.19 shows the results of the MPC approach for the OPT++ Case [4]. The use of the MPC approach has a very different outcome in this case, reducing the total percentage of completely accurate trajectories from 20% with the continuous trajectory approach shown in Fig. 5.6(b) to a mere 1%. The percentage of total trajectories in which $\zeta \leq 0.2$ is reduced to 5% from 36%, and the upstream only percentages that meet the $\zeta = 0$ and $\zeta \leq 0.2$ criteria are reduced to 2% and 7% from 21% and 35%, respectively. These numbers are very similar to the percentages for the CONMIN Case [4] in Table 5.2.

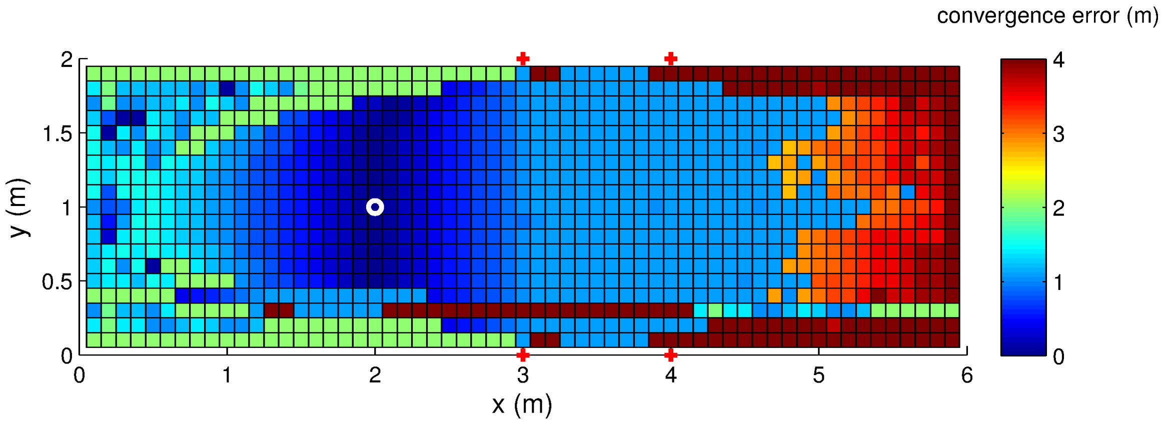


Figure 5.19: Case [4], OPT++ results using the MPC approach of restarting the trajectory after each optimization iteration to correct for the discrete vs. continuous parameter discrepancy.

One reason that the MPC approach leads to a worse outcome in this case can again be explained by the particulars of the OPT++ quasi-Newton algorithm. As has been discussed, the OPT++ optimizer tends to take a larger jumps in its trajectory to sample distant points, especially when the optimizer is unable to retrieve sufficient information after several consecutive iterations. When the optimization is restarted at every iteration the algorithm does not recognize when it is struggling and hence does not initiate the jumps in trajectory that in many cases allow the optimization to converge to a greater degree of accuracy. Since the functional in Case [1] is more ideal, the optimizer did not have to initiate the large steps and therefore the MPC approach was beneficial. However, the functional in Case [4] is the most difficult of the four test cases to optimize with respect to both solvers and in this case the OPT++ optimizer frequently needs the benefit of the continuous trajectory approach in order to proceed towards finding the minimum.

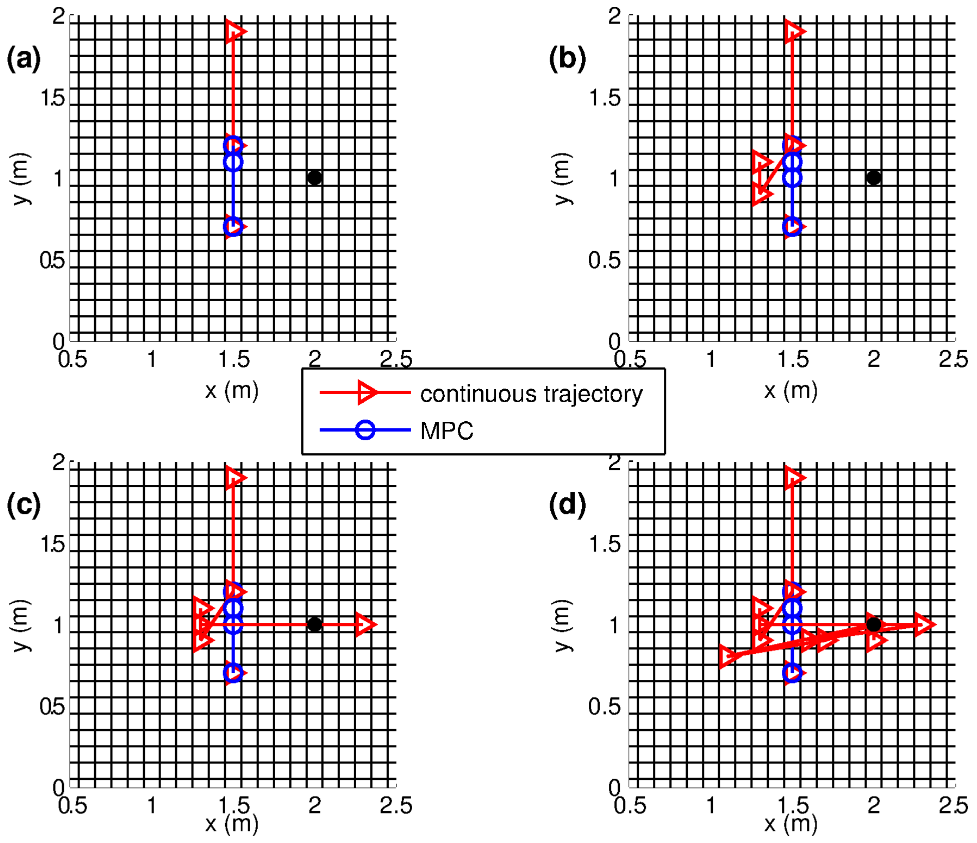


Figure 5.20: Example of the difference in trajectories using the continuous trajectory approach vs. the MPC approach of restarting the optimization after every iteration at the nearest cell center to endpoint of the previous step.

An example of this is shown in Fig. 5.20, which compares the trajectories for the MPC and continuous trajectory methods in Case [4] from a starting point of $(1.5, 0.7)$. Subfigures (a)-(d) correspond to 2, 4, 6 and 21 trajectory steps for the continuous optimization method. The MPC approach terminates after 6 iterations and thus is shown at 2, 4 and 6 steps in subfigures (a)-(c), with no change in its trajectory from subfigure (c) to (d). After 2 iterations (Fig. 5.20(a)), the MPC trajectory lands at $(1.5, 1.1)$ while the continuous trajectory ends up one cell above at $(1.5, 1.2)$. After 4 iterations, the continuous trajectory is actually further from the minimum (denoted by the black dot) at coordinates of $(1.3, 1.1)$, while the MPC trajectory has gotten stuck at the point $(1.5, 1)$. The MPC trajectory stays at this point until the optimization is terminated after 3 iterations of the same values being returned. The continuous optimization trajectory however, continues to progress. The trajectory lands on the minimum after 9 iterations, tests several nearby points, and converges correctly (Fig. 5.20(d)).

Aside from illustrating the tendency of the continuous optimization to take larger steps due to the algorithm's assessment of the recent iteration values, this example also suggests that small discrepancies between the optimization and CFD trajectories can actually be beneficial for problematic functionals, as they keep the trajectory progressing instead of prematurely terminating due to the optimizer getting stuck in a single cell. That is, even if the optimization is temporarily sent in the opposite direction of the minimum, the trajectory may end up in a location that allows for it to converge correctly.

It also must be noted that the MPC method tested here did not incorporate the nudging scheme implemented in the above, continuous trajectory results, and hence the less accurate results in this case can also be in part attributed to this omission.

## 5.7 Summary and Discussion

In this chapter, the performance of several optimization approaches were compared for continuous and instantaneous releases in a two-dimensional channel. The discretization problems demonstrated in the previous chapter in addition to the occurrence of non-ideal functional surfaces have been shown to lead to significant difficulties with the gradient-based methods in some cases. The problem parameters, including the Péclet number, number and location of sensors, are factors that greatly influence the shape of the functional and thus its inherent optimizability. As a consequence, gradient-based source inversion methods exhibit various degrees of reliability and robustness. For some cases, such as a mid-range Péclet number with the presence of an adequate number and location of sensors, both the tested gradient-based inversion algorithms (CONMIN and OPT++) are quite robust, especially when considering starting points in the region upstream of the most downstream sensors. For other cases, such as lower Péclet numbers and anisotropic diffusivities, gradient-based methods in general are only reliable when the initial guess is very near the true source.

Overall, the quasi-Newton algorithm implemented in OPT++ was found to perform better than the FRCG method implemented in CONMIN for the tested source inversion problems. Considering the results in the previous chapter for the Ackley function this finding is somewhat surprising. In that example, CONMIN was shown to be more accurate and reliable for the Ackley functional, and given that the flatter outer regions of that functional more closely resemble the problematic functional surfaces in this chapter, it could be conjectured that CONMIN would also outperform OPT++ for the source inversion problems. However, the Ackley function also was uniformly graded in the x- and y-dimensions, which significantly facilitates the optimization procedure. One possible explanation for the more erroneous results with CONMIM is that the CONMIN algorithm struggles more with the asymmetry in the gradient field than OPT++. This issue of asymmetry can likely be improved by implementing additional manual adjustment of scaling of the $x$ and $y$ parameters in the DAKOTA framework, although outside of the automatic scaling that DAKOTA offers, this possibility is not a very feasible one from a practical point of view. This is because while methods of tuning the optimization routine to a specific problem can greatly increase its performance, in the source inversion problem it is assumed that very little is known about the functional a priori, and therefore tailoring the optimizer to the problem is not a viable possibility in a realistic source inversion scenario. Instead, methods need to be developed that are as robust and reliable as possible for the most general of source inversion problems.

While OPT++ is the more accurate of the two tested gradient-based methods, this algorithm also takes a longer average time to converge in nearly every tested case. While efficiency is a highly sought-after goal of the source inversion algorithm, its value is negated if the results of the inversion are highly unreliable. Therefore, CONMIN's faster convergence times are not benefits in light of the lower overall performance of that method.

The overall run-times for each trajectory using gradient-based methods were in the range of roughly 3-6 minutes, although as has been stressed previously, these times could likely be greatly improved by running the trajectories on a faster processor and with no other processes being run at the same time.

A hybrid source inversion optimization approach similar to the method used by Adepalli et al. for their Gaussian Plume source inversion was extended in this work to the CFD model source inversion. When using the hybrid approach, the differences between the CONMIN and OPT++ algorithms are largely eliminated, as when the gradient-based trajectories are initiated in the near-source region their behavior is similar. Surprisingly, the overall run-times using a single iteration of the EA and a conservative population number of 10 were on average shorter compared to the gradient-based only mean run-times. Hence for a small-scale problem at least, the benefits of starting the gradient-based optimizer in a more favorable location outweigh the costs of running the initial EA run. However, the results of the hybrid method in many cases, particularly in regards to the 'difficult' functionals in which the transverse diffusivities were lowered, are found to be highly variable and unpredictable. This is in part due to the random nature of the EA algorithm, in which the various starting population points may or may not lead to a starting guess for the gradient-based optimizer that is in a favorable region of the functional. This is more likely to be a problem with the functionals that do not resemble an ideal, convex surface. In these cases, the presence of local minima and/or regions of low functional values can pose a challenge to the derivative-free methods finding a point within a convex region surrounding the minimum. Also, the extremely small population and number of iterations used in these tests undoubtedly is a major factor leading to some of the poorer trajectory results. Increasing these parameters however, will likely lead to an unacceptable convergence time.

The effect of regularly correcting for the errors between the optimization trajectory and the cell-center tested points in the CFD model was also explored in this chapter. While the results were greatly improved for an optimization problem characterized by a more ideal functional surface, the results for a problematic functional inversion were much less robust than the continuous trajectory approach. One problem with this approach when employing the OPT++ solver is that the underlying quasi-Newton algorithm, as has previously been shown, tends to take large jumps in the trajectory. This skipping around frequently occurs after the trajectory oscillates back and forth between two nearby points for several iterations. Instead of terminating the trajectory when adequate progress is not made as CONMIN tends to do, OPT++ recognizes when the trajectory is 'stuck' and takes a large step in an attempt to gather new information. When the optimizer is re-started at every iteration, it doesn't have the previously stored records of failed attempts at the same locations, and thus it not signaled to make the large step that frequently sends it to a more favorable region of the domain. For this reason, updating the trajectory with the true test point at every iteration actually may lead to poorer results for the OPT++ algorithm with problematic functional surfaces.

# CHAPTER VI

# Source Inversion in Higher Dimensions

The previous chapter dealt with the contaminant source inversion problem in two spatial dimensions, making the assumption that the source strength and release time were known in addition to the other flow parameters. While these results demonstrate several of the numerical challenges encountered in the discretized source inversion algorithm and provide insight into the differences in its performance as a function of optimization method and starting guess, in a realistic problem additional parameters must be recovered from the source inversion routine in order to fully reconstruct the contaminant plume. Depending on the specifics of the contaminant release scenario, the additional required parameters are likely to include the time of release, the source strength, and the third spatial dimension. This chapter addresses the inclusion of additional parameters to the source inversion routine and the particular challenges that are faced in regards to optimization with respect to larger parameter spaces.

From a programming standpoint, the inclusion of additional source inversion parameters is the relatively simple matter of adding the variables to the optimization routine along with their respective limits, and, in the case of applying gradient-based optimization, including the estimation of their gradients. However, in practice there are two significant impediments to inversely solving the contaminant transport problem with increasingly large parameter spaces. The first is a computational one:

regardless of the inversion technique, simply increasing the parameter space of the inversion problem will add to the computational burden of the optimizer. Further, for gradient-based optimization, each additional unknown parameter requires an additional run of the forward model in order to approximate the gradient of the objective function with respect to that parameter. This leads to the source inversion run-times becoming increasingly less feasible for real-time applications as the number of inversion parameters increases. The third spatial dimension in particular adds a significant amount of run-time to the inversion process. The problem of increased run-time and required computational resources could be dealt with by exploring methods of improving the efficiency of the model, such as using adjoint variables to compute the gradient field. This is a possibility that is discussed in Chapter IX.

The second, and more serious, problem involves the tractability of the functional itself. With each additional parameter added to the optimization process, the topography of the functional becomes increasingly complex. As has been shown in the previous chapter, the source inversion process is very sensitive to the characteristics of the functional, and gradient-based methods in particular require a well-posed functional that has the properties of continuity and convexity to facilitate successful convergence. Extended to the higher-parameter space source inversion functionals, additional methods such as restricting the search domain and improving the starting guess are necessary in order to avoid many of these issues. However, even taking these measures will not guarantee that the functional is well-posed, as will be shown in this chapter.

Several examples are shown in this chapter that illustrate the performance of the source inversion algorithm for problems of three or more parameters, and results are shown for different combinations of inversion parameters and varying methods of optimization. Functionals incorporating four or more unknown parameters are impossible to plot, and it is therefore more difficult to visualize the problems that are created by the more complex functionals that result from these cases. However, functionals of three variables can be plotted and analyzed in a similar manner as the two-dimensional functionals when examined in the previous chapter, and several are included below for this purpose.

## 6.1 Modeling Specifics

This section defines the modeling details used in the following source inversion simulations. The time-step implemented is $\Delta t = 0.04$ s, which maintains the Courant number at around 0.80. The computational grid is the same as the channel used in Chapter V, shown in Fig. 3.2.

As mentioned above, the inclusion of additional parameters in the source inversion problem requires that they be added to the optimization routine with their respective bounds. Gradient estimates of $J$ with respect to these parameters must be carried out when using the gradient-based solvers, and therefore a step-size $h$ also must be defined for these additional parameters. The importance of the initial guess has been stressed in previous chapters, and although it has also been shown that several random choices for the initial parameters may result in zero convergence error, the most robust optimization results tended to occur when the initial guess is relatively close to the true parameters. Thus, in order to expedite the optimization process and to increase the chances of a successful inversion, it is desirable to reduce the parameter space as much as possible using the information that is known about the source. To this end, information deduced from the sensor data can be used to make a logical first guess for the optimization trajectory, and the assumption that the ambient flow field is known and steady can be also used to improve the initial trajectory guesses. With this in mind, the bounds for the parameters are estimated using the information assumed to be known about the problem a priori, and the logic behind these choices is explained herein.

### 6.1.1 Parameter Bounds and Gradient Step-Size

We begin with the bounds on the time and spatial dimensions, which can be conservatively restricted using logic in regards to what is known about the problem. For the spatial parameters, as was discussed in Chapter V, in a uni-directional flow field it is expected that the source will have originated upstream of the x-coordinate of the sensor with the highest $C$ peak as determined by the simulated sensor data. This coordinate represents the maximum value for $x_s$ and is denoted as $x_{max}$. Likewise, the time of release cannot exceed the time of the first detectable sensor measurement. This time provides a maximum for the bounds of $t_s$, and is denoted as $t_{max}$. However it has been found in this work that the optimizer performs better with the three-parameter space problems when it is allowed to test points slightly above the time designated by $t_{max}$, and for this reason a small value $\delta t$ is added to $t_{max}$ in some of the following examples to increase the upper bound slightly above this limit.

The gradient step-sizes for the spatial dimensions are given by the cell-to-cell distance **ds** as before. With respect to the time dimension, it is also logical to use the time-step, $\Delta t$, as the interval over which to approximate the gradient, as this is the minimum possible $h$ for this discretized parameter. In the time-dimension the magnitude of the gradients tends to be significantly smaller than the gradients in the spatial dimensions for many evaluated parameter combinations. It is therefore necessary to use scaling with this parameter to force its evaluation relative to a common scale with respect to the spatial and source strength parameters.

Unlike the spatial and temporal parameters, the initial source strength $q_s$ is treated as a continuous variable in the optimization routine, and hence its bounds and gradient step-size $h$ are arbitrary as they are not restricted by the discretization of the problem or the physical bounds of the system. However the choices for the maximum and minimum source strength and gradient step-size should be chosen so that adequate information in terms of the sensor measurements and the gradient estimations is provided to the optimizer. That is, the true source strength should be strong enough to be detected by the sensors, and the gradient step-size used in the optimization process should be large enough to result in a significantly measurable change in the objective function with each gradient evaluation. However, a step-size too large will lead to a poor representation of the gradient field, and there is a risk of leading the trajectory in an erroneous direction based on a large gradient step, especially when dealing with the complex, higher parameter-space functionals. The choice of $h = 20$ units used for the examples below has shown to generally be small enough to prevent extreme errors in the gradient field while large enough to lead to a measurable change in the objective function.

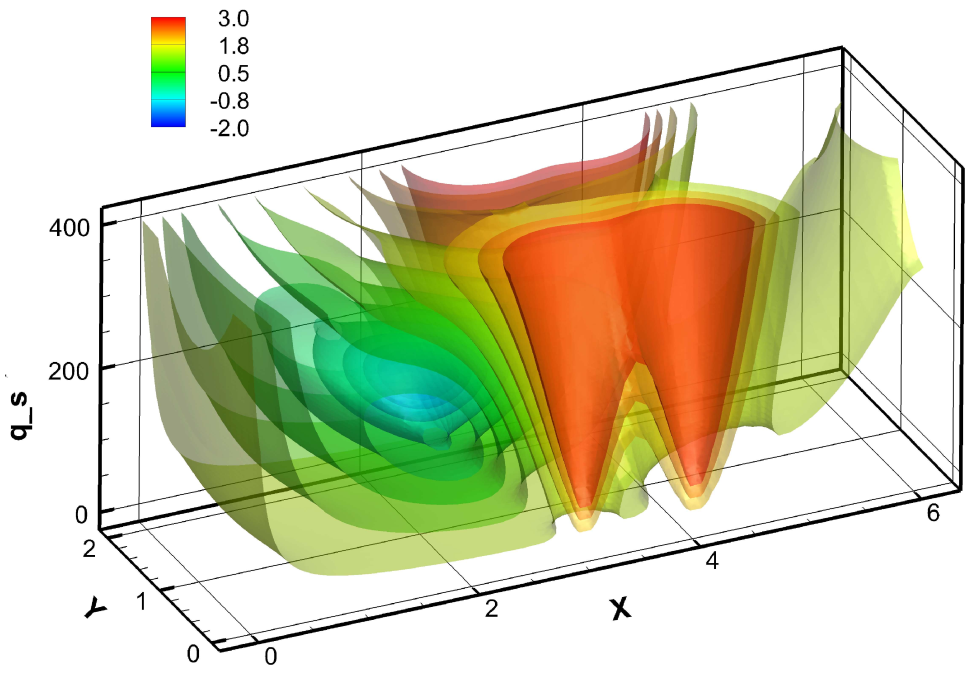


Figure 6.1: Functional corresponding to a case with parameters $\mathbf{f} = (x_s, y_s, q_s)$, and true parameters $\mathbf{f_T} = (2\text{ m}, 1\text{ m}, 400\text{ units})$

An example of a functional with parameters $x_s$, $y_s$ and $q_s$ is shown in Fig. 6.1. This functional corresponds to an instantaneous source release of 200 units that occurs at spatial coordinates of $(2, 1)$ with an isotropic diffusivity of $D_T = D_L = 1\text{ m}^2/\text{s}$. Notable in this example and in the subsequent higher parameter-space functional plots is the color scaling. In this case, the colorbar has been adjusted to show variations within the range of $-2 \leq J \leq 3$. While the minimum value is well below 0 due to taking the natural log of the functional, the majority of the functional values, excluding those within the immediate vicinity of the minimum or the local maxima caused by the sensors, fall within a very small range. The use of an $h$ of 20 units for example gives resulting numerical gradients with respect to the source strength typically on the order of $10^{-4}$ to $10^{-6}$. As these gradients are frequently 2-5 orders of magnitude smaller than the spatial gradients, the scaling option must be used to ensure that the source strength gradient information is large enough in magnitude to sufficiently guide the optimization trajectory towards the minimum. Even with scaling however, there are regions of the functional whereby the objective function is not changing appreciably due to the lack of information being supplied by the sensors. A trajectory that is initiated or converges to these regions will have a difficult time reaching the true minimum.

A summary of the parameter bounds are gradient step-sizes implemented in this chapter are provided in Table 6.1.

Table 6.1: Source inversion parameter values

| | $x_s$ (m) | $y_s$ (m) | $t_s$ (s) | $q_s$ (units) |
|---|---|---|---|---|
| min | 0.1 | 0.1 | 0 | 0 |
| max | $x_{max}$ | 1.9 | $t_{max} + \delta t$ | 800 |
| $h$ | 0.1 | 0.1 | 0.04 | 20 |

### 6.1.2 Scaling

In previous examples, automatic scaling was shown to have inconsistent results, and in some cases actually degraded the performance of the optimization. Hence, for the two-dimensional source inversion problems in which the relative gradients in the x- and y-dimensions did not consistently vary by several orders of magnitude, the scaling option was not implemented in DAKOTA. However, as mentioned above, the addition of the time and source strength parameters leads to a vector of numerical gradients $\nabla J(\mathbf{f})$ containing elements that frequently vary by many orders of magnitude, and hence scaling in these cases is required. For the reasons discussed in Chapter III, the user-defined scaling option, while it allows for greater control and manipulation of the optimization trajectory step-sizes, is not desirable for use in the source inversion problem. This is mainly because the user-defined scaling must be performed on a case-by-case basis, and therefore requires specific customization of the solver to a particular functional. Instead, the inversions carried out in this chapter make use of the automatic scaling option in DAKOTA, which maps the parameters space and respective bounds to the interval [0 1].

### 6.1.3 Turbulence Modeling and Boundary Conditions

The previous chapter's source inversion simulations were all carried out assuming laminar flow in the CFD modeling. In this chapter, the RANS $k-\epsilon$ model described in Chapter II is implemented to allow for more realistic simulations of the flow.

The CFD model boundary conditions, including the RANS parameters $k$ and $\epsilon$, are shown in Table 6.2. The kinematic viscosity is set at a value appropriate for water at 20 $^oC$, $\nu = 1 \times 10^-6$ m$^2$/s. The values for $k$ and $\epsilon$ at the inlet are estimated using relations (2.12) and (2.13). In these calculations, $U_{ref}$ is estimated as the average velocity at the inlet, $U_{ref} = \frac{2}{3}U_{max} = 1\frac{1}{3}$ m/s, and the turbulent intensity is estimated as 5% of the average inlet velocity, $I = 0.05U_{ref}$. The turbulent length scale is taken to be 7% of the characteristic inlet length, $l = 0.07 * 2$. With these values $k = 1.2 \times 10^{-2}$ m$^2$/s$^2$ and $\epsilon = 1.5 \times 10^{-3}$ m$^3$/s$^2$.

Table 6.2: Boundary conditions for the k-$\epsilon$ simulations.

| | p (m$^2$/s$^2$) | U (m/s) | C (units) | k (m$^2$/s$^2$) | $\epsilon$ (m$^s$/s$^3$) |
|---|---|---|---|---|---|
| inlet | ZG | parabolic, $U_{max} = 2$ | 0 | $1.2 \times 10^{-2}$ | $1.5 \times 10^{-3}$ |
| outlet | 0 | ZG | ZG | ZG | ZG |
| walls | ZG | 0 | ZG | standard wall function | standard wall function |

ZG=zero-gradient.

### 6.1.4 Initial Guess

With the addition of time as an optimization parameter, the initial starting guess of the spatial and temporal guesses should be chosen with forethought based on the information that is known about the problem. For example, consider a problem in which $\mathbf{f} = (x_s, y_s, t_s)$. The maximum flow in the channel, which is assumed known, is 2 m/s, and the sensor data that is obtained from 4 boundary sensors at (3,0), (3,2), (4,0) and (4,2) is given in Fig. 6.2.

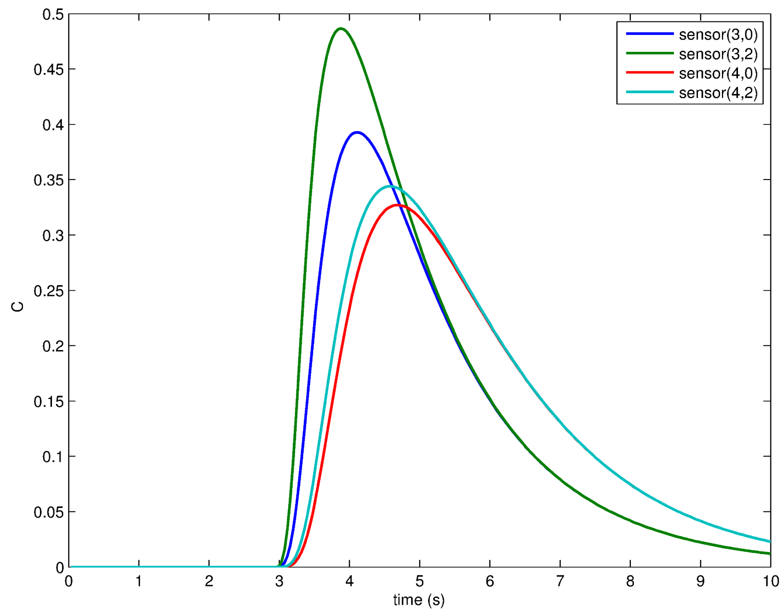


Figure 6.2: Example of sensor data for an instantaneous point release.

The peak value of $C$ occurs in this case at $x_{max} = 3$ at $t_p = 3.88$ s. We also have from this information that $t_{max} = 3.04$ s, based on a nominal minimum detectable contaminant level of $\delta_s$ of 0.001 unit. From this, the bounds for $x_s$ and $t_s$ are [0.1, 3] m and [0, 3.04] s, respectively. Now suppose that the initial guess for $x_s$ is chosen to be 0.1 m, With this value of $x_s$, it would not be logical to choose a corresponding first guess for $t_s$ that is 3 seconds, for example. Given the limiting flow velocity of the channel, this is essentially 'guessing' that the plume traveled approximately $x_{max} - x_s = 2.9$ m in $t_p - t_s = 0.88$ s, resulting in an approximate velocity of 3.3 m/s, which exceeds the known maximum velocity. Hence in order to provide starting guesses that are physically consistent within the bounds of the problem and therefore are likely to be closer to the true parameter values, an approximate guess for the release time can be calculated as a function of the guess in the x-direction, using the average flow velocity in the channel as a reference velocity. This relationship is:

$$t_{guess} = t_p - \frac{x_{max} - x_{guess}}{U_{avg}}. \tag{6.1}$$

While this relationship is dependent on the average of characteristic velocity on the flow instead of a limiting value, it provides an approximate starting guess that should fall within an region of the functional that contains adequate sensor information.

The ramifications of not carefully choosing consistent temporal and spatial parameters can be seen visually by inspecting the functional contours corresponding to the three-parameter space of $\mathbf{f} = (x_s,\, y_s,\, t_s)$. Figure 6.3 shows such a functional, created in this case for a larger domain of length 80 meters and width 20 meters. Four sensors are located at coordinates of (40, 0), (60, 0), (40, 20) and (60, 20). The diffusivity is assumed isotropic and equal to 1. The plotted functional corresponds to an inversion problem in which the true parameters are $\mathbf{f_T} = (10 \text{ m}, 10 \text{ m}, 10 \text{ s})$. Looking closely at the region surrounding the functional minimum visible in the contour slice at $x = 10$ m, it is clear that the functional is actually *increasing* in value as the time approaches the true value of $t_s$. Therefore, a guess for example of $(10 \text{ m}, 10 \text{ m}, 40 \text{ s})$, which would not be feasible given the maximum flow velocity in this channel of 2 m/s, would lead to a starting trajectory that has a *positive* gradient in the direction of decreasing time values. In other words, the numerical gradient of $J$ with respect to the time-dimension would be negative in the opposite direction of the true $t_s$.

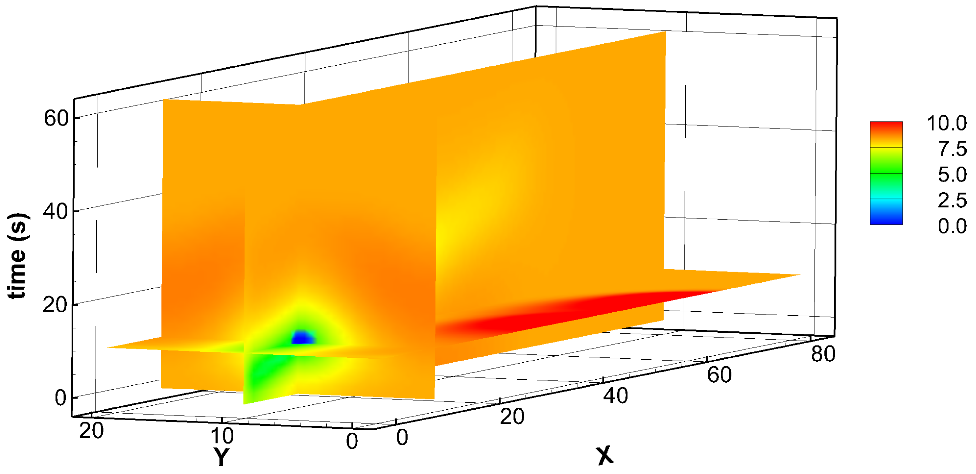


Figure 6.3: Functional contours for source inversion of $\mathbf{f} = (x_s, y_s, t_s)$ and true parameters $\mathbf{f_T} = (10 \text{ m}, 10 \text{ m}, 10 \text{ s})$

Figure 6.4 shows a zoomed-in example of this problem, where the color range has been narrowed to more clearly show the increase in the functional values above approximately 20 seconds in the range $5 < y < 15$. Estimating the initial $t_s$ using Eqn. (6.1) as a function of the starting value of $x_s$ and placing appropriate bounds on the inversion parameters based on the physics of the problem help to avoid the situation in which a trajectory encounters a region of the functional such as this.

### 6.1.5 Optimization Approach

Several optimization possibilities were explored for the two-dimensional source inversion problems in the previous chapter, including two gradient-based only solvers, a hybrid optimization method, and a Model Predictive Control approach for correcting the optimizer-CFD model discrepancies. Based on the general superior performance of the OPT++ optimizer in the two-dimensional source inversion applications in Chapter V, this solver is used exclusively as the gradient-based (GB) optimizer in the

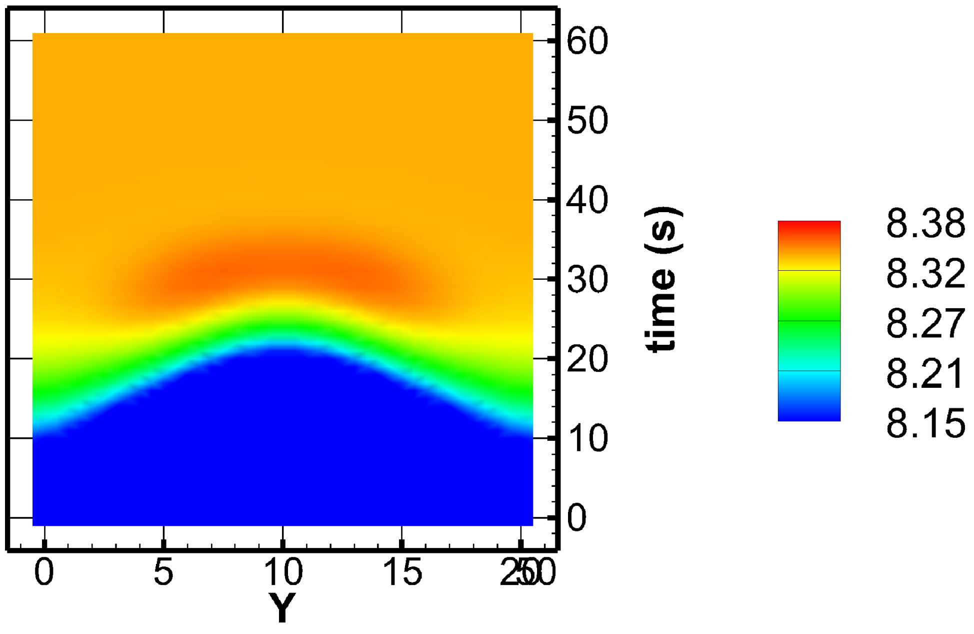


Figure 6.4: Cross-sectional contours of $J$ at x=0 m for the functional shown in Fig. 6.3.

following examples.

Due to the difficulties with the poor functionals that result from inversion with a higher number of parameters, it is reasonable to conjecture that the hybrid method might fare better for these problems as it allows for a greater exploration of the parameter space and has less of a dependence on the gradient-field. In order to test this, the hybrid technique is implemented in some of the examples below.

The MPC approach has been shown to work well in the case of a more idealized functional, however it may actually lower the performance of the optimization in more difficult cases due to the loss of past trajectory information that results from the stopping and restarting of the optimization process. Due to the limitations of this alternative method, it is not used in the source inversion problems below.

## 6.2 Functionals of Three Parameters

As it was insightful to visualize the topography of the functional surfaces for the two-dimensional source inversion problems, there is much to be learned about the optimization behaviors and limitations by inspecting functionals in higher dimensions. Of course, it is not easy to comprehend a functional corresponding to four or more unknown parameters, as such a surface is not able to be plotted. However, functionals in three-parameter space can be plotted by calculating the objective function value at a range of grid points and graphing the interpolated results as was done for the source inversion problems with two unknown parameters.

First, it is interesting to consider what an 'ideal' functional looks like in three-parameter space. As was shown in Fig. 5.9(b), in an isotropic situation, the functional that results from sensor measurements obtained at every point in the domain results in a smoothly convex surface with relatively uniform gradients with respect to either dimension. The analogous functional in the case of three unknown parameters $\mathbf{f} = (x_s, y_s, t_s)$ is shown in Fig. 6.5 for a continuous point source at (10 m, 10 m) that is initiated at 10 s. The observation time in this case is 60 seconds and the isotropic diffusivity is $D_T = D_L = 1$ m$^2$/s. In this case, the addition of time as an unknown parameter leads to the creation of a 'wormhole' which contains the minimum and corresponds to the true source parameters.

The color bar shows that the majority of the functional values fall within a very narrow range, about 16-17. However due to sensors being located at every grid point, there is information obtained at every point in the functional space. This available information is reflected in the filling of the entire domain with functional values that are continually varying in time and space. This is in contrast to the functional shown in Fig. 6.6, which is the functional obtained from this scenario with only four boundary sensors. There is a greater overall range of values in this case leading to strong gradients in some regions. But there are also regions of the parameter space

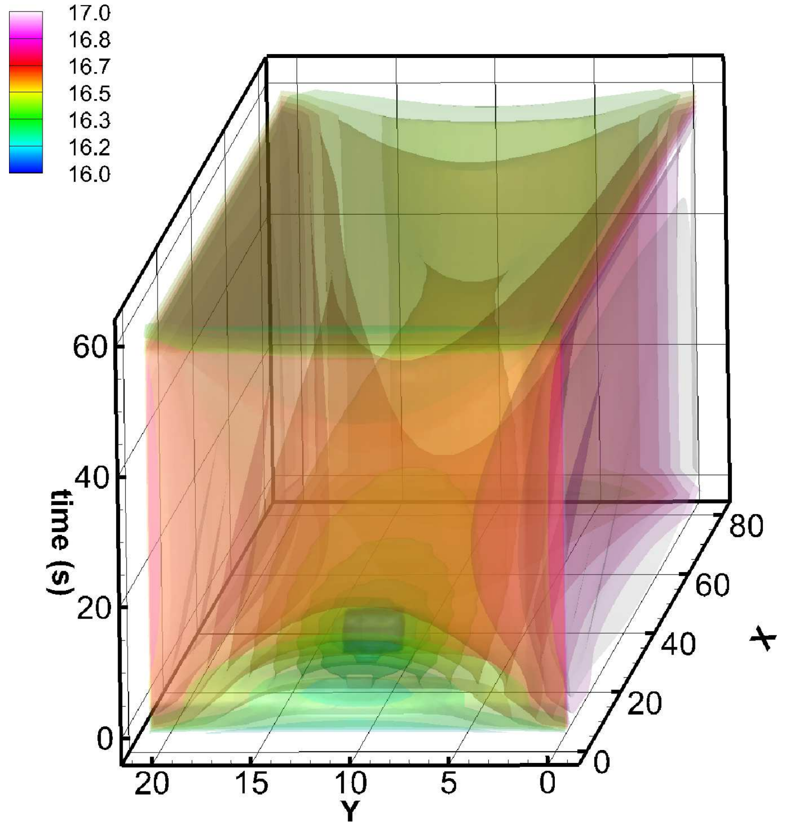


Figure 6.5: Functional isosurfaces resulting from sensors placed at every point for source inversion with respect to parameters $\mathbf{f} = (x_s, y_s, t_s)$, where $D_T = 1$ m$^2$/s. True parameters are $\mathbf{f_T} = (10\text{ m}, 10\text{ m}, 10\text{ s})$.

in which no useful information is obtained, reflected by a constant value of $J$ over several points in time and space. These regions however are concentrated above the time of release and downstream of the x-coordinate of the release. Therefore, if care is taken to adequately restrict the parameter space and to make an intelligent starting guess, these intractable regions of the functional can largely be avoided.

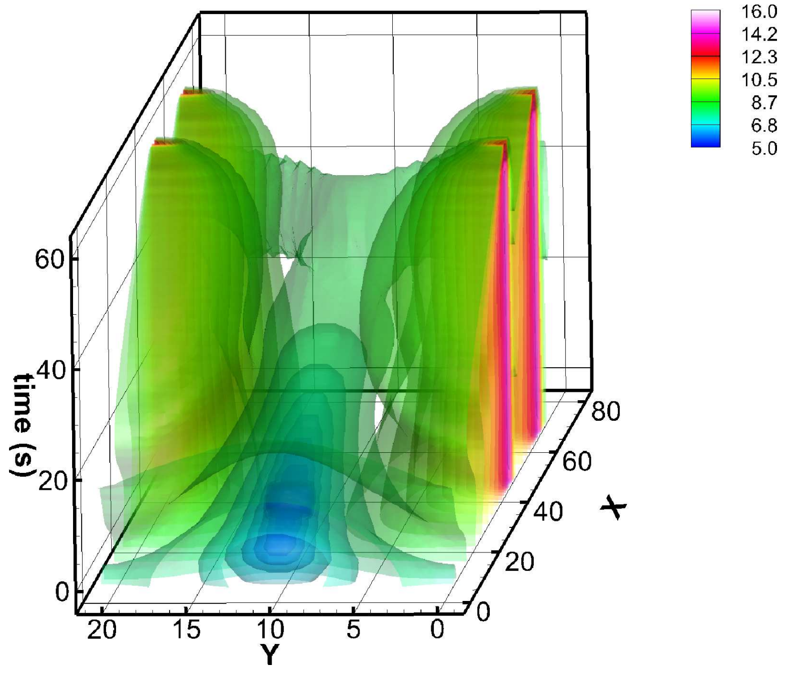


Figure 6.6: Functional isosurfaces resulting from 4 boundary sensors for source inversion with respect to parameters $\mathbf{f} = (x_s, y_s, t_s)$, where $D_T = 1$ m$^2$/s. True parameters are $\mathbf{f_T} = (10 \text{ m}, 10 \text{ m}, 10 \text{ s})$.

The final example functional in this section shows the same source inversion functional as Fig. 6.6 for the case of a smaller transverse diffusivity, $D_T = 0.1$ m$^2$/s. Similar to the two-parameter space source inversion problems in the previous chapter, lowering the transverse diffusivity results in a weaker signal obtained from the boundary sensors and hence results in a more sparse field of isocontours, occurring over a more limited range of values. Keeping in mind the difficulties encountered in the previous chapter when optimizing source inversion functionals, the challenge of optimizing a complex functional such as the above examples can be more fully appreciated.

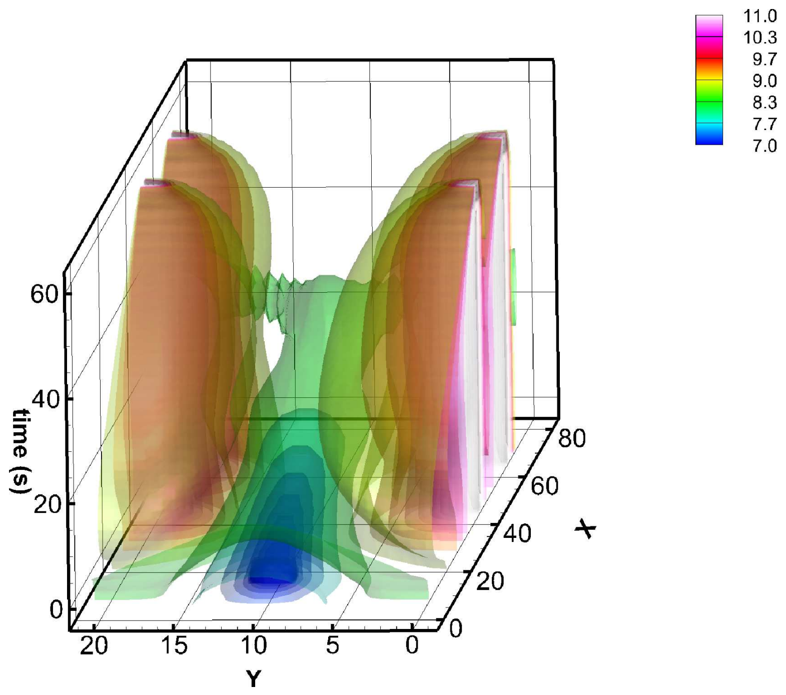


Figure 6.7: Functional isosurfaces resulting from 4 boundary sensors for source inversion with respect to parameters $\mathbf{f} = (x_s, y_s, t_s)$, where $D_T = 0.1$ m$^2$/s. True parameters are $\mathbf{f_T} = (10\text{ m}, 10\text{ m}, 10\text{ s})$.

## 6.3 Results with Three Parameters

### 6.3.1 Source Inversion with Respect to $\mathbf{f} = (x_s, y_s, t_s)$

First, we examine the case of the two-dimensional channel that has been used in previous examples, where again four sensors are located at the boundary at coordinates (3,0), (3,2), (4,0) and (4,2). However in this case the time of release $t_s$ is included as an additional optimization parameter. An instantaneous release of 400 units is initiated at the coordinates of (2,1) and at $t = 3$ s, where the time is in reference to an arbitrary observational start time, in this case defined as $t_0 = 0$ s. The sensor data corresponding to this scenario is shown in Fig. 6.8. From this data, it can be discerned that the first detectable contaminant data point is at $t_{max} = 3$ s at $x_{max} = 3$ m where in this example 'measurable' is arbitrarily defined as $C > 0.001$ unit. This initial detection time coincides with the time of release because of the instantaneous spreading that occurs with the relatively high diffusivity in this example, which is set at 1 $m^2/s$ in both the transverse and longitudinal directions. From this data, the maximum bound on $x_s$ is determined to be 3 meters. As mentioned above, it is found that the optimizer performs better in three-parameter space problems when it can test slightly above the maximum release time, so that the gradients can be calculated on either side of the true value and the optimizer can better 'zero in' on the minimum. With this in mind the time maximum is increased slightly to 3.2 seconds in this case, which still falls in a zone where the gradients are negative in the direction of the true time parameter, avoiding the issues illustrated in Fig. 6.4.

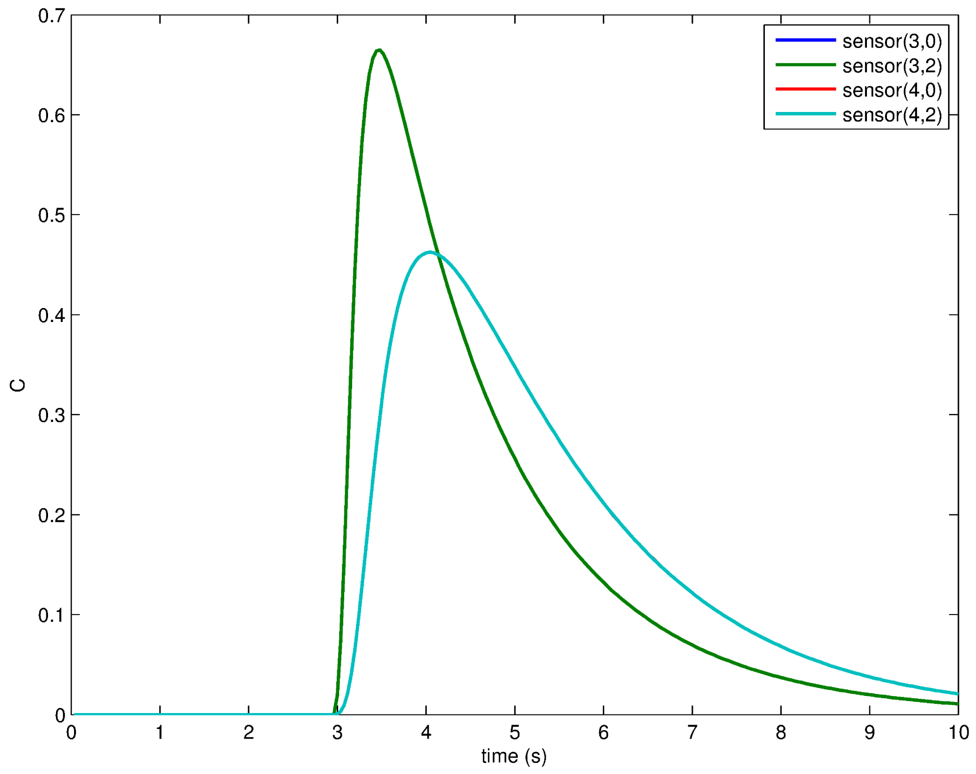


Figure 6.8: Simulated sensor data for source inversion with parameters $\mathbf{f} = (x_s, y_s, t_s)$.

The observational time is defined as a total of 10 seconds, starting from $t_0 = 0$. The functional corresponding to this scenario is shown in Fig. 6.9 in the form of isosurfaces of $J$. Most notable again in this figure is that there are large regions of the domain in which the functional is changing very little, resulting in a narrow range in the time-dimension in particular in which the functional is well-defined. Again, these regions are concentrated in the downstream and infeasible regions of the domain, that is, the regions in which the time and spatial parameters combinations are not consistent with the physics of the problem. For example, restricting the maximum possible time to values less than 3.2 seconds and $x$ less than or equal to 3 meters cuts out much of the troublesome regions of the functional.

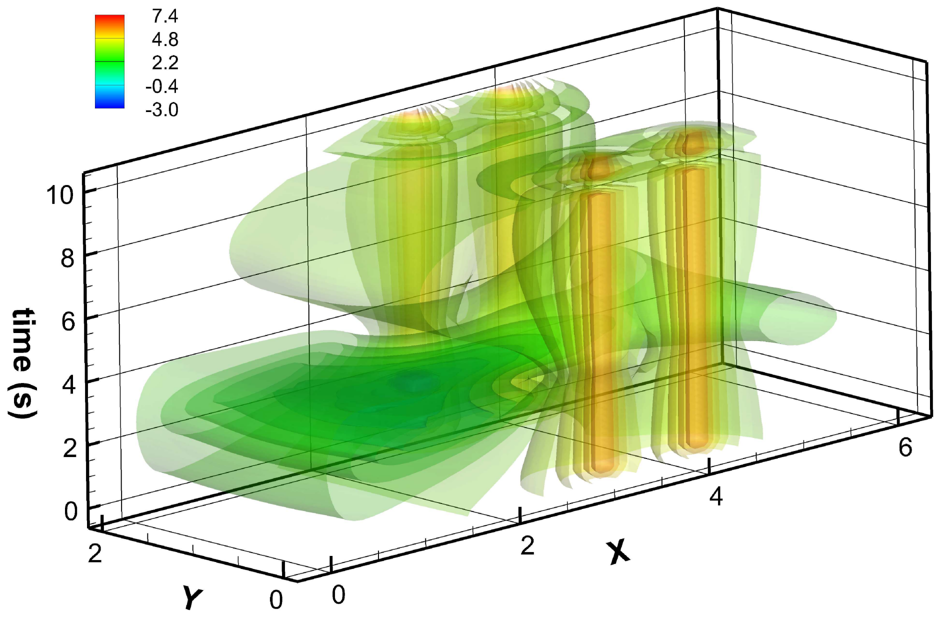


Figure 6.9: Functional corresponding to source inversion with respect to parameters $\mathbf{f} = (x_s, y_s, t_s)$, where the true parameters are $\mathbf{f_T} = (2\text{ m}, 1\text{ m}, 3\text{ s})$.

The initial spatial parameter points used in these source inversion tests are shown in Fig. 6.10. The respective guesses for the time parameter for each of these 142 points were calculated using Eqn. (6.1). Because the average velocity is used in this equation, some of these times will exceed the upper bound placed on the release time parameter. In these cases, the release time guess is automatically reset to the maximum bound in DAKOTA.

The results of the source inversion simulations for this case are shown in Fig. 6.11,

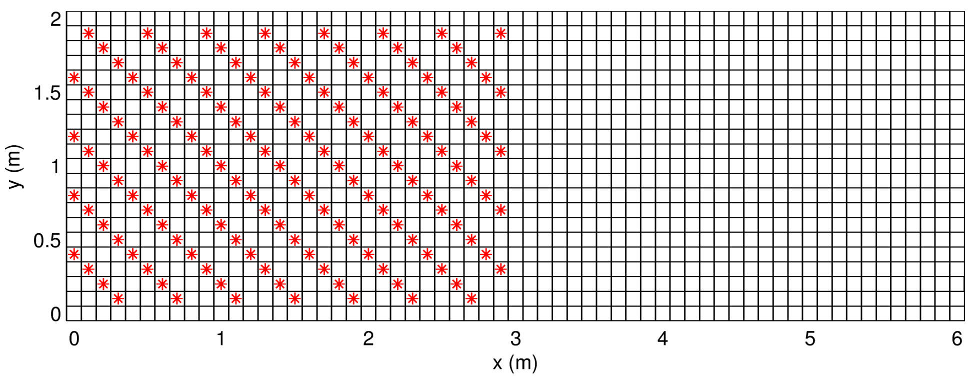


Figure 6.10: Initial test points for source inversion of three parameters.

where the white diamonds indicate the final converged parameters from the optimization process. The red circles denote the starting points which correspond to completely accurate trajectories in this case. Somewhat frustratingly, there is no discernible pattern to these successful trajectory starting points, indicating that there doesn't appear to be definitively reliable region of the parameter space in which to begin the optimization process.

This test case was repeated using the hybrid optimization approach as well. An initial population of $P_0 = 30$ and a conservative maximum iteration limit of three was imposed in order to reduce the convergence times. The hybrid source inversion was also run a total of 142 times and the results, including the previously found straight gradient-based results for comparison, are plotted in Fig. 6.12.

The average convergence errors with respect to the spatial distance from the true source location and to the discrepancy between the true and inverted release times for both methods are shown in Table 6.3, compiled for 142 trials of each method. In general, the time-dimensional inversion was more accurate than the spatial; the average release time error was less than a tenth of a second for both the GB and hybrid approaches. As is visible in the GB results in Fig. 6.11, the spatial errors tend to be higher with regards to the x-dimension. This is perplexing, as the gradients in the longitudinal direction are strongly negative in this region and thus the optimizer

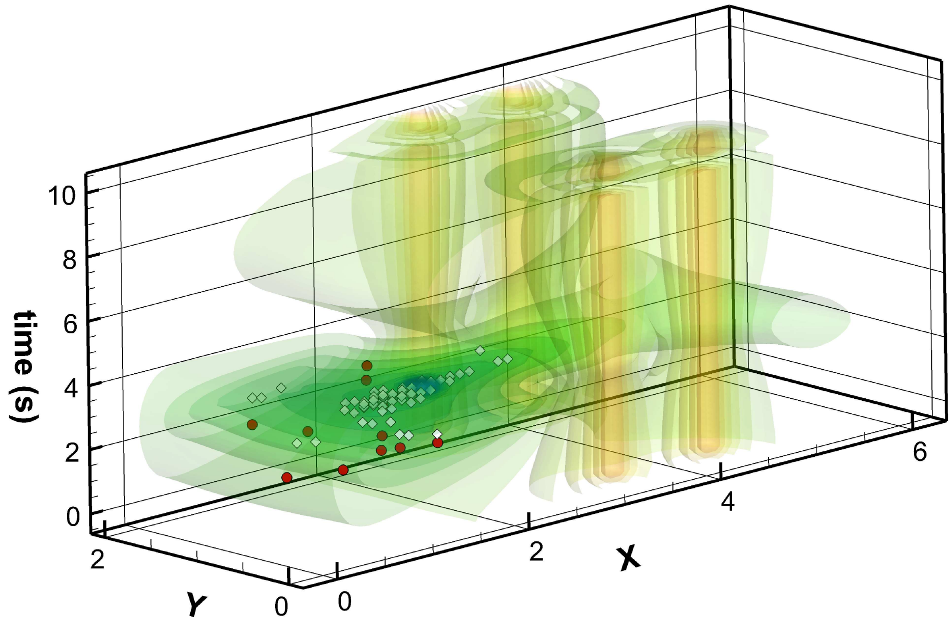


Figure 6.11: OPT++ results (white diamonds) of source inversion with respect to parameters $\mathbf{f} = (x_s, y_s, t_s)$, where the true parameters are $\mathbf{f_T} = (2\ \mathrm{m}, 1\ \mathrm{m}, 3\ \mathrm{s})$. Initial points resulting in perfect convergence are shown as red circles.

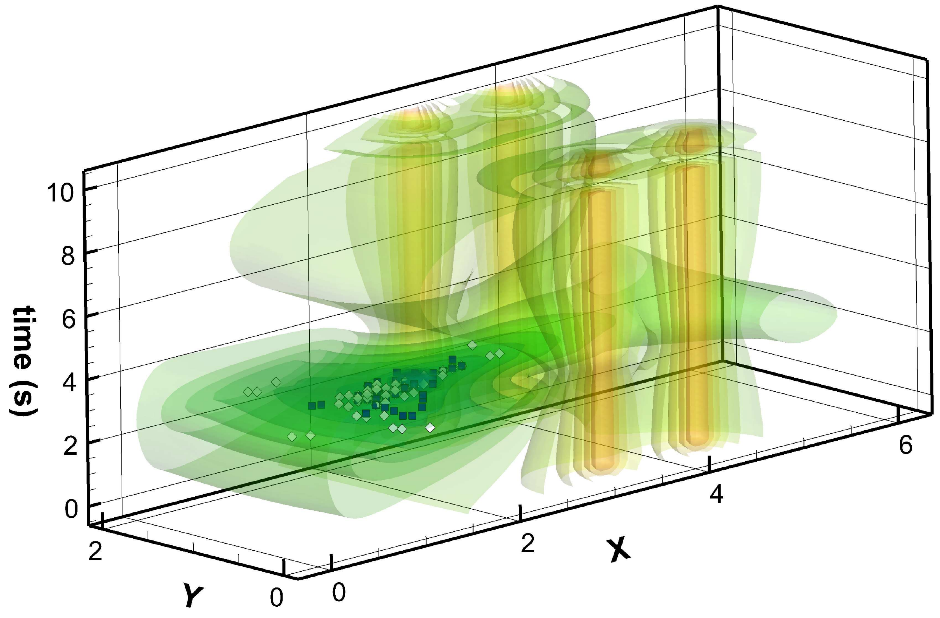


Figure 6.12: Hybrid (blue squares) and OPT++ (white diamonds) results of source inversion with respect to parameters $\mathbf{f} = (x_s, y_s, t_s)$, where the true parameters are $\mathbf{f_T} = (2\ \mathrm{m}, 1\ \mathrm{m}, 3\ \mathrm{s})$.

should have ample information in which to choose the correct trajectory direction. Likely, the trajectories are getting stuck in this region because when the gradients are relatively high the algorithm chooses an $\alpha$ that is correspondingly small in order to avoid overshooting. This leads to the previous problems of the trajectory getting stuck and being unable to move out of a cell.

Between the two approaches, on average the hybrid results were more accurate than the straight gradient-based results, with smaller mean and maximum errors with respect to the errors in the spatial distance to the true minimum and to the release time discrepancies. While the GB approach led to a slightly higher percentage of completely accurate trajectories, these numbers were small (less than 10%) for both methods. Increasing the acceptable errors to less than or equal to 0.2 m in distance and to within 0.2 s from the true release time results in a third of the trajectories being 'successful' with the GB approach and about 41% with the hybrid approach.

Table 6.3: Error statistics for source inversion with respect to $\mathbf{f} = (x_s, y_s, t_s)$.

| | | **mean($\zeta$)** | **max($\zeta$)** | %($\zeta = 0$) | %($\zeta \leq 0.2$) | %(**total** $\zeta = 0$) | %(**total** $\zeta \leq 0.2$) |
|---|---|---|---|---|---|---|---|
| **GB** | **space** | 0.38 | 1.3 | 9.9 | 33.1 | 7.7 | 33.1 |
| | **time** | 0.09 | 0.76 | 16.9 | 88.7 | | |
| **hybrid** | **space** | 0.28 | 0.92 | 7.6 | 41.7 | 7 | 40.8 |
| | **time** | 0.08 | 0.44 | 15.3 | 93.8 | | |

In the previous chapter, it was found that the biggest drawback with the hybrid approach was that its non-deterministic nature made the results unpredictable and did not allow for an improved chance of convergence based on a good initial starting guess. However, the results of the straight gradient-based approach are also unpredictable for this case and do not exhibit a clear region in which the optimization is consistently successful, and thus overall the hybrid approach is a good choice in this case.

A comparison of the convergence run-times for both the GB and hybrid approaches is included in Table 6.4. As expected, the hybrid approach takes longer to converge on average, 15.7 minutes per inversion vs. 10.9 minutes for the GB trials. The minimum hybrid convergence time was also about twice as long as the minimum GB convergence time, and the maximum run-time was almost 40 minutes compared to about half an hour for the GB method.

Table 6.4: Run-times for source inversion with respect to parameters $\mathbf{f} = (x_s, y_s, t_s)$.

| | **run-times (min)** | | |
|---|---|---|---|
| | **mean** | **max** | **min** |
| **GB** | 10.9 | 27.5 | 4.9 |
| **hybrid** | 15.7 | 37.3 | 9.9 |

### 6.3.2 A Second Example with a New $\mathbf{f_T}$

The above case was modified to reflect a different true source location and release time, corresponding to true parameters $\mathbf{f_T} = (1 \text{ m}, 0.5 \text{ m}, 2 \text{ s})$. The simulated sensor data for this case is shown in Fig. 6.13. Following the logic in the previous case, $x_{max} = 3$ m and $t_{max}$ is now 2 seconds. Again, the upper bound for the optimization of the time parameter is adjusted to slightly above this value at 2.2 seconds. The same points shown in Fig. 6.10 were used to test this case using the GB approach. The results are shown in Fig. 6.14. As with the previous case, the largest convergence errors in this case occur in the x-dimension, and there is no clear relationship between the points that result in perfect convergence and their proximity to the actual source parameters in the functional space.

The GB results for this source are on average more accurate than the first tested case. While the percentage of completely successful trajectories is the same (7.7%), 57% of the trajectories came within a distance of 0.2 m and a source time of 0.2 s from the true source time in this case. Clearly, the results of the optimization will vary with differing combinations of the true parameters, which corresponds to changes in the nature of the functional. In this case, it could be that the lack of symmetry for the source located at (1, 0.5) m leads to a more well-posed functional. Also, since the

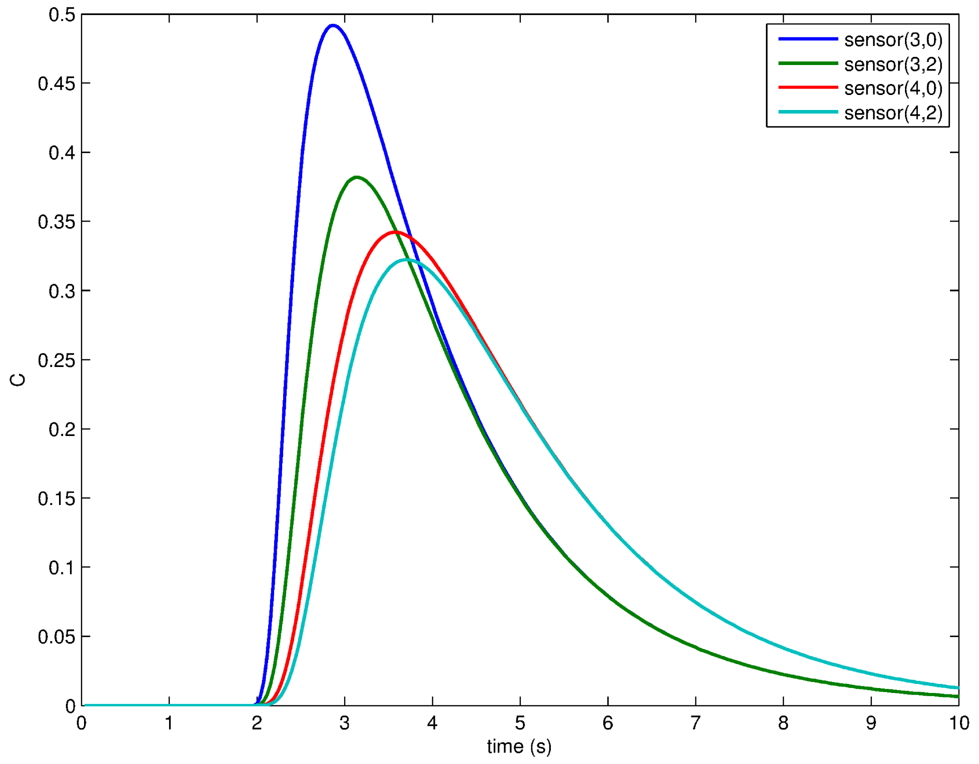


Figure 6.13: Simulated sensor data for source inversion with respect to parameters $\mathbf{f} = (x_s, y_s, t_s)$ and true parameters $\mathbf{f_T} = (1\text{ m}, 0.5\text{ m}, 2\text{ s})$.

observation time is held constant in these cases regardless of the initial detection time, the improved trajectory outcomes could be the result of using a longer observation record for the source inversions (8 s for $t_s = 2$ s vs. 7 s for $t_s = 3$ s).

### 6.3.3 Source Inversion with Respect to $\mathbf{f} = (x_s, y_s, q_s)$

Using the same 142 spatial points tested in the previous three-parameter space examples (Fig. 6.10), the results were repeated for the case of the three unknown parameters consisting of $x_s$, $y_s$ and $q_s$. The true parameters are given by $\mathbf{f_T} = (2\text{ m}, 1\text{ m}, 400\text{ units})$, and the time of release is assumed known at $t_s = 3$ s. The initial guess in each of these cases is set at 200 units. The four sensors are located at the same coordinates along the boundaries as in the previous cases, and the corresponding sensor data is shown in Fig. 6.15.

The results for the GB source inversion of $x_s, y_s$ and $q_s$ are shown in Fig. 6.16.

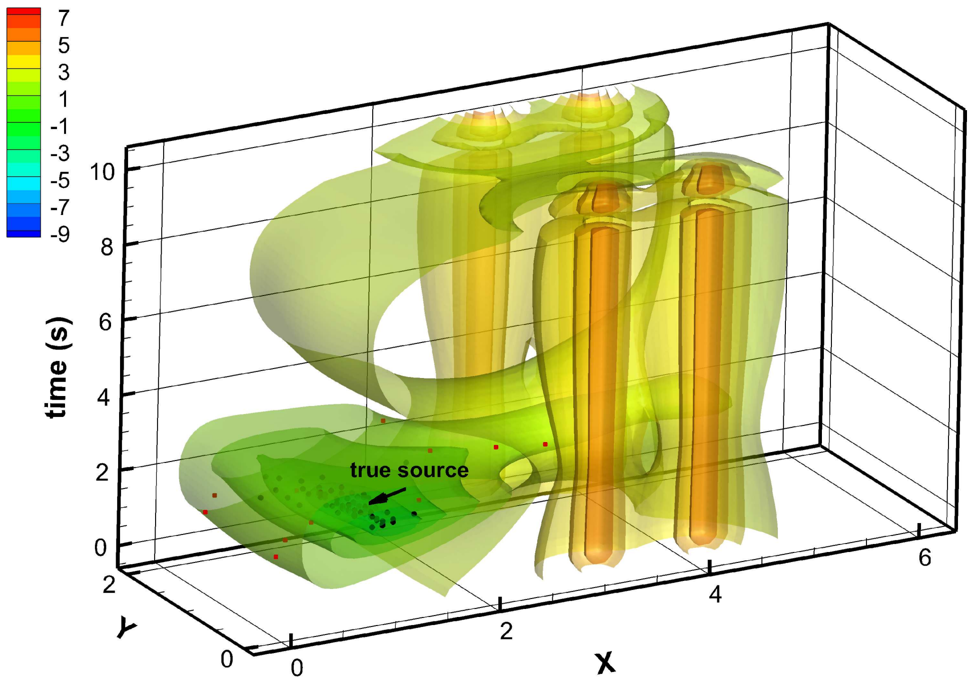


Figure 6.14: OPT++ (black circles) results of source inversion with respect to parameters $\mathbf{f} = (x_s, y_s, t_s)$ and true parameters $\mathbf{f_T} = (1\text{ m}, 0.5\text{ m}, 2\text{ s})$. Initial points resulting in perfect convergence are shown as red circles.

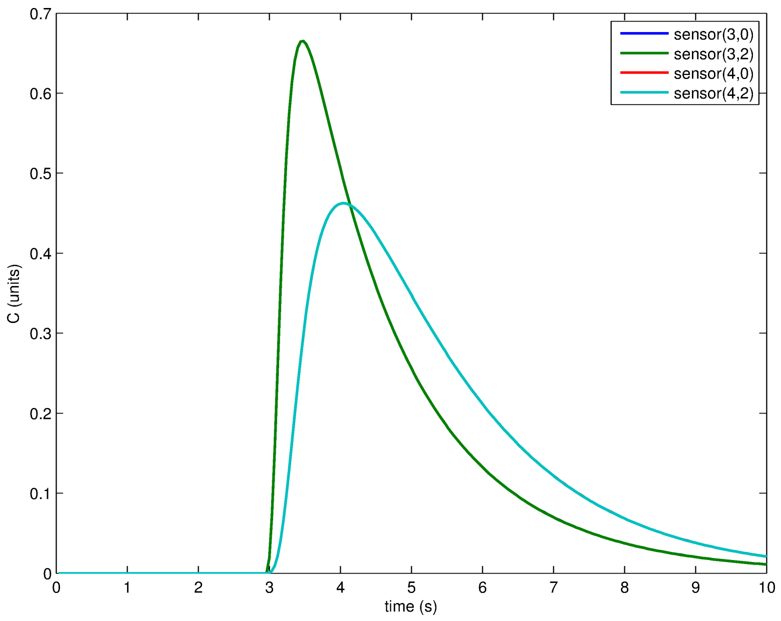


Figure 6.15: Simulated sensor data corresponding to the source inversion example with true parameters $\mathbf{f_T}$ = (2 m, 1 m, 400 units).

The 'successful' trajectory initial points are also shown, symbolized by the green squares. Successful trajectories in this case were arbitrary defined as those which came within 0.2 m distance from the true source and within 20 units of the true source strength. As with the previous three-parameter space examples, there appears to be no clear radius of convergence in the parameter space that leads to consistently reliable optimization. The average convergence run-time for these 142 trials was 8.1 minutes.

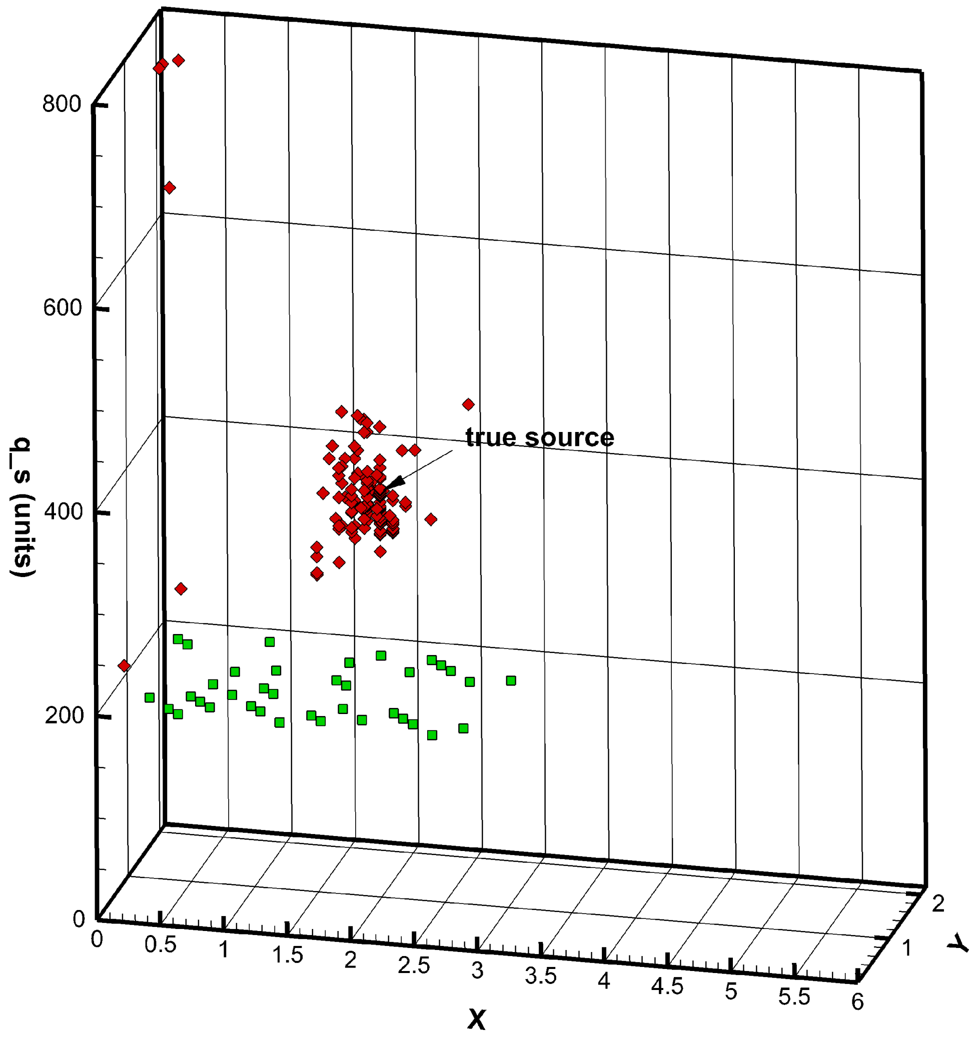


Figure 6.16: OPT++ results (red diamonds) of source inversion with respect to parameters $x_s, y_s$ and $q_s$, where the true parameters are $\mathbf{f_T} = (2 \text{ m}, 1 \text{ m}, 400 \text{ units})$. Initial points resulting in convergence to within 0.2 meters distance and 20 units of source strength are shown as green squares.

A summary of the convergence errors with respect to the spatial distance from the true source and the discrepancy between the converged upon and actual source strength are shown in Table 6.5. Most of the trajectories get closer in spatial distance to the true source location when optimizing with respect to the source strength vs. the case of optimizing with respect to the release time. While it would be expected that the treatment of the initial source strength as a continuous variable would allow for more accurate optimization of this parameter, this does not appear to be the case. This is likely the result of the forward differencing scheme and the chosen step size. Using a smaller gradient step-size $h$ may allow for better convergence with respect to $q_s$, however this will also likely increase the overall convergence time.

Table 6.5: Convergence errors for parameters $x_s, y_s$ and $q_s$.

| | **mean($\zeta$)** | **max($\zeta$)** | **%($\zeta = 0$ m)** | **%($\zeta \leq 0.2$ m)** |
|---|---|---|---|---|
| **spatial distance (m)** | 0.24 | 1.4 | 12 | 57.7 |
| | **mean($\zeta$)** | **max($\zeta$)** | **%($\zeta \leq 5$ units)** | **%($\zeta \leq 20$ units)** |
| **$\mathbf{q_s}$ (units)** | 54.3 | 400 | 14.8 | 45.1 |

## 6.4 Source Inversion of Four Parameters

### 6.4.1 Gradient-based Results

We now consider a case in which the optimized parameters are $\mathbf{f} = (x_s, y_s, t_s, q_s)$ and the true source is given by $\mathbf{f_T} = (2 \text{ m}, 1 \text{ m}, 3 \text{ s}, 400 \text{ units})$. The GB and hybrid methods are both implemented for 20 randomly generated spatial starting points. The guesses for $t_s$ are calculated as before, and the initial guess for the source strength $q_s$ is again 200 units for every point. The upper limit on the time bounds in this case is chosen to be equal to $t_{max}$ instead of adding a small amount to the upper bound as previously was done. With the higher number of parameters to invert, it is found that the best results are obtained in this case when the maximum time is not allowed to exceed the initial detection time. Additionally, for these trials, a maximum number of iterations equal to 50 is imposed in the GB optimization trials. This limit is imposed because in many cases when the trajectory gets stuck in a 'bad' region of the parameter space, it tends to repeat a pattern of evaluating the same few points again and again in a circuitous manner. In order to reduce the run-times and cut short trajectories that have effectively dead-ended, this maximum number of iterations is enforced.

The resulting spatial convergence errors of the 20 GB trials are shown along with the initial spatial test points in Fig. 6.17. As can be seen, few of the trajectories end up in the spatial vicinity of the true source, with several points ending up towards the boundaries of the domain. The average and maximum convergence times are 36.8 minutes and 78.6 minutes, respectively.

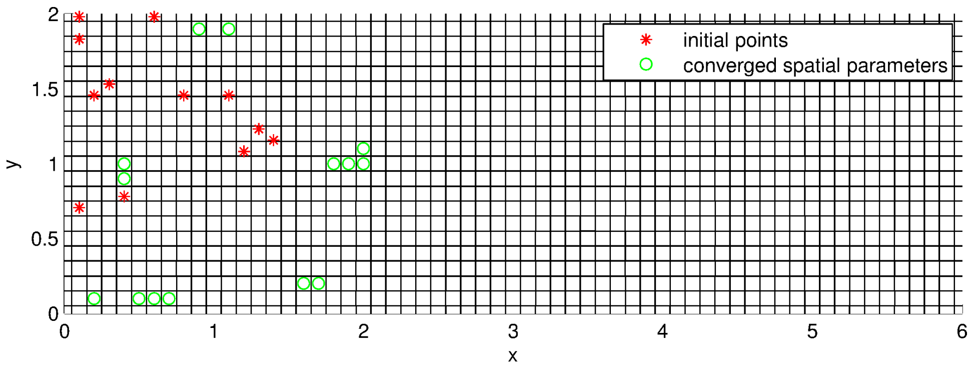


Figure 6.17: Initial spatial test points (red) and final convergence spatial parameters (green) for source inversion of four parameters.

Figure 6.18 provides a graphical representation of the final convergence errors with respect to the spatial parameters, release time, and initial source strength. Notable in this figure is the correlation between the optimization accuracy with respect to the individual parameters. That is, when one parameter converges within a small error bound of the true parameter, typically the other parameters do as well. For example, of the 8 points in which the distance in error from the true source is less than or equal to 0.2 m, the corresponding error in the time dimension is less than 0.05

seconds for all 8 trajectories and the error in the initial source strength is less than 30 units in all except one of these instances. This finding reflects the fact that the gradient-based optimization success is limited by the complexity of the functional, and with certain combinations of parameters the divergent gradient field will lead the trajectory astray.

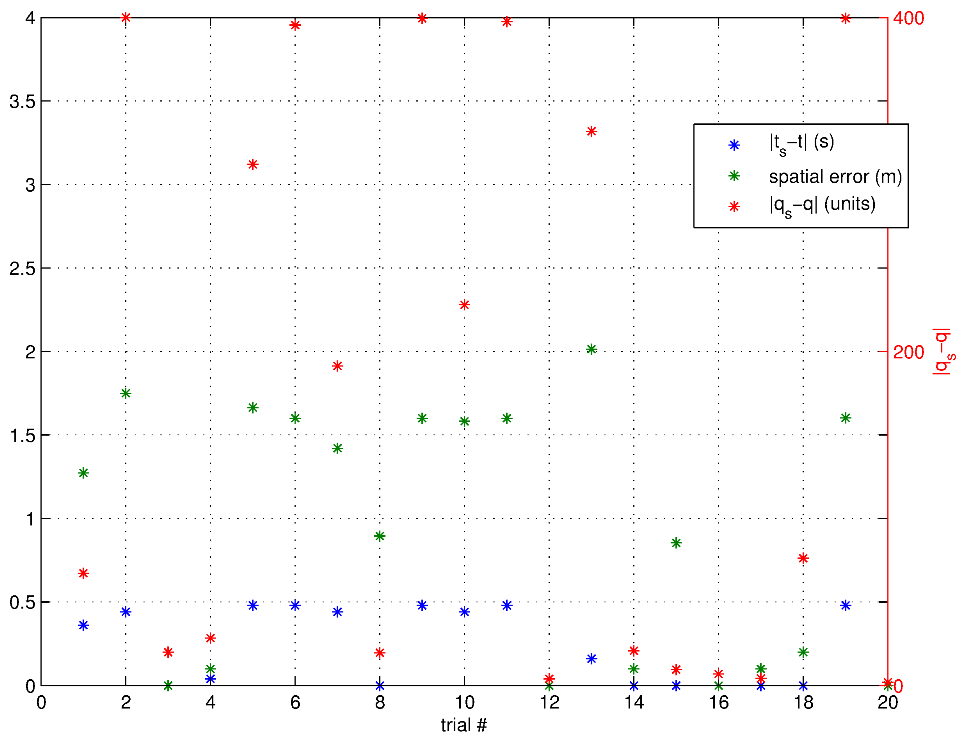


Figure 6.18: Convergence errors for GB source inversion with respect to $\mathbf{f} = (x_s, y_s, t_s, q_s)$.

To illustrate this, Table 6.6 shows the parameter guesses and associated gradients corresponding to ten iterations of the GB optimization with initial guess $\mathbf{f}_0 = (2.7 \text{ m}, 1.8 \text{ m}, 2.64 \text{ s}, 200 \text{ units})$. The highlighted values in the table are those in which the gradients point away from the direction of the true parameter at that iteration. As can be seen, this is a problem in particular with respect to the release time and source strength parameters. Because of the divergent gradient field, for certain tested combinations of parameters the objective function *decreases* overall as the parameters are *further* from the true parameters. For example, the objective function in iteration 1 is 3.15, which is greater than the objective function that is evaluated at iteration 10, where all of the variables are at a greater distance in the parameter space from the true parameters.

Table 6.6: Trajectory parameters and associated gradients for inversion of $\mathbf{f} = (x_s, y_s, t_s, q_s)$.

| | | $\mathbf{x_s}$ | $\mathbf{y_s}$ | $\mathbf{t_s}$ | $\mathbf{q_s}$ | **J** |
|---|---|---|---|---|---|---|
| **iter.** | **true parameters** | 2 | 1 | 3 | 400 | -70 |
| **1** | **parameters** | 2.7 | 1.8 | 2.64 | 200 | 3.15 |
| | **gradients** | 2.59 | 1.95 | -0.572 | 1.88E-04 | |
| **2** | **parameters** | 0.1 | 1.6 | 2.8 | 110 | 3.00 |
| | **gradients** | -0.554 | 4.88E-02 | 3.42E-02 | -3.52E-05 | |
| **3** | **parameters** | 0.1 | 0.9 | 2.96 | 373 | 2.73 |
| | **gradients** | -2.14 | -1.98E-02 | 0.221 | -4.20E-05 | |
| **4** | **parameters** | 0.1 | 0.1 | 2.96 | 671 | 2.60 |
| | **gradients** | -3.26 | -0.25 | 0.333 | -3.12E-05 | |
| **5** | **parameters** | 0.2 | 0.1 | 2.88 | 800 | 2.03 |
| | **gradients** | -3.96 | -0.46 | 1.02 | -5.84E-005 | |
| **6** | **parameters** | 0.1 | 0.1 | 2 | 800 | 2.47 |
| | **gradients** | -4.28 | -0.315 | -0.332 | -3.46E-05 | |
| **7** | **parameters** | 0.2 | 0.1 | 2.56 | 800 | 1.79 |
| | **gradients** | -6.85 | -0.67 | 0.376 | -8.83E-05 | |
| **8** | **parameters** | 0.1 | 0.1 | 2.4 | 800 | 2.37 |
| | **gradients** | -5.87 | -0.4 | -5.09E-02 | -4.30E-05 | |
| **9** | **parameters** | 0.2 | 0.1 | 2.56 | 800 | 1.79 |
| | **gradients** | -6.85 | -0.67 | 0.376 | -8.83E-05 | |
| **10** | **parameters** | 0.1 | 0.1 | 2.56 | 800 | 2.38 |
| | **gradients** | -5.84 | -0.4 | 0.138 | -4.27E-05 | |

In this case, the ill-posedness of the functional has a numerical instead of theoretical basis, as it is in part the result of the boundary conditions used for the contaminant concentration $C$. Due to the zero-gradient boundary conditions and the numerical treatment of the source term, some mass of contaminant is lost when the source is placed near the boundaries of the domain. In order to better see this, the modeled data corresponding to iterations 1 and 10 are compared in Fig. 6.19 to the simulated sensor data representing the true parameters. As can be seen, the parameters tested in iteration 10 lead to data that is a better match with the sensor data than the data obtained using the parameters in iteration 1 because when the tested source is placed near the boundary, the modeled concentrations at the sensor locations are artificially low. Compensating for the loss of mass at the boundaries in this case would improve the results of the optimization, however in many cases the issue of non-uniqueness still will exist, which is illustrated in the following section.

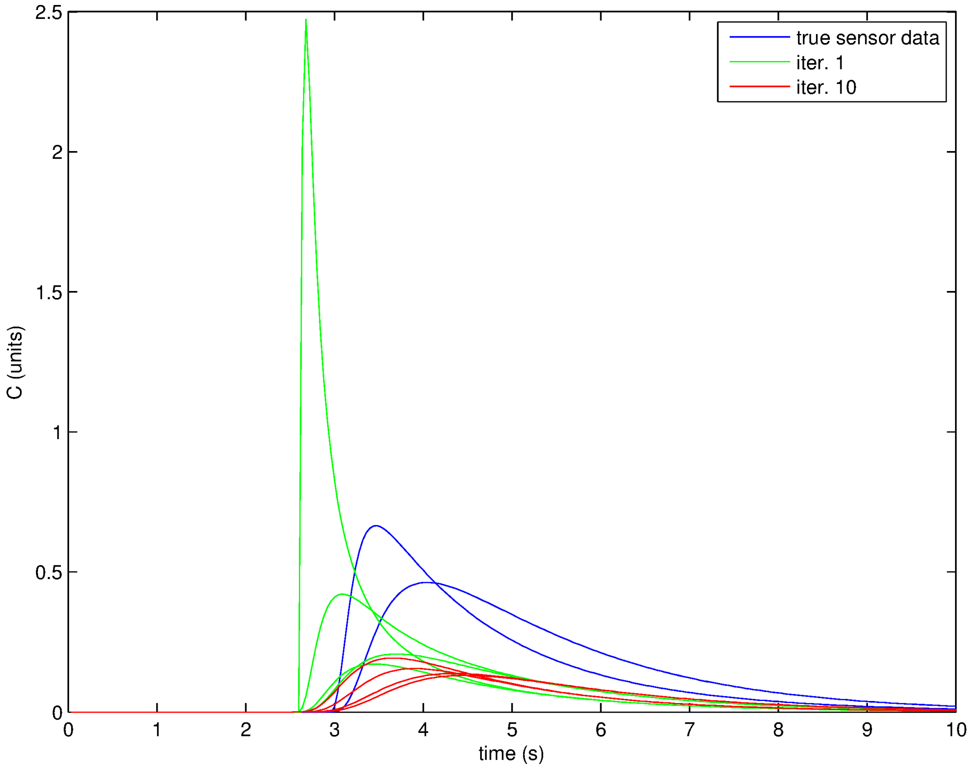


Figure 6.19: True sensor data compared to that obtained using the parameters in iteration 1 and 10 in Table 6.6.

### 6.4.2 Hybrid Results

This case was also tested with 20 runs of the hybrid method, and the results shown in Fig. 6.20. The same ill-posedness problems occur with the hybrid method because the divergent gradient field occurs with certain parameter combinations even with points that are relatively close to the true parameters. Further, as the initial parameters are chosen randomly in the hybrid method, a physically consistent starting guess cannot be stipulated. This likely explains why there is less of a correlation between the successful convergence of the individual parameters in this case. For example, as with the GA approach, 8 of the 20 inversions resulted in a spatial error of 0.2 m or less. However in this case the corresponding release times and source strengths are not similarly accurate.

Additionally, the run-times for the hybrid method are about twice as long as the GA method; an average of 60.2 minutes for a single inversion and a maximum of 98 minutes. While increasing the number of the EA iterations may lead to an improvement of the hybrid method, the lengthy run-time detracts from its usefulness.

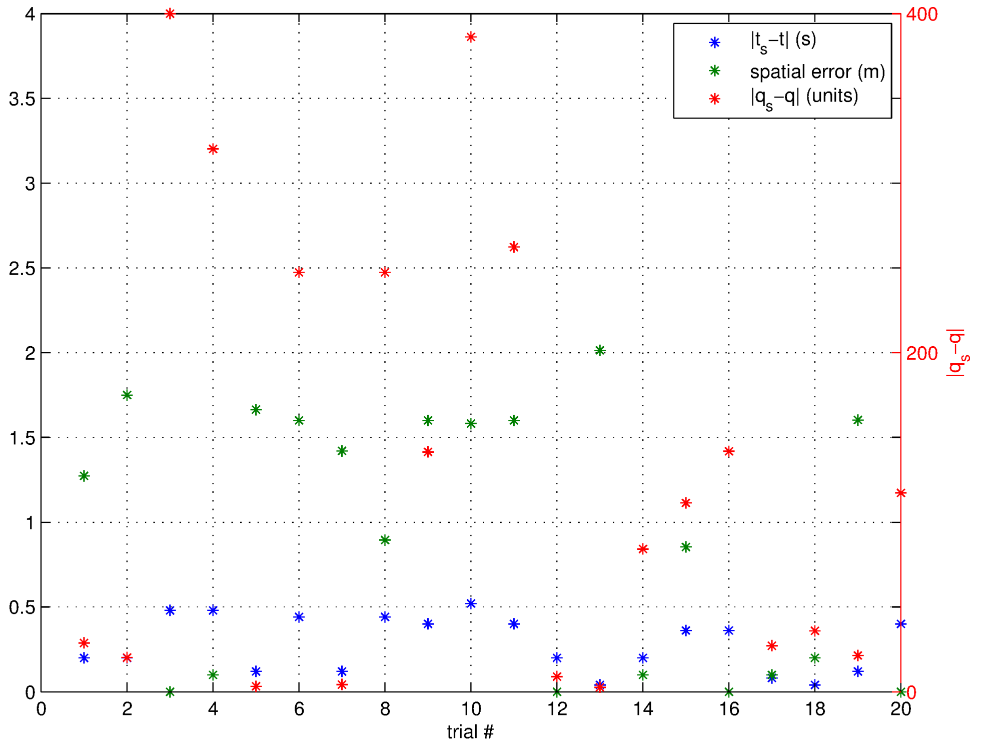


Figure 6.20: Convergence errors for hybrid source inversion with respect to $\mathbf{f} = (x_s, y_s, t_s, q_s)$.

## 6.5 Non-Uniqueness

We now examine a case in which the source inversion problem is ill-posed due to non-uniqueness of the functional. In this example, a three-dimensional domain that is 6 m long, 2 m wide and 1 m tall contains four boundary sensors, located mid-height at coordinates (3, 0, 0.5), (3, 2, 0.5), (4, 0, 0.5) and (4, 2, 0.5). The source is an instantaneous release of 800 units at coordinates (2 m, 1 m, 1 m). The functional isosurfaces relating to this problem are shown in Fig. 6.21.

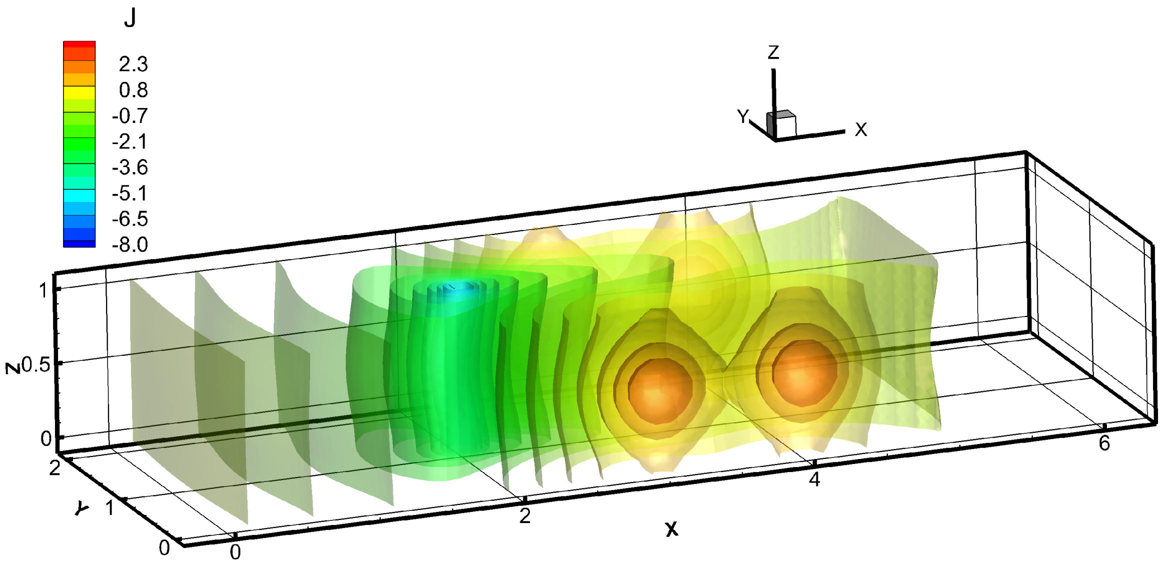


Figure 6.21: Functional isosurfaces corresponding to a true source located at spatial coordinates (2, 1, 1).

As can be seen in this figure, instead of the 'wormhole' like functional zone leading to the minimum in the case of the parameter space consisting of $x_s, y_s$ and $t_s$, the contours of $J$ surrounding the minimum form a nearly hourglass shape. Closer inspection of the functional contours (Fig. 6.22(a)) shows that this functional has two distinct minimum at coordinates (2, 1, 0) and (2, 1, 1). Looking at the two-dimensional x-y cross-section of this functional (Fig. 6.22(b)), the contours approximate those shown in the previous chapter's two-dimensional source inversion results. Due to the symmetry of the problem, this makes sense: a source released equidistant from the sensors will result in the same concentration values measured by the four sensors. Without additional information in regards to the likely source coordinates in this case, it is not possible to discern the source parameters in this case via the optimization source inversion approach. In regards to the complex surfaces that result from source inversion of two-spatial parameters in addition to a third parameter, be it time or source strength, it is easy to imagine how these problems can become intractable. Furthermore, adding additional sensors will not necessarily create a unique functional in this case. That is, adding sensors to every single grid point will still result in the functional having two minima.

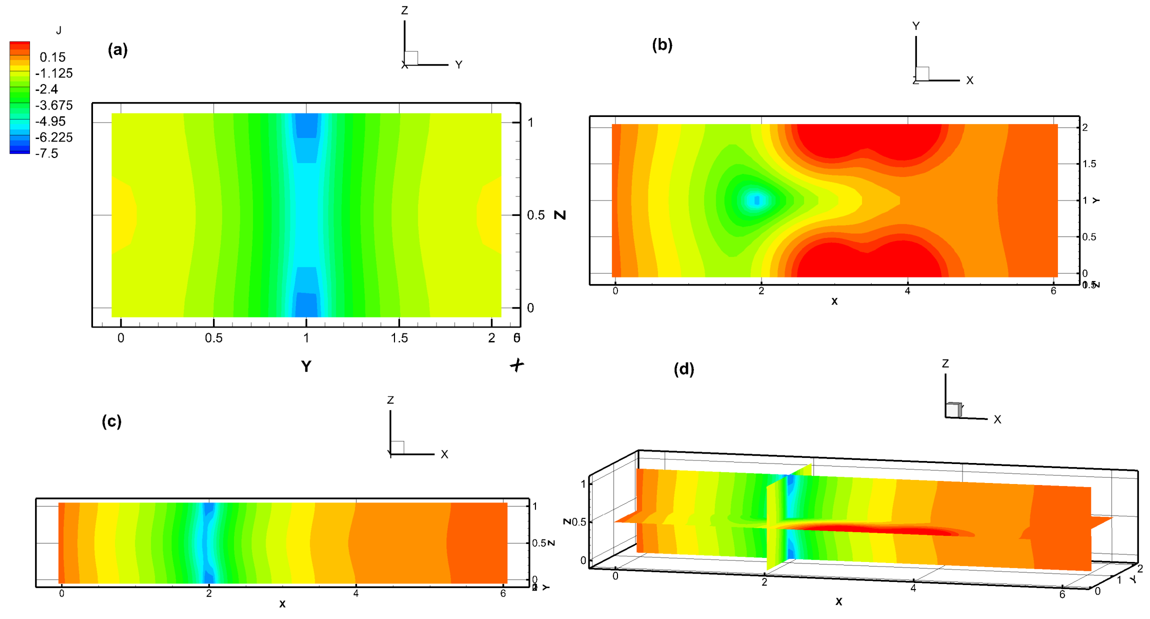


Figure 6.22: Functional surface for source inversion in the x, y, and z dimensions where the true source is located at (2,1,1).

Unfortunately, some variation of this problem is likely to occur in cases of source inversion problems with more than two unknown parameters, and with the inclusion of the third spatial dimension in particular. Simple methods of source regularization, which aim to recast an ill-conditioned problem as a well-conditioned one, do not address problems of non-uniqueness such as this example shows. The Tikhonov regularization that was introduced in Chapter III and is implemented in this work serves to smooth the functional surface and avoid small errors due to noise in the data. However, the smoothing that is implemented will do nothing to mitigate the symmetry problem that is shown above. In order for the optimization is cases such as these to be successful, further information about the solution must be incorporated into the problem either by more sophisticated methods of regularization or by further restricting the domain, ideally to a set of possible points such as is typically done in groundwater source inversion problems.

## 6.6 Summary and Discussion

This section has provided analysis of several inversion test cases with three or more optimization parameters. The results for the optimization of three parameters (two spatial parameters and the third comprising of the release time or initial source strength) show no obvious dependence on the starting trajectory locations within the parameter space.

In all cases, it is important to use the information that is known about the problem to narrow down the parameter space to a more tractable region as well as to make an initial guess that physically makes sense. Failure to account for these limitations can lead to a trajectory that begins or iterates to a region of the parameter space that is unsuitable for gradient-based optimization methods.

The hybrid approach was tested for the case of three and four optimization parameters. For the case of three parameters, the hybrid method gave results that were on a whole more reliable than that of the straight gradient-based method using OPT++ . It is likely that these hybrid results could further be improved by increasing the number of iterations for the global EA search. However, the convergence times are also higher for the hybrid approach. Additionally, the use of the hybrid scheme does not allow for a user-defined physically consistent set of initial parameter guesses, leading to more inconsistent results in the four-parameter space tested case.

Even with a good starting guess, serious problems occur with higher-number parameter spaces in regards to ill-posedness and non-uniqueness of the functional. These issues are well-known and significant limitations of the source inversion problem and inverse problems in general. As a result, gradient-based schemes are difficult to implement for source inversion with respect to several parameters. The functionals in these cases are frequently not amenable to gradient-based methods, as the gradient field associated with ill-posed problem may be divergent or contain multiple minima. This problem also occurs with hybrid methods, as the initial starting guess from the derivative-free solver is apt to still fall in a region of the functional that is not amenable to gradient-based optimization.

# CHAPTER VII

# Boundary Control of Hazardous Plumes

In the introduction, the automatic detection and control algorithm for hazardous plumes was described as consisting of two distinct phases. The first phase, the source inversion step, was presented in the previous three chapters of this dissertation. The outcome of the source inversion phase ideally includes the complete parameters necessary to accurately model the release and propagation of the plume in time and space. This information then becomes the initial conditions specified in the forward model for the boundary control phase, which is the subject of this chapter.

The boundary control algorithm is developed in the following sections, and its implementation using a Model Predictive Control approach is described. Several examples of are given for various scenarios using a two-dimensional domain and six ports that serve as actuators.

## 7.1 Optimization Approach for Boundary Control

As has been previously discussed, the goal of the boundary control phase is to implement control actions (actuator suction and/or blowing) that will meet a predetermined set of criteria that defines 'protecting' the domain from the hazardous plume. This criteria can be defined in several ways: perhaps the objective is to protect the entire domain, and thus the control actions implemented by the boundary ports need to remove as much of the contaminant as is physically feasible. At the other end of the spectrum, the goal may be to protect a single point in space, most likely requiring a more localized control action in order to deflect the plume away from the designated protected point.

While at first glance the above described boundary control algorithm may appear distinct from the source inversion problem, these two phases of the detection and control algorithm have in common that they can both be formulated as optimization problems. Hence the methodology employed for defining and solving the source inversion problem can be directly extended to the boundary control problem, allowing for the same basic model framework to be implemented with few alterations. In fact, the boundary control optimization problem is in many ways simpler than the source inversion problem, as will become apparent shortly.

The boundary control optimization problem, like the source inversion problem, is bound-constrained and non-linear in nature. Thus the same gradient-based optimizers available through DAKOTA that were applied to the source inversion problem are also appropriate for use in the boundary control algorithm. As before, these methods are the conjugate-gradient based CONMIN algorithm and the quasi-Newton based OPT++ algorithm, both of which are tested herein.

The optimization approach with respect to the boundary control phase again requires the definition of the objective function, the optimization parameters, and their respective bounds. Where the bounds in the source inversion problem took the form of spatial, temporal and source strength constraints, in the boundary control problem they are defined as the feasible limits on the port velocities. The objective function in this case can be formulated in a simpler manner than the source inversion problem (Eqn. (3.3)). This is because the goal in this problem is no longer to minimize the difference between a series of observations and the modeled data, but rather to minimize the concentration values of the modeled contaminant at the target protection point or points. Thus, the objective function need not be formulated in terms of a least-squares function. In addition, there is no need for regularization in a conventional optimization problem such as this. The objective function can instead be formulated simply as the sum of the concentration values at the protected points as are predicted to occur due to the source release over the prediction period $T$:

$$J(\mathbf{f}) = \sum_{p=1}^{N_p} \int_0^T C(\mathbf{x}_p)dt, \tag{7.1}$$

subject to

$$|U_i| \leq U_{max}, \tag{7.2}$$

where $\mathbf{x}_p$ is a vector of coordinates corresponding to the location of the $p^{th}$ protected point and $N_p$ denotes the total number of protected points in the domain. The vector of unknown parameters $\mathbf{f}$ in this case consists of the velocities at the ports. For a total of $i$ ports, this results in $\mathbf{f} = (U_1, U_2, ..., U_i)$. The constraints given by Eqn. (7.2) require that the predicted control velocities do not exceed the limits of the physical actuators. As with the source inversion problem, the minimization of Eqn. (7.1) is subject to the underlying equations of the CFD model defined in Chapter II.

It is worth noting that the objective function given in Eqn. (7.1) and the associated constraints are the most basic control formulation for this problem. Additional terms, called *penalty* terms, could be included in Eqn. (7.1) to incorporate additional constraints directly into the objective function, such as specifying a maximum number of ports that can be used at any given time or giving priority to certain protected points over others. Penalty terms typically are weighted to adjust for their relative importance in the optimization goal. While these applications are not explored in the current work, the inclusion of additional terms in the optimization problem would not change the algorithm that is utilized herein.

At this point, we can denote some fundamental differences between the previously described source inversion optimization problem and the boundary control optimization problem. First of all, while the source inversion optimization sought a single, global minimum to be successful, in the boundary control problem a global minimum does not necessarily exist nor is it required. This is because depending on the source and port locations, there will likely exist more than one control strategy that will serve to 'adequately' control the plume. For instance, consider the domain with 6 boundary ports as shown in Fig. 7.1. This domain is 10 meters long, 1 meter wide, and contains 6 actuators (ports) that are 0.2 m in diameter. In the case of a source that is released at coordinates of $(1, 0.5)$ with an optimization goal of protecting a single point $x_p$ at coordinates $(5, 0.5)$, blowing from either port 1 or port 4 could be initiated to deflect the plume away from this point with equal success due to the symmetry of the problem.

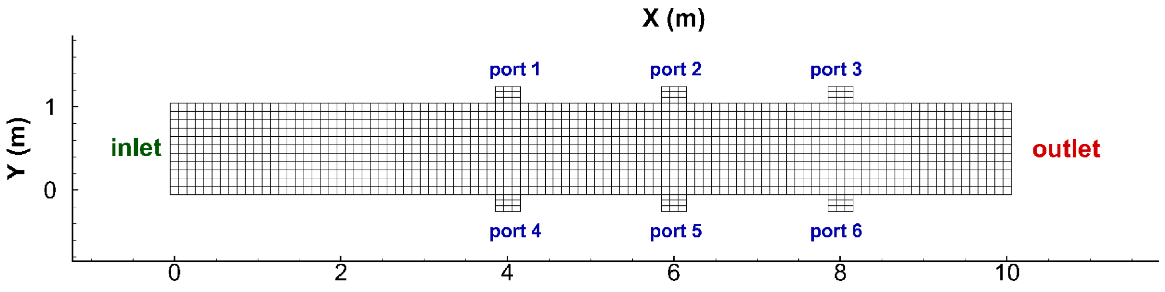


Figure 7.1: Computational grid for the boundary control problem with 6 ports.

Further, there is more leniency in the criteria that can be specified defining what 'success' is for the plume mitigation. Whereas one set of parameters was sought in the source inversion problem that would result in a (non-regularized) value of $J = 0$, the value of $J$ that is acceptable for the boundary control problem is a function of the maximum tolerance level for a given contaminant. Thus, the control may be deemed successful if the concentration at the protected points is reduced to below a designated 'hazardous' level. We can therefore specify convergence criteria that reflects this allowable limit, such as a maximum allowable $J$, denoted as $\delta$. The higher the value of $\delta$, the more flexibility in the combination of parameters that will effectively meet the convergence criteria and thus meet this criteria. In other words, in this case it is not necessary to find *the* optimal control strategy for the plume, but rather *a* control strategy which achieves the pre-defined objective function goal. Further, it cannot be expected that the optimization will be able to reduce the objective function adequately in all cases, requiring additional restrictions on the convergence criteria, such as a maximum time frame over which the optimization is carried out.

A second significant difference between the source inversion and boundary control optimization problems is due to the nature of the unknown parameters being optimized. In the source inversion problem, several issues arose due to the discretized variables in time and space and the resulting discrepancies between the test parameters stipulated by the optimizer and the available test points defined in the forward (CFD) model. However, with with respect to the boundary control problem the port velocities can be assumed to be continuous variables, and therefore can take on any value within their respective limits and thus avoiding the discretization issues. Even if discrete values were to be specified for the optimal control problem, because there may be several possible combinations of parameters that result in an acceptable outcome, the restriction of matching the CFD and optimizer requested values is not as stringent. In other words, the main difficulty in the source inversion problem was that a single, unique solution was sought, leaving little room for error. With the boundary control problem, small errors in the optimization trajectory are unimportant as long as the end result achieves the goal of sufficiently mitigating the contaminant plume.

Another simplifying factor in the boundary control problem is in regards to the numerical estimation of the gradients. In the previous source inversion chapters, it was shown that the intrinsic derivative estimation routines supplied by DAKOTA were not appropriate for that problem, due to the method of scaling the gradient step-size $h$ as a function of the magnitude of the currently evaluated parameters. With respect to the vector of unknown velocities in the boundary control problem, this method is not logical either, as tested port velocities closer to zero do not indicate the need for a smaller gradient step-size. However, because there is a more narrow range of possible parameter values in this optimization problem , the use of the scaled gradient step-sizes is not a detriment to the boundary control algorithm, and in fact, there are several benefits to using this method over the user-calculated gradients.

The main benefit to using the internally calculated vs. user calculated gradients is a matter of simplicity with respect to the linked optimization-CFD framework. That is, when specifying 'internal gradients' in the DAKOTA input file, the previous requirements of replacing the pertinent input files with the perturbed values, re-running the CFD model, and formatting the output to be read into DAKOTA are bypassed. Instead, these steps are automatically carried out in the DAKOTA optimization procedure.

As was discussed in Chapter IV, the use of internally calculated gradients requires a scale-factor value to be defined by the user. Because of the use of continuous-valued parameters for the port velocities, there is no restriction on the minimum $h$ as was the case for the discretized time and space parameters in the source inversion problem. Hence the choice of the scale factor $\delta_d$ is not limited by modeling constraints. However, its selection will affect the convergence speed of the optimization. A value too small will lead to longer convergence times, while a value too large may lead to erroneous gradient estimations. For the subsequent boundary control examples in this chapter, a value of $\delta_d = 100$ was experimentally found to be a good choice for this parameter, although it is not claimed that this is the optimal choice.

The ports in the ensuing boundary control simulations are assumed to encompass the same range of possible velocities, and thus, the elements in $\mathbf{f}$ are all relatively equal in magnitude. This is an additional factor that facilitates the optimization process, and negates the necessity of scaling the parameters. However, the use of scaling may increase the efficiency of the optimization process, and while this is not an aspect that is considered in the present work, it could be investigated in future studies.

A final point of comparison between the present problem and the source inversion phase is with respect to the initial starting guess $\mathbf{f_0}$. It was previously shown that the source inversion results were very sensitive to the choice of $\mathbf{f_0}$, although there was not always a predictable range of these values which resulted in a reliable outcome. In the boundary control optimization, the choice of the initial guess is less of a deciding factor in the outcome of the optimization. This is in part because the range of possible values for the parameters in the current problem formulation, which are given by the maximum possible velocities of the actuators, is a relatively narrow range of values compared to some of the unknown parameter ranges required for the source inversion problem. Secondly, the possibility of more than one adequate control strategy which provides more flexibility in the optimization trajectory leads to a lesser dependence of the boundary control optimization on the initial starting guess. Although the choice of $\mathbf{f_0}$ will inevitably still have an effect on the particular measures that are implemented to carry out the boundary control, it is less critical to achieving the end goal of this optimization problem.

## 7.2 Problem Description

Due to the differences discussed above, the boundary control problem is overall an easier optimization problem to solve numerically than the source inversion problem. However, as the goal of the boundary control problem is time-dependent, there are additional considerations in this optimization problem that must be addressed. Specifically, in the source inversion problem the results of the optimization were a single set of parameters characterizing the initial conditions of the contaminant plume at a single point in time. By contrast, the output that is desired from the boundary control algorithm needs to be updated over a number of optimization periods.

For example, consider the following scenario: a contaminant plume is instantaneously released in the channel depicted in Fig. 7.1 with the following initial conditions: $\mathbf{f_{source}} = (x_s, y_s, q_s, t_s) = (0.25$ m, $0.5$ m, $200$ units, $0$ s$)$. As before, it is assumed that the basic flow characteristics (ambient velocity, diffusivity) are known. Let's also assume that the outcome of the source inversion process is completely accurate, so that the forward model is able to simulate the propagation of the plume in time and space with no measure of uncertainty. Given this information, we wish to carry out a boundary control optimization in which the goal is to protect 15 arbitrarily determined points, shown as red diamonds in Fig. 7.2. This requires minimizing the functional given by Eqn. (7.1) in order to optimize the parameters that represent the 6 port velocities, where the bounds for this scenario are arbitrarily defined as $\pm 3$ m/s on all ports.

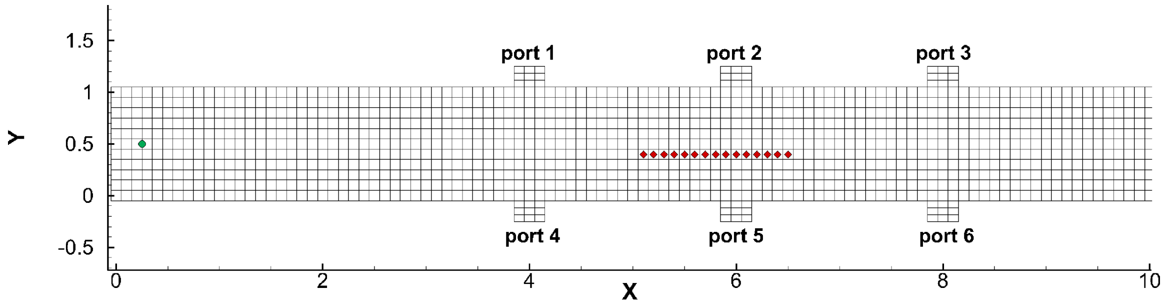


Figure 7.2: Boundary control example with 15 protected points.

The simplest way to implement this boundary control optimization would be to assume that the ports are running continuously and with constant velocities, in which case a single optimization would be performed over a defined period of time $T$. The objective function is minimized by evaluating the simulated concentration values at the protected points as a function of the tested port velocities in $\mathbf{f}$ until the predefined termination criteria is reached. As noted above, the choice of the termination criteria is a matter of the desired accuracy and efficiency of the boundary control algorithm, and is discussed further below. For the time being, we can stipulate that the algorithm terminate after either (1) the objective function value falls below the maximum allowable threshold $\delta$ or (2) after a maximum number of iterations $N_{max}$ is reached. This allows for the optimization to terminate even in the case that the objective function is not able to be significantly minimized given the problem conditions. For example, had the plume been release directly at a protected point, in which no control action could be taken to protect the point fully at all times. In this example the termination values are set at $\delta = 10^{-3}$ and $N_{max} = 3$ and whichever criteria is reached first will signal the optimization process to terminate.

With sufficiently high blowing and suction capabilities of the ports, this single-optimization framework will allow for nearly complete mitigation of the plume via the upstream ports, as can be seen in the time series of the contaminant concentrations within the domain (Fig. 7.3). The velocity controls implemented at the boundary ports corresponding to this simulation are graphed in Fig. 7.4(a). Note that in this figure and the other velocity plots in this chapter, positive velocities correspond to the positive direction on the y-axis, and thus represent suction (fluid flow out of the domain) for ports 1-3 and blowing (fluid pushed into the domain) for ports 4-6. The converse is true for negative velocities. In this example the CONMIN algorithm was used as the optimizer.

The corresponding sum of the contaminant mass that passes through the protected points, with and without boundary control, is shown in Fig. 7.4(b). Note that in this scenario, the plume is released 3.65 meters upstream of the first set of boundary ports, and the maximum center-line velocity of the ambient flow is 1 m/s. These conditions provide adequate time and space for the ports to nearly completely mitigate the plume using velocities with the specified limits, reducing the total contaminant mass that reaches the downstream protected points to a negligible amount.

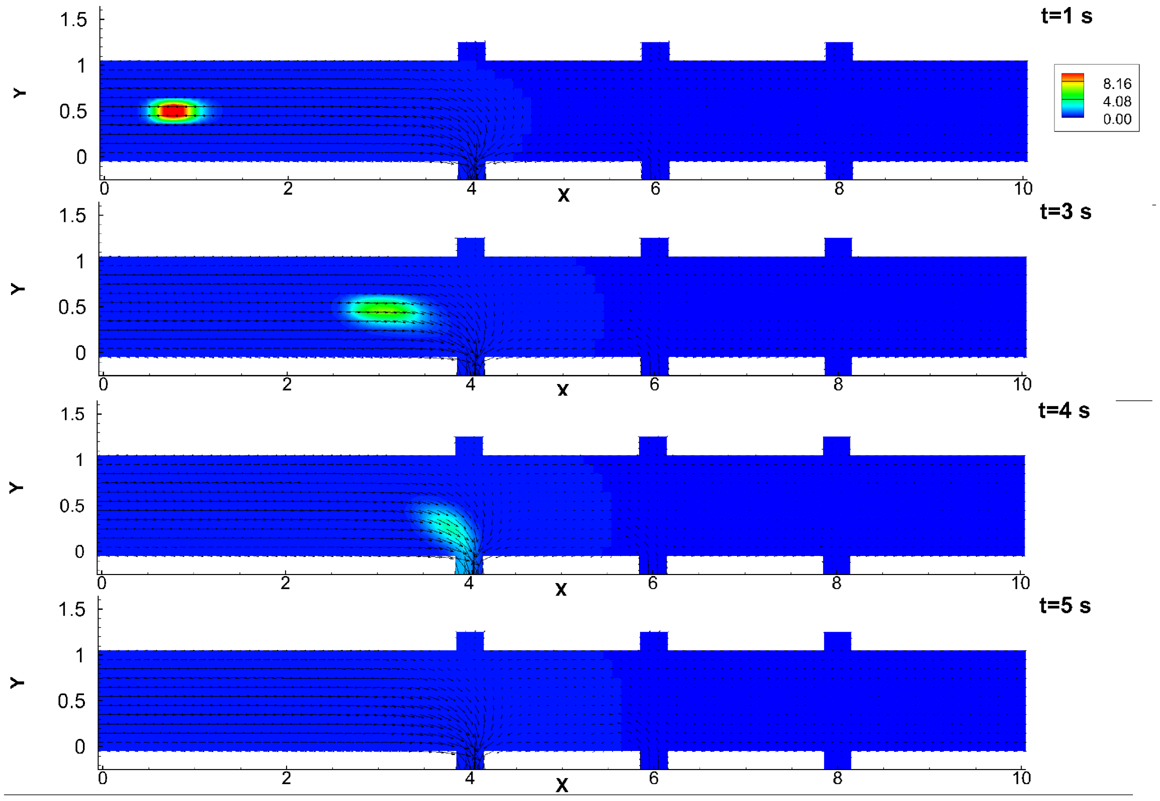


Figure 7.3: Boundary control by continuous port action where $T = 10$ seconds and maximum port velocity is $\pm 3$ m/s for a source release at coordinates (4.5, 0.5) and using CONMIN optimization.

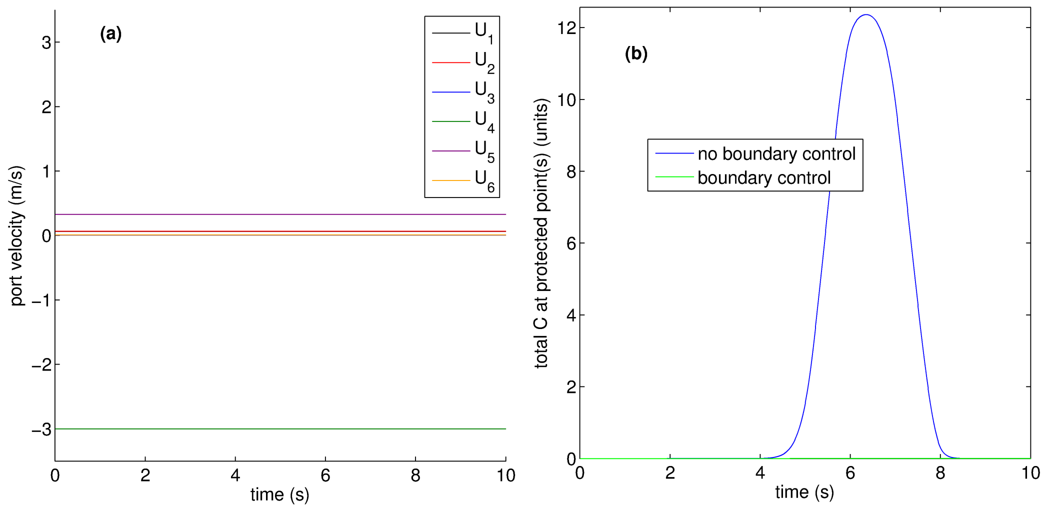


Figure 7.4: Port velocities (a) and the total contaminant amounts measured at all protected points with and without the controls implemented (b) for the case of continuous boundary control using CONMIN.

For comparison, the above scenario can be repeated with the location of the source release moved downstream to $x = 4.5$ m. Unlike the previous example, where the upstream ports had adequate time to remove the plume completely via suction, the control actions in this case serve to deflect the plume away from the protected points. This deflection occurs almost entirely by the blowing action implemented at Port 5 (Figs. 7.6 and 7.5(b)). Because the release occurs so close to the first protected point, the mitigation is not completely successful, as can be seen in the contaminant values that are measured at the protected points in Fig. 7.6(b). However, the amount of contaminant at these points is greatly reduced compared to the amount that would have resulted without any boundary control actions implemented.

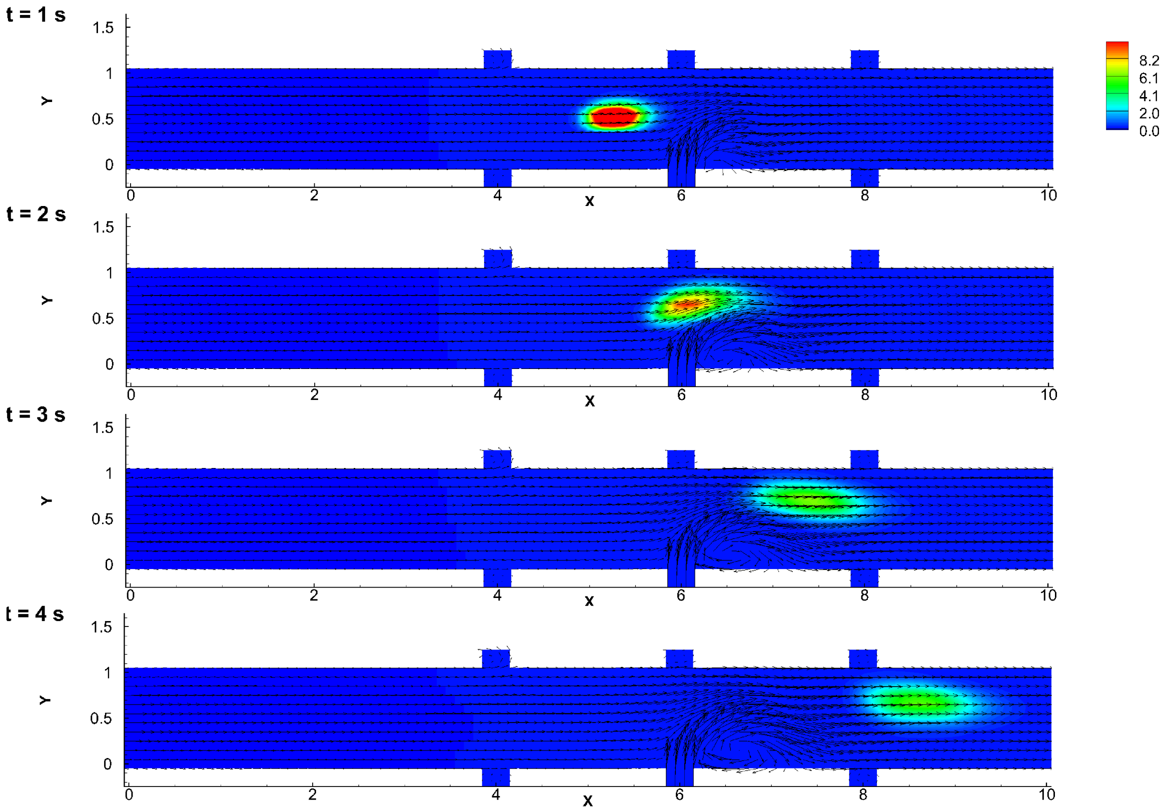


Figure 7.5: Boundary control by CONMIN continuous port action where $T = 10$ seconds and maximum port velocity is $\pm 3$ m/s for a source release at coordinates (4.5, 0.5).

While the continuous port action method implemented in the above two examples

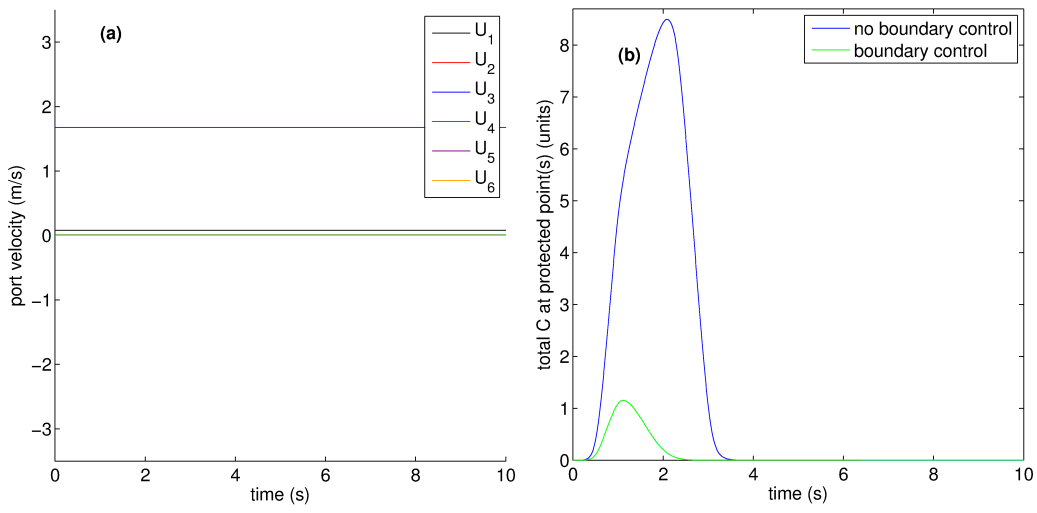


Figure 7.6: Port velocities (a) and the total mass of contaminant measured at the protected points with and without the controls implemented (b) for the case of continuous boundary control with source release at coordinates (4.5, 0.5).

is effective, there are two reasons why it cannot be adopted for the boundary control algorithm. The first reason is a logistical one: assuming that the actuators run continuously is not a realistic supposition for a real-life scenario. Instead, it is necessary to develop an optimization algorithm that allows for the ports to be turned on and off as a function of the contaminant levels that are present in the domain at a given time.

Secondly, it cannot be assumed in a realistic scenario that the model predicted trajectory will be completely accurate. That is, there will be errors in the predictive model as a function of variations in the flow field, noise in the data, and any other unforeseen affecting factors. Determining a single control strategy to be implemented over one time-frame does not allow for correction of the control measures in light of these uncertainties. Thus, it is desirable to develop a control strategy framework that allows for re-evaluation of the current state and adjustment of the controls based on their effectiveness.

In order to provide these dynamic capabilities, more than one simulation period needs to be carried out, with the controls updated as a function of the current conditions at specified intervals. This stepped-control approach can be achieved using a Model Predictive Control framework. The basic MPC approach has been given a cursory introduction with respect to the source inversion problem in Chapter V. In the following section, the MPC approach as it relates to the boundary control problem is developed and described. It should be stressed that a rigorous application of MPC in respect to dynamical systems theory is not implemented in this work, and thus the reader is referred to the texts by Grune and Pannek (2001) and Kwon (2005) for a more in-depth treatment of the topic.

## 7.3 Model Predictive Control

Model Predictive Control, which is also known as receding horizon control, has its roots in control theory and dynamical systems. The central idea of MPC is to steer a system towards a desired state by minimizing an objective function (typically referred to as a *cost functional* in the field of MPC) over a *predictive control horizon.* In order to flesh out this concept, the analogy to a game of chess can be employed. In chess, it is advantageous to always be thinking several moves ahead, with the goal ultimately being to put one of your pieces in a position to capture your opponent's king. However, even while planning your moves several steps ahead, you are only able to implement the first move of your planned strategy at every turn. Further, your 'optimal trajectory' is dynamically changing in response to the moves made by your opponent. Thus, in order to win the game, you must re-evaluate your strategy regularly, and correct it to reflect the information that is available at the present moment. With a little more imagination, we can stretch this analogy to encompass the MPC concept of feedback as well. Suppose that after an initial time period of playing your opponent, you find you are able to better guess the moves *he* will make in response to your moves. This physiological game-playing aspect has a corollary in the MPC framework, whereby past information can be accounted for in the trajectory in order to improve the future predictions. The main steps required for this process are enumerated as follows:

1. Estimate the current system state (appraise the current layout of the board).
2. Find the optimal parameters for a designated time-frame that spans the present time to some point in the future, based on the current state (plan your moves from now until you are able to capture the king, or a similar advantage position. Note that this time-frame, $T$, is arbitrary depending on how far ahead you desire or are able to plan, and in the chess analogy is not in direct reference to 'time' but rather the number of future 'steps'. Also note that incorporated in your decision making at this step is any prior knowledge of your opponents strategies).
3. Apply the determined actions at the present time through the next stipulated application time (when it is your turn, apply the first move of your strategy. In chess, this *advancement period*, $T_a$, is automatically defined from one turn to the next).
4. Repeat at the next evaluation point (move ahead to the next series of turns, and repeat the process from Step (1)).

This general process is illustrated in Fig. 7.7. The above described process includes several components that are essential for any MPC procedure. These components include:

- The finite prediction horizon $T$: This is the defined time-frame over which the controls are optimized, and remain constant throughout the MPC process. When evaluating the problem, the future behavior of the system is predicted from the current time, $t$ to the time $t + T$.

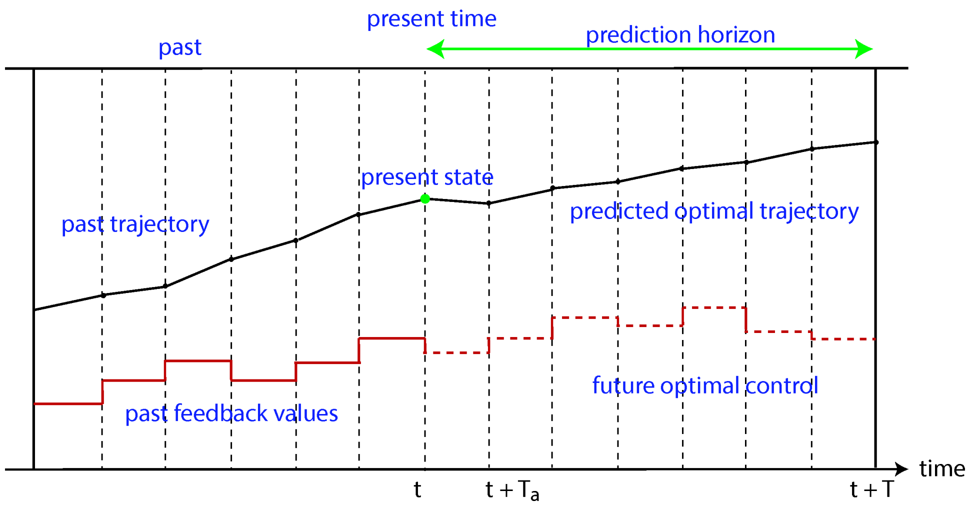


Figure 7.7: Concept of Model Predictive Control.

- The advancement time, $T_a$: This time is how far ahead the controls are applied at every evaluation. By definition, the advancement time should not exceed $T$. The controls determined at each prediction horizon evaluation will then be applied from $t$ to $t + T_a$, and the present evaluation time will be moved ahead, $t = t + T_a$. Figure 7.8 illustrates this concept of the time advancement window.

- The predictive (forward) model: The more accurate this model is, the more accurate the control strategy will be.

- The objective function $J$: This function is formulated to provide a metric for evaluated the effectiveness of the control actions, and thus is formulated to best steer the system to the desired state.

With the basic MPC process defined, we can now relate the above described concepts to the current boundary control algorithm, where in this application the system is being 'steered' towards the state of having no detectable levels of contaminant at the protected points of the domain. As has been discussed, this goal is not always possible, but each 'move' that is made by the control strategy attempts to adjust the system to be as close as possible to this state given the information available over

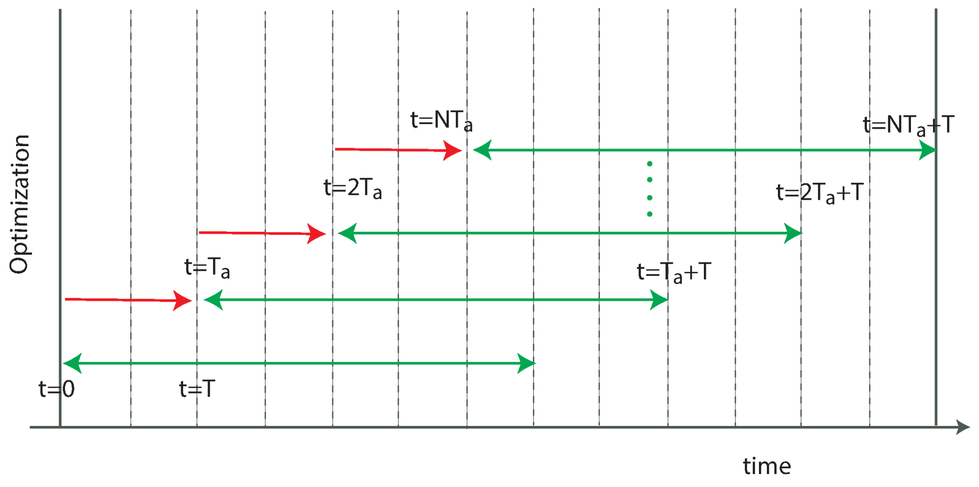


Figure 7.8: Schematic illustrating the advancing prediction horizon for MPC.

the current evaluation time-frame. In step (1) above, the system evaluation requires modeling the flow field, including the contaminant values $C$ at the protected points, over a prediction horizon $T$. Step (2) requires solving the optimization problem for the parameters $\mathbf{f}$ that will minimize the objective function (a measure of how much contaminant affects the protected points) over the evaluation time period. Step (3) applies the optimized control parameters $\mathbf{f}$ for a time period $T_a$, which must be equal to or less than the optimization time period, $T$. Having completed the current 'move', the evaluation period advances to $t = t + T_a$ and the next optimization period spans $t$ through $t + T$. The process terminates when the specified termination criteria is met.

As previously noted, the MPC framework introduced above allows for the incorporation of past control action information in the form of feedback into the current and future control actions. In the current boundary control problem, feedback is not applied, although this framework provides the possibility of re-engineering the detection and control algorithm in order to allow for model predictive feedback to be used. The main advantage to doing so, as discussed above, would be to allow for a more accurate control simulation in the case of unsteady or unpredictable flow conditions. For example, in a realistic flow there will be perturbations to the ambient velocity, and thus the forward model of the contaminant would be unable to model the trajectory of the plume with complete accuracy. The model feedback would allow for corrections to the optimization trajectory that incorporate the deviations in the trajectory from the model predicted values. As the examples herein assume steady-state and known flow conditions, the use of the feedback model-based control is not currently employed.

The algorithm for the hazardous plume boundary control can now be explicitly outlined. This algorithm is shown below in Algorithm 5.

**Algorithm 5:** Boundary control algorithm

```
Initialization: set $T$, $T_a$, $T_f$, $N_{max}$, $\delta$, $i = 0$
while ($iT_a < T_f$) do
    if $J > \delta$ then
        optimize $\mathbf{f_i}$ over $iT_a$ to $iT_a + T$;
        apply $\mathbf{f_i}$ for period $iT_a$ through $iT_a + T_a$;
    else
        set $\mathbf{f_i} = \vec{0}$;
        apply $\mathbf{f_i}$ for period $iT_a$ through $iT_a + T_a$;
    end
    i=i++
end
```

Note that this algorithm continues running until a termination time, $T_f$, has been reached, even after the objective function termination criteria is met. The reason for this is that depending on the length of the time horizon, there may be evaluation periods in which the objective function is very low because the contaminant plume has not yet reached the protected points. Thus, a $J$ below $\delta$ cannot be deemed a sufficient condition to terminate the optimization. Instead, when this occurs the optimization step is by-passed, the velocities are set to zero at the ports, and the evaluation period is advanced as normal. The choice of $T_f$ then must be long enough to span the period from the initial release $t_0$ to the time when the plume is assuredly no longer a threat, due to either the control actions at the port or its advection out of the domain of interest.

As an initial example of how Algorithm 5 is applied to the boundary control problem, we repeat the first test case from Section 7.2, which was initially carried out using a single run of the optimization over 10 seconds ($T = 10$ s, $T_f = T_a = T$) using $T = 10$ s, $T_a = 4$ s and a $T_f$ of 20 seconds. The velocities at the ports and corresponding total contaminant levels for this scenario using the MPC approach are shown in Fig. 7.9(a) and Fig. 7.9(b), respectively. Selected snapshots of the plume mitigation corresponding to this control sequence given by Fig. 7.9(a) are shown in Fig. 7.10.

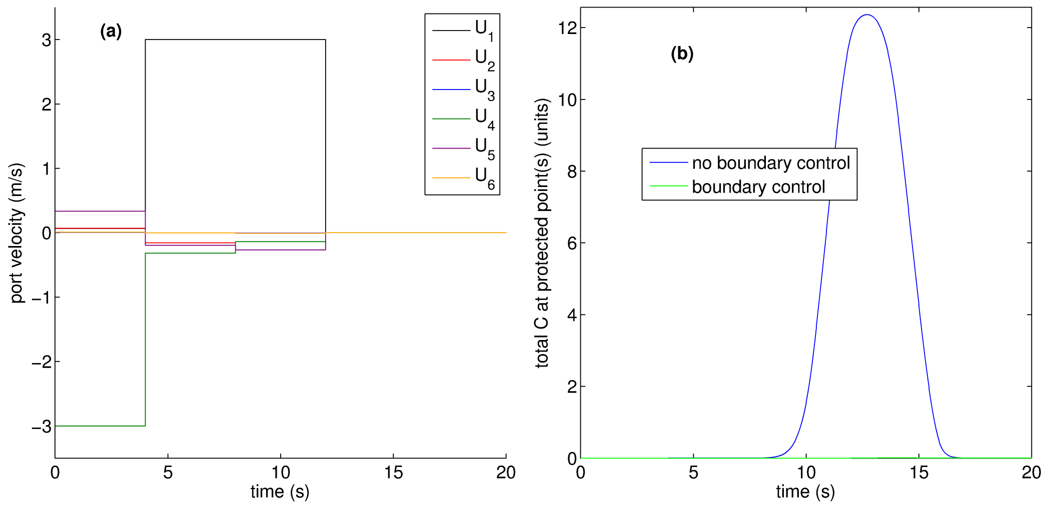


Figure 7.9: Port velocities (a) and the total contaminant amounts measured at all protected points with and without the controls implemented (b) for the case of MPC continuous boundary control.

As can be seen in this example, the main control actions in this case are carried out by port 4 and port 1. Initially, port 4 is turned on and uses suction to reduce the

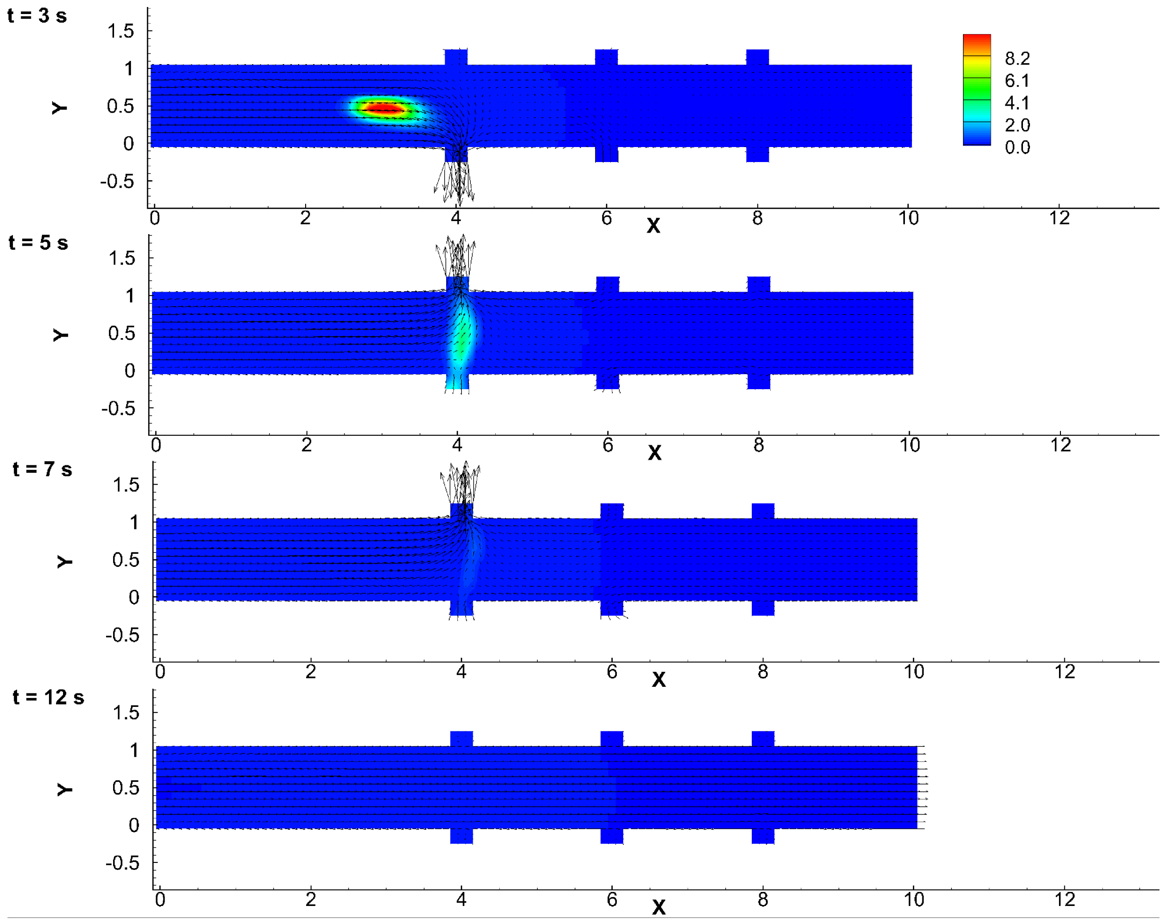


Figure 7.10: MPC boundary control by CONMIN continuous port action where $T =$ 10 seconds and maximum port velocity is $\pm 3$ m/s for a source release at coordinates (0.25, 0.5)

contaminant and to slow its propagation towards the downstream protected points. During the second implemented control phase (beginning at $t = 4$ s), the velocity at port 4 is reduced and suction is implemented at port 1 which serves to remove the majority of the remaining plume. By the forth optimization phase, the termination criteria are met and the ports are turned off.

## 7.4 Model Parameters

Before presenting further boundary control examples, details on the computational set-up for these simulations and the modeling and optimization considerations particular to the boundary control phase are provided in this section.

### 7.4.1 CFD Specifics

An additional modeling concern for the boundary control algorithm is the choice of time-step for the CFD model. Because the maximum CFL number in the domain is a function of the velocities, and the velocities at the ports are not known a priori, a conservatively small time-step must be chosen to ensure the maximum CFL number stays below one at all times. The scenario corresponding to the smallest possible required time-step is the case when all of the ports are operating at their maximum velocities. For the following scenarios this approach is used in order to estimate a limiting time-step to be used as a function of the bounds placed on the actuators. For the bounds given by $|U_i| \leq 3$ m/s, a uniform time-step of $\Delta t = 0.01$ s is used in order to guarantee numerical stability.

The k-$\epsilon$ RANS model is used in the following simulations as in the previous chapter's source inversion simulations, where the initial and boundary conditions are calculated using the relations in Chapter II as before. Again, the kinematic viscosity is set at a value appropriate for water at 20 $^oC$, $\nu = 1 \times 10^-6$ m$^2$/s. The values for $k$ and $\epsilon$ at the inlet are again estimated using relations (2.12) and (2.13), where in this case the reference velocity is $U_{ref} = \frac{2}{3}U_{max} = \frac{2}{3}$ m/s, and the turbulent intensity is estimated as 5% of the average inlet velocity, $I = 0.05U_{ref} = 3.33 \times 10^{-2}$. The turbulent length scale is again taken to be 7% of the characteristic inlet length. With these values $k = 1.2 \times 10^{-2}$ m$^2$/s$^2$ and $\epsilon = 1.5 \times 10^{-3}$ m$^3$/s$^2$.

A summary of the boundary conditions used in the simulations in this chapter is shown in Table 7.1. The 'normal fixed value' boundary condition that is specified at the ports is used to enforce the optimized parameter values to be velocities with direction strictly normal to the ports.

Table 7.1: Boundary conditions for the boundary control simulations.

| | p (m$^2$/s$^2$) | U (m/s) | C (units) | k (m$^2$/s$^2$) | $\epsilon$(m$^2$/s$^3$) |
|---|---|---|---|---|---|
| inlet | ZG | parabolic, $U_{max} = 1$ | 0 | $7.4 \times 10^{-4}$ | $4.7 \times 10^{-5}$ |
| outlet | 0 | ZG | ZG | ZG | ZG |
| walls | ZG | 0 | ZG | standard wall function | standard wall function |
| ports | ZG | normal fixed value | ZG | ZG | ZG |

ZG=zero-gradient.

### 7.4.2 Optimization Settings

In the boundary control simulations herein, a maximum number of iterations is imposed on the optimization algorithm. The main reasons for enforcing this limit are two-fold: First of all, as with the source inversion process, a goal of this research is to investigate the feasibility for applying the automatic detection and control algorithm to a real-time, realistic scenario. As was shown in the chapters on the source inversion phase, the optimization run-time can greatly exceed real-time capabilities, especially for the cases involving several unknown parameters. However, it was also noted that there are possibilities for reducing the overall run-time that are not explicitly investigated in the present work. The goal of the present work is to investigate the effects of various approaches on the over-all accuracy and efficiency of the methods tested. In the boundary control problem, it is found that without imposing a limit of the optimization of the port velocities, the optimizer has a difficult time converging. This is in part due to the non-uniqueness of the boundary control functional. That is, as discussed above, there may be several combinations of port velocities that achieve the goal of adequately minimizing the objective function. Without adding additional constraints and requirements to the optimization problem, in the form of penalizing higher velocities for example, this leads to the optimizer have a difficult time 'choosing' the best scenario out of several. In other words, this is an under-constrained problem with many degrees of freedom.

As emphasized earlier, because the optimization goal is this work is simply to mitigate the plume within the physical constraints placed on the actuators, the output of the optimization need not be the most efficient control method, so long as the job gets done. Hence, ideally the optimization should be allowed long enough to find a combination of port velocities that will result in the sufficient amount of reduction in contaminant concentration at the specified points. In some cases, as shown for the first illustrative case in Section 7.2, the specified points are able to be completely protected from the hazardous plume, meaning the control actions either fully divert the plume from the protected points or remove it from the domain before it propagates to the points of interest. In others, such as the second example in Section 7.2, the control actions will only be able to reduce the mass of contaminant. In either case, the number of iterations is an important consideration as it provides a measure of control over the relative importance of accuracy and efficiency in the boundary control algorithm.

Along these same lines, the choice of convergence criteria is an important factor to consider, also affecting the accuracy and efficiency of the algorithm. As discussed above, the current algorithm is configured to run for a time period of $T_f$, but allows for the possibility of by-passing this optimization process if $J < \delta$. In the following examples, the value of $\delta$ is taken to be $10^{-5}$. This rather stringent termination criteria leads to the ports staying on longer than may be necessary in some cases, as is exhibited in some of the results below.

Finally, as discussed above, the initial starting guess $\mathbf{f_0}$ is not as much of an inhibiting factor in the optimization problem as it was in the source inversion phase, however the starting guess will still alter the particular control strategy that is determined by the MPC process as well as the total convergence time. The present research does not present a formal study of these effects. Instead, all of the simulations in this chapter the optimization are initiated with starting guess $f_0 = \vec{0}$, indicating all ports are turned off. The reason for this is that initial experiments indicate that choosing higher values of the initial port velocities results in overall more extreme optimized control actions. From a practical standpoint, it is desirable to use a control strategy that is economical from the perspective of using the lowest possible velocities to control the port. One reason for this is to reduce the amount of turbulence in the flow system that will be induced due to the port suction and blowing. Turbulence will further spread the plume and possibly negate the control actions. Further, in a human-inhabited domain, extreme velocities could cause harm to the by-standers, and thus should be avoided if possible.

### 7.4.3 OPT++ vs. CONMIN

One of the conclusions drawn from the source inversion problem explored in the previous chapters is that the OPT++ algorithm was consistently more reliable than CONMIN for the majority of the tested source inversion scenarios. The main reason for this is related to the underlying OPT++ algorithm, which tends to make larger steps in the optimization trajectory, allowing for the optimizer to 'escape' intractable regions of the functional and to continue progressing instead of getting stuck due to the discretized parameters. However, it was also determined that the OPT++

algorithm was typically slower to converge than CONMIN for the tested problems. For the source inversion phase, where a unique solution is sought with as little error as possible, the lengthier run-time of the OPT++ algorithm was deemed preferable to the lesser reliability of the CONMIN algorithm.

For the boundary control problem, as has been discussed, we are not seeking a *unique* solution but rather an *adequate* solution, hence there is more flexibility in the outcome of the optimization routine. It follows then that as long as either algorithm is able to determine an effective control strategy, the optimizer that gives a more efficient result is preferable, where 'efficient' refers to both the run-time of the algorithm and the prudent usage of the ports. In this respect, the previously found tendency of the OPT++ algorithm to jump around and test extreme values of the parameters is not always beneficial or necessary in regards to the present optimization problem. As an illustration of this, compare the results in Fig. 7.11 and Fig. 7.12 to the above continuous port boundary control carried out using CONMIN shown in Fig. 7.3 and 7.4. The control actions determined by either optimizer in this case are sufficient to completely protect the specified points from the plume. However, the actions determined via OPT++ are less judicious. Although the plume is removed at port 4 as with the CONMIN results, the OPT++ controls also specify blowing at full capacity to be carried out by port 5. This action serves to deflect any remaining amount of contaminant that makes it downstream from the protected points and possibly to retard the propagation of the plume towards this region, however the action does not seem necessary since the plume is almost completely mitigated upstream by port 4 and the amount that is left to propagate downstream is potentially negligible, as is indicated by the CONMIN results.

While in this example the control actions determined by OPT++ are excessive, in less ideal situations the algorithm is able to better control the plume than CONMIN in the same number of iterations. This is illustrated by repeating the second example in

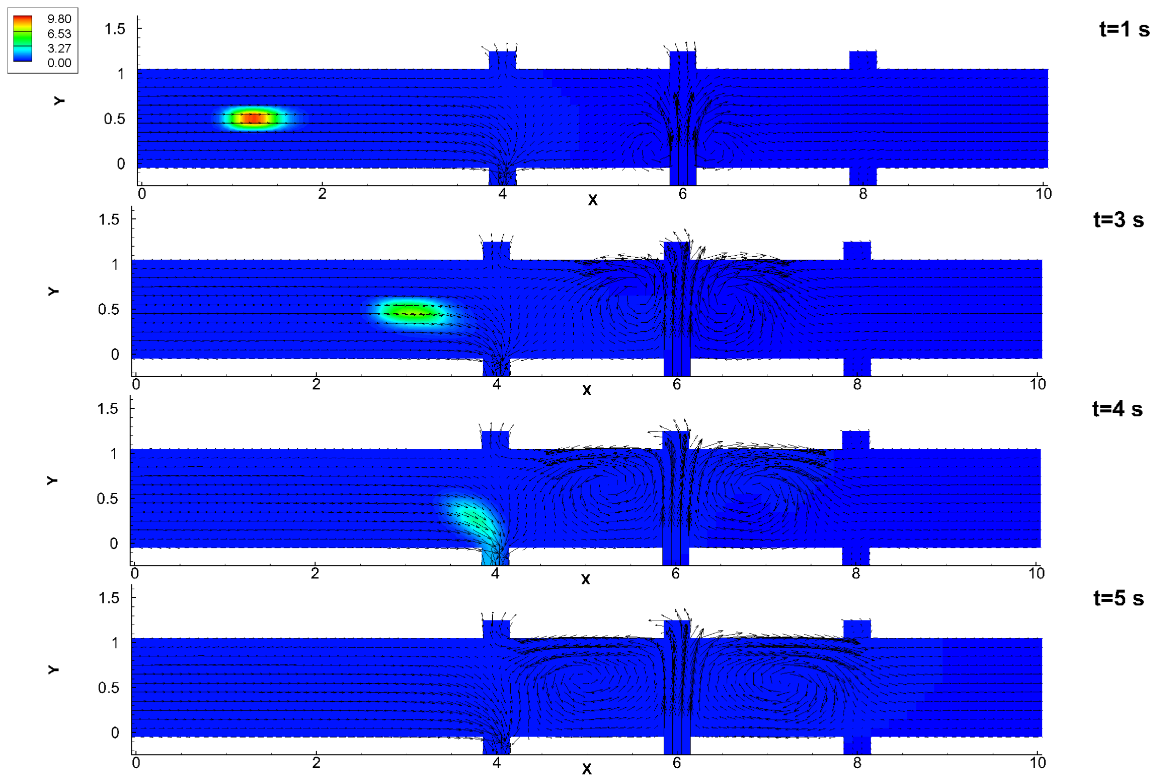


Figure 7.11: Boundary control by OPT++ continuous port action where $T = 10$ seconds and maximum port velocity is $\pm 3$ m/s whereby the source is released at coordinates (0.25, 0.50).

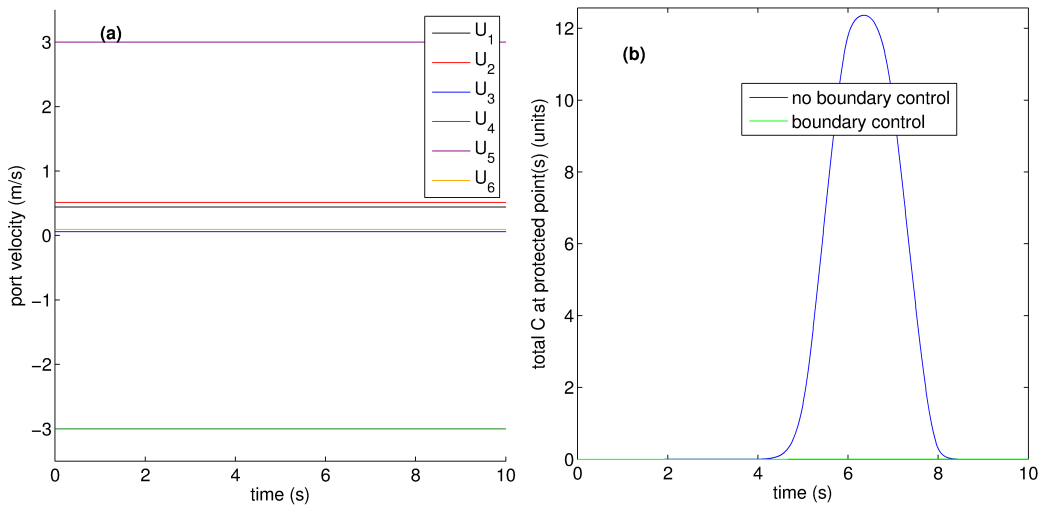


Figure 7.12: Port velocities (a) and time history of the total contaminant concentrations passing through the protected points with and without the controls implemented (b) for the case of continuous boundary control using the OPT++ optimizer and source release at coordinates (0.25, 0.50).

Section 7.2, whereby the source was released at a shorter distance from the protected points in the domain, using OPT++ instead of CONMIN. The port controls and subsequent reduction in the amount of contaminant that makes it to the protected points using CONMIN optimization were shown in Figs. 7.5 and 7.6. The analogous results using OPT++ are shown in Fig. 7.13 and Fig. 7.14. As with the CONMIN results, due to the lack of time and space for the controls to mitigate the plume, the OPT++ determined controls are not able to fully protect the specified points either, resulting in a maximum concentration of 0.35 units measured over the protected region. However, the OPT++ algorithm is more effective than CONMIN, which led to a peak of $C = 1.15$ units in the affecting contaminant concentration, and overall resulting in a higher degree of reduction of the amount of contaminant mass that reaches the protected points.

With respect to the computational efficiency of the two optimizers, as with the source inversion problem, the conjugate-gradient based CONMIN algorithm tends to be more efficient than the quasi-Newton OPT++ algorithm. For the first scenario

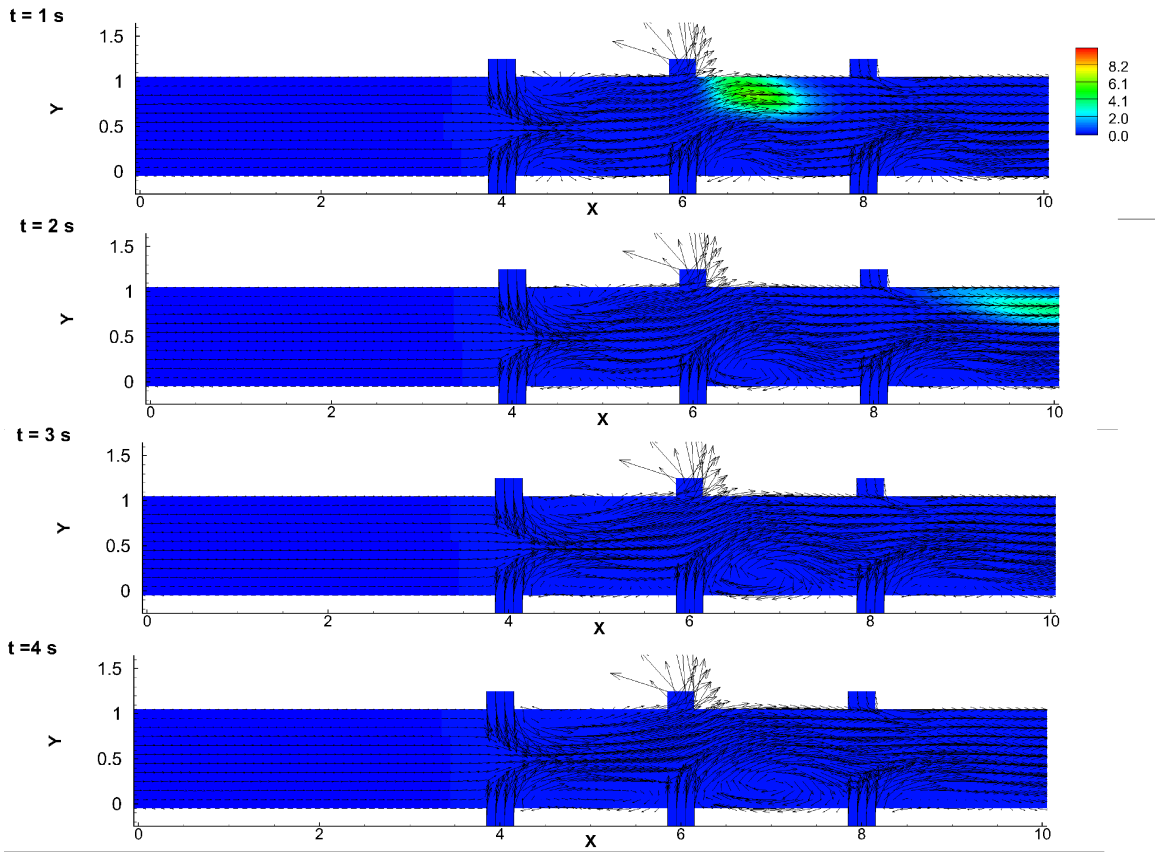


Figure 7.13: Boundary control by OPT++ continuous port action where $T = 10$ seconds and maximum port velocity is $\pm 3$ m/s whereby the source is released at coordinates (4.5, 0.50).

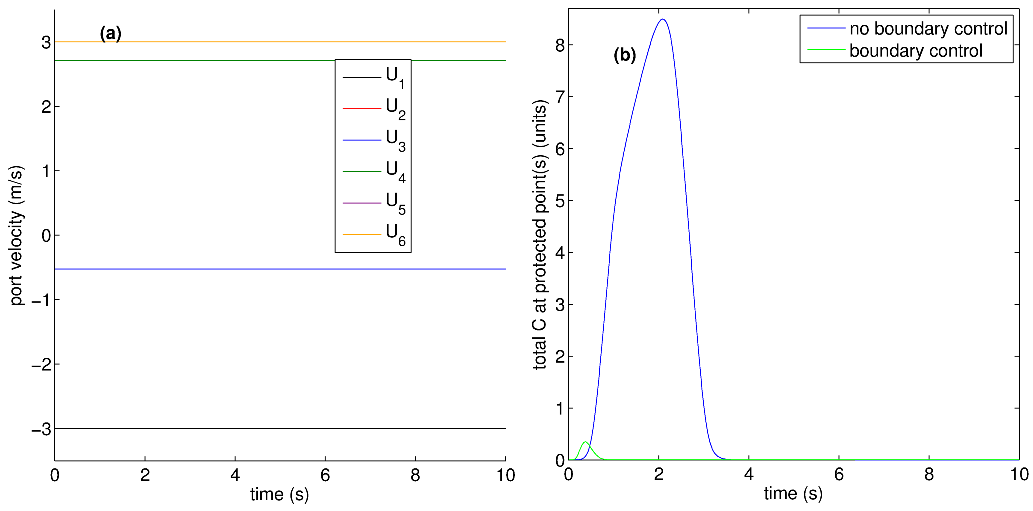


Figure 7.14: Port velocities (a) and time history of the total contaminant mass passing through the protected points with and without the controls implemented (b) for the case of continuous boundary control using the OPT++ optimizer and source release at coordinates (4.5, 0.50).

in which the plume is release at (0.25, 0.5), the CONMIN method converged in 9.8 minutes, while OPT++ took nearly twice as long at 18.4 minutes. In the second scenario with the contaminant plume released further downstream at (4.5, 0.5), the difference in optimization times were much less significant; 318 seconds for CONMIN vs. 338 seconds for OPT++. Note that all of these simulations enforced a maximum number of iterations $N_{max}$ of 3.

Overall, in regards to the choice of optimization method as well as the determination of the maximum allowable number of iterations and the maximum acceptable $J$, $\delta$, we face the typical choice between accuracy and efficiency. The present research does not endeavor to parse the relative importance of these two criteria in regards to either the source inversion or boundary control phases of the automatic detection and control algorithm. The present work instead is focused on exploring the differences in methods and their respective impacts on the efficiency and accuracy of the algorithm components. However, based on numerical experiments conducted via the models developed in this work, determining allowable thresholds for error with respect to both phases of the problem, as well as defining acceptable levels of contaminants, could provide guidance for making choices in regards to these criteria. In the following examples, several combinations of these criteria are used in order to provide insight into their relative effects on the boundary control algorithm.

## 7.5 Downstream Protected Region

The following results are for the two-dimensional channel with 6 ports as shown in Fig. 7.1. First, we investigate the boundary control actions for a case defined as follows: the initial, instantaneous source is determined to have the following release conditions from the source inversion phase: $\mathbf{f_{source}} = (0.25 \text{ m}, 0.5 \text{ m}, 0 \text{ s}, 200 \text{ units})$. The region of the domain that is to be protected includes all points encompassed within $5 < x \leq 9$ m and $0.1 < y \leq 0.8$; 280 finite volume cells. With this case set-up, we will compare the relative boundary control performance for CONMIN and OPT++ considering various scenarios.

### 7.5.1 Maximum Number of Iterations

It would be expected that decreasing the maximum number of iterations $N_{max}$ will increase the efficiency of the boundary control algorithm while decreasing the effectiveness of the contaminant mitigation. In order to test this assumption, the control strategies as determined by CONMIN and OPT++ using values of $N_{max} = 3$ and $N_{max} = 10$ are compared, while holding the other MPC parameters at $T = 10$ s, $T_a = 4$ s, and $T_f = 30$ s. First, the velocities and sum of the contaminant concentrations over the protected region for the CONMIN and OPT++ optimizers corresponding to $N_{max} = 3$ iterations are shown in Fig. 7.15 and Fig. 7.16, respectively.

With this imposed maximum of $N_{max}$, the control actions determined by the CONMIN algorithm are initially carried out by port 4. The first optimization period specifies maximum blowing capacity at this point, apparently to deflect as much of

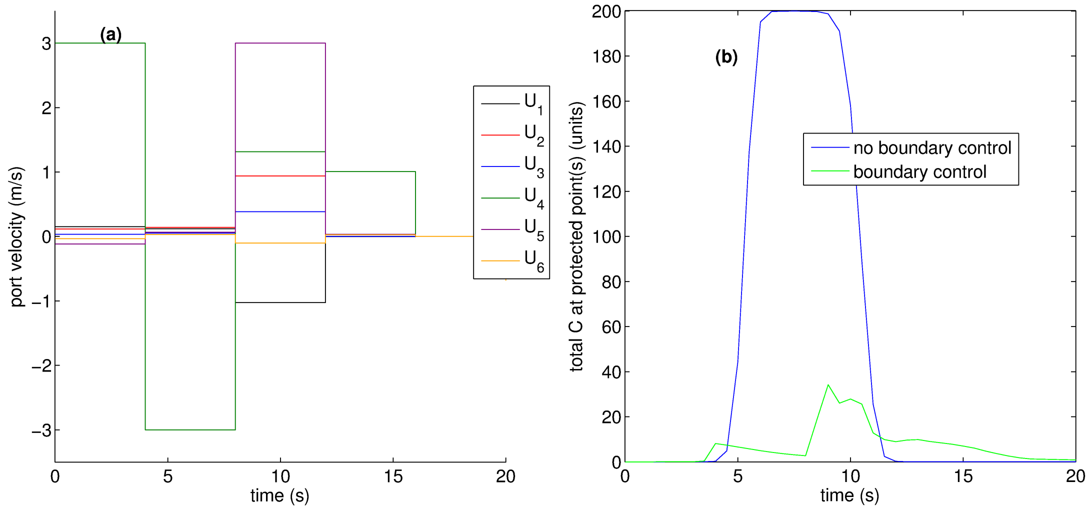


Figure 7.15: Port velocities (a) and time history of the total contaminant mass passing through the protected points with and without the controls implemented (b) for the case of continuous boundary control using the CONMIN optimizer and source release at coordinates (0.25, 0.50).

the plume as possible away from the downstream protected points. This action is also shown in the top subfigure of Fig. 7.17. At $t = 4$ s, the direction of the velocity at this port abruptly switches, instigating suction that serves to remove the majority of the plume. At the third optimization period beginning at $t = 8$ s, all of the ports are operating at some capacity, with suction from ports 2,3, and 6 and blowing from ports 1, 4 and 5. These control actions are hard to explain, as the blowing in particular from ports 1 and 4 acts to effectively usher the remaining contaminant towards the protected region and leading to an increase in the mass of contaminant over this region, seen in Fig. 7.15(b).

The control actions dictated by the OPT++ algorithm are superior to CONMIN's in this case, leading to complete protection of the designated area from the contaminant plume (Fig. 7.16(b)). The main control actions in this case are carried out by ports 1 and 4, which both are initiated at maximum capacity to remove the plume via suction before it reaches the downstream protected region. During the first control phase ($t = 0 - 4$ s), all ports are operating at some level of suction. This results

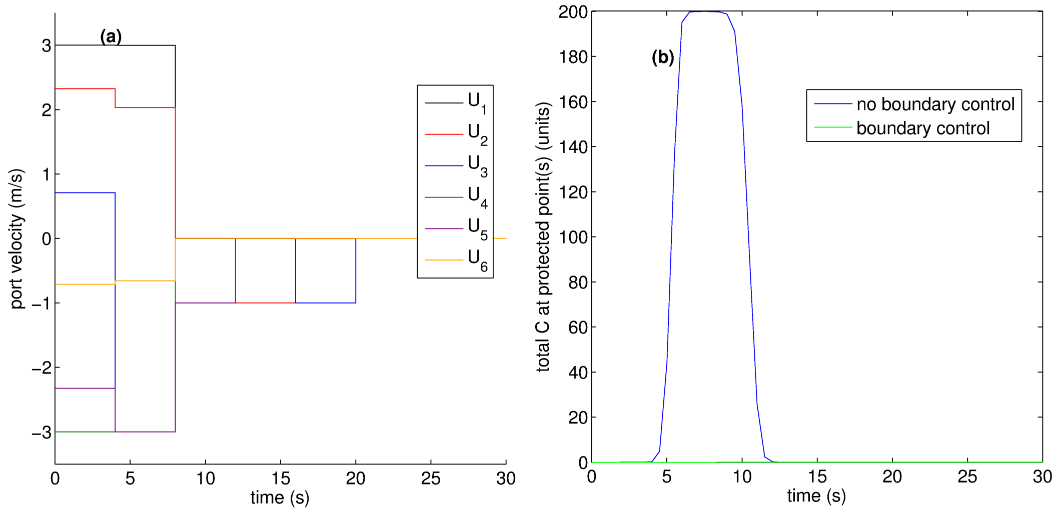


Figure 7.16: Port velocities (a) and time history of the total contaminant mass passing through the protected points with and without the controls implemented (b) for the case of continuous boundary control using the OPT++ optimizer and source release at coordinates (0.25, 0.50).

in a reverse of the flow in the channel, allowing the upstream ports more time to effectively mitigate the plume (top subfigure of Fig. 7.18). The second control phase ($t = 4 - 8$ s), the main control actions do not change much except for port 3, which switches from suction to blowing. Because of the small $\delta$ of $10^{-5}$ that is used in this example, the optimization continues even after the plume has largely been removed from the domain, due to the minute amounts of residual mass that are simulated at the protected points.

These simulations were repeated using $N_{max} = 10$. Interestingly, the increased allowable number of iterations did not change the results for either optimizer. This finding would be expected in the OPT++ case, where the controls were already effectively using the smaller $N_{max}$. However it is surprising that the increased number of allowable iterations did not improve the effectiveness of the CONMIN determined controls. Seeing that the increase in $N_{max}$ does not change the results and that the choice of $N_{max} = 3$ is adequate, at least in the OPT++ case, to control the plume, it is worth investigating the possibility of imposing a lower limit on the number of

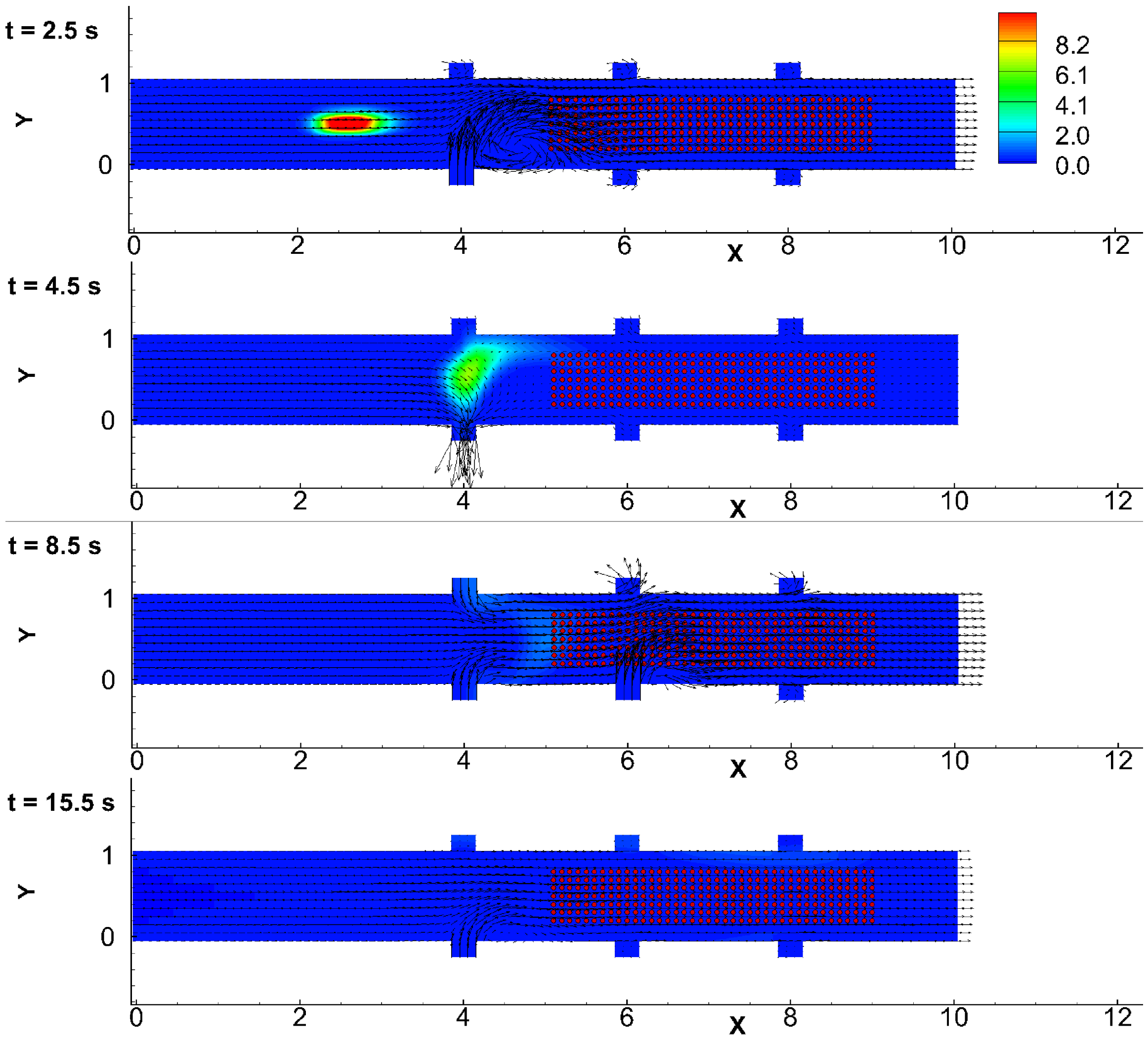


Figure 7.17: Selected snapshots of the plume mitigation via boundary ports corresponding to the velocities shown in Fig. 7.15(a).

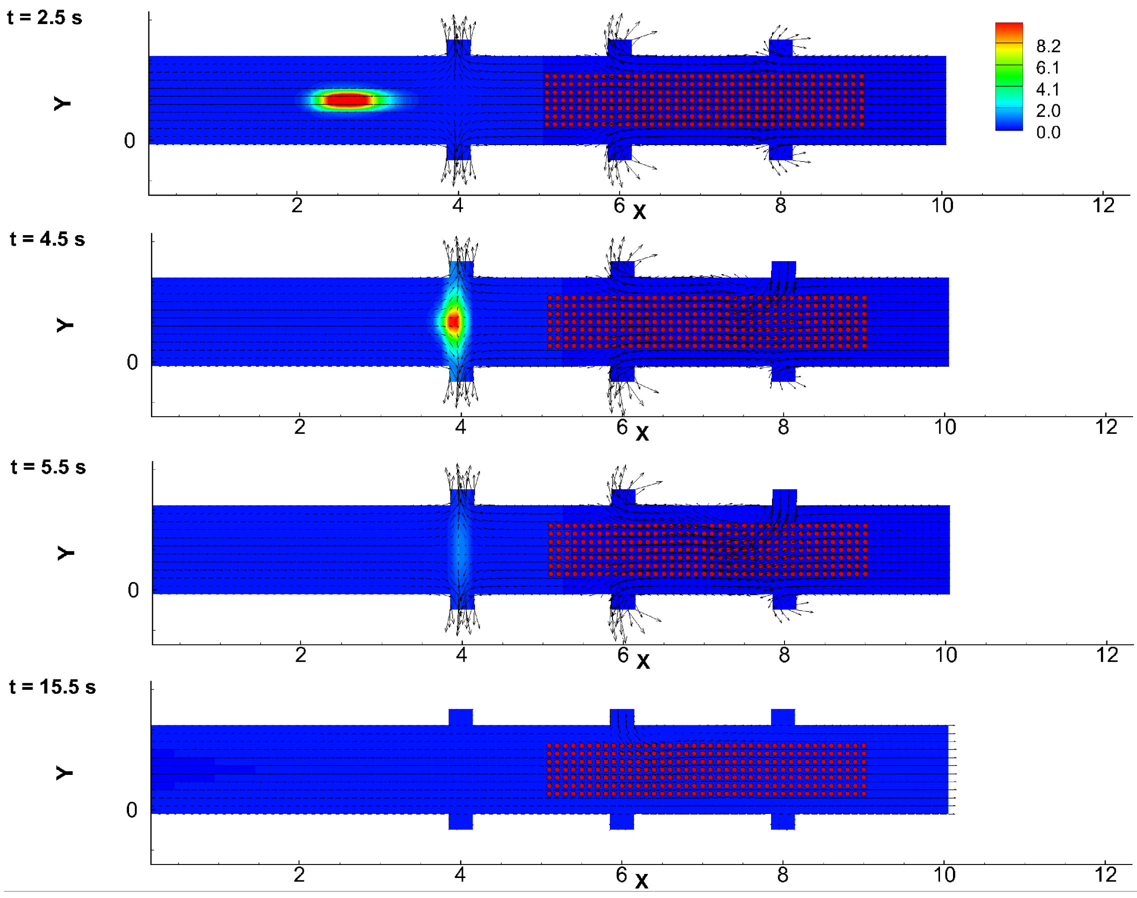


Figure 7.18: Selected snapshots of the plume mitigation via boundary ports corresponding to the velocities shown in Fig. 7.16(a).

maximum iterations. Thus, the CONMIN and OPT++ algorithm were both tested using $N_{max} = 2$ for this case as well. Again, there was no change in the control strategies determined by either optimization method. The only impact of the three tested values of $N_{max}$ on the boundary control process was in respect to the run-times of the CONMIN algorithm. As anticipated, the run-times decreased with fewer number of maximum iterations. However, the OPT++ algorithm did not possess a significant difference in run-times with differing values of $N_{max}$. In fact, the run-times were almost exactly the same in every case.

The findings above signify that the OPT++ algorithm converges to the optimal control strategy at every control evaluation in the MPC process during the first two iterations, negating the need, in this case at least, for more iterations of the optimizer. The CONMIN algorithm on the other hand appears to continue searching for a better strategy up to the maximum allowable number of iterations, however the optimal control strategy is still found in one of the initial first two iterations of the optimizer.

### 7.5.2 Effect of Prediction Horizon

In MPC, the ideal prediction horizon is infinite, which allows for optimal prediction of the control trajectory. However this is not a realistic possibility in modeling, hence a decision must be made as to an appropriate time-frame for $T$. In the case of a finite domain such as is considered here, an ideal $T$ should be long enough to span the time-frame from the release to the complete exit of the plume from the domain of interest. In the absence of unforeseen variations in the flow-field and the assumption that the forward modeling simulations are perfect, this allows for a control strategy to be mapped out for the plume from its release to the point where it is no longer a threat, either because it has been mitigated via port actions or has exited the domain due to advection.

For the previous cases, the choice of $T = 10$ s was shown to be sufficient for complete plume mitigation in many cases. By reducing this finite prediction horizon, we effectively make the MPC algorithm 'short-sighted'. That is, the controls are being determined while only considering the propagation of the plume roughly mid-way through the channel instead of its complete trajectory until exiting the domain. Whether or not this will decrease the effectiveness is dependent on the degree of this short-sightedness, as in general the shorter the finite prediction horizon the less effective the overall control strategy. However, the results are also affected by the choice of the optimizer, as shown in this section, where we now consider the case of $T = 5$ s. Correspondingly, we decrease the advancement time to $T_a = 2$ s. The control strategies and contaminant mitigation histories for this shorter prediction horizon scenario, optimized with CONMIN and OPT++, are shown in Fig. 7.19 and Fig. 7.20, respectively.

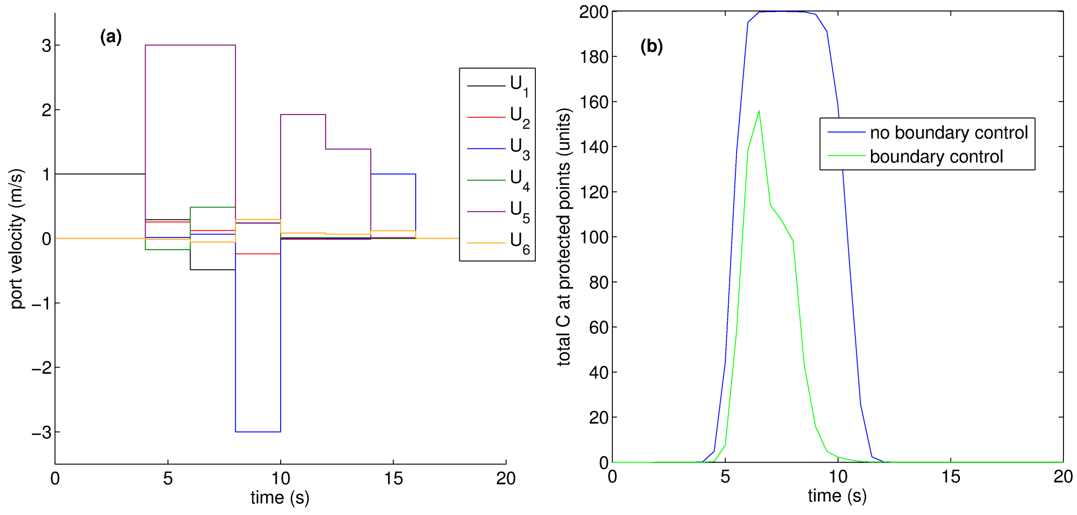


Figure 7.19: Control actions (a) and total contaminant mass at the protected points (b) using a shorter $T$ and CONMIN optimization for the case of the downstream protected region.

The controls in the CONMIN case, which were unable to completely mitigate the plume even in the case of $T = 10$ s, are as expected less effective with the shorter $T$. We can understand where the optimization process errors by further analyzing the control strategy. In Fig. 7.19 it can be seen that port 1 is initiated to provide suction, which acts to remove some of the plume and further, to slow its propagation to the downstream protected points. Thus, over the evaluated time period spanning length $T = 5$ s, the objective function is significantly minimized. However, this control strategy cannot account for the fact that the plume is still largely present in the domain since it is not able to predict its propagation more than 5 s ahead. As the initial control action was only sufficient to impede its trajectory, the plume continues propagating towards the protected region, and the subsequent designated controls are unable to adequately mitigate it from affecting these points.

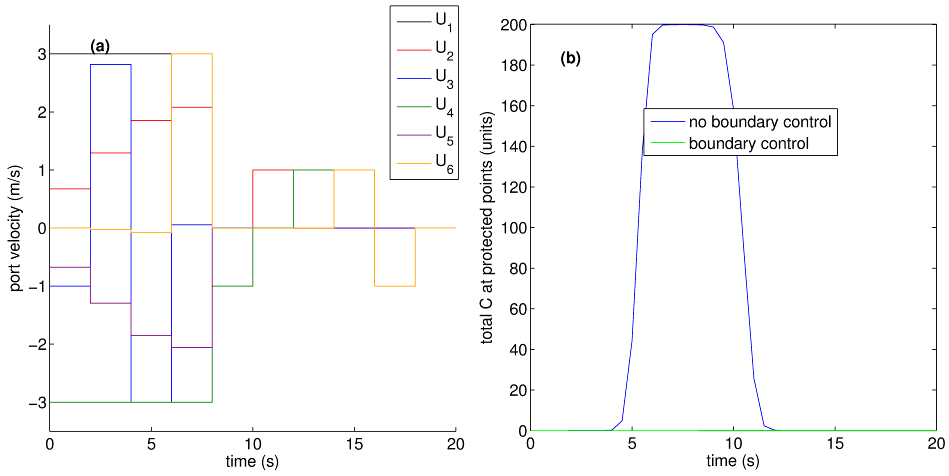


Figure 7.20: Control actions (a) and total contaminant mass at the protected points (b) using a shorter $T$ and OPT++ optimization for the case of the downstream protected region.

The OPT++ algorithm, by contrast, is able to successfully mitigate the plume even with the reduced prediction horizon. Overall, the essential control strategy determined via this optimization method remains the same as that which was determined with $T = 10$ s (Fig. 7.16(b)); the upstream ports 1 and 4 are both turned on to their maximum capacity to suck the plume out of the domain.

A final note on the effect of the length of the prediction horizon is in regards to the run-times. Picking a shorter $T$ may allow the optimization to proceed significantly faster. In these experiments, the CONMIN convergence time was reduced from 37 minutes to 17 by halving the length of $T$, while the OPT++ convergence time was reduced from roughly 57 to 37 minutes. This time-effectiveness is beneficial in the case of the OPT++ method, while the poor performance with CONMIN makes this method in general undesirable for either choice of $T$.

## 7.6 Entire Domain Protection

It is interesting to consider how the MPC will handle the case of every single point in the domain being designated a protected point. In this section, the OPT++ algorithm is applied to the MPC for this scenario, using $N_{max} = 3$ iterations for a plume again found to have initial properties given by $\mathbf{f}_{\mathbf{source}} = (0.25\text{ m}, 0.5\text{ m}, 0\text{ s}, 200\text{ units})$. The prediction horizon and advancement times are defined as $T = 10$ s and $T_a = 4$ s. The results of this boundary control simulation are shown in Fig. 7.21 and Fig. 7.22.

The control actions in this case are similar to the above OPT++ controls for the case of the downstream protected points, in which the two nearest ports are turned on with their maximum suction capacity. However in this case, it can be seen that strong suction is also enacted by the downstream ports, which acts to reverse the direction of the flow and allows the upstream ports more time to remove the plume so as to protect a greater portion of the domain.

### 7.6.1 Impact of the Port Velocity Limits

The suction in the above example is sufficient to remove the plume entirely using the two upstream ports (1 and 4). However, if the maximum velocities on the ports were to be limited to a smaller range, the control strategy will have to be altered to accommodate the lesser suction capacity of the actuators. To investigate this scenario, the entire-domain protection case was repeated reducing the maximum port velocities to $|U_{max}| = 2$ m/s and $|U_{max}| = 1$ m/s and holding all other simulation factors

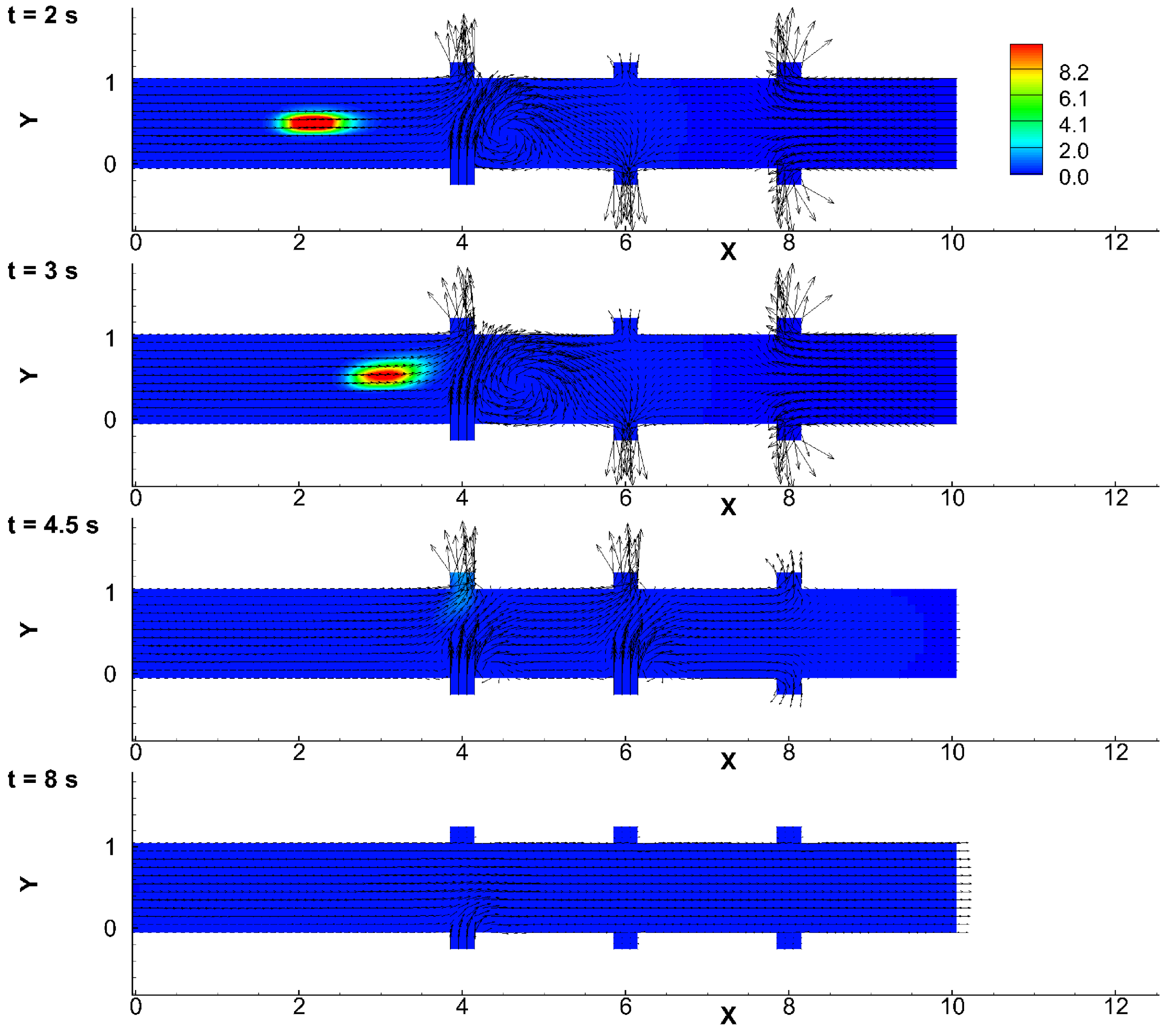


Figure 7.21: Selected snapshots of the plume mitigation via boundary ports for the protection of entire domain.

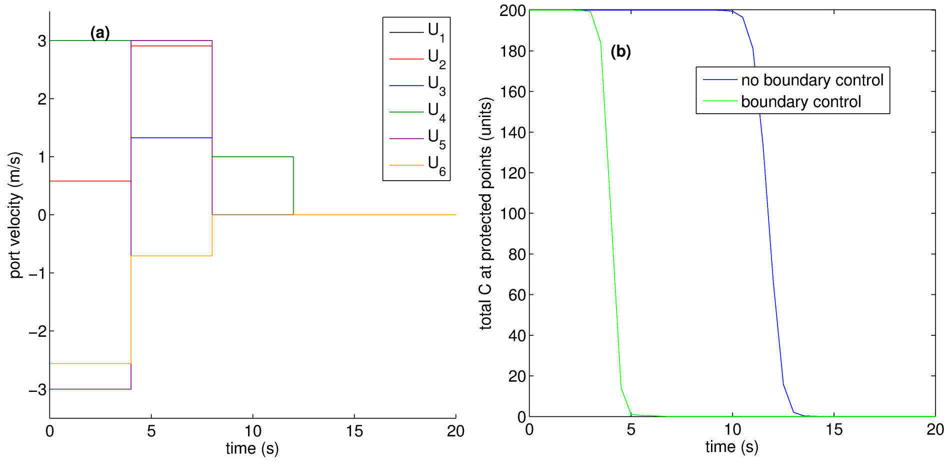


Figure 7.22: Port velocities (a) and time history of affecting contaminant mass at protected points (b) for the case of entire domain protection.

constant. The effectiveness of the plume mitigation for the three maximum velocities is compared in Fig. 7.23. In this scenario, a maximum port velocity of 2 m/s is nearly as effective as 3 m/s. However reducing the velocities to 1 m/s significantly degrades the performance, as well as increases the run-time of the optimization process, as it is more difficult to minimize the objective function with respect to the more stringent parameter constraints. The run-times for maximum velocities of 2 and 3 m/s were roughly 16 and 22 minutes, respectively, while the maximum of 1 m/s required nearly three times as long to complete at 46 minutes.

## 7.7 Comparison of Run-Times and Efficiency

A compilation of relevant data regarding several combinations of effecting boundary control model factors is shown in Table 7.2. This table notes the specific details in each scenario along with the total boundary control algorithm run-time and the percentage reduction in the total affecting contaminant mass as summed over the protected points.

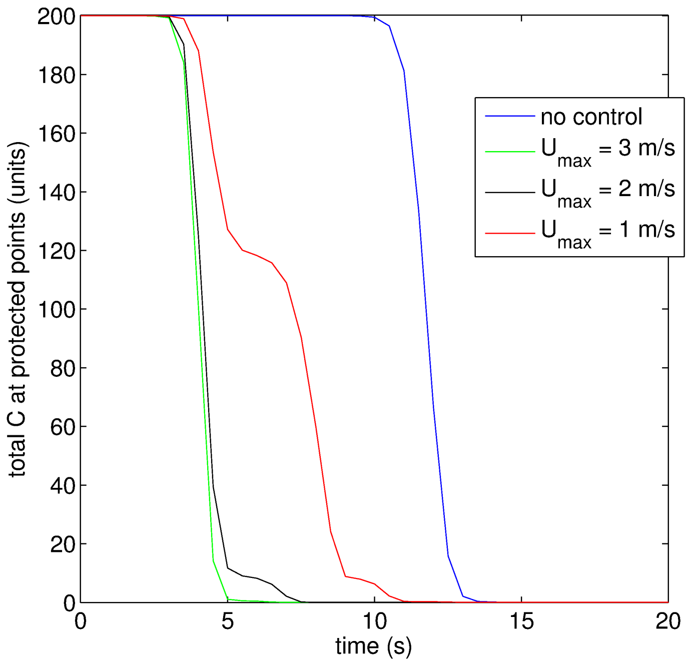


Figure 7.23: Time history of total affecting contaminant mass at protected points for the case of entire domain protection for the case of 3 maximum port velocities, using OPT++ optimization.

Table 7.2: Comparison of run-times for various boundary control scenarios.

| | | | | | **Run-time (mins)** | | **% reduction in total $C(\mathbf{x_p})$** | |
|---|---|---|---|---|---|---|---|---|
| **Protected Area** | $U_{max}$ | $\delta$ | $T$ , $T_a$ **(s)** | $N_{max}$ | **CONMIN** | **OPT++** | **CONMIN** | **OPT++** |
| Scenario 2 | 3 m/s | $10^{-5}$ | 10, 4 | 2 | 28.3 | 57.2 | 85.9 | 100 |
| Scenario 2 | 3 m/s | $10^{-5}$ | 10, 4 | 3 | 36.7 | 57.2 | 85.9 | 100 |
| Scenario 2 | 3 m/s | $10^{-5}$ | 10, 4 | 10 | 54.5 | 57.3 | 85.9 | 99.6 |
| Scenario 2 | 3 m/s | $10^{-5}$ | 5, 2 | 3 | 17.2 | 39 | 63.5 | 100 |
| Scenario 1 | 3 m/s | $10^{-5}$ | 10, 10 | 3 | 12.3 | 14 | 92 | 99.2 |
| Scenario 3 | 3 m/s | $10^{-5}$ | 10, 4 | 3 | 47.7 | 22.2 | 43.3 | 64.6 |
| Scenario 3 | 2 m/s | $10^{-5}$ | 10, 4 | 3 | 41.8 | 15.6 | 34.8 | 62.6 |
| Scenario 3 | 1 m/s | $10^{-5}$ | 10, 4 | 3 | 41.3 | 45.6 | 24.7 | 43.0 |

Scenario 1: 15 protected points (shown in Fig. 7.2); Scenario 2: 280 protected points (shown in Fig. 7.17); Scenario 3: all (1111) points protected points.

In every case of the boundary control simulations the OPT++ algorithm determined a control strategy that was able to better able to protect the designated regions in the domain. While in regards to the scenario with the downstream points deemed protected (Section 7.5, the increased effectiveness of the OPT++ algorithm corresponded to a longer run-times, the opposite situation occurs for the case of all domain points considered protected (Section 7.6). As such, it is hard to draw a conclusion as to the relationship between the boundary control run-times and the choice of the gradient-based optimization method.

The fastest simulations with either optimizer occurred with the continuous port operation approach, in which a single optimization period was required. This is to be expected, as every control evaluation required by the MPC finite prediction horizon approach requires restarting the optimization procedure. As a shorter prediction horizon corresponds to faster forward simulations, it also makes sense that decreasing the length of the finite prediction horizon leads shorter run-times for the boundary control algorithm. As discussed above, the shorter $T$ implemented in OPT++ for the case of downstream domain protection did not decrease the effectiveness of the control strategy while providing the benefit of a shorter run-time. Thus, restricting the finite prediction horizon is one factor that may lead to increased efficiency of the boundary control model.

With respect to the overall run-times, it is clear that the algorithm in its present form needs additional work before it is real-time operational. As shown in Table 7.2, these examples are used a convergence criterion of $J < 10^{-5}$. With this relatively stringent criteria, the optimization continues even after the majority of the contaminant has been deflected from the protected region or advected out of the domain in many of these test cases. An additional possibility for reducing the run-times therefore could be to increase the value of $\delta$, allowing for the optimization to conclude at an earlier point.

## 7.8 Summary and Conclusions

In this section, several examples of boundary control were presented using a number of possible choices for the necessary optimization parameters, protected regions and port operation limits for two gradient-based optimization methods. The formulation of the boundary control problem is an easier one to implement in the optimization framework, however requires additional considerations as to the prediction horizon and advancement times of the Model Predictive Control.

As with the source inversion problem, the OPT++ optimizer is found to perform more accurately and reliably than the CONMIN algorithm, all other factors held the same. Again this appears to be the result of the underlying BFGS quasi-Newton method and its tendency to explore the parameter space to a greater degree than the conjugate-gradient based CONMIN algorithm.

In general, it was found that the use of more evaluation periods and longer prediction horizons leads to increased run-times of the model. The most efficient run-times occurred with a single evaluation period, however this is not a realistic approach nor does it allow for the incorporation of feedback into the boundary control model. Although in the present formulation the measurements and forward model are assumed perfect and thus feedback is not required or implemented, in a realistic scenario the use of feedback in the MPC algorithm would be beneficial to the determination of the optimal control strategy. In any case, the extended run-times resulting from more optimization evaluation periods will need to be weighed against the increased accuracy that this approach would provide. In other words, in order to achieve real-time detection and control, decisions will have to be made as to the minimum possible modeling requirements in order to adequately mitigate the hazardous plume. A detailed risk-analysis, which is beyond the scope of the present work, would be necessary in order to quantify these requirements further.

An interesting finding in this study is that very few iterations of the optimizer at over every evaluation period may be required in order to determine an adequate control strategy for plume mitigation. This is advantageous from the perspective of developing a real-time detection and control system, although further work will need to be done before the boundary control model is operable in real-time. Several possibilities could be implemented to increase the efficiency of the model. For example there are many optimization controls, such as the gradient scaling factor, the number of allowable iterations, and scaling of the parameters, that could be adjusted to decrease the overall required run-times. Further, with additional information on the allowable contaminant levels in the domain, the convergence criteria could be loosened in order to expedite the boundary control process.

# CHAPTER VIII

# Boundary Control Applied to The McNamara Terminal

In this chapter an example of plume mitigation in a realistic domain, the McNamara Terminal at the Detroit Metropolitan Airport, is given. The terminal, shown in its entirety in Fig. 8.1, is approximately 1500 meters in length with a maximum height and width of approximately 12.3 and 30 meters, respectively. The layout of the terminal includes a series of shops lining the interior of the building, above which is a tram that spans the majority of the length of the terminal. This tram runs on a track that is approximately 6 m above the floor and is roughly 6 meters wide, as shown in the left figure of Fig. 8.2.

The exterior side of the terminal contains the boarding waiting areas, and mainly consists of groups of chairs. The exterior wall has a series of windows that extend up to the height of the wall; a little over 6 meters. Near the intersection of the exterior wall and the ceiling are a number of vents for heating and cooling purposes, which can be seen in the right figure in Fig. 8.2. These vents are estimated to be about 0.4 meters in diameter and spaced three meters apart.

A finite volume grid has been generated for a 90 m long subsection of the terminal (Fig. 8.3). This grid is composed of 770,702 finite volume cells. The cells in the interior of the domain are of the hexagonal type that are 0.4 meters on each side. The

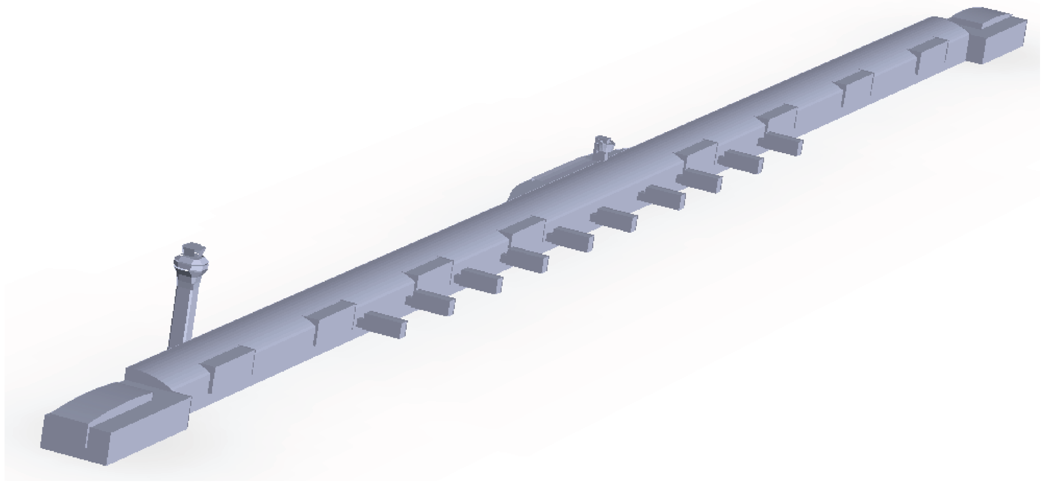


Figure 8.1: McNamara Terminal at the Detroit Metropolitan Airport.

Figure 8.2: Interior views of the McNamara Terminal.

region abutting the exterior wall is discretized into smaller cells in order to adequately resolve the boundary ports, and is made up of tetrahedral elements roughly 0.2 m on each side.

It should be noted that in this example the mesh is generated to be relatively coarse and does not include the small-scale interior features of the terminal. For higher accuracy and flow detail, the mesh would need to include much finer resolution, especially near the boundaries and the vents. Boundary layers would need to be included as were discussed in Chapter II. Also, for a fully representative simulation of the flow, minor features would need to be included in the grid such as the escalator, chairs, et cetera. However, a mesh of this scale with fine enough resolution to accurately capture these features and to completely resolve the boundary layer would require an extremely long simulation time. As the goal of this simulation is to illustrate how the boundary control works in a realistic domain, it is preferable to keep the mesh as coarse and neglect the small-scale features.

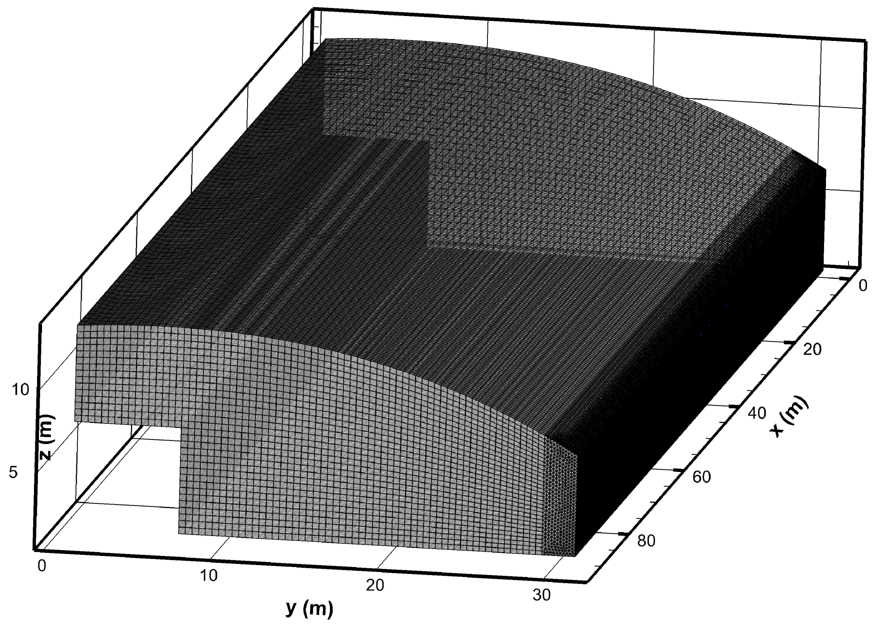


Figure 8.3: Computational grid for a 90 m long section of the McNamara Terminal.

The ambient flow velocity in the terminal is assumed to be 0.3 m/s, with the flow moving in the positive x-direction. This flow speed is within reasonable estimates for an airport terminal (*Piechowski and Rowe* (2007)). A plume is assumed to be detected by the boundary sensors with initial properties $\mathbf{f_{source}}$ =(x = 21 m, y = 30 m, z = 5 m, $q_s$ = 220 units, $t_s$ = 0 s).

The choice of the protected points is arbitrary, and depends on the region(s) of the domain that are deemed to be high risk to the passengers. For example, considering an airport terminal, a hazardous plume would pose the greatest threat to humans when it is at the level of the human occupants. This is roughly between 0-2 m from the floor, assuming, as this study does, that the hazardous plume has neutral buoyancy. Thus for safety, it is reasonable to designate protected regions that are well above the height of the tallest occupant in order to ensure that the control actions will direct the plume away from this region.

In this example, we choose the protected region to be within the bounds $20 < x < 26$, $y > 28$, and $z < 5$. A total of 321 points in this region are designated 'protected', as are shown in Fig. 8.4 along with the initial location of the plume. The protected region in this example is relatively small, based on the assumption that the plume is traveling at the ambient flow velocity of 0.3 m/s, and thus at this relatively slow speed there is adequate time to control the plume before it reaches a region that does not contain protected points. Logistically, the addition of more protected points is a simple matter of changing the input file to the boundary control model to include the other points. However, for a realistic, large-scale domain, stipulating every point in the protected region to be included in the vector $\mathbf{x_p}$ will lead to very large storage requirements for the boundary control simulations. Thus is it preferable to pick points that span the protected region at a density that is high enough accurately guide the minimization.

Another consideration for a large-scale boundary control simulation scenario is the number of actuators to include in the optimization process. This number will greatly

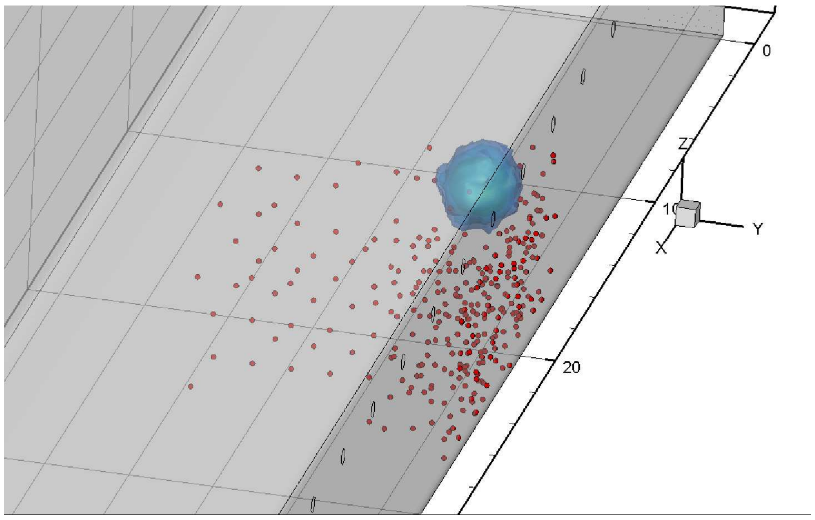


Figure 8.4: Plume location at t=0 and protected points for McNamara simulation.

affect the run-time of the model, as the gradient-based optimization method requires running the forward model $N + 1$ times per optimization evaluation, where $N$ is the length of the unknown parameters $\mathbf{f}$. Even for a restricted number of optimization iterations, the number of evaluations can easily be on the order of 20-50. In the previous chapter's examples for the small-scale, two-dimensional domain, the forward model ran in a matter of seconds, and therefore restricting the number of 'potentially active' ports was not necessary for these examples. However, in this example, where the forward run-time is on the order of hours, it is easy to see how optimization several port velocities could lead to very long simulation times for the boundary control model. It will be necessary to develop a method of a maximum number of optimization ports in the the boundary control model based on the estimated location of the plume and the known ambient velocity at which it is traveling. The low velocity of the airport terminal domain makes it even easier to restrict the potentially active ports, as the plume is moving slow enough to allow for mitigation by one or two ports. For example, the boundary control scenario shown in Fig. 8.5 shows the removal of the plume in this example by steady suction of 20 m/s at the two ports nearest the plume.

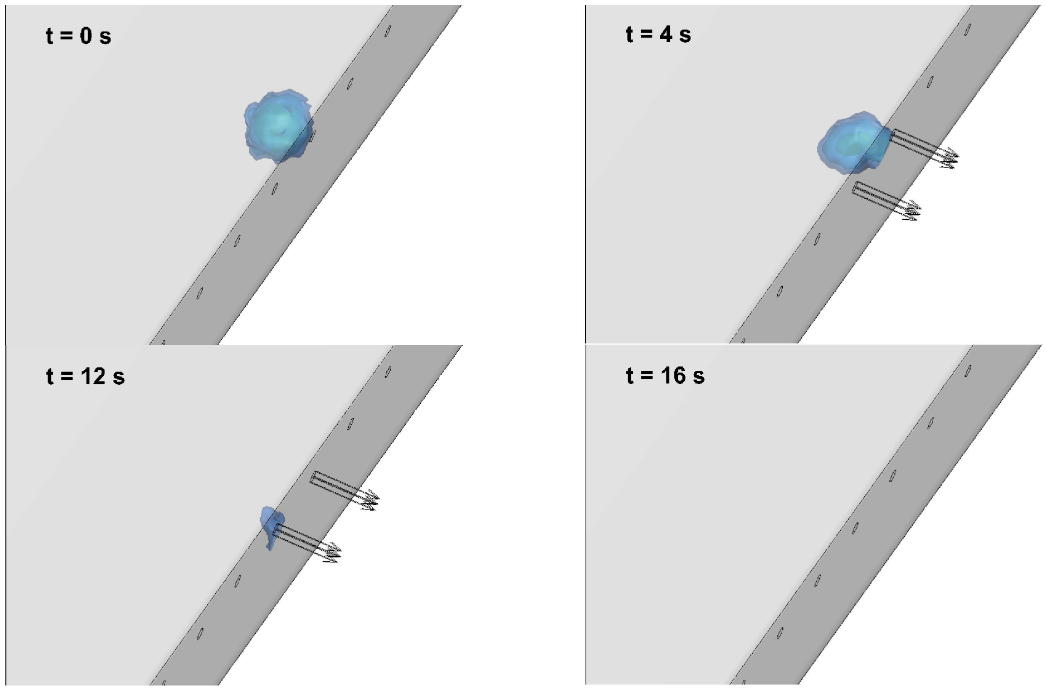


Figure 8.5: Plume mitigation by two vents.

The iso-contours of $C$ in this figure are plotted for three values, 0.5, 0.2 and 0.1. The time-step is restricted in this scenario to $\Delta t = 0.001$ s, which maintains the CFL number around 0.6-0.7 for the majority of the 20 second simulation. As can be seen, the control actions of two vents are sufficient in this case to lift the plume away from the protected region and to remove the majority of the contaminant within the first 12 seconds. However, small quantities of the contaminant still remain in the protected area, as can be seen in Fig. 8.6, which shows a comparison of the total affecting contaminant mass over the protected points with and without mitigation by the boundary ports. The undulations in the contaminant levels are due to the uneven distribution of the protected points in this example. Due to the limits on the ports, which are estimated to be within realistic bounds for actual airport terminal vents, complete mitigation of the plume is not possible.

In light of the fact that few vents are needed to control the plume in this case, another possibility for reducing the optimization time is to reduce the optimization

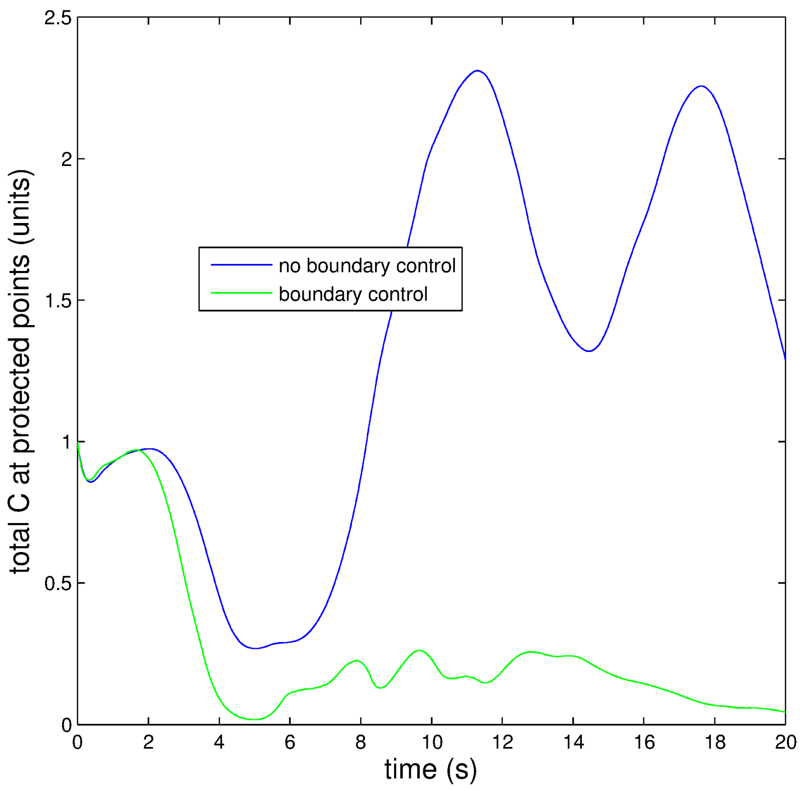


Figure 8.6: Time history of total affecting contaminant mass at protected points for McNaramara Terminal plume mitigation via two vents.

parameters to a single unknown velocity. That is, the optimization can be carried out assuming that the velocities on the active vents have the same magnitude and direction over any optimization period. For example, picking two of the nearest vents to the plume to be 'active' and then optimizing for a single velocity by requiring that the two vents administer the same control actions ($U_1 = U_2$) will require only two forward model runs per optimization evaluation vs. three if the two velocities are optimized individually. It follows that the optimized velocities using this approach will likely not be the most efficient or effective control strategy possible, however given the lengthy run-time required for a large-scale simulation, simplifications of this nature will be necessary in order to work towards a feasible execution time for the boundary control model.

# CHAPTER IX

# Adjoint Methods for Optimization

Thus far, this dissertation has provided many examples of the capabilities and limitations of gradient-based optimization methods as applied to the source inversion and boundary control optimization problems. It has also been shown that traditional gradient-based methods require several runs of the forward model in order to estimate the gradients at every evaluation of the optimizer. While in the small-scale domains tested in the majority of this work the forward model runs in a matter of seconds, more complicated real-world domains would require exponentially longer run-times. And as each additional parameter requires an additional run of the model in order to estimate the gradient with respect to that parameter, the greater the number of unknown parameters the longer the overall optimization process will take. This presents a severe restriction on the gradient-based optimization method's feasibility in a real-time detection and control algorithm.

This section presents an alternative method of obtaining the gradient field for the optimization process which has the potential to greatly reduce the computational requirements of the gradient-based optimization methods. This approach involves using *Adjoint Sensitivity* theory in order to obtain the numerical gradients at all points in the parameters space at each iteration of the optimizer in an efficient manner, as will be described below. This chapter provides an overview of the theory behind adjoint methods, followed by a discussion of the adjoint sensitivity method and its costs and benefits in relation to the automatic control and detection algorithm.

## 9.1 Introduction to Adjoint Methods

Adjoint methods have been used in a wide range of scientific and engineering fields for applications to parameter estimation, design optimization, optimal control, data assimilation, process sensitivity, and experimental design. While this section does not attempt to provide a thorough literature review of the entire body of research that employs adjoint theory, a brief overview of relevant works in a variety of disciplines is provided below.

One of the earliest applications of adjoint methods was for nuclear reactors theory in the 1950s (*Marchuk* (1995)). Adjoint methods have been applied extensively to the field of meteorology for data assimilation and sensitivity analysis of atmospheric models (*Hall* (1986); *Talagrand* (1991); *Courtier et al.* (1993)). Also in the atmospheric sciences, Houweling (1999) used adjoint methods to inversely model the sources and sinks of methane in the atmosphere using data from global monitoring stations.

In the field of groundwater hydrology, adjoint equations have been used for parameter estimation (*Wilson and Metcalfe* (1985); *Sykes et al.* (1985); *Yeh and Sun* (1990)), contaminant remediation (*Merckx* (1991)) and optimal well placement in petroleum engineering (*Zandvliet et al.* (2008); *Zhang et al.* (2010)). Adjoint theory has been used for design optimization in aerospace applications (*Jameson* (1995); *Nadarajah et al.* (2006); *Nadarajah and Tatossian* (2010)) and for fluid animation (*McNamara et al.* (2004)). In the biomechanics field, adjoint equations have been used to study the optimal design of prosthetic bypasses using simplified two-dimensional blood flow models (*Agoshkov et al.* (2006); *Quarteroni and Rozza* (2003); *Rozza* (2005)). Additionally, adjoint theory has been used to determine the optimal shape of fluid-embedded objects (*Okumura and Kawahara* (2000); *Ochiai and Kawahara* (2001);

*Duta et al.* (2002); *Yagi and Kawahara* (2007); *Ishiyama and Kawahara* (2008)). In the geophysical field, adjoint methods have been employed to recover the source parameters related the undersea earthquakes (*Pires and Miranda* (2005)).

As mentioned in the introduction, adjoint methods have been used by other researchers in approaching the source inversion problem (*Akcelik et al.* (2003), *Michalak and Kitanidis* (2004), *Neupauer and Wilson* (1999), *Neupauer et al.* (2000), *Neupauer* (2011)).

The use of adjoint methods in optimal control of fluid flows has received increased attention in recent years. Piasecki and Katopodes (1999) developed a method to inversely solve for stream dispersion coefficients using the adjoint sensitivity methods. Piasecki and Katopodes (1997a, 1997b) applied adjoint theory to the problem of optimal control of hazardous releases in channels. Sanders and Katopodes (2000) used adjoint equations to show the ability to control shallow-water waves and mitigate flooding of populated areas by means of selective flooding. Shichitake (2008) used an adjoint sensitivity method to develop a method of optimally control fluid flows, with applications to optimal control discharge determining the optimal control of a sluice gate and the optimal control discharge for water quality purication. Alvarez-Vàzquez et al. (2009) used adjoint equations in determining an optimal control strategy for injecting clean water into the one-dimensional section of a polluted waterway in order to reduce the contaminant level beneath a given threshold.

## 9.2 Restrictions

While adjoint methods have many benefits that have led to their widespread usage amongst many disciplines, they also have strict modeling requirements that must be heeded. First and foremost they require that the functional be continuous and differentiable over the entire domain. Additionally, the adjoint method is more apt to converge on local minima and require a good starting guess. It is therefore important when choosing the adjoint approach to consider the likely properties of the functional. With an adequate number and distribution of sensors and model grid points it is reasonable to assume that the objective functional can be approximated as a continuous surface with calculable gradients and a distinct minimum at the location of the source and hence meet these requirements.

A further drawback is the need for computational storage. The adjoint method requires a forward run and an adjoint run; the adjoint run derives its equation coefficients via the forward run. Ideally the variables are stored at every time-step during the forward run and accessed by the adjoint calculation during the reverse integration. In practice however, this is impractical for large-scale simulations, and shortcuts must be implemented to allow for the efficient implementation of the adjoint model. This topic is discussed further below.

## 9.3 Advection-Diffusion Examples

In this section, simple examples based on the advection-diffusion equation are derived in order to demonstrate the motivation behind the use of adjoint methods. For a more detailed introduction to the adjoint equations the reader is referred to the references by Marchuk and Giles & Pierce (2000).

First, imagine that we want to put a waste-water treatment plant on a stretch of a river. We want to minimize the effect of the waste-water release on a house located some distance downstream at location $x_T$. For simplicity, we imagine that the river is narrow enough to be considered one-dimensional and that its flow velocity is a constant $u$. What location $x_0$ of the treatment plant would result in the minimal impact to the house? The governing equation for this example is the 1D, steady state advection-diffusion equation, which is a simplified version of Eqn. (2.3):

$$D\frac{\partial^2 C}{\partial x^2} - u\frac{\partial C}{\partial x} + f = 0, \tag{9.1}$$

where $C$ is the concentration of the contaminant release, $D$ is the diffusion coefficient, $u$ is the mean velocity, and $f$ is a known forcing function. The boundary conditions over the domain $x(0, L)$ are $C(0) = C(L) = 0$. We define a cost functional, which describes the parameters we wish to minimize. In this case, the cost functional is simply the contaminant concentration at the distance from the plant to the house, $x_T$:

$$J = \int_0^L r(x)C(x)dx, \tag{9.2}$$

where $r$ is the Dirac-Delta function evaluated at $x_T$:

$$r = \delta(x - x_T). \tag{9.3}$$

The brute force method of solving this problem, known as the Direct Sensitivity Method (DSM), would be to solve the governing equation at each point along the river and then integrate the cost functional. The alternate approach is to use the adjoint equation, which is derived as follows:

We define a continuous and twice differentiable function, $C^*$. This function is multiplied by the forward equation and then integrated by parts:

$$\int_0^L C^* \left( -u\frac{\partial C}{\partial x} + D\frac{\partial^2 C}{\partial x^2} + f \right) dx = 0, \tag{9.4}$$

$$-u\int_0^L C^* \frac{\partial C}{\partial x} dx + \int_0^L \left( C^* D \frac{\partial^2 C}{\partial x^2} \right) dx + \int_0^L C^* f dx = 0 \tag{9.5}$$

Integrating once more and applying the boundary conditions, we get:

$$\int_0^L \left( DC\frac{\partial^2 C^*}{\partial x^2} dx + uC\frac{\partial C^*}{\partial x} + C^* f \right) dx = 0 \tag{9.6}$$

Since this equation is equal to zero, it can be added to the adjoint function and rearranged to give:

$$J = \int_0^L C \left( r + u\frac{\partial C^*}{\partial x} + D\frac{\partial^2 C^*}{\partial x^2} \right) dx + \int_0^L C^* f dx \tag{9.7}$$

The quantity in the first term is known as the *adjoint operator*. Given that

$$D\frac{\partial^2 C^*}{\partial x^2} + u\frac{\partial C^*}{\partial x} + r = 0, \tag{9.8}$$

the objective function can also be represented as

$$J = \int_0^L f(x)C^*(x)dx. \tag{9.9}$$

The ability to represent $J$ by equation (9.2) and (9.9) is known as the *Lagrange duality principle* and has powerful implications. With this relationship, in order to find the optimal location to build the waste water treatment plant the objective function can be solved directly after solving the adjoint equation only one time. Although the results for this simple example are trivial (it is clear that the optimal location of the plant should be as far as possible from the house) this example illustrates the significant savings in computations for an optimization problem.

## 9.4 Unsteady Advection-Diffusion Examples

We define an adjoint operator, $C^*$, multiply (9.11) by this operator, and then integrate over space and time:

$$\int_0^T \int_0^L C^* \left( \frac{\partial C}{\partial t} - D\frac{\partial^2 C}{\partial x^2} \right) dxdt \tag{9.10}$$

In this section numerical examples of the unsteady advection-diffusion equation are computed in OpenFOAM for both the direct and adjoint methods in order to better illustrate the advantages and accuracy of the adjoint method. The unsteady advection-diffusion equation is given as

$$\frac{\partial C}{\partial t} + u\frac{\partial C}{\partial x} - \frac{\partial^2 C}{\partial x^2} = f(x,t), \tag{9.11}$$

with boundary conditions $C(0) = C(L) = 0$. For the first example we consider a one-dimensional problem that spans the x-axis. Imagine we have a target point $x_T$ at which we are interested in measuring the concentration of a contaminant which has been released at a given source location $x_S$ and time $t_T$. This quantity of interest can be expressed as a functional with the form

$$J = \int_0^T \int_0^L r(x,t)dxdt, \tag{9.12}$$

where $f(x,t) = C\delta(x - x_S)\delta(t - t_T)$ and $r(x,t) = C\delta(x - x_T)\delta(t - t_T)$. To obtain this functional by the direct method (9.11) is solved and then (9.12) evaluated sequentially $N$ times, once for each possible source location. The resulting functional is a vector of $C$ values at $x_T$ and $t_T$ that result from the pulse delivered at each possible $x_S$. The alternative is to solve the adjoint transport equation derived using the method outlined above for the 1-D diffusion equation. The adjoint unsteady advection-diffusion equation is given as

$$-\frac{\partial C^*}{\partial t} - u\frac{\partial C^*}{\partial x} - \frac{\partial^2 C^*}{\partial x^2} = r(x,t) \tag{9.13}$$

with the dual functional given by

$$J = \int_0^T \int_0^L f(x,t)C^*(x,t)dxdt \tag{9.14}$$

A grid composed of 201 cells with a $\Delta x$ of 0.1 m is used to compare the direct and adjoint functionals for this case. A unit contaminant pulse is released at $t = 0.1s$. The diffusion coefficient is $D = 10^{-2}\text{m}^2/\text{s}$ and $t_T = 2s$ For the first case we test diffusion only. The direct functional is obtained by running the OpenFOAM model 201 times and evaluating (9.12). The results are compared to the functional (9.14) evaluated after running the model once solving for $C^*$. Figure 9.1 demonstrates the perfect match between the two functionals. For the case of diffusion only, the maximum $J$ occurs at the target location as would be expected. The second example includes a steady left to right velocity of 1 m/s. A comparison of the results is shown in Figure 9.2, again illustrating the ability of the adjoint model to match the direct computational results. In this case the maximum $J$ will occur 2 m upstream of $x_T$.

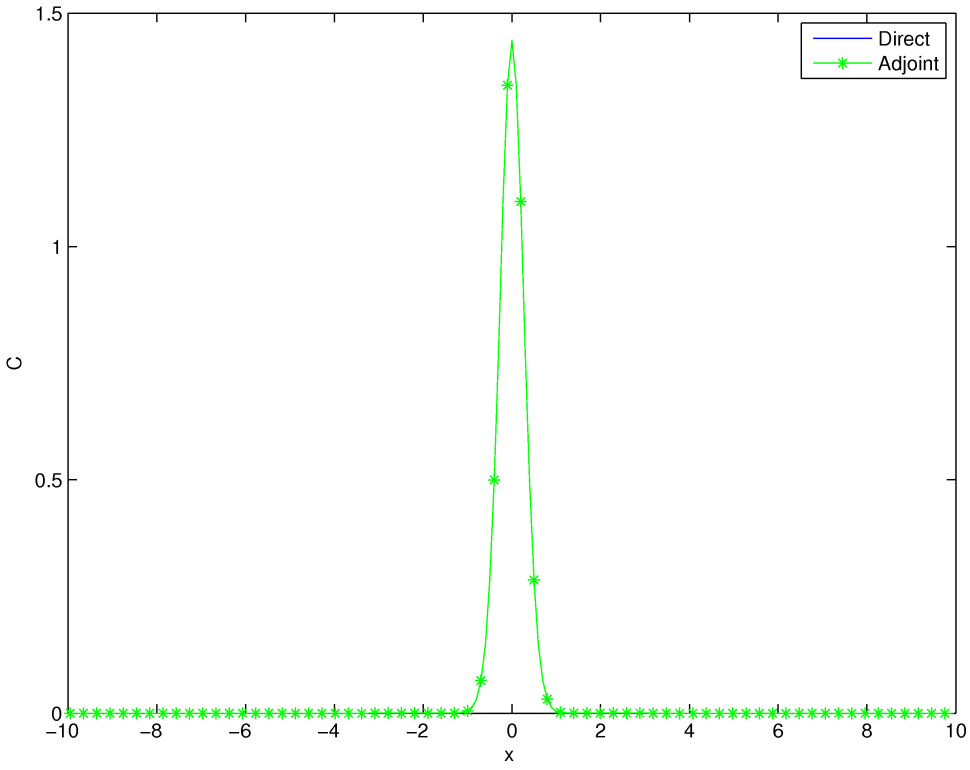


Figure 9.1: Comparison of the functionals obtained by the direct and adjoint methods for pure 1-D diffusion.

This method is very easily extended to two-dimensions, in which the functional takes the form of a surface as shown in Figures 9.3 and 9.4. Finally we extend the method to three-dimensions, where the advantage of using the adjoint method

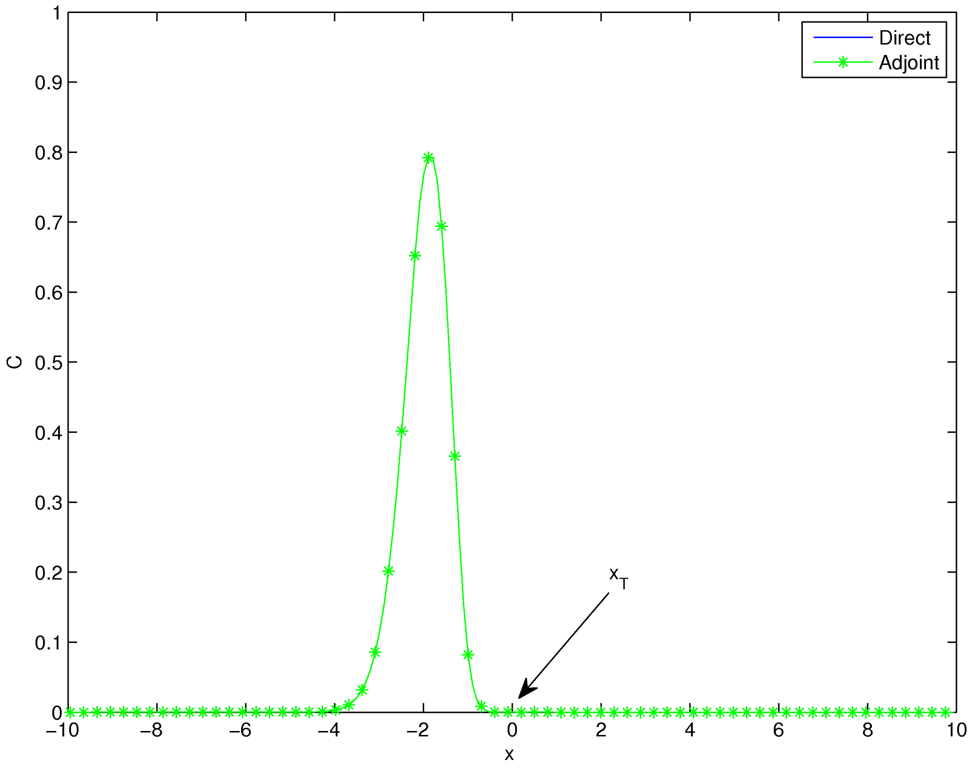


Figure 9.2: Comparison of the functionals obtained by the direct and adjoint methods for 1-D advection-diffusion.

becomes very clear. The direct method requires 15,545 runs of the forward model followed by 15,625 evaluations of (9.12) for this relatively small domain of a $5 \times 5m$ box. As shown in Figure 9.5 the adjoint method is able to match the direct results with a single run of the forward model and evaluation of (9.14). Again, the results show the perfect match between the direct and adjoint methods. The time savings for even a simplistic case such as this are on the order of several hours when running the model on a single processor.

## 9.5 The Adjoint Sensitivity Method

The above introduction into adjoint equations illustrated how taking the adjoint of a forward operator effectively solves the system in backwards time, allowing for a single adjoint run to retrieve the entire objective function field. For the problem of

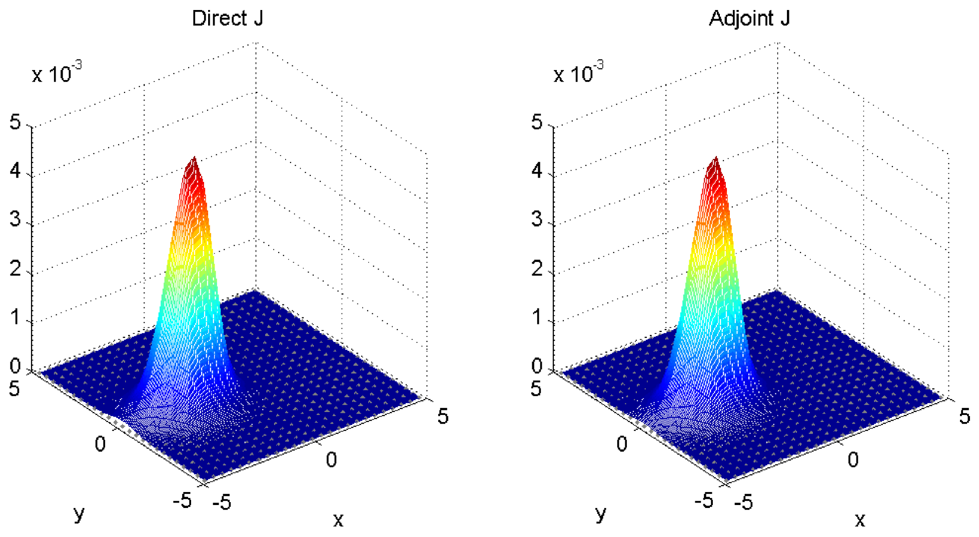


Figure 9.3: Surfaces depicting $J$ obtained by the direct and adjoint methods for the case of 2-D advection-diffusion.

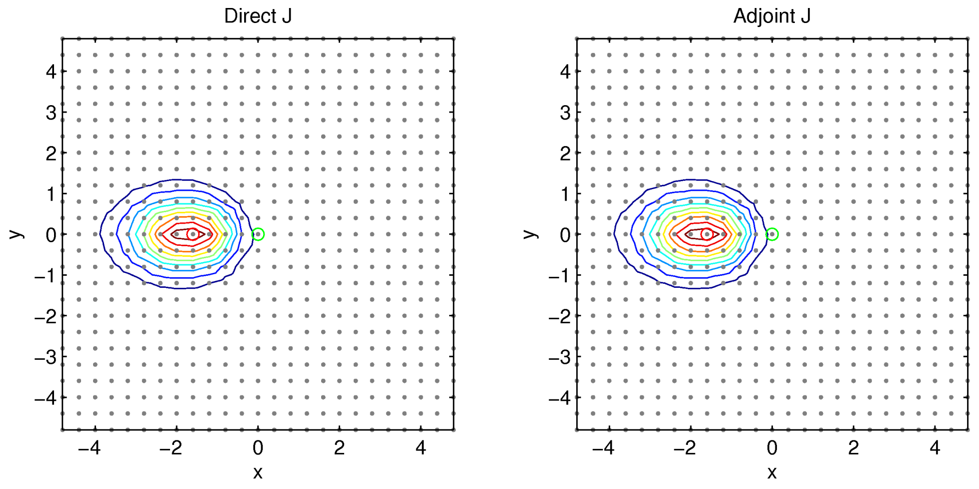


Figure 9.4: Contours of $J$ obtained by the direct and adjoint methods for the case of 2-D advection-diffusion.

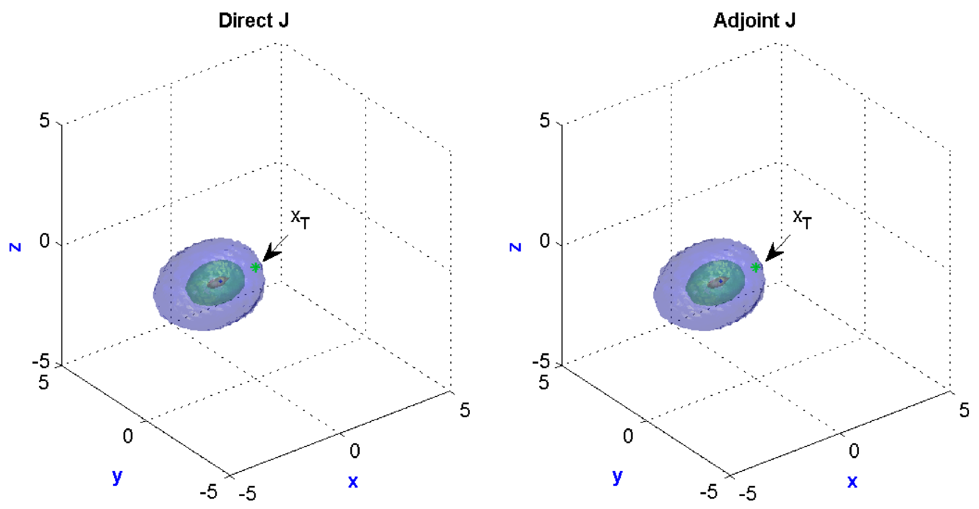


Figure 9.5: Isosurfaces of $J$ obtained by the direct and adjoint methods for the case of 3-D advection-diffusion. The maximum $C$ is shown by the blue dot in each figure.

flow control it is not the objective function for which we seek information, but rather the sensitivities of the objective function. The sensitivities define how the objective function will respond to variations in the flow conditions throughout the domain, thus knowledge of the sensitivities of the system to changes in flow parameters allow for the assessment of the effectiveness of a control action on the cost functional, in the present case, how effective a control action will be to reduce the contaminant concentration at the specified points. The identification of the sensitivities thus allows for a control to be specified which will result in the minimization of the objective function.

The computational savings illustrated by the previous examples can be extended to the case of calculating sensitivities. Similar to the problem of optimization of the cost functional, direct methods are frequently employed to determine sensitivities. In this case there are two direct methods: the influence-coefficient method, which involves perturbing each of the parameters in turn, and the sensitivity-equation method, which requires evaluating the derivatives of each variable with respect to every independent parameter in the governing equations. The adjoint method allows for the sensitivities of the model with respect to any number of parameters to be determined by a single simulation of the adjoint model. For problems in which the number of pertinent parameters is much greater than the number of model outputs of interest, the ASM will generally be more efficient than direct sensitivity methods. For example, a one-dimensional problem discretized into N nodes will have N sensitivities to be determined at each time-step. By the DSM, the methodology to finding these sensitivities involves perturbing the variables at each of the N points and then solving the governing equations to determine the effect of the perturbation on the objective function. For a high resolution and/or large domain, one can imagine the extreme number of calculations required by a direct approach at every time-step. By contrast, the ASM approach only requires solving the adjoint equation once before evaluating the sensitivities.

Taking the variational approach we are able to solve for the *sensitivities* of the adjoint problem for our purpose of solving the inverse contaminant problem described in Chapter III. It is no longer the functional that we are after but the gradient of the functional with respect to the flow parameters, in the inverse case this functional is the sum of squares. To illustrate how this method is applied to the source inversion problem, consider the unregularized, continuous release version of the objective function presented in Chapter III:

$$J(\mathbf{f}) = \frac{1}{2}\sum_{m=1}^{N_s}\int_{0}^{T}\int_{\Omega}\left[\delta(\mathbf{x}-\mathbf{x}_m)(C-\hat{C})^2\right]d\Omega dt \tag{9.15}$$

The forward system of equations this objective function is subject to is given by:

$$\frac{\partial C}{\partial t} + \nabla\cdot(\mathbf{U}C) - \nabla^2(\Gamma_e C) - f = 0 \text{ in } \Omega, \tag{9.16}$$

$$\nabla C\cdot\vec{n} = 0 \text{ on the boundary,} \tag{9.17}$$

$$C(t_0, x) = 0 \text{ in } \Omega. \tag{9.18}$$

The corresponding adjoint system is given by:

$$\frac{-\partial C^*}{\partial t} - \nabla \cdot (\mathbf{U}C^*) - \nabla^2(\Gamma_e C^*) = r(t) \text{ in } \Omega, \tag{9.19}$$

$$(\nu \nabla C^* + C^* \mathbf{U}) \cdot \vec{n} = 0 \text{ on the boundary,} \tag{9.20}$$

$$C^*(t_f, x) = 0 \text{ in } \Omega, \tag{9.21}$$

where $\nabla_f J = -\int_t C * dt$ and

$$r(t) = \sum_{m=1}^{N_s} \int_0^T \int_\Omega \left[ \delta(\mathbf{x} - \mathbf{x}_m)(\hat{C} - C)^2 \right] d\Omega. \tag{9.22}$$

With this system of equations, the gradients with respect to the parameters in **f** are retrieved after a forward calculation of Eqn. (9.16) and a backwards-in-time calculation of Eqn. (9.19). Since as in the examples before, the basic equation require very little alteration in order to switch from the forward to adjoint methods, the same computational model can be used for both, simply switching from one set of equations to the other and integrating either forward or backwards in time. However, because Eqn. (9.19) is dependent on (9.16), the coefficients of the forward run must be saved in order to perform the adjoint calculations. Saving these values at every time-step can become computationally expensive. To mitigate this large storage requirement, it is typical to implement a *checkpointing* scheme. This process requires only storing the state variables at specific intervals over the forward run. During the backward pass, the system state at the final time $T$ is reconstructed by re-running the forward model from the nearest available checkpoint to $T$, and the adjoint state in determined by integrating backward up to the nearest checkpoint. Having calculated the adjoint at this point, the process continues up to the next checkpoint in the backwards time direction and continues until $t = 0$.

With the benefits and restrictions of the ASM in mind, we can now consider its applicability to the automatic detection and control algorithm. It is clear that the savings from employing the ASM in place of the direct method are dependent on the number of unknown parameters (source properties), in the case of source inversion problem, or the number of optimization parameters (port velocities) in the boundary control problem. In the two-dimensional source inversion examples given in Chapter V for example, there were two unknown parameters to be solved for, $x_s$ and $y_S$. In this case, there were three required forward runs for each gradient-based optimization evaluation: one to obtain $J(\mathbf{f})$, and two more to estimate $\partial J/\partial x$ and $\partial J/\partial y$. In this case, the use of the adjoint method would theoretically save a single run of the forward model, only requiring a single forward and backward run, respectively. However, the use of the checkpointing scheme will add approximately one forward run to the overall requirements of the adjoint sensitivity implementation, and thus in this case there is not a measurable benefit to using the ASM over the direct method.

Now consider the case of the inversion problems presented in Chapter VI, where the number of unknown source parameters in $\mathbf{f}$ was at least three. In this case, the benefits of the ASM become apparent. For the larger parameter-space problems, not only does the direct method require $N + 1$ forward runs, where $N$ is the number of unknown parameters, but the time and computational resources to run a single forward one also are much larger. The cost of the ASM however, is still at most that of two forward runs and a single backwards run, regardless of the size of $N$. Moreover, in the boundary control problem, where in a realistic system there could exist a large number of ports and thus $N$ would become very big, the use of the ASM to calculate the gradients with respect to all the port velocities in three runs could greatly reduce the optimization time requirements.

In conclusion, the implementation of the ASM method in the optimization-CFD model for use in the automatic detection and control algorithm has potential to greatly expedite the inversion and optimization processes. This is a further step that could be taken to achieve the goal of a real-time detection and control methodology for hazardous plumes.

# CHAPTER X

# Summary and Conclusions

In summary, this work has developed the numerical tools necessary to study and simulate the automatic control and detection algorithm, for the purpose of mitigating harm caused to the environment and human health due to the threat of air- or water-borne hazardous plumes. The model developed herein is comprised of a linked CFD-optimization framework, based on the open-source CFD software, OpenFOAM, and the DAKOTA Optimization Toolkit distributed by Sandia National Laboratories. The two components have been linked via a loosely-coupled interface, and customization of the two components has been carried out in order to include the scalar transport equation in the CFD software and allow for the evaluation of the numerical gradients via OpenFOAM when this information is necessary to be supplied to the optimization routine.

The linked CFD-optimization model has been used for both the source inversion and boundary control components of the automatic detection and control algorithm. Although there are several modifications that must be made in regards to the input/output files for both of the CFD and optimization components when switching from one phase to the other, the underlying model framework is essentially the same for the source inversion and boundary control components.

It is worth reiterated at this point that the study conducted in this dissertation contains several limitations, most of which could be addressed with minimal modifications in subsequent studies. The predominant limitations are summarized as follows:

- This study assumes a single plume release. Should this assumption not be accurate, the current framework will fail. The main hindrance to implementing more than one source release is with respect to the source inversion problem. The issues of non-uniqueness, which were illustrated in Chapter VI, impose considerable constraints on the source inversion phase in general, and would make the inversion of more than one contaminant plume very unreliable using the current method.

- This study assumes perfect sensors that are able to detect minute concentrations. As already emphasized, an error analysis was not one of the goals of this study, however it is necessary to consider the effects that realistic, noisy, measurements would have on the source inversion and boundary control phases.

- The run-times that are cited in this work are not indicative of the best-case-scenario run-times, and are included mainly for relative comparison purposes. Subsequent work must be performed to optimize and stream-line the existing code, and several options could be carried out to reduce the overall run-times regardless of the optimizer choice and other affecting factors, such as the possibility of using adjoint equations for the gradient-based optimization methods, as was discussed in Chapter IX.

- The tested domains for all chapters in this work are composed of regular grids. Modifications could be made to the model that will allow for the inclusion of irregular grids. This will mainly require alterations to the current method of estimating the gradients in the source inversion phase. Also, the restriction to two-dimensional flows is an additional limitation of this work. As referenced in the first point discussed above, the third spatial dimensional poses a significant challenge in regards to the source inversion problem in terms of its theoretical feasibility. The increased run-time required with the extension to three-dimensions is an additional limitation that would affect both the source inversion and boundary control components.

- Assumption of steady-state ambient flow field. In the absence of this assumption, the source inversion problem would require additional parameters with respect to the velocities in the flow. Even for a uni-directional flow field, this would present an additional complication to the already complicated source inversion process. In many realistic scenarios however, the assumption of a steady-state flow field may be adequate for the automatic detection and control algorithm to operate effectively.

Of the two tested gradient-based approaches, the quasi-Newton based OPT++ algorithm results in more robust and accurate results in nearly all tested cases compared to the conjugate-gradient based CONMIN optimization method. However this algorithm also took on average a longer time to converge, many times more than twice as long. Similarly, other factors such as the maximum number of optimization iterations, spatial/temporal resolution, the method of estimating the gradients, and the scaling and gradient step-size factors, have the potential to greatly alter the efficiency/accuracy balance. As stressed, the aim of this work was primarily to develop the modeling tool to provide a complete numerical framework for simulating and investigating the automatic detection and control algorithm. Further work should be carried out in order to make decisions as to the allowable uncertainty in the source inversion and boundary control phases, both of which are subject to the specific scenario, domain, and relative risk that is posed by a hazardous release. The following sections summarize the conclusions and recommendations from this work specific to each component.

## 10.1 Source Inversion Conclusions and Recommendations

Of the two developed components of the automatic detection and control algorithm, the source inversion phase is the weaker link. Overall, it was found that even with analytical functions that exhibit ideal optimization characteristics, the optimization algorithms applied to discretized parameters are generally not reliable. The issues that arise due to the discritization and approximation of continuous variables from discrete ones becomes more difficult in the case of optimizing a source inversion functional, as was shown in Chapter V. Further, the addition of each unknown parameter led to an increased chance of encountering problems with ill-posedness and non-uniqueness. These are typical problem in source inversion, and are frequently dealt with by reducing the possible choices for the parameters to few possible choices, such as is frequently done in groundwater source inversion problems, where a number of potential source locations is specified a priori. This is not a realistic assumption for the source inversion scenarios that are discussed in this work, and thus there is a conundrum as to how to best perform the source inversion phase. Despite these general concerns, there are some useful conclusions and recommendations that can be made from the source inversion research presented in this work, which are summarized henceforth:

- The analysis of two gradient-based methods as applied to idealized test functions reinforces the statement that an optimization method cannot be considered robust simply because it performs well with a small number of test cases. Indeed, tuning of the algorithms in terms of adjusting the scale values, the gradient calculations and other model parameters is typically necessary to get reliable results from the optimizer. Further, gradient-based methods are highly sensitive to the initial starting guess and to noise in the data and other factors that affect the quality of the functional. These facts severely limit the applicability of this method to the source inversion problem.

- The hybrid optimization approach used herein shows promise, especially when the parameter space is adequately restricted. It is found that even a minimal amount of iterations and a conservative population size for the initial evolutionary algorithm model is sufficient in many cases to recover the source inversion parameters within an acceptable level of error. An additional draw of the hybrid method is that its initial phase could be carried out in parallel, potentially reducing the total computational time.

- Future work in this area will need to focus on increasing the reliability of the source inversion process. Strategic methods of reducing the parameters space by making a good approximate initial guess will be necessary, however in many cases they are not sufficient to guarantee accurate recovery of the defining source characteristics.

- Feasibility of real-time control with respect to the source inversion component will depend on coarseness of mesh, size of domain and number of inverted parameters. Future studies will need to investigate ways of reducing the run-time by using as large as possible of a resolution, and reducing the parameter space as much as possible using the information that is known about the problem.

## 10.2 Boundary Control Conclusions and Recommendations

The boundary control problem overall is a more tractable problem using gradient-based optimization methods than the source inversion problem. Again, the OPT++ optimizer was shown to consistently give better results than CONMIN. The boundary control phase of the algorithm also has potential for improvement via further investigations. As discussed in Chapter VII, additional terms and constraints could be added to the objective function in order to specify addition boundary control requirements.

Further, interesting simulations could be run using a more sophisticated turbulence model in order to investigate the affects of the turbulence created by the control actions on the spreading of the plume. It is logical to conclude that these turbulence effects could interfere and possibly defeat the purpose of the control efforts in some cases. Thus, high resolution simulations of the turbulent effects could provide valuable insight into the requirements of the boundary control problem.

As with the source inversion phase, work will need to be done in order to make the boundary control algorithm operational in real-time, especially when extending the boundary control to three-spatial dimensions. Again, studies will need to be performed to determine maximum allowable errors and allowable coarseness of the grid that will result in an adequate control strategy. Additionally, the length of the prediction horizon is a determining factor in the overall boundary control run-time, and it may be possible to reduce the run-times by decreasing this parameter. Further investigations determining a lower limit for the prediction horizon length as a function of the estimated source properties and the specifics of the flow properties and domain in question could be performed in order to better quantify the requirements for the prediction horizon.

Finally, the current framework for the boundary control algorithm could be amended to include feedback in the form on additional sensor and updated flow-field information, making the algorithm more suitable for real-world, unsteady-state scenarios.

# APPENDIX

# APPENDIX A

# Sample Input Files

## A.1 Sample DAKOTA Input File

```
strategy,
  single_method

method,
        conmin_frcg
        convergence_tolerance 1e-6
        scaling

variables,
          continuous_design = 2
    cdv_initial_point     5.5    0.2
          cdv_lower_bounds    0.1    0.1
          cdv_upper_bounds    5.90   1.9
          cdv_descriptor       'x'    'y'
          scale_types        = 'auto'  'auto'
          scales = 1.0         1.0

interface,
        system
          analysis_driver = 'openfoam.sh'
          parameters_file = 'params.in'
          results_file    = 'results.out'
          work_directory directory_tag
          template_directory = 'templatedir'
           named 'workdir' file_save  directory_save
          aprepro
          deactivate active_set_vector

responses,
        num_objective_functions = 1
        analytic_gradients
        no_hessians
```

## A.2 OpenFOAM controlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
libs ("myBClib.so");
application     pisoFoam;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         10;
deltaT          0.04;
writeControl    runTime;
writeInterval   1;
purgeWrite      0;
writeFormat     ascii;
writePrecision  16;
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable true;

//  ************************************************************************* //
```

## A.3 OpenFOAM-DAKOTA Interface File

```
#!/bin/bash
# $1 is params.in from Dakota
# $2 is results.out returned to Dakota

# --------------
# Pre-processing
# --------------
cp -r ../testChannel .
cp ../last_cell.dat testChannel/
cp ../grad.dat testChannel/

# Update OpenFOAM parametric file by reading from template using Dprepro utility
dprepro $1 sourceProperties.template sourceProperties
cp sourceProperties testChannel/constant/

# --------
# ANALYSIS
# --------
cd testChannel
cp ../../script_OF.py .
python script_OF.py

# ---------------
# POST-PROCESSING
# ---------------
# extract function value from the simulation output
mv output.dat $2
mv $2 ../

cd  ../../
```

## A.4 OpenFOAM Boundary Condition File

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0 0 0);
boundaryField
{
    wall{
        type            fixedValue;
        value           uniform (0 0 0);
    }
    inlet
                {
        type            parabolicVelocity;
                                n                 (1 0 0);
                                y                  (0 1 0);
        maxValue        2;
        value           uniform (2 0 0);
    }
    outlet
                {
        type            zeroGradient;
    }
    frontAndBack
                {
        type            empty;
    }
}
// ************************************************************************* //
```

# BIBLIOGRAPHY

# BIBLIOGRAPHY

Adams, B. M., K. R. Dabley, M. S. Eldred, L. P. Swiler, W. Bohnhoff, J. P. Eddy, and D. M. Vigil (2009), Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantifcation, and sensitivity analysis version 5.2 user's manual.

Addepalli, B., K. Sikorski, E. R. Pardyjak, and M. S. Zhdanov (2011), Source characterization of atmospheric releases using stochastic search and regularized gradient optimization, *Inverse Problems in Science and Engineering*, *19*(8), 1097–1124, doi: 10.1080/17415977.2011.589901.

Agoshkov, V., A. Quarteroni, and G. Rozza (2006), A mathematical approach in the design of arterial bypass using unsteady stokes equations, *Journal of Scientific Computing*, *28*(2-3), 139–165.

Akcelik, V., G. Biros, O. Ghattas, K. R. Long, and B. V. Waanders (2003), A variational finite element method for source inversion for convective-diffusive transport, *Finite Elements in Analysis and Design*, *39*(8), 683–705, doi:10.1016/s0168-874x(03)00054-4.

Alapati, S., and Z. J. Kabala (2000), Recovering the release history of a groundwater contaminant using a non-linear least-squares method, *Hydrological Processes*, *14*(6), 1003–1016.

Allen, C. T., G. S. Young, and S. E. Haupt (2007), Improving pollutant source characterization by better estimating wind direction with a genetic algorithm, *Atmospheric Environment*, *41*(11), 2283–2289, doi:10.1016/j.atmosenv.2006.11.007.

Alvarez-Vzquez, L., A. Martnez, M. Vzquez-Mndez, and M. Vilar (2009), An application of optimal control theory to river pollution remediation, *Applied Numerical Mathematics*, *59*(5), 845 – 858, doi: http://dx.doi.org/10.1016/j.apnum.2008.03.027.

Aral, M. M., J. B. Guan, and M. L. Maslia (2001), Identification of contaminant source location and release history in aquifers, *Journal of Hydrologic Engineering*, *6*(3), 225–234, doi:10.1061/(asce)1084-0699(2001)6:3(225).

Atmadja, J., and A. C. Bagtzoglou (2000), Groundwater pollution source identification using the backward beam equation method, *Computational Methods in Water*

*Resources, Vols 1 and 2: Computational Methods for Subsurface Flow and Transport - Computational Methods, Surface Water Systems and Hydrology*, pp. 397–404.

Atmadja, J., and A. C. Bagtzoglou (2001a), Pollution source identification in heterogeneous porous media, *Water Resources Research*, *37*(8), 2113–2125, doi: 10.1029/2001wr000223.

Atmadja, J., and A. C. Bagtzoglou (2001b), State of the art report on mathematical methods for groundwater pollution source identification, *Environmental Forensics*, *2*(3), 205 – 214, doi:http://dx.doi.org/10.1006/enfo.2001.0055.

Ayvaz, M. T. (2010), A linked simulation-optimization model for solving the unknown groundwater pollution source identification problems, *Journal of Contaminant Hydrology*, *117*(1-4), 46–59, doi:10.1016/j.jconhyd.2010.06.004.

Babcock, D., C. Lee, B. Gupta, J. Kim, and R. Goodman (1996), Active drag reduction using neural networks, *International Workshop on Neural Networks for Identification, Control, Robotics, and Signal/Image Processing - Proceedings*, pp. 279–287.

Bagtzoglou, A. C., and J. Atmadja (2003), Marching-jury backward beam equation and quasi-reversibility methods for hydrologic inversion: Application to contaminant plume spatial distribution recovery, *Water Resources Research*, *39*(2), doi: 10.1029/2001wr001021.

Bagtzoglou, A. C., and S. A. Baun (2005), Near real-time atmospheric contamination source identification by an optimization-based inverse method, *Inverse Problems in Science and Engineering*, *13*(3), 241–259, doi:10.1080/10682760412331330163.

Berggren, M. (1998), Numerical solution of a flow-control problem: Vorticity reduction by dynamic boundary action, *SIAM Journal on Scientific Computing*, *19*(3), 829–860, doi:10.1137/S1064827595294678.

Bergmann, M., and L. Cordier (2008), Optimal control of the cylinder wake in the laminar regime by trust-region methods and pod reduced-order models, *J. Comput. Phys.*, *227*(16), 7813–7840, doi:10.1016/j.jcp.2008.04.034.

Berkooz, G., P. Holmes, and J. L. Lumley (1993), The proper orthogonal decomposition in the analysis of turbulent flows, *Annual Review of Fluid Mechanics*, *25*, 539–575.

Bewley, T., and P. Moin (1994), Optimal control of turbulent channel flows.

Bewley, T. R., P. Moin, and R. Temam (2001), Dns-based predictive control of turbulence: an optimal benchmark for feedback algorithms, *Journal of Fluid Mechanics*, *447*, 179–225.

Boggs, P. T., K. R. Long, S. B. Margolis, and P. A. Howard (2006), Rapid source inversion for chemical/biological attacks, part 1: The steady-state case, *Siam Journal on Optimization*, *17*(2), 430–458, doi:10.1137/040603036.

Butera, I., M. G. Tanda, and A. Zanini (2013), Simultaneous identification of the pollutant release history and the source location in groundwater by means of a geostatistical approach, *Stochastic Environmental Research and Risk Assessment*, *27*(5), 1269–1280, doi:10.1007/s00477-012-0662-1.

Cheng, W. P., and Y. Jia (2010), Identification of contaminant point source in surface waters based on backward location probability density function method, *Advances in Water Resources*, *33*(4), 397 – 410, doi: http://dx.doi.org/10.1016/j.advwatres.2010.01.004.

Choi, H., P. Moin, and J. Kim (1994), Active turbulence control for drag reduction in wall-bounded flows, *Journal of Fluid Mechanics*, *262*, 75–110.

Choi, H. C., R. Temam, P. Moin, and J. Kim (1993), Feedback-control for unsteady-flow and its application to the stochastic burgers-equation, *Journal of Fluid Mechanics*, *253*, 509–543.

Chow, F. K., B. Kosovic, and S. Chan (2008), Source inversion for contaminant plume dispersion in urban environments using building-resolving simulations, *Journal of Applied Meteorology and Climatology*, *47*(6), 1553–1572, doi: 10.1175/2007jamc1733.1.

Courant, R., E. Isaacson, and M. Rees (1952), On the solution of nonlinear hyperbolic differential equations by finite differences, *Communications on Pure and Applied Mathematics*, *5*(3), 243–255, doi:10.1002/cpa.3160050303.

Courtier, P., J. Derber, R. Errico, J. F. Louis, and T. Vukicevic (1993), Important literature on the use of adjoint, variational-methods and the kalman filter in meteorology, *Tellus Series a-Dynamic Meteorology and Oceanography*, *45A*(5), 342–357.

Datta, B., D. Chakrabarty, and A. Dhar (2009), Simultaneous identification of unknown groundwater pollution sources and estimation of aquifer parameters, *Journal of Hydrology*, *376*(12), 48–57, doi:http://dx.doi.org/10.1016/j.jhydrol.2009.07.014.

de Villiers, E. (2006), The potential of large eddy simulation for the modeling of wall bounded flows.

Duta, M. C., M. B. Giles, and M. S. Campobasso (2002), The harmonic adjoint approach to unsteady turbomachinery design, *International Journal for Numerical Methods in Fluids*, *40*(3-4), 323–332.

Fletcher, R., and C. M. Reeves (1964), Function minimization by conjugate gradients, *Computer Journal*, *7*(2), doi:10.1093/comjnl/7.2.149.

Giles, M. B., and N. A. Pierce (2000), An introduction to the adjoint approach to design, *Flow Turbulence and Combustion*, *65*(3-4), 393–415.

Gorelick, S. M., B. Evans, and I. Remson (1983), Identifying sources of groundwater pollution - an optimization approach, *Water Resources Research*, *19*(3), 779–790, doi:10.1029/WR019i003p00779.

Graebel, W. P. (2001), *Engineering fluid mechanics*, xv, 676 p. pp., Taylor & Francis, New York.

Grune, L., and J. Pannek (2011), *Nonlinear Model Predictive Control Theory and Algorithms*, Communications and Control Engineering,0178-5354, Springer-Verlag London Limited, London.

Hall, M. C. G. (1986), Application of adjoint sensitivity theory to an atmospheric general-circulation model, *Journal of the Atmospheric Sciences*, *43*(22), 2644–2651.

Hammond, E. P., T. R. Bewley, and P. Moin (1998), Observed mechanisms for turbulence attenuation and enhancement in opposition-controlled wall-bounded flows, *Physics of Fluids*, *10*(9), 2421–2423.

Hart, W. E. (2007), The coliny project.

Houweling, S., T. Kaminski, F. Dentener, J. Lelieveld, and M. Heimann (1999), Inverse modeling of methane sources and sinks using the adjoint of a global transport model, *Journal of Geophysical Research-Atmospheres*, *104*(D21), 26,137–26,160, doi:10.1029/1999jd900428.

Ishiyama, H., and M. Kawahara (2008), Shape optimization of body located in incompressible viscous flow, *International Journal of Computer Mathematics*, *85*(10), 1515–1530.

Issa, R. (1986), Solution of the implicitly discretised fluid flow equations by operator-splitting, *Journal of Computational Physics*, *62*, 40–65, doi:10.1016/0021-9991(86)90099-9.

Jameson, A. (1995), Optimum aerodynamic design using cfd and control theory, *COMPUTATIONAL FLUID*, p. 495.

Jasak, H. (1996), Error analysis and estimation for the finite volume method with applications to fluid flows, Thesis, Imperial College of Science, Technology and Medicine, University of London.

Jha, M., and B. Datta (2013), Three-dimensional groundwater contamination source identification using adaptive simulated annealing, *Journal of Hydrologic Engineering*, *18*(3), 307–317, doi:10.1061/(asce)he.1943-5584.0000624.

Jha, M. K., and B. Datta (2011), Simulated annealing based simulation-optimization approach for identification of unknown contaminant sources in groundwater aquifers, *Desalination and Water Treatment*, *32*(1-3), 79–85, doi:10.5004/dwt.2011.2681.

Khemka, A., C. A. Bouman, and M. R. Bell (2006), Inverse problems in atmospheric dispersion with randomly scattered sensors, *Digital Signal Processing*, *16*(5), 638–651, doi:10.1016/j.dsp.2005.03.002.

Kolmogoroff, A. (1941), The local structure of turbulence in incompressible viscous fluid for very large reynolds numbers, *Comptes Rendus De L Academie Des Sciences De L Urss*, *30*, 301–305.

Kwon, W. H., S. Han, and SpringerLink (2005), *Receding horizon control: model predictive control for state models*, Advanced textbooks in control and signal processing,1439-2232, xiv, 380 p. pp., Springer, Berlin ; London.

Lee, C., J. Kim, D. Babcock, and R. Goodman (1997), Application of neural networks to turbulence control for drag reduction, *Physics of Fluids*, *9*(6), 1740–1747.

Lee, C., J. Kim, and H. Choi (1998), Suboptimal control of turbulent channel flow for drag reduction, *Journal of Fluid Mechanics*, *358*, 245–258.

Leonard, B. P. (1979), Stable and accurate convective modeling procedure based on quadratic upstream interpolation, *Computer Methods in Applied Mechanics and Engineering*, *19*(1), 59–98, doi:10.1016/0045-7825(79)90034-3, times Cited: 2048 2077.

Liu, C. X., and W. P. Ball (1999), Application of inverse methods to contaminant source identification from aquitard diffusion profiles at dover afb, delaware, *Water Resources Research*, *35*(7), 1975–1985, doi:10.1029/1999wr900092.

Liu, X., and Z. Zhai (2007), Inverse modeling methods for indoor airborne pollutant tracking: literature review and fundamentals, *Indoor Air*, *17*(6), doi:10.1111/j.1600-0668.2007.00497.x.

Liu, X., and Z. Zhai (2008), Location identification for indoor instantaneous point contaminant source by probability-based inverse computational fluid dynamics modeling, *Indoor Air*, *18*(1), 2–11, doi:10.1111/j.1600-0668.2007.00499.x.

Liu, X., and Z. J. Zhai (2009), Prompt tracking of indoor airborne contaminant source location with probability-based inverse multi-zone modeling, *Building and Environment*, *44*(6), 1135–1143, doi:10.1016/j.buildenv.2008.08.004.

Long, K. J., S. E. Haupt, G. S. Young, and C. T. Allen (2006), Characterizing contaminant source and meteorological forcing using data assimilation with a genetic algorithm, *86 AM MET SOC ANN M*.

Mahar, P. S., and B. Datta (1997), Optimal monitoring network and groundwater-pollution source identification, *Journal of Water Resources Planning and Management-Asce*, *123*(4), 199–207, doi:10.1061/(asce)0733-9496(1997)123:4(199).

Mahar, P. S., and B. Datta (2000), Identification of pollution sources in transient groundwater systems, *Water Resources Management*, *14*(3), 209–227, doi:10.1023/a:1026527901213.

Mahar, P. S., and B. Datta (2001), Optimal identification of ground-water pollution sources and parameter estimation, *Journal of Water Resources Planning and Management-Asce*, *127*(1), 20–29, doi:10.1061/(asce)0733-9496(2001)127:1(20).

Mahinthakumar, G. K., and M. Sayeed (2006), Reconstructing groundwater source release histories using hybrid optimization approaches, *Environmental Forensics*, *7*(1), 45–54, doi:10.1080/15275920500506774.

Marchuk, G. I. (1995), *Adjoint Equations and Analysis of Complex Systems*, Kluwer Academic Publishers.

McNamara, A., A. Treuille, Z. Popovic, and J. Stam (2004), Fluid control using the adjoint method, *Acm Transactions on Graphics*, *23*(3), 449–456.

Merckx, C. (1991), Optimal pumping strategy for groundwater decontamination, *International Journal of Control*, *53*(4), 889–905.

Meza, J. C., R. A. Oliva, P. D. Hough, and P. J. Williams (2007), Opt++: An object-oriented toolkit for nonlinear optimization, *ACM Trans. Math. Softw.*, *33*(2), doi:10.1145/1236463.1236467.

Michalak, A. M., and P. K. Kitanidis (2004), Estimation of historical groundwater contaminant distribution using the adjoint state method applied to geostatistical inverse modeling, *Water Resources Research*, *40*(8), doi:10.1029/2004wr003214.

Moin, P., and T. Bewley (1994), Feedback Control of Turbulence, *Applied Mechanics Reviews*, *47*, 3–+, doi:10.1115/1.3124438.

More, J. J., B. S. Garbow, and K. E. Hillstrom (1981), Testing unconstrained optimization software, *Acm Transactions on Mathematical Software*, *7*(1), 17–41, doi:10.1145/355934.355936.

Nadarajah, S. K., and C. Tatossian (2010), Multi-objective aerodynamic shape optimization for unsteady viscous flows, *Optimization and Engineering*, *11*(1), 67–106.

Nadarajah, S. K., A. Jameson, and J. Alonso (2006), An adjoint method for the calculation of remote sensitivities in supersonic flow, *International Journal of Computational Fluid Dynamics*, *20*(2), 61–74.

Neupauer, R. M. (2011), Adjoint sensitivity analysis of contaminant concentrations in water distribution systems, *Journal of Engineering Mechanics-Asce*, *137*(1), 31–39.

Neupauer, R. M., and R. H. Lin (2006), Identifying sources of a conservative groundwater contaminant using backward probabilities conditioned on measured concentrations, *Water Resources Research*, *42*(3), doi:10.1029/2005wr004115.

Neupauer, R. M., and J. L. Wilson (1999), Adjoint method for obtaining backward-in-time location and travel time probabilities of a conservative groundwater contaminant, *Water Resources Research*, *35*(11), 3389–3398, doi:10.1029/1999wr900190.

Neupauer, R. M., and J. L. Wilson (2001), Adjoint-derived location and travel time probabilities for a multidimensional groundwater system, *Water Resources Research*, *37*(6), 1657–1668, doi:10.1029/2000wr900388.

Neupauer, R. M., B. Borchers, and J. L. Wilson (2000), Comparison of inverse methods for reconstructing the release history of a groundwater contamination source, *Water Resources Research*, *36*(9), doi:10.1029/2000wr900176.

Nocedal, J., and S. J. Wright (2006), *Numerical optimization*, Springer series in operations research and financial engineering, Springer, New York.

Ochiai, T., and M. Kawahara (2001), Shape optimization problem for incompressible viscous flow based on optimal control theory, *International Symposium on Multi-Phase Flow and Transport Phenomena*, pp. 78–85.

Okumura, H., and M. Kawahara (2000), Shape optimization of body located in incompressible navier-stokes flow based on optimal control theory, *Cmes-Computer Modeling in Engineering & Sciences*, *1*(2), 71–77.

Ott, E., C. Grebogi, and J. A. Yorke (1990), Controlling chaos, *Phys. Rev. Lett.*, *64*(11), 1196–1199, doi:10.1103/PhysRevLett.64.1196.

Piasecki, M., and N. D. Katopodes (1997a), Control of contaminant releases in rivers. 1: Adjoint sensitivity analysis, *Journal of Hydraulic Engineering-Asce*, *123*(6), 486–492, doi:10.1061/(asce)0733-9429(1997)123:6(486).

Piasecki, M., and N. D. Katopodes (1997b), Control of contaminant releases in rivers. 2: Optimal design, *Journal of Hydraulic Engineering-Asce*, *123*(6), 493–503, doi:10.1061/(asce)0733-9429(1997)123:6(493).

Piasecki, M., and N. D. Katopodes (1999), Identification of stream dispersion coefficients by adjoint sensitivity method, *Journal of Hydraulic Engineering-Asce*, *125*(7), doi:10.1061/(asce)0733-9429(1999)125:7(714).

Piechowski, M., and A. Rowe (2007), Building design for hot and humid climates - implications on thermal comfort and energy efficiency, *Building Simulation 2007, Vols 1-3, Proceedings*, pp. 122–126.

Pires, C., and P. M. A. Miranda (2005), Adjoint inversion of the source parameters of near-shore tsunamigenic earthquakes, *Tsunamis: Case Studies and Recent Developments*, *23*.

Quarteroni, A., and G. Rozza (2003), Optimal control and shape optimization of aorto-coronaric bypass anastomoses, *Mathematical Models & Methods in Applied Sciences*, *13*(12), 1801–1823.

Ravindran, S. S. (2000), A reduced-order approach for optimal control of fluids using proper orthogonal decomposition, *International Journal for Numerical Methods in Fluids*, *34*(5), 425–448.

Rebbeck, H., and K. S. Choi (2001), Opposition control of near-wall turbulence with a piston-type actuator, *Physics of Fluids*, *13*(8), 2142–2145, doi:10.1063/1.1381563.

Rozza, G. (2005), On optimization, control and shape design of an arterial bypass, *International Journal for Numerical Methods in Fluids*, *47*(10-11), 1411–1419.

Sanders, B. F., and N. D. Katopodes (2000), Adjoint sensitivity analysis for shallow-water wave control, *Journal of Engineering Mechanics-Asce*, *126*(9), 909–919.

Semeraro, O., S. Bagheri, L. Brandt, and D. S. Henningson (2011), Feedback control of three-dimensional optimal disturbances using reduced-order models, *Journal of Fluid Mechanics*, *677*, 63–102.

Shanno, D. F. (1970), Conditioning of quasi-newton methods for function minimization, *Mathematics of Computation*, *24*(111), 647–656, doi:10.2307/2004840, times Cited: 1011 1032.

Shichitake, M., and M. Kawahara (2008), Optimal control applied to water flow using second order adjoint method, *International Journal of Computational Fluid Dynamics*, *22*(5), 351–365, doi:10.1080/10618560802077814.

Singh, R. M., and B. Datta (2006), Identification of groundwater pollution sources using ga-based linked simulation optimization model, *Journal of Hydrologic Engineering*, *11*(2), 101–109, doi:10.1061/(asce)1084-0699(2006)11:2(101).

Singh, R. M., and B. Datta (2007), Artificial neural network modeling for identification of unknown pollution sources in groundwater with partially missing concentration observation data, *Water Resources Management*, *21*(3), 557–572, doi: 10.1007/s11269-006-9029-z.

Singh, R. M., B. Datta, and A. Jain (2004), Identification of unknown groundwater pollution sources using artificial neural networks, *Journal of Water Resources Planning and Management-Asce*, *130*(6), 506–514, doi:10.1061/(asce)0733-9496(2004)130:6(506).

Sivanandam, S. N., and S. N. Deepa (2008), *Introduction to Genetic Algorithms*, Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg.

Skaggs, T. H., and Z. J. Kabala (1994), Recovering the release history of a groundwater contaminant, *Water Resources Research*, *30*(1), 71–79, doi:10.1029/93wr02656.

Skaggs, T. H., and Z. J. Kabala (1995), Recovering the history of a groundwater contaminant plume - method of quasi-reversibility, *Water Resources Research*, *31*(11), 2669–2673, doi:10.1029/95wr02383.

Skaggs, T. H., and Z. J. Kabala (1998), Limitations in recovering the history of a groundwater contaminant plume, *Journal of Contaminant Hydrology*, *33*(3-4), 347–359.

Snodgrass, M. F., and P. K. Kitanidis (1997), A geostatistical approach to contaminant source identification, *Water Resources Research*, *33*(4), 537–546, doi:10.1029/96wr03753.

Sun, A. Y. (2007), A robust geostatistical approach to contaminant source identification, *Water Resources Research*, *43*(2), doi:10.1029/2006wr005106.

Sykes, J. F., J. L. Wilson, and R. W. Andrews (1985), Sensitivity analysis for steady-state groundwater-flow using adjoint operators, *Water Resources Research*, *21*(3), 359–371.

Talagrand, O. (1991), The use of adjoint equations in numerical modeling of the atmospheric circulation, in *1st Workshop on Automatic Differentiation*, pp. 169–180, Siam.

Tu, J., G. H. Yeoh, C. Liu, and ScienceDirect (2012), *Computational Fluid Dynamics A Practical Approach*, Butterworth-Heinemann [Imprint] Elsevier Science & Technology Books., San Diego.

Vanderplaats, G. (1973), Conmin: A program for constrained function minimization (1973) user's manual, *NASA TMX-62282*.

Vogel, C. R. (2002), *Computational methods for inverse problems*, Frontiers in applied mathematics, xvi, 183 p. pp., Society for Industrial and Applied Mathematics, Philadelphia.

Wagner, B. J. (1992), Simultaneous parameter-estimation and contaminant source characterization for coupled groundwater-flow and contaminant transport modeling, *Journal of Hydrology*, *135*(1-4), 275–303, doi:10.1016/0022-1694(92)90092-a.

Wilson, J. L., and D. E. Metcalfe (1985), Illustration and verification of adjoint sensitivity theory for steady-state groundwater-flow, *Water Resources Research*, *21*(11), 1602–1610.

Woodbury, A., E. Sudicky, T. J. Ulrych, and R. Ludwig (1998), Three-dimensional plume source reconstruction using minimum relative entropy inversion, *Journal of Contaminant Hydrology*, *32*(1-2), 131–158, doi:10.1016/s0169-7722(97)00088-0.

Woodbury, A. D., and T. J. Ulrych (1996), Minimum relative entropy inversion: Theory and application to recovering the release history of a groundwater contaminant, *Water Resources Research*, *32*(9), 2671–2681, doi:10.1029/95wr03818.

Yagi, H., and M. Kawahara (2007), Optimal shape determination of a body located in incompressible viscous fluid flow, *Computer Methods in Applied Mechanics and Engineering*, *196*(49-52), 5084–5091.

Yeh, H.-D., T.-H. Chang, and Y.-C. Lin (2007), Groundwater contaminant source identification by a hybrid heuristic approach, *Water Resources Research*, *43*(9), doi:10.1029/2005wr004731.

Yeh, W. W. G., and N. Z. Sun (1990), Variational sensitivity analysis, data requirements, and parameter-identification in a leaky aquifer system, *Water Resources Research*, *26*(9), 1927–1938.

Zandvliet, M. J., M. Handels, G. M. van Essen, D. R. Brouwer, and J. D. Jansen (2008), Adjoint-based well-placement optimization under production constraints, *Spe Journal*, *13*(4), 392–399.

Zhang, K., G. M. Li, A. C. Reynolds, J. Yao, and L. M. Zhang (2010), Optimal well placement using an adjoint gradient, *Journal of Petroleum Science and Engineering*, *73*(3-4), 220–226.

Zhang, T. F., and Q. Chen (2007), Identification of contaminant sources in enclosed environments by inverse cfd modeling, *Indoor Air*, *17*(3), 167–177, doi: 10.1111/j.1600-0668.2006.00452.x.