Abstract

Kavanagh, Kathleen Rose. Nonsmooth Nonlinearities in Applications From Hydrology. (Under the direction of C.T. Kelley)

This work has two parts; simulation of unsaturated flow and optimization of remediation problems. For the unsaturated flow simulation, we propose an adaptive time stepping scheme based on error control for Richards' equation, a model for flow in unsaturated porous media. The motivation for this work is a ground and surface water simulator being developed by the U.S. Engineering Research Development Center called the **AD**aptive **H**ydrology Model. ADH uses unstructured, adaptive finite elements. ADH advances in time implicitly, solving the nonlinear equations with an inexact—Newton method with a two-level domain decomposition preconditioner. The nonlinearity in Richards' Equation can be non-Lipschitz and nonsmooth. Standard theory for temporal integration may not apply for certain physical parameters. We consider a method for error estimation and control for temporal adaption.

In the optimization section, we investigate a suite of test problems from the literature that are intended for benchmarking purposes and comparison of optimization

algorithms. The objective functions can be nonsmooth, nonconvex, or have several minima that may trap standard gradient based methods. We apply the implicit filtering algorithm to some such problems.

NONSMOOTH NONLINEARITIES IN APPLICATIONS FROM HYDROLOGY

BY

KATHLEEN R. KAVANAGH

A DISSERTATION SUBMITTED TO THE GRADUATE FACULTY OF

NORTH CAROLINA STATE UNIVERSITY

IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

APPLIED MATHEMATICS, COMPUTATIONAL MATHEMATICS CONCENTRATION

RALEIGH, NORTH CAROLINA

JULY 2003

APPROVED BY:

C. T. KELLEY
CHAIR OF ADVISORY COMMITTEE

C. T. MILLER

P. A. GREMAUD

S. E. HOWINGTON

E. L. STITZINGER

Biography

Kathleen Rose Kavanagh was born on October 27, 1973 in New York, NY. She grew up in Plattsburgh, NY and attended high school at Beekmantown Central School, graduating in 1991. She spent two years at Paul Smith's College in the heart of the Adirondack mountains where she originally studied to one day own her own restaurant. However, her interest in Mathematics lead her to obtain a B.S. in Mathematics from SUNY Plattsburgh in 1996. She began graduate school at North Carolina State University in 1998 and obtained her M.S. in Computational Applied Mathematics in 2000 and her Ph.D in Mathematics in 2003. She will begin a tenure track assistant professor position at Clarkson University in Potsdam, NY in the Fall of 2003. She still loves to cook and is pleased to be returning to the North Country.

Acknowledgments

This research was supported by a Department of Education GAANN Computational Science Fellowship, the National Science Foundation, and the Army Research Office. I would like to thank these organizations for their financial support. I would especially like to thank Stacy Howington, Jackie Hallberg, and Charlie Berger at the U.S. Army Corps of Engineers Waterways Experimentation Station in Vicksburg, Mississippi for their guidance and patience as I worked on ADH and for making my visits to Vicksburg, MS most enjoyable. I'd like to thank Casey Miller from UNC-Chapel Hill for his efforts in teaching a mathematician about groundwater engineering, for answering numerous off-the-wall questions, and for all the thought experiments. I'd like to thank Dr. Campbell, Dr. Ipsen, Dr. Gremaud and Dr. Meyer for all the constructive criticism given in the numerical analysis seminars. I'd like to thank Astrid Battermann for all her help while I was beginning to learn about optimization–without "The Astrid Problem" I wouldn't have much to talk about. I owe a huge thanks to my peers for answering many questions related to classes, mathematics, hydrology,computing and life in general; especially Chris Kees, Joerg Gablonsky, Todd Coffey, Lea Jenkins, and my office-mates–Brian Adams, Jill Reese, and last but not least Jordan Massad (DJSUDB). Another huge thank-you goes to Mark Reed at the North Carolina Supercomputing Center for teaching me how to program more efficiently and for helping me get my work running in parallel so I could graduate in finite time.

On a more personal note, I'd like to thank my family–especially my mother, father, and brother, who either directly or indirectly lead me to graduate school and for your constant support and unconditional love. A thank-you goes to my roommates Brian Lewis, Greg Hicks, Bill Nealis, and Pat Zabst for all the dinners at the table, Christmas pictures, oil changes, and making my home-life how should I say it, special. A thank-you to my friends back in NY–the Judges, Kristi, Shannon, Christina, and Bleisse for the constant encouragement and for giving me something to come home to. An immense thank-you goes to Karen Bliss, Karen Yokley, Tracey Westbrook, Vicky Williams, and Jill Reese. There are no words to express how much I appreciate you–you are the best! Thank you Karen Bliss for being on the other end of that phone at any time of the day or night–I will always love you (sung to the tune by Whitney Houston). A massive thank-you goes to Vicky Williams–who has been with me since recruitment weekend, in numerous classes, during qualifying exams and our rigorous 18 hours-a-day study schedule, for preliminary orals, applying for jobs, and during the stress of writing a thesis. How will I survive without you? I want to thank Jennifer Fowler for introducing me to her brother, Reyna and your entire family. I am eternally grateful to Robert Fowler for everything you do to make my life easier,

more fun, and much more fulfilling than I ever imagined it could be. I have to give props to coffee and beer, both of which made grad school bearable.

Lastly, an ∞–sized thank-you goes to my advisor Tim Kelley for being an exemplary role model, building my confidence, telling me when I am not thinking, giving me sound advice, training me to become a professional, sending me to numerous conferences around the world, and introducing me to countless people in the field. Thanks for being patient with me over the last five years–especially these last few weeks. I have nothing but the deepest gratitude for you.

Table of Contents

List of Tables

List of Figures

Chapter 1

Overview

This thesis has two parts. The first part deals with simulation of groundwater flow in unsaturated porous media. We are especially interested in error estimation and control to be used for an adaptive temporal integration scheme. The second part deals with optimization of groundwater design problems involving groundwater flow and contaminant transport. The focus of this work is to investigate the nonsmooth, nonlinearites that arise naturally from the governing equations for groundwater flow and remediation design.

1.1 Simulation of Unsaturated Flow

One commonly used model for unsaturated groundwater flow is Richards' equation (RE). RE is a nonsmooth, nonlinear PDE and the constitutive relations needed to close the system can be non–Lipschitz continuous and nondifferentiable for certain physical parameters. We have implemented an adaptive temporal integration scheme based on error estimation and control in a production code used by the U.S. Army Corps of Engineers. The scheme is based on smooth functions; however performs well despite the nonsmoothness associated with RE in some types of media. We test the adaption by simulating flow through one–dimensional homogeneous media and three–dimensional heterogeneous media. Both tests involve porous media and boundary conditions that make the constitutive laws non–Lipschitz continuous and nondifferentiable at the water table. In Chapters 2 and 3 we discuss the simulation of unsaturated flow and address the issue of nonsmooth nonlinearites.

1.2 Remediation Design

Contaminated groundwater is expensive and difficult to clean up. We rely on accurate mathematical models and numerical software to aid in the remediation process. In the context of contaminated groundwater sites the environmental engineering community requires remediation design that can reduce the amount of a contaminant present at the lowest possible cost. Mathematically, this leads to an optimal control problem. The objective functions and constraints that are typically used depend on the concentration of the contaminant and the flow field either directly or through constraints. Evaluation of the objective function requires the solution to a set of partial differential equations. The solution to the PDE is found with a call to a groundwater simulator. Due to the formulation of the objective function and constraints, the optimization landscapes can be nonsmooth and nonconvex. Because we only have information on function values, not gradients, we advocate using sampling methods for the optimization.

In [28], the authors present an extensive review of optimal remediation design efforts from the literature and address the challenges remaining in this field. These challenges include limitations on existing models, realization of the subsurface, simulators, and optimization techniques. In this work, we consider a set of community problems proposed in [28] designed for benchmarking purposes. This suite of problems can be used to compare optimization techniques for groundwater management and remediation problems. In Chapter 4 we investigate the performance of the implicit filtering algorithm on subset of the community problems.

Chapter 2

Temporal Integration and Richards' Equation

2.1 Introduction

In this chapter, we consider temporal integration of a nonsmooth, nonlinear, partial differential equation called Richards' equation (RE) [37]. RE models flow in unsaturated porous media. Although some analytic solutions exist for RE [42], the solution is usually approximated numerically. From a mathematical standpoint, RE is challenging for numerical integrators due to the formation of sharp wetting fronts that may result from an initial transient or time dependent boundary conditions. An accurate solution for RE requires a particularly small time step while the front is forming. Moreover, the constitutive relations used to close the system can be nonsmooth and not Lipschitz continuous for certain physical parameters. Traditional theory for temporal adaption is based on error estimation and control derived from Taylor series analysis. Taylor series based error estimation assumes that the function at hand is smooth; however with RE some derivatives may not exist for certain realistic problems. In this chapter, we derive a simple method using linear extrapolation for temporal adaption that is based on smooth functions yet experimentally works well for RE. We provide results for one dimensional flow in homogeneous media and three dimensional flow in heterogeneous media. The motivation for this work is a three dimensional ground and surface water simulator being developed by the U.S. Army Corps of Engineers called the **AD**aptive **H**ydrology Model [40]. We give an overview of ADH in §2.2. In §2.3 we develop the model, RE, and in §2.4, §2.5 and §2.6, we describe the discretization, nonlinear and linear solvers, and preconditioners used in ADH. In §2.7 we present the method for error estimation and control used for temporal adaption and give numerical results in §2.8 and §2.9.

2.2 The Adaptive Hydrology Model

ADH solves saturated and unsaturated flow and transport problems, as well as surface water–groundwater interactions. ADH discretizes the continuous problem using three–dimensional, unstructured finite elements that are adaptive in space and implicit temporal integration leading to a large, nonlinear system of algebraic equations that must be solved at each time step. ADH advances in time with backward Euler and a user specified time time step that can grow or shrink based on the performance of the nonlinear solver. The downfalls of the current time stepping method are that the user has no control over the desired accuracy and a small time step may not be computationally efficient for an entire simulation. Our goal is to implement a time adaption method in ADH such that the user can specify the desired temporal accuracy and solve the problem as efficiently as possible. The final version of ADH will regrid in space after each time step thus an attractive approach for temporal error control is to use linear extrapolation. We describe the first-order method we implemented for in §2.7. Although ADH uses low order Galerkin finite elements that are not locally mass conservative, discontinuous Galerkin (DG) methods, which do conserve mass and can provide higher order approximations, are a current area of research for unsaturated flow. The finite element discretization is not the topic of this work however and DG methods have not, to our knowledge, been implemented in a full three-dimensional production code for unsaturated flow. The purpose of this work is that, upon completion, ADH will be a fully spatial and temporal adaptive simulator for saturated and unsaturated flow and transport as well as surface water and groundwater interaction. In this work we focus only on the temporal adaption.

The nonlinear system that arises from the implicit temporal discretization is solved with Newton–Krylov–Schwarz method, which is an inexact–Newton method that uses a Krylov method to approximate the Newton step. Convergence of the Krylov method is accelerated with Schwarz domain–decomposition preconditioners [39]. ADH offers a choice of Jacobi preconditioning, one–level additive Schwarz, two–level additive Schwarz, or a hybrid preconditioner. For the two–level preconditioners, ADH recomputes the coarse mesh component every time step and on every processor. At the time the preconditioners were implemented, it was observed that the construction of the coarse mesh problem could be computationally expensive [21]. Timings were improved by only updating the coarse mesh part of the preconditioner every ten nonlinear iterations. We investigated a time dependent update for the coarse mesh component of the preconditioner that is based on ideas implemented in the Lawrence Livermore National Lab ODE code, PVODE [19], and include numerical results in §2.10.1.

2.3 Model Development

2.3.1 Basic Concepts

In this section we introduce the concepts from groundwater engineering to develop Richards' equation. The basis for developing the model is the statement of mass conservation;

$$\text{change in mass with time} = \text{mass inflow rate - mass outflow rate.} \tag{2.1}$$

Consider (2.1) applied to a volume, V of porous media. The void space not occupied by a solid could be occupied for example, by water, air, or a pollutant. We refer to these different substances as phases, denoted with a subscript α; $\alpha = s$ for the solid phase, $\alpha = a$ for the aqueous phase, $\alpha = g$ for the gas phase, and $\alpha = n$ for the NAPL phase, or nonaqueous phase liquid (pollutant). Moreover, within each phase there may exist different species, for example if several gases are present at once. We denote the different species within a phase with a superscript ι. We define the following quantities; the porosity,

$$\eta = \frac{\text{volume of void space}}{\text{total volume}},$$

volume fraction of the α phase,

$$\theta_\alpha = \frac{\text{volume of } \alpha \text{ phase}}{\text{total volume}},$$

mass fraction of species ι in the α phase ,

$$\omega_\alpha^\iota = \frac{\text{mass of species } \iota}{\text{mass of } \alpha \text{ phase}},$$

a nonadvective flux term for species ι in the α phase, for example representing diffusion, $\mathbf{j}_\alpha^\iota$, density of the α phase, ρ_α, and velocity of the α phase, $\mathbf{v}_\alpha$. Let $\mathcal{R}_\alpha^\iota$ denote reactions of species ι in the α phase, $\mathcal{I}_\alpha^\iota$ denote mass transfer for species ι to the α

phase, and $\mathcal{S}_\alpha^\iota$ denote external sources for species ι in the α phase. Summing over all the species yields the following identities;

$$\sum_\iota \omega_\alpha^\iota = 1,$$

$$\sum_\iota \mathcal{R}_\alpha^\iota = 0,$$

$$\sum_\iota \mathcal{S}_\alpha^\iota = \mathcal{S}_\alpha,$$

$$\sum_\iota \mathcal{I}_\alpha^\iota = \mathcal{I}_\alpha,$$

$$\sum_\iota \mathbf{j}_\alpha^\iota = 0.$$

2.3.2 Formulation of RE

The starting point for deriving Richards' equation is the species mass balance statement. The system is closed by modifying Darcy's law, which describes saturated flow. Let M_α^ι denote the mass of species ι per volume of the α phase. The species mass balance statement is

$$\begin{aligned} \text{change in} M_\alpha^\iota \text{ over time} \;\; = \;\; & M_\alpha^\iota \text{ into } V - M_\alpha^\iota \text{ out of } V \\ & + M_\alpha^\iota \text{ gained from interphase transfer} \\ & + M_\alpha^\iota \text{ gained from reactions} \\ & + \text{ external sources of } M_\alpha^\iota. \end{aligned} \tag{2.2}$$

Using the previously defined notation we can express (2.2) as

$$\frac{\partial}{\partial t}(\theta_\alpha \rho_\alpha \omega_\alpha^\iota) = -\nabla \cdot (\theta_\alpha \rho_\alpha \mathbf{v}_\alpha \omega_\alpha^\iota) - \nabla \cdot \mathbf{j}_\alpha^\iota + \mathcal{I}_\alpha^\iota + \mathcal{R}_\alpha^\iota + \mathcal{I}_\alpha^\iota. \tag{2.3}$$

We can sum over ι and apply the identities and definitions in §3.3.1 to get

$$\frac{\partial}{\partial t}(\theta_\alpha \rho_\alpha) = -\nabla \cdot (\theta_\alpha \rho_\alpha \mathbf{v}_\alpha) + \mathcal{S}_\alpha^\iota + \mathcal{I}_\alpha^\iota. \tag{2.4}$$

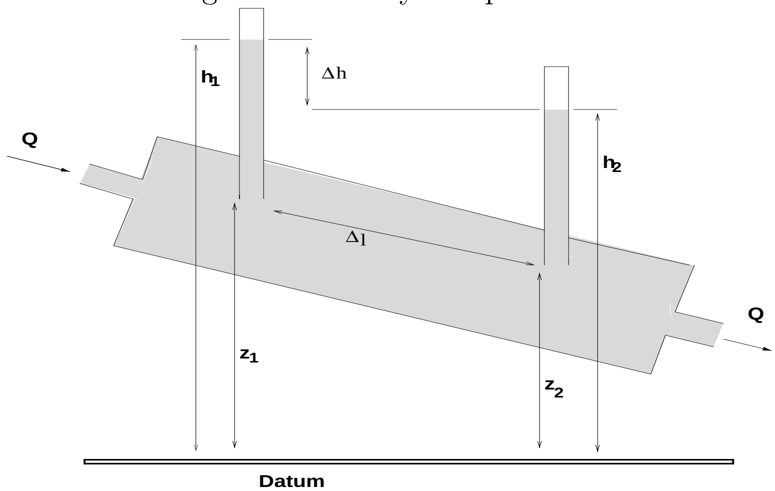
For flow in the unsaturated zone without the presence of a contaminant, the existing phases are the aqueaous phase, the gas phase, and the solid phase. We assume the solid phase is immobile and that the gradient of the velocity in the gas phase is negligible. Hence for Richards' equation, we consider only the aqueous phase, $\alpha = a$, equation (2.4)

$$\frac{\partial}{\partial t}(\theta_a \rho_a) = -\nabla \cdot (\theta_a \rho_a \mathbf{v}_a) + \mathcal{S}_a^\iota + \mathcal{I}_a^\iota. \tag{2.5}$$

To close the system we modify a relationship from saturated flow called Darcy's law. Henry Darcy, a civil engineer, studied flow through saturated sand in hopes of improving the water purification system in Dijon, France [17]. In 1856 his experimental results were published. He used an apparatus similar to that in Figure 2.1.

The apparatus consists of a cylinder of wet sand with two manometers attached to the cylinder. A manometer is a small open tube and when interested into saturated porous media, the water will rise in the tube. This elevation of the water is a measure

Figure 2.1: Darcy's Experiment



of the groundwater pressure at the inlet of the manometer and is called the hydraulic head, denoted h with units of length (L). Mathematically, the hydraulic head is expressed as

$$h = z + \frac{P_a}{\rho_a g} = z + \psi, \tag{2.6}$$

where g is the gravitational constant and p_a is the pressure of the fluid. The expression $\psi = \frac{P_a}{\rho_a g}$ is called the pressure head. For Darcy's experiment, we denote the cross sectional area of his cylinder as $A(L^2)$ and the distance between the manometers as Δl. In his experiment Darcy knew the volumetric flow rate, $Q(L^3/T)$ through the cylinder and the height of the water in the manometers, h_1 and h_2. Darcy found that the volumetric flow rate per unit surface area was proportional to the hydraulic gradient. That is

$$\frac{Q}{A} = -K_s \frac{(h_2 - h_1)}{\Delta l} = -K_s \nabla h.$$

The proportionality constant, K_s, is called the saturated hydraulic conductivity. The quantity Q/A is called the specific discharge and denoted $\mathbf{q}_a$. Since fluid only flows through the pore spaces in the media, it is useful to relate $\mathbf{q}_a$ to the more realistic velocity of the fluid by $\mathbf{q}_a = \theta_a \mathbf{v}_a$. Darcy's law can then be expressed as

$$\mathbf{q}_a = \theta_a \mathbf{v}_a = -K_s \nabla h = -K_s \nabla (z + \psi). \tag{2.7}$$

The proportionality constant, K_s, has dimensions (L/T) and is a measure of how easily water flows through the porous medium. K_s can further be defined as

$$K_s = \frac{k \rho_a g}{\mu_a}, \tag{2.8}$$

where k, a property of the porous media, is called the intrinsic permeabilty and μ_a is the viscosity of the fluid.

Darcy's law was developed for flow in saturated porous media, hence (2.7) has to be modified for flow in the unsaturated zone. To do this, we replace (2.8) with

$$K = k_r(\psi) K_s, \tag{2.9}$$

where $k_r(\psi)$ is called the relative permeability. When the medium is fully saturated, $k_r(\psi) = 1$, otherwise $0 < k_r(\psi) < 1$. We now have a modified version of Darcy's law,

[32],

$$\theta_a \mathbf{v}_a = \mathbf{q}_a = -K\nabla(z+\psi)$$

that we can substitute into (2.5) to get

$$\frac{\partial}{\partial t}(\theta_a \rho_a) = -\nabla \cdot [\rho_a K(z+\psi)] + \mathcal{S}_a + \mathcal{I}_a.$$

Chain out the derivative in the right-hand side to get

$$\frac{\partial}{\partial t}(\theta_a \rho_a) = -\underbrace{\nabla \rho_a \cdot K[\nabla(z+\psi)]}_{\text{negligible}} + \rho_a \nabla \cdot [K\nabla(z+\psi)] + \mathcal{I}_a + \mathcal{S}_a. \tag{2.10}$$

Assume that changes in density with respect to space are negligible and ignore the first term on the right-hand side. We chain out the time derivative on the left side in (2.10) and divide through by ρ_a to get

$$\frac{\partial \theta_a}{\partial t} + \frac{\theta_a}{\rho_a}\frac{\partial \rho_a}{\partial t} = \nabla \cdot [K\nabla(z+\psi)] + \mathcal{I}_a + \mathcal{S}_a. \tag{2.11}$$

The saturation of the aqueous phase, S_a, is a measure of the volume of the aqueous phase per volume of the void space. For saturated flow we would have that $\theta_a = \eta$, and $S_a = 1$, however we must take into account that air is present in our model. We express S_a, as a function of ψ by

$$\eta S_a(\psi) = \theta_a. \tag{2.12}$$

Thus, in (2.11) we have

$$\frac{\partial \theta_a}{\partial t} = \eta \frac{\partial S_a(\psi)}{\partial t} \tag{2.13}$$

Next consider

$$\frac{\theta_a}{\rho_a} \frac{\partial \rho_a}{\partial t} \tag{2.14}$$

in the left hand side of (2.11). If we assume that fluid is slightly compressible, with linear compression and compressibility coefficient β_a, then the density would change with respect to the fluid pressure, thus (2.14) becomes

$$\frac{\theta_a}{\rho_a} \frac{\partial \rho_a}{\partial p_a} \frac{\partial p_a}{\partial t} = \theta_a \beta_a \frac{\partial p_a}{\partial t}. \tag{2.15}$$

Using (2.12) and $\psi = p_a/(\rho_a g)$, we can relate (2.15) to changes in pressure head and (2.15) becomes

$$S_a(\psi)[\eta \rho_a g \beta_a] \frac{\partial \psi}{\partial t} = S_s \frac{\partial \psi}{\partial t}, \tag{2.16}$$

where S_s is the bracketed expression and is called the specific storage coefficient. S_s takes into account that fluid density decreases when fluid pressure decreases and vice versa, thus a saturated volume of water would release water from storage as the pressure head is decreased.

W substitute (2.16),(2.13), and (2.12) into (2.11) and assume $\mathcal{I}_a = \mathcal{S}_a = 0$ to get the pressure head form of Richards' equation

$$S_s S_a(\psi)\frac{\partial \psi}{\partial t} + \eta\frac{\partial S_a(\psi)}{\partial t} = \nabla\cdot\left[K\nabla\left(z+\psi\right)\right] \tag{2.17}$$

In one dimension Richards' equation is

$$S_s S_a(\psi)\frac{\partial \psi}{\partial t} + \eta\frac{\partial S_a}{\partial t} = \frac{\partial}{\partial z}\left[K_s k_r(\psi)\left(1+\frac{\partial \psi}{\partial z}\right)\right]. \tag{2.18}$$

2.3.3 Constitutive Relations

To close the system of equations we require the appropriate boundary and initial conditions. In addition we require constitutive relations to describe the pressure–saturation–relative permeability relationships. For this work we model the saturation and relative permeability with the van Genuchten and Mualem formulas. The van Genuchten relation [43], given by

$$S_a(\psi) = S_r + \frac{(1-S_r)}{[1+(\alpha|\psi|)^n]^m}, \psi \leq 0, \tag{2.19}$$

is used to model the saturation. The Mualem relation [33], given by

$$k_r(\psi) = \frac{[1-(\alpha|\psi|)^{n-1}[1+(\alpha|\psi|)^n]^{-m}]^2}{[1+(\alpha|\psi|)^n]^{m/2}}, \psi \leq 0, \tag{2.20}$$

is used to model the relative permeability. We also have that

$$S_a(\psi) = 1 = k_r(\psi), \psi > 0.$$

Here S_r is the residual saturation and α and n are properties of the porous media. $\alpha > 0$ is a measure of the mean pore size and $n > 1$ is a measure of the pore size uniformity. m is a function of n given by $m = 1/(1-n)$.

Note that these relations are used only when the media is unsaturated, that is $\psi < 0$. Once saturated conditions develop, $\psi > 0$, both these relations obtain a value of one and remain constant. In this case, RE reduces to the saturated flow equation. Although alternatives to (2.19) and (2.20) exist, such as the approach taken by Brooks and Corey [10], the van Genuchten and Mualem relationships are widely used with experimentally determined parameters.

One interesting aspect of the constitutive relations is that for values of n such that $1 < n < 2$, (2.19) and (2.20) are not Lipschitz continuous. Also of interest is that at the water table, $\psi = 0$, the constitutive laws are not even differentiable for these values of n. This is a physically realistic scenario. For example, consider a nonuniform media that begins drained to equilibrium and infiltration from ponding occurs at the top of the domain. The domain will be saturated in some regions and unsaturated in others, resulting in a solution with a sharp, propagating front. In the context of temporal adaption which is based on Taylor series analysis, traditional theory may not hold for a problem of this nature. We address this issue in more detail in Chapter

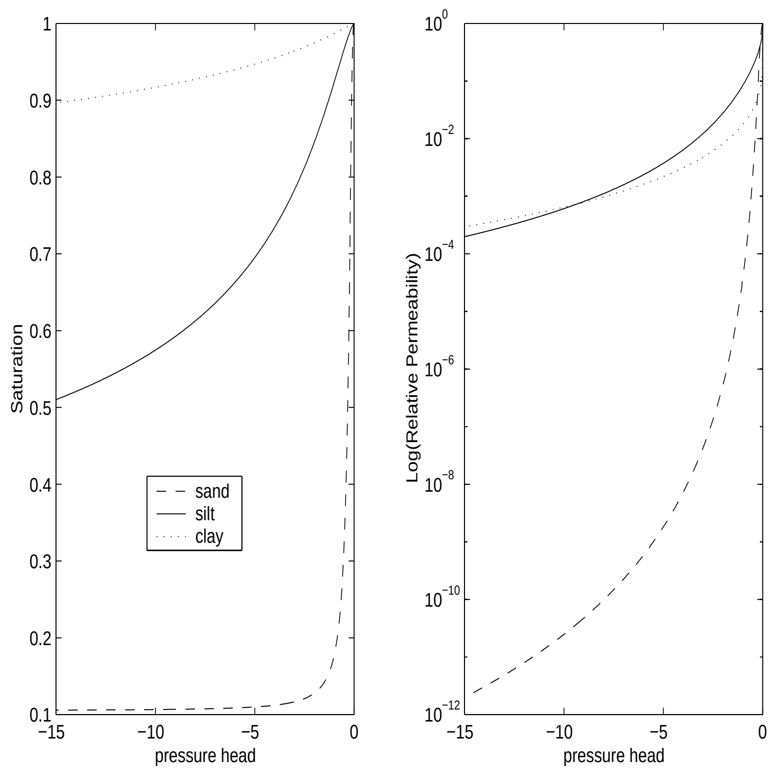
3.

Figure 2.2 shows (2.19) and (2.20) for silt, sand, and clay for the parameter values given in Table 2.1 which are used in the numerical experiments in §2.9. The values were taken from [12].

Table 2.1: Soil Parameters

Soil type	S_r (-)	α (1/ft)	n (-)
Sand	0.105	4.420	2.68
Silt	0.074	0.487	1.37
Clay	0.179	0.244	1.09

Figure 2.2: Constitutive Equations



2.4 Finite Element Approach

In ADH, the equation for RE given by (2.17) is discretized in space using piecewise linear Galerkin finite elements. Backward Euler is used to advance the solution in time. The objective of this section is to describe the finite element discretization of Richards' equation. We begin by discretizing a simple parabolic example [22]. We then derive the weak formulation of RE. We include a term by term derivation of the discretization on an arbitrary element for one-dimensional RE given by (2.18) in Appendix B. The motivation for the one-dimensional derivation is that we developed a one-dimensional finite elements code structured as similarly as possible to ADH to perform feasibility studies. We also use the spatial discretization on the one–dimensional RE to study the effects of the nonsmooth nonlinearities in Chapter 3. The assembly of the system of equations are identical in one and three dimensions and can be understood in ADH by considering the process in one dimension, which we do next.

2.4.1 Parabolic Model Problem

In this section, we describe the finite element discretization used in ADH by demonstrating on a parabolic model problem. Let $u(x, t) \in \Omega \times (0, T)$ where Ω is a convex polygonal domain with boundary Γ, $x \in \Omega$. Our model problem is

$$\begin{aligned} u_t - \Delta u = f &\quad \text{in } \Omega \times (0,T) \\ u = 0 &\quad \text{on } \Gamma \times (0,T) \\ u(x,0) = u^0 &\quad x \in \Omega \end{aligned} \tag{2.21}$$

Define the bilinear forms,

$$(v,w) = \int_\Omega vw dx \text{ and } a(v,w) = \int_\Omega \nabla v \nabla w dx.$$

Let $H_0^1(\Omega) = \{v : v, \nabla v \in L^2(\Omega), v = 0 \text{ on } \Gamma\}$. The weak form of (2.21) is found by multiplying by $v \in H_0^1$ and integrating over Ω. Simplifying yields

$$(u_t, v) + a(u,v) = (f,v),$$

$\forall v \in H_0^1, t \in I$.

Consider V_h, a finite dimensional subspace of H_0^1 with basis $\{\phi_1, ..., \phi_m\}$, consisting of piecewise linear functions defined on a uniform discretization of Ω with size h. We can now define our discrete problem as to find $u_h \in V_h$ so that $\forall v_h \in V_h, t \in (0,T)$,

$$((u_h)_t, v_h) + a(u_h, v_h) = (f, v_h).$$

Using backward Euler to advance in time can be viewed as using finite elements that are piecewise constant in time or as a method of lines (MOL). MOL discretizes the time dependent system in space and then uses an ODE integrator to advance in time. Let $0 = t_0 < t_1 < ... < t_N = T$, $I_k = (t_{k-1}, t_k)$, and $\Delta t_k = t_k - t_{k-1}$. Our problem is now to find u_h^k an approximation to $u(\cdot, t_k)$ such that for $k = 0, ..., N$ and $\forall v_h \in V_h$ we have

$$\left(\frac{u_h^k - u_h^{k-1}}{\Delta t_k}, v_h \right) + a(u_h^k, v_h) = (f(t_k), v_h). \tag{2.22}$$

To obtain the resulting linear system, we express u_h^k in terms of the basis functions for V_h. We have

$$u_h^k = \sum_{i=1}^{m} u_i^k \phi_i, \tag{2.23}$$

where $u_i^k \in \mathcal{R}$ for $i = 1, ..., m$ are the time dependent coefficients. Since (2.22) must hold for all ϕ_j and applying (2.23) we have that

$$\sum_{i,j} \int_\Omega \phi_i \phi_j (u_i^k - u_i^{k-1}) dx + \Delta t_k \sum_{i,j} \int_\Omega \nabla \phi_i \nabla \phi_j u_i^k dx = \Delta t_k \int_\Omega f \phi_j dx,$$

for $j = 1, ..., m$. The linear system is

$$(M + \Delta t_k K) U^k = \Delta t_k F + M U^{k-1},$$

where

$$U_i^k = u_i^k,$$

$$M_{i,j} = \int_{\Omega} \phi_i \phi_j dx,$$

$$K_{i,j} = \int_{\Omega} \nabla\phi_i \nabla\phi_j dx,$$

and

$$F_i = \int_{\Omega} f\phi_i dx.$$

2.4.2 Accuracy

Error estimates have been derived for the backward Euler scheme and when V_h consists of piecewise linear polynomials as above. For example, if we assume a quasi-uniform mesh then it has been shown in [22], [18] that we have the following error estimate. Suppose there is a constant γ such that the time steps satisfy $\Delta t_{k-1} \le \gamma \Delta t_k$, $k = 1, 2..., N$ and let U^k be the approximation to $u(t_k)$, then there exists a constant C depending on γ and the regularity of the mesh such that

$$\max_{t \in I} ||u - U^k|| \le C(\max_{k \le N} \Delta t_k ||u_t||_{\infty, L^2(\Omega)} + \max_{t \in I} h^2 ||u(t)||_{\infty, H^2(\Omega)}). \qquad (2.24)$$

To test the performance of the one-dimensional finite element code, we performed a grid refinement study on a nonlinear parabolic problem, structured similarly to (2.18), of the form

$$f(u)\frac{\partial u}{\partial t} + \frac{\partial f}{\partial t} = \frac{\partial}{\partial z}\left[g(u)\left(\frac{\partial u}{\partial z} + 1\right)\right] + Q,$$

where $u(z,t), g(u), f(u)$ were known functions so that the error could be computed exactly. The error estimate, (2.24), implies second order accuracy in space and first order accuracy in time. We examined the error in the scaled Euclidean norm as the mesh and time step were refined. Let $E(h) = (1/\sqrt{N})||u - u_h||_2$ be the error with mesh size h, N nodes, and time step Δt and let $E(h/2)$ be the error with mesh size $h/2$ and time step $\Delta t/4$. We can expect from (2.24) that $E(h)/E(h/2) \approx 4$. We include the results from one of these grid refinement studies where the true solution was $u(z,t) = \frac{zsin(\pi z)}{1+t}$ and $f(u) = g(u) = u$ for $z \in \Omega = [0,1]$ and $I = [0,1]$. We then have that

$$uu_t + u_t - u_z^2 - u_z - uu_{zz} = Q.$$

Table 2.2 shows the mesh sizes, time steps, error, and ratios of errors. The fourth column confirms the order of discretization and performance of the code.

Table 2.2: Order Verification: 1-D

h	Δt	$E(h)$	$E(h)/E(h/2)$
1/10	1.2500e-01	4.4294e-03	
1/20	3.1250e-02	1.1822e-03	3.7467
1/40	7.8125e-03	2.9885e-04	3.9558
1/80	1.9531e-03	8.0028e-05	3.7430

2.4.3 Weak Form of RE

In this section we use the ideas from §2.4.1 to derive the weak form of Richards' equation. Note that the full discretization for 1-D Richards' equation is given in

Appendix B. We denote the residual form of Richards' equation as

$$RE(\psi) = S_s S_a(\psi)\frac{\partial \psi}{\partial t} + \eta \frac{\partial S_a}{\partial t} - \nabla \cdot [K(\psi)\nabla(z+\psi)] = 0 \tag{2.25}$$

The weak form of Richards' equation is found by multiplying (2.25) by an appropriate weight function and integrating over time and space. ADH uses a Galerkin method [20] with piecewise linear polynomials in space and piecewise constant polynomials in time. Let Ω_k be the current spatial mesh with boundary Γ_k over the time interval $[t^{k-1}, t^k]$. We assume Dirichlet boundary conditions, $\psi = \bar{\psi}$ on Γ_k. Our finite element interpolation spaces are

$$V_h = \{v_h | v_h \in H^{1,h}(\Omega_k), v_h = 0 \text{ on } \Gamma_k\},$$

$$\hat{V}_h = \{\hat{v}_h | \hat{v}_h \in H^{1,h}(\Omega_k), \hat{v}_h = \bar{\psi} \text{ on } \Gamma_k\}.$$

We pose our formulation as: given $\hat{\psi}^{k-1}$ find $\hat{\psi}^k$, an approximation to $\psi(t^k)$ such that

$$\int_{t^{k-1}}^{t^k} \int_{\Omega_k} RE(\hat{\psi}^k) v_h d\Omega_k dt = 0$$

$\forall \hat{\psi}^k \in \hat{V}_h$, $\forall v_h \in V_h$. The weak formulation can be simplified using integration by parts to get

$$\int\limits_{\Omega_k} S_s S_a(\hat{\psi}^k)[\hat{\psi}^k - \hat{\psi}^{k-1}] v_h dz + \int\limits_{\Omega_k} \eta [S_a(\hat{\psi}^k) - S_a(\hat{\psi}^{k-1})] v_h dz \tag{2.26}$$

$$+\Delta t \int_{\Omega_k} [K_s k_r(\hat{\psi}^k) \nabla(\hat{\psi}^k + z)] \nabla v_h dz = 0.$$

2.5 Newton–Krylov Methods

In §2.4 we discretized the governing equations using finite elements to obtain a system of nonlinear equations to be solved at each time step. That is, to advance from t^k to t^{k+1} we must solve the nonlinear equation given by (2.26).

In both the 1-D and 3-D results presented, Newton's method is used to solve the nonlinear equations. Newton's method iteratively solves problems of the form

$$F(x) = 0$$

where $F : \mathbb{R}^N \to \mathbb{R}^N$ is a nonlinear function with the ith component denoted by f_i. Let $F'(x)$ denote the Jacobian of $F(x)$ with

$$F'(x)_{ij} = \frac{\partial f_i}{\partial x_j}(x).$$

We make the following assumptions on F;

$$\begin{aligned}
&\text{A solution } x^* \text{ exists,} \\
&F'(x) : \mathbb{R}^N \to \mathbb{R}^N \times \mathbb{R}^N \text{ is Lipschitz continuous, and} \\
&F'(x^*) \text{ is nonsingular.}
\end{aligned} \tag{2.27}$$

Let x_c be the current Newton iterate. Newton's method computes the next iterate by

$$x_+ = x_c - F'(x_c)^{-1}F(x_c). \tag{2.28}$$

Let $s = -F'(x_c)^{-1}F(x_c)$ be the Newton step, then (2.28) can be written as $x_+ = x_c + s$. The Newton step is found by solving the linear system

$$F'(x_c)s = -F(x_c). \tag{2.29}$$

Let $B(\delta) = \{x : ||e|| < \delta\}$ be the ball of radius δ about x^* with $e = x - x^*$. The error in the current Newton iterate is $e_c = x_c - x^*$. The convergence result for Newton's method is given in the following theorem [25].

Theorem 2.1. *Assume* (2.27) *are satisfied. Then there are* $C > 0$ *and* $\delta > 0$ *such that if* $x_c \in B(\delta)$ *then the Newton iteration given by* (2.28) *converges quadratically, that is*

$$||e_+|| \leq C||e_c||^2.$$

In the 1-D results presented in §2.8 and §3.3.4, (2.29) is solved with Gaussian elimination. For the 3-D problems however, the number of unknowns is often large and a direct solve is not sensible. In ADH, since storage and computational efficiency are required, an inexact–Newton's method is used. Inexact–Newton methods approximate s by solving (2.29) iteratively with a Krylov method such that the Newton step satisfies

$$||F'(x_c)s + F(x_c)|| \leq \eta_c||F(x_c)||. \tag{2.30}$$

This can be viewed as a reduction in the relative residual and η_c is called the forcing term. We have the following convergence theory for inexact-Newton methods [25].

Theorem 2.2. *Assume* (2.27) *and* (2.30) *are satisfied. Then there exists* δ *and* C_I *such that if* $x_c \in B(\delta)$ *and* s *satisfies* (2.30) *and* $x_+ = x_c + s$ *then*

$$||e_+|| \leq K_I(||e_c|| + \eta_c)||e_c||.$$

ADH uses BiCGstab as the Krylov method. Although BiCGStab can break down and is not supported by the convergence theory of GMRES or CG, the method uses less storage than GMRES and the system need not be symmetric positive definite. [25].

2.6 Schwarz Preconditioners

The convergence of Krylov methods is accelerated by the use of preconditioning. ADH applies right preconditioning to (2.29) such that the new system is

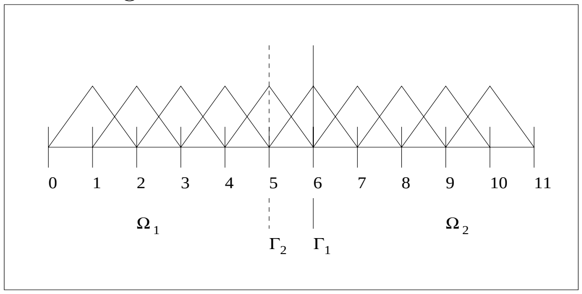
$$F'(x_c)B\hat{s} = -F(x_c), \tag{2.31}$$

with $s = B\hat{s}$. The hope is that B is close in a sense to $F'(x_c)^{-1}$ thus making (2.31) easier to solve than the original system (2.29). ADH uses Schwarz domain decomposition preconditioners [39] because they are readily implemented in parallel. Schwarz preconditioners can be viewed as smoothers which incorporate error correction. To understand the general idea behind error correcting preconditioners, consider solving the linear system $Au = f$ iteratively to obtain an approximation v. If $r = f - Av$ is the residual, and $e = u - v$ is the error, then $Ae = r$. If we could solve $Ae = r$ exactly we could "correct" our approximate solution v to obtain u.

Schwarz domain decomposition involves dividing the domain Ω into p overlapping subdomains, Ω_i, $i = 1, ..., p$. We then restrict the linear operator on each subdomain, solve each subdomain problem, and then reconstruct the solution on the entire domain. The overlapping of subdomains creates new artificial boundaries. Figure 2.3 shows a domain,Ω, divided into two subdomains Ω_1 and Ω_2 with an overlap of width δ. We now have two artificial boundaries, Γ_1 the boundary of Ω_1 lies within Ω_2 and Γ_2 the boundary of Ω_2 lies within Ω_1. In ADH, the overlap is the width of a single element. The number of nodes is divided evenly among processors with any remaining nodes given to the last processor. The user specifies the number of subdomains per processor, and then on each processor, the nodes are divided evenly again. To close the system of equations on a subdomain, the Schwarz method defines zero Dirichlet boundary conditions for the artificial boundaries.

ADH offers three choices of domain–decomposition preconditioners; one-level ad-

Figure 2.3: Two Subdomains



ditive Schwarz, two-level additive Schwarz, and a two level hybrid, which we describe below. Define $R_i = [0I0]$ to be the restriction operator for subdomain i, $i = 1, ..., p$, where I is the identity operator on subdomain i. Let

$$B_i = R_i^T (R_i A R_i^T)^{-1} R_i.$$

One-level additive Schwarz is a block Jacobi preconditioner and the preconditioner is

$$B = \sum_{i=1}^{p} B_i.$$

A two-level method constructs a coarse mesh problem $A_C u_C = f_C$, where the subscript C denotes "coarse mesh." We now refer to the original system on all of Ω as the "fine mesh" problem, denoted with a subscript F. The idea behind a two level method is to approximate the error, e_C, on the coarse mesh by solving $A_C e_C = r_C$, where r_C is the coarse mesh residual. Then we interpolate e_C to the fine mesh to obtain e_F, and then correct the approximated solution with $v \leftarrow v + e_F$. If we let R_0 denote the restriction interpolator to the coarse mesh, then R_0^T is the interpolation operator back to the fine mesh. The two level error correction is then expressed as

$$v \leftarrow v + R_0^T A_C^{-1} R_0 (f_C - A_C u_C).$$

A two level additive Schwarz method combines a coarse grid correction with a one-level additive Schwarz preconditioner. The preconditioner is

$$B = B_0 + \sum_{i=1}^{p} B_i.$$

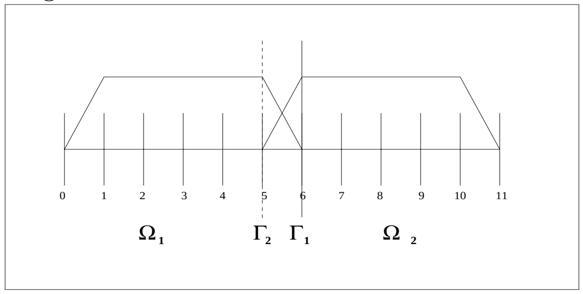
The two level hybrid preconditioner also implemented in ADH is

$$B = B_0 + \sum_{i=1}^{p} B_i (I - AB_0).$$

The two level preconditioners require the construction of and solution to a coarse mesh problem. ADH incorporates ideas from algebraic multigrid [9],[44]. The coarse mesh basis functions are obtained using the fine mesh basis functions by aggregation. This fits well into the framework of ADH because the meshes are unstructured, making interpolation between meshes difficult. The coarse mesh basis functions have a constant value in each subdomain except in the overlap region as in Figure 2.4.

The applications of the preconditioners require solutions to the subdomain and coarse mesh problems. The subdomain solves are done with a sparse direct solver and the coarse mesh problems are done with a full LU factorization.

Figure 2.4: Coarse Mesh Basis Functions



Bounds on the condition number of the precondioned system have been proven for a general elliptic problem. For the two level aggregation preconditioner [21], the condition number of the preconditioned system is $\mathcal{O}([1 + H/\delta]^2)$, where H is the width of the subdomain and δ is the size of the overlap. If a method such as conjugate gradient is used, then the theory suggests that the number of iterations for convergence will be proportional to the square root of the condition number of the system. So for the two level preconditioner this means that if $\delta = h$, the width of the fine mesh, and we keep H/h constant then the iteration count would remain constant. Testing was done to verify the condition number of the two level preconditioner in [21].

2.7 Error Estimation and Time Step Control

Adaptive temporal integration is needed in ADH for several reasons. The solution of RE often results in the formation of a sharp front, requiring a small time step during formation. A fixed time step method is not computationally efficient. Moreover, the user has no control over the accuracy of the solution. Variable time–stepping based on error control will improve the robustness of the code and allow the user to supply an error tolerance for desired accuracy [5]. We consider an error control method, which we derive under the assumption that the nonlinearities (2.19) and (2.20) are smooth. Recall, ADH advances in time with backward Euler and regrids after each time step. We seek to estimate the local truncation error by using linear extrapolation. We then compute the next step size in hopes of bounding the error in the next step by some tolerance, TOL.

Most approaches to solving Richards' equation use low-order temporal integration and a heuristic method for changing the time step such as in [36]. By a heuristic method, we mean the time step changes based on the performance of the nonlinear solver—growing by a certain factor if the nonlinear solver converges in a set number of iterations, or shrinking by a certain factor if the nonlinear solver fails to converge. The advantage of this method is that it is easily implemented, but it lacks theoretical support and does not attempt to measure the temporal error. Moreover, experiments must be done to determine an optimal growth factor for the time step and the growth factor is likely to be problem dependent. A similar heuristic method was previously used in ADH and we often observed a saw–tooth effect in the size of the step where the time step would grow and shrink repeatedly due to repeated failure and success of the nonlinear solver. It has been shown that adaptive temporal integration based on error control can improve the performance of the nonlinear solver for Richards' equation [41].

Several approaches exist for controlling the temporal error for various forms of

Richards' equation. In [41] the authors use a variation of the variable step size, variable order differential algebraic equation (DAE) solver called DASPK [11]. DASPK uses backward difference formulas (BDFs) to discretize in time and can vary from first to fifth order. The order and step size depend on an approximation to the local truncation error using a predictor–corrector method. In [41] the DAE formulation of the pressure head form of (2.17) is discretized with a method of lines (MOL). This method is shown to be robust for difficult 1-D infiltration problems as described in §2.3.3 when high accuracy is required. However, low order methods are competitive with higher order methods when low accuracy is required, which is the case with ADH.

In [23] the author works with the moisture based for of Richards' equation given by

$$\frac{\partial \theta}{\partial t} - \nabla \cdot D(\theta) + \frac{\partial K(\theta)}{\partial z} = 0, \tag{2.32}$$

where θ is the volumemetric moisture content, $K(\theta)$ is the permeability, and $D(\theta)$ is the diffusivity. The authors derive a method for error control using a backward Euler approximation and advancing in time with local extrapolation equivalent to a second order Thomas-Gladwell scheme. Their approach is appealing in that second order accuracy is obtained by solving the linear system from the backward Euler discretization. They formulate the problem so that the unknown is $\dot{\theta}^{n+1}$ and recover $\theta^{n+1} = \theta^n + \Delta t \dot{\theta}^{n+1}$, which is a different formulation then what is implemented in ADH.

In [24], the authors of [23] apply a similar second order method to control the local truncation error for the mixed form of Richards' equation given by

$$\frac{\partial \theta}{\partial t} + \frac{\theta}{\eta} S_s \frac{\partial \psi}{\partial t} - \nabla \cdot K(\psi) \nabla \psi + \frac{\partial K(\psi)}{\partial z} = 0, \tag{2.33}$$

where here ψ is the pressure head, η is the porosity, S_s is the specific storage, and $K(\psi)$ is the permeability. The solution advances in time with a second order Adams–Moulton scheme. The backward Euler and second order Thomas–Gladwell approximations to ψ are then backed out and used to approximate the error in the pressure variable. The estimated error for ψ is then used to select a time step to advance the entire system.

Our approach differs from [41],[24], and [23] in several ways.

• In[41] the authors show that solutions for RE in the nonsmooth regime, that is where $1 < n < 2$, can be found using adaptive temporal integration based on error control. However, the step size is chosen so that it either doubles, is cut in half, or remains constant. In our approach we choose a time step that is a function of the estimated error in the next step.

• Our approach for computing the time step is more similar to the second order error approximation in [24], [23] although they do not apply their method to problems in the nonsmooth regime, or allow saturated conditions to develop in the domain.

• We implement and analyze a first order method to the pressure based form of RE and apply the method successfully to problems in the nonsmooth regime.

• We present numerical results for three dimensional heterogeneous media where saturated and unsaturated conditions develop.

2.7.1 Theory for Error Control

We describe the theory by considering a scalar ODE. Consider

$$y'(t) = f(t, y), t^0 < t^f, y(t^0) = y_0,$$

where $f(t, y)$ is a smooth function. Let $\Delta t_k = t^k - t^{k-1}, y^{k-1}$ approximate $y(t^{k-1})$ such that we seek y^k, an approximation to $y(t^k)$. Assume we advance in time with backward Euler and let $y^k_{BE} = y^{k-1} + \Delta t_k y'(t^k)$ be the backward Euler approximation. The local truncation error for y^k_{BE} is

$$E_k = |y^k_{BE} - y(t^k)| = \frac{\Delta t_k^2}{2}|y''(\xi_k)| + \mathcal{O}(\Delta t_k^3), \tag{2.34}$$

where $\xi_k \in (t^{k-1}, t^k)$. Our first goal is to approximate $|y''(\xi_k)|$ using a simple, explicit predictor. We let C_k denote the approximation to $|y''(\xi_k)|$.

Using C_k and (2.34) we can bound the estimated local truncation error by TOL if the next step size, Δt_{k+1} satisfies

$$\frac{\Delta t_{k+1}^2}{2} C_k < TOL.$$

This inequality is satisfied if

$$\Delta t_{k+1} = \sqrt{\frac{2TOL \cdot frac}{C_k}}, \tag{2.35}$$

where $frac < 1$ is a user specified safety parameter to try to ensure that the next step is not a failure.

To obtain C_k we define the predictor polynomial as $y^k_{PRE} = y^{k-1} + \Delta t_k y'^{k-1}_{PRE}$, where $y'^{k-1}_{PRE} = (y^{k-1} - y^k)/\Delta t_{k-1}$ is just a forward difference approximation to $y'(t^{k-1})$. The analysis shows that

$$|y^k_{PRE} - y^k_{BE}| = |\Delta t_k^2 - \frac{\Delta t_k \Delta t_{k-1}}{2}||y''(\xi_k)|.$$

Hence,

$$C_k = 2\frac{|y^k_{PRE} - y^k_{BE}|}{|\Delta t_k(2\Delta t_k - \Delta t_{k-1})|} \approx |y''(\xi_k)|. \tag{2.36}$$

2.7.2 Implementation

To implement the adaptive scheme in ADH, we combined the empirical methods for changing the time step already in place with the new method based on error control. This is the approach taken in the ODE code PVODE [19] and is common in practice [11, 35]. Full implementation is described in algorithmic form below.

algorithm: error_control

• Try to compute y_{BE^k} with Newton's method

if Nonlinear solver or linear solver fail to converge **then**
$\Delta t_k/4 \leftarrow \Delta t_k$ and try again

else Compute E_k using y^k_{BE} and y^k_{pre}

if $E_k > TOL$ **then**
$\Delta t_k/4 \leftarrow \Delta t_k$ and try again

else $E_k < TOL \rightarrow$ Compute Δt_{k+1}

if $\Delta t_{k+1}/\Delta t_k > 10$ **then**
$\Delta t_{k+1} = 3 * \Delta t_k$.

• Move on in time

2.8 Numerical Results: 1-D Grid Refinement Study

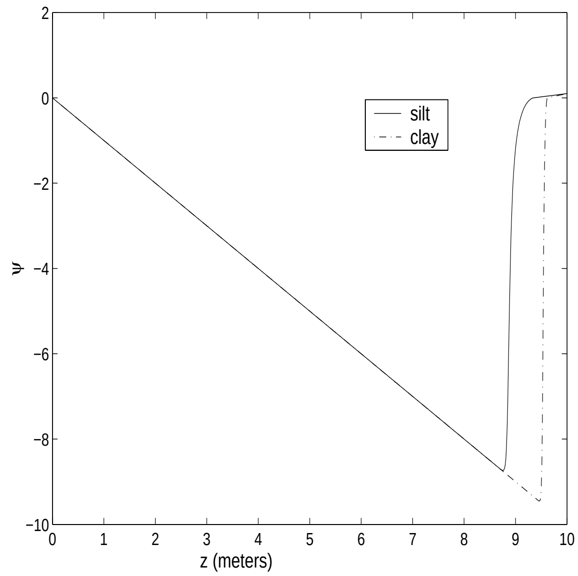
For RE, the assumption that the nonlinearities are smooth may not hold for certain physical parameters, meaning a Taylor series expansion may not even exist for the continuous problem. The effect would be that the error estimator could depend on the spatial mesh. As the mesh is refined, we may experience erratic behavior in the temporal integrator. To test the effect of the nonsmooth nonlinearities we performed a one dimensional grid refinement study. We considered two infiltration problems– one with clay and one with silt. Both of these soils have values of n making (2.19) and (2.20) not Lipschitz continuous. Table 2.3 gives a complete description of the media parameters for each simulation. Both problems are solved over a 10 meter long domain. We begin with the media unsaturated and fix a boundary condition at the top so that saturated conditions develop and propagate through the domain. Both problems result in the formation of a sharp front and require an initial time step of $10^{-9} days$ to capture the front accurately. Clearly a fixed time step method would not be efficient. The plot of the solutions for clay and silt are provided in Figure 2.5 to show the sharp infiltration front that develops between the saturated and unsaturated parts of the domain.

Table 2.3: Media Properties

Parameter	clay	silt
n	1.09	1.37
$\alpha(1/m)$	0.244	0.478
S_r	0.179	0.074
η	0.33	0.40
$K_s(m/day)$	1.10808e-5	1.1801e-03
t_f	600 days	90 days
$max\Delta t$	10 days	1 day

The 1-D simulator used was structured similarly to ADH except that the linear solve for the Newton step was performed with Gaussian elimination. Since the saturation and relative conductivity relations are expensive to evaluate, the one dimensional simulator and ADH both use piecewise linear splines to approximate (2.19) and (2.20). This is common in practice and has proven to be more robust than exact evaluation [31]. For these infiltration problems, we sampled 5000 spline knots on the interval $-15 < \psi < 0$. The spline is extended to be constant outside of this range. In the 1-D simulation we considered meshes of size $\Delta z = 1/100, 1/200, 1/400$, and

Figure 2.5: Clay and Silt



1/800. The 1-D simulations were done on an Ultra 10 workstation with a 440MHz processor using cc: Workshop Compilers 5.0 98/12/15 C 5.0. For these results we used $TOL = 10^{-3}$, $frac = 0.5$, and an initial time step of 10^{-9}. To observe the behavior of the error indicator as the mesh was refined, we tabulated C_k, given by (2.36), as a function of time. Figures 2.6 and 2.7 show that C_k appears to be independent of the mesh size.

2.9 Numerical Results: ADH

To test the performance of the temporal adaption in ADH, we simulated flow through three unstructured columns of heterogeneous media. The three problems differ only in their spatial mesh. The three grids are described in Table 2.4 and we refer to them as het_column_0, het_column_1, and het_column_2. The columns consist of layers of

Figure 2.6: C_k for Clay

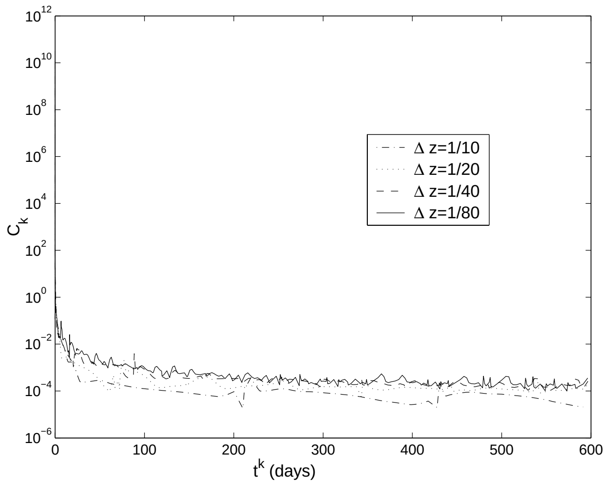


Figure 2.7: C_k for Silt

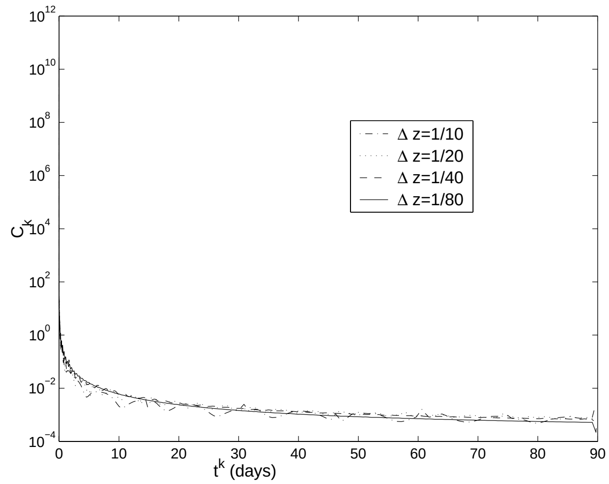
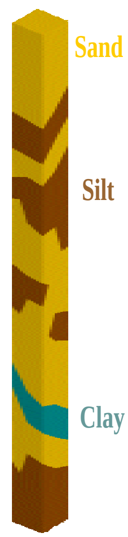


Table 2.4: Three Grids

Problem	nodes	elements
0	845	3840
1	5881	30720
2	43889	245760

sand, silt, and clay and are 10 meters long. This time, the media begins fully saturated and a boundary condition is set so the column drains from the bottom. We began with an initial time step of 10^{-8} days and let the simulation run for 30 days with a maximum allowable time step of 0.5 day. For the 3-D runs we used $TOL = 0.01$ and $frac = 0.5$. Figure 2.8 shows the material domain. All simulations were done with fixed spatial grids.

Figure 2.8: Heterogeneous Column



We present one set of results with Jacobi preconditioning on one processor and one set of results with the two level hybrid preconditioner. In Table 2.5 we tabulate the iteration history for the runs done with Jacobi preconditioning and in Table 2.6 we present the results for the two level hybrid preconditioner. For the two level preconditioner, we also include the number subdomains used, Ω_i, and the number of processor we ran on. The platform was an IBM RS/6000 SP, located at the North Carolina Supercomputing Center. This IBM SP is equipped with 180 nodes and each node has four 375 MHz Power3-II processors. It runs C for AIX Compiler, Version 5.

Table 2.5: ADH: Iteration history, Jacobi Preconditioning

het_column_	time	No. TS	Ave TS
0	30	234	1.67e-01
1	519	579	8.17e-02
2	1875	221	1.36e-01

Table 2.6: ADH: Iteration history, Two Level Hybrid Preconditioning

het_column_	time	No. TS	Ave TS	Ω_i/No. Procs.
0	42	216	1.39e-01	4/1
1	55	147	2.04e-01	32/4
2	340	225	1.33e-01	128/16

In Tables 2.5 and 2.6 the times are in seconds. We also give the total number of time steps taken and the average time step. The adaptive algorithm grows the time step seven orders of magnitude and the average time step is nearly the same for all the problems, regardless of the preconditioner.

Figures 2.9 and 2.10 show the approximation C_k as in (2.36). Again, we do not see erratic behavior or sudden growth in the approximation due to the nonsmoothness caused by the physical parameters.

Figure 2.9: C_k in ADH, Jacobi Preconditioning

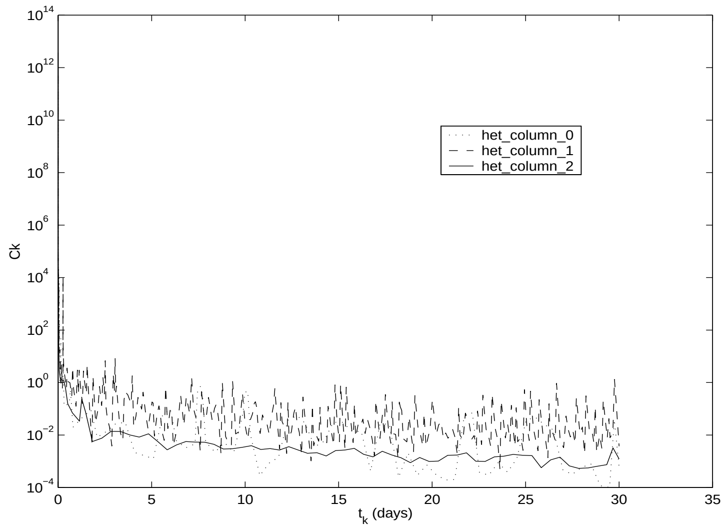
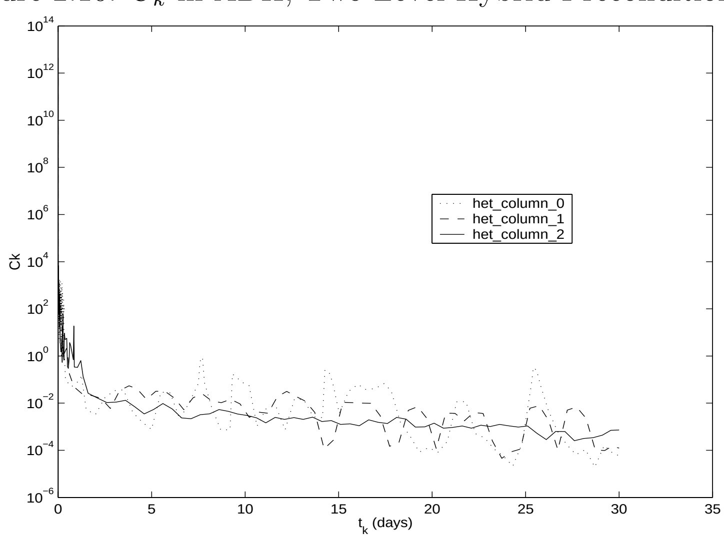


Figure 2.10: C_k in ADH, Two Level Hybrid Preconditioning



Figures 2.11 and 2.12 show growth of the time step. In both cases the stepsize eventually reaches the maximum allowable time step.

Figure 2.11: Δt, Jacobi Preconditioning

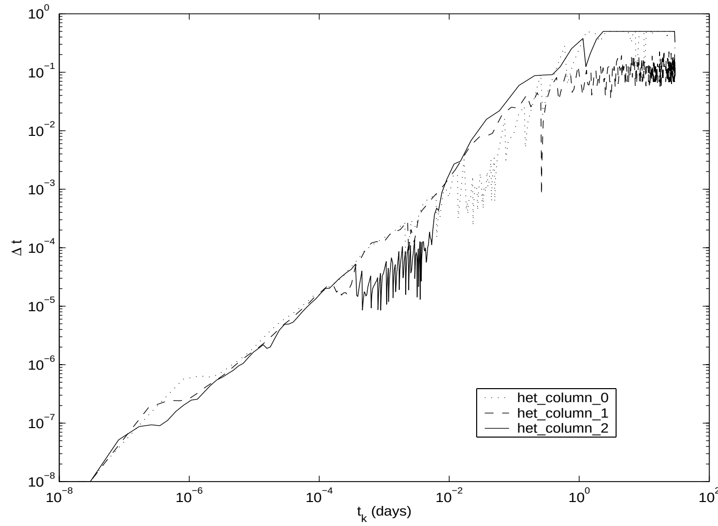
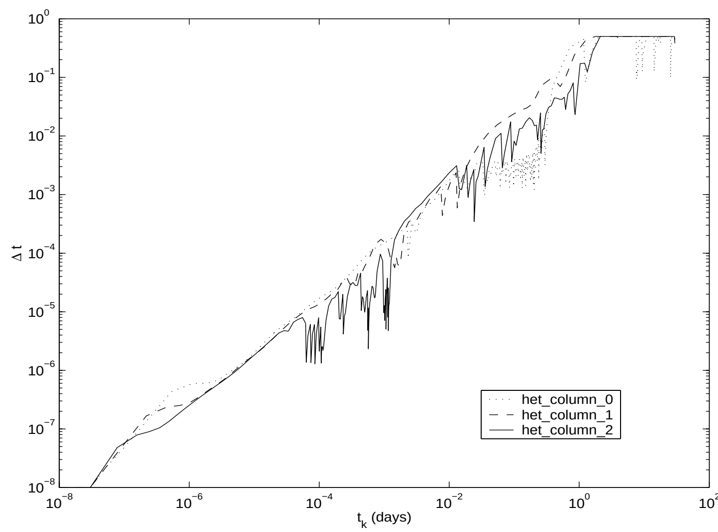


Figure 2.12: Δt, Two Level Hybrid Preconditioning



2.9.1 Preconditioner Management

The two level preconditioners in ADH require the construction and solution to a coarse mesh problem. Prior to the implementation of error control, the coarse mesh component of the preconditioners was computed every time step. It was shown in [21] that timings were improved if the coarse mesh part was only formed and factored every 10 nonlinear iterations, which we refer to as lagging the preconditioner for 10 nonlinear iterations. The results in [21] were obtained by studying problems where the linear and nonlinear solvers did not fail, a large fixed time step was used, and $S_s = 0$ in (2.17). This method may not be sensible for problems where $S_s \neq 0$ and when the time step is varying by orders of magnitude over the coarse of the simulation. We implemented a time–dependent update strategy based on the error control in ADH. Updating was done if the approximated error failed to meet the user specified tolerance, the linear or nonlinear solver failed to converge, or *lag* time steps occurred, where lag is a user specified parameter. We present the results for het_column_2 and the two level hybrid preconditioner in table 2.7.

The runs were also done on the IBM SP2 at the North Carolina Supercomputing Center. The SP has 180 nodes, each with 4 processors. A user can submit a job by specifying the number of nodes needed for a job and the number of processors per node. We tested the performance of the update strategy by looking at 128 subdomains on 8 processors. For Test 1, we used two nodes with four processors per node and for TEST 2 we used 8 nodes with one processor per node. This was done to see if communication between processors would significantly affect the timings. Table 2.7 shows the timings in seconds where time (1) corresponds to Test 1 and time (2) refers to Test 2. We also tabulate the number of linear and nonlinear iterations, the total number of time steps taken, the average time step, and the number of times the time step was cut due to a failure to meet the error tolerance or linear and nonlinear convergence criteria.

Table 2.7: Preconditioner Management

lag	time (1)	time (2)	L its	NL its	No. TS	Ave TS	E cuts	L cuts
0	330	339	1240	266	225	1.33e-01	40	0
2	433	412	1964	329	257	1.17e-01	72	0
5	424	390	1720	325	249	1.20e-01	71	0
10	360	331	1486	269	227	1.32e-01	39	0
15	337	317	1190	264	224	1.34e-01	39	0
20	329	322	1068	262	223	1.35e-01	39	0

Table 2.7 shows that regardless of the amount of updates for the coarse mesh component of the preconditioner the algorithm uses nearly the same number of time steps and step size throughout the simulation. For the runs done on two nodes, there is no significant improvement in the timings and for the runs done on 8 nodes, the improvement is around 6%. In both cases, lagging the preconditioner can result in a slight increase in the timings, however neither the penalty nor the benefit of lagging the preconditioner is significant. We conclude that in the case of adaptive time stepping, preconditioner management is no longer needed.

2.10 Conclusions: Temporal Adaption

In this chapter we have presented a method for temporal adaption based on error control for Richards' equation. RE is a nonlinear PDE that models variably saturated flow. RE can be nonsmooth for certain, realistic physcial scenarios. The method for error control is based on linear extrapolation and was derived under the assumption that the nonlinearities in RE are smooth, yet the method works well for problems in the nonsmooth regime. The method was implemented in the three dimensional subsurface simulator, ADH. With error control in place, ADH is able to solve a wider range of problems more accurately. Moreover, replacing the heuristic method for changing the time step with one based on error control has improved the performance of the linear and nonlinear solvers in ADH. We also performed a study to see if updating the coarse mesh component of the preconditioners based on the error estimator would improve run times. We found that with the error control mechanism in place, that time dependent or heuristic methods of lagging the preconditioner were no longer needed, as the convergence of the linear solver has improved. In Chapter 3, we investigate the numerical approach taken in ADH to understand the effects of nonsmooth nonlinearites on the integrator and nonlinear solver.

Chapter 3

Non–Lipschitz Nonlinearities

3.1 Introduction

We have proposed a first order method for temporal error control for the pressure head form of Richards' equation in Chapter 2. The method uses an approximation to the second derivative to estimate the local truncation error and compute appropriate time step to control the error. Recall the nonlinearities given in (2.19) and (2.20) respectively;

$$S_a(\psi) = S_r + \frac{(1 - S_r)}{[1 + (\alpha|\psi|)^n]^m}, \psi < 0,$$

$$k_r(\psi) = \frac{[1 - (\alpha|\psi|)^{n-1}[1 + (\alpha|\psi|)^n]^{-m}]^2}{[1 + (\alpha|\psi|)^n]^{m/2}}, \psi < 0,$$

with

$$S_a(\psi) = 1 = k_r(\psi), \psi \geq 0,$$

and $m = 1 - 1/n$. If $1 < n < 2$ then $S_a(\psi)$ and $k_r(\psi)$ are not differentiable at $\psi = 0$. The lack of smoothness will affect the nonlinear solver and the error control mechanism for $1 < n < 2$, [31].

The nonlinearities in Richards' equation for this work have been approximated with a piecewise linear spline. In [31] the authors performed numerical experiments to study the effects of replacing the nonlinearities in RE with spline approximations and compared the performance of direct evaluation of (2.19) and (2.20) to using piecewise linear splines, log linear splines, cubic splines, and Hermite splines. The authors found that Hermite splines worked best when high accuracy is required; however linear splines are competetive when low accuracy is required, which is the case with ADH. Moreover, in ADH, a finite difference Jacobian is used within an inexact–Newton method. In this chapter we address the effects of the spline and approximate Jacobian on the integrator and the nonlinear solver.

In §3.2 we reformulate our method for error control on a smooth problem to show the dependence of the error estimate on the Lipschitz constant. In §3.3 we review concepts from the nonsmooth analysis literature which we use to analyze the numerical approach taken in ADH. We study the use of a centered difference Jacobian in the context of generalized Jacobians. Throughout this chapter we consider a nonsmooth model problem from the reaction–diffusion literature. The analysis on the model problem gives insight as to why the nonlinear solver in ADH performs well. We relate the results for the model problem to one-dimensional Richards' equation and we conclude in §3.4.

3.2 Error Control Reformulation

Consider again the scalar model problem

$$y'(t) = f(t, y), t^0 < t^f, y(t^0) = y_0.$$

Let $y^k = y(t^k)$ and $\Delta t_k = t^k - t^{k-1}$. Suppose that f is Lipschitz continuous, and hence y is Lipschitz continuous, with corresponding Lipschitz constants L_f and L_y. The fundamental theorem of calculus states that

$$\begin{aligned} y^k &= y^{k-1} + \int_{t^{k-1}}^{t^k} f(y(\tau))d\tau \\ &= y^{k-1} + \Delta t_k f(y^k) + \int_{t^{k-1}}^{t^k} [f(y(\tau)) - f(y^k)]d\tau. \end{aligned}$$

We can bound the last term on the right since

$$\int_{t^{k-1}}^{t^k} |f(y(\tau)) - f(y^k)|d\tau \leq L_y L_f \Delta t_k^2/2.$$

Let $L_y L_f \Delta t_k^2/2 = C_k \Delta t_k^2/2$. The subscript k is to indicate that C_k does depend on time however, we assume C_k doesn't vary significantly with time. Let y_{BE}^k denote the backward Euler approximation to y^k. The error in the backward Euler approximation is $C_k \Delta t_k^2/2$.

Recall the predictor used in the error control method presented in §2.7 was given by $y^k_{PRE} = y^{k-1} + \Delta t_k y'^{k-1}_{PRE}$, where $y'^{k-1}_{PRE} = (y^{k-1} - y^k)/\Delta t_{k-1}$ is a forward difference approximation to $y'(t^{k-1})$. We have that

$$\begin{aligned} y^k_{PRE} &= y^{k-1} + \Delta t_k \left(\tfrac{y^{k-1}-y^{k-2}}{\Delta t_{k-1}}\right) \\ &= y^{k-1} + \int_{t^{k-1}}^{t^k}\int_{t^{k-2}}^{t^{k-1}} \tfrac{f(y(\tau))}{\Delta t_{k-1}} d\tau \\ &= y^{k-1} + \Delta t_k f(y^{k-1}) + \int_{t^{k-1}}^{t^k}\int_{t^{k-2}}^{t^{k-1}} \left[\tfrac{f(y(\tau))-f(y^{k-1})}{\Delta t_k - 1}\right] d\tau. \end{aligned}$$

We can bound the last term on the right with

$$\int_{t^{k-1}}^{t^k}\int_{t^{k-2}}^{t^{k-1}} \left|\frac{f(y(\tau)) - f(y^{k-1})}{\Delta t_{k-1}}\right| d\tau \le \frac{\Delta t_k \Delta t_{k-1}}{2} C_k.$$

Thus the difference in the backward Euler approximation and the predictor is

$$\begin{aligned} |y^k_{PRE} - y^k_{BE}| &= |y^{k-1} + \Delta t_k f(y^{k-1}) + \tfrac{\Delta t_k \Delta t_{k-1}}{2} C_k - y^{k-1} - \Delta t_k f(y^k)| \\ &= |\Delta t_k (f(y^{k-1}) - f(y^k)) - \tfrac{\Delta t_k \Delta t_{k-1}}{2} C_k| \\ &\le |\Delta t_k^2 C_k - \tfrac{\Delta t_k \Delta t_{k-1}}{2} C_k| \\ &= |\Delta t_k^2 - \tfrac{\Delta t_k \Delta t_{k-1}}{2}||C_k| \end{aligned}$$

We have shown that, for Lipschitz continuous problems, we can view the method for error control as a function of the Lipschitz constants of y and f. In the case of Richards' equation the nonlinearities are not Lipschitz continuous if $1 < n <$

2, however we replace the nonlinearities with a piecewise linear spline. The spline approximation is still nonsmooth but is at least locally Lipschitz continuous. We consider a one-dimensional model from the reaction-diffusion literature [13], [6],[7] and show how the new locally-Lipschitz problem is discretized and corresponding Jacobian entries are bounded. This in turn means that the error estimate will be well defined within the temporal adaption scheme.

3.2.1 1-D Example

Consider $\Omega = [-1, 1]$ and the model problem,

$$\begin{aligned} u_t &= -u_{xx} + \lambda f(u) && x \in \Omega, \\ u &= 1 && x \in \partial\Omega, \\ u(\cdot, 0) &= u_0 && \text{where} \end{aligned} \tag{3.1}$$

$$f(u) = \begin{cases} u^p & u \geq 0 \\ 0 & u < 0 \end{cases} \tag{3.2}$$

and $p \in (0, 1)$ and $\lambda \in \mathbb{R}$. The above model is used to describe an irreversible reaction. The unknown, u represents the concentration of the reactant and at steady-state, λ is a measure of the ratio of reaction to diffusion rates and $f(u)$ is a measure of the reaction rate. We consider solving

$$F(u) = u_t + u_{xx} - \lambda f(u) = 0.$$

Note that $F(u)$ is not Lipschitz continuous due to (3.2).

This steady state model has been widely studied in the literature. Several numerical approaches for solving the steady state (3.1) have been proposed in which (3.2) is replaced with a smooth, Lipschitz function [6], [7]. The authors prove error bounds for a finite element approach that depends on the smoothing parameters used for (3.2).

In [13], the author approximates (3.1) with a locally–Lipschitz problem and views the Jacobian of the new problem as a Clarke-generalized Jacobian, which we define in §3.3. Using the tools from nonsmooth analysis and a smoothing method, the authors prove superlinear convergence for Newton's method applied to the new problem. We will also fit our analysis into the framework from the nonsmooth literature.

We consider a piecewise linear approximation, $f_s(u)$, to $f(u)$ given by (3.2) for $u \geq 0$. We then solve the locally–Lipschitz problem

$$F_s(u) = u_t + u_{xx} - \lambda f_s(u) = 0.$$

Consider $I = [0, \bar{u}]$ divided into N subintervals of length Δ such that $u_0 = 0$, $u_N = (\bar{u})$,

$I_l = [u_l, u_{l+1}] = [l\Delta, (l+1)\Delta]$, and $I = \cup_{l=0}^{N-1} I_l$. We define $f_s(u)$ by

$$
\begin{aligned}
f_s(u) &= \begin{cases} \frac{u_{l+1}^p - u_l^p}{\Delta}(u - u_l) + u_l^p, & u \in [u_l, u_{l+1}], l = 0 : N-1 \\ u_l^p, & u = u_l, l = 0 : N \end{cases} \\
&= \begin{cases} \frac{(l+1)^p \Delta^p - l^p \Delta^p}{\Delta}(u - l\Delta) + l^p \Delta^p, & u \in [u_l, u_{l+1}], l = 0 : N-1 \\ u_l^p, & u = u_l, l = 0 : N \end{cases} \\
&= \begin{cases} ((l+1)^p - l^p)\Delta^{p-1}(u - l\Delta) + l^p \Delta^p, & u \in [u_l, u_{l+1}], l = 0 : N-1 \\ u_l^p, & u = u_l, l = 0 : N. \end{cases}
\end{aligned}
$$

So we have that

$$f_s'(u) = ((l+1)^p - l^p)\Delta^{p-1} \leq \Delta^{p-1}.$$

If we discretize $F_s(u)$ in space, with finite elements for example, and in time with backward Euler, we get

$$F_s(u^k) = u^k - u^{k-1} + \Delta t_k A u^k + \Delta t_k \lambda f_s(u^k) = 0, \tag{3.3}$$

Where A is the linear operator for the discretized Laplacian term. The corresponding Jacobian is

$$F_s'(u^k) = I + \Delta t_k A + \Delta t_k \lambda L(u^k), \tag{3.4}$$

where L is a diagonal matrix with diagonal elements,

$$L_{ii}(u^k) = ((l+1)^p - l^p)\Delta^{p-1} \text{ for } u_l^k \in [u_l, u_{l+1}].$$

Note that although the spline approximation will keep C_k bounded, we are now solving an approximation to the original nonlinear problem. In [31] the authors highlight the fact that even if $F(u^*) = 0$ we still have that $||F_s(u^*)||_\infty = \mathcal{O}(\Delta^{p-1})$.

3.3 Nonsmooth Analysis

In §3.2 we replaced the non-Lipschitz problem with a locally-Lipschitz one by using a piecewise linear spline approximation to the nonlinearity in the model problem. However, a piecewise linear spline, although being locally Lipschitz continuous, still lacks the smoothness required for convergence of Newton's method. In the remainder of this chapter we consider theory from the nonsmooth analysis literature to study the use of a finite difference approximation to the Jacobian for the model problem. The underlying idea is that the approach taken in ADH for Richards' equation also fits into this framework.

3.3.1 Background

Nonsmooth equations appear naturally in applied mathematics, optimization, and as we have shown, in applications from hydrology. Nonsmooth equations are challenging since most classical methods assume smoothness for convergence theory. Work has been done over the last several decades to extend the existing ideas for smooth equations to nonsmooth equations. In [16] the concept of a generalized gradient and generalized Jacobian was introduced for problems that are nonsmooth. In [34] the authors used ideas presented in [16] to prove superlinear convergence of Newton's method for nonsmooth equations when a generalized Jacobian is used. In [34] the authors apply a concept called *s*emismoothness, originally introduced for functionals in [30] to prove convergence.

In this section, we define the generalized gradient, the generalized Jacobian, and semismooth and provide examples. We then state the convergence theory for nonsmooth Newton's method and the extension to nonsmooth inexact-Newton's method that was proposed in [27]. The reader should keep in mind that we intend to fit the reformulation of the model problem into the framework provided here on nonsmooth analysis.

Generalized Gradients and Jacobians

As in [16], we first consider $F : X \subset \mathbb{R}^N \to \mathbb{R}$ where X is a Banach space. We assume F is locally Lipschitz continuous near x, that is F satisfies

$$|F(y) - F(y')| \leq K||y - y'||,$$

and y, y' are in a neighborhood of x. Rademacher's theorem, [16], states that if F is locally Lipschitz continuous then F is differentiable almost everywhere. Let D_F be the set of points where F is differentiable and $< \cdot, \cdot >$ denote the inner product in X. The *generalized-gradient* of F at x is given by

$$\partial F(x) = conv\{\lim_{n\to\infty} \nabla F(x_n); x_n \in D_F, x_n \to x\}, \tag{3.5}$$

where *conv* denotes the convex hull. It was shown in [16] that $\partial F(x)$ is a nonempty, convex, compact set. In [16], they prove that a mean-value theorem holds in the context of generalized gradients, as given in the following theorem.

Theorem 3.1. *Let y and z be in a convex subset of $\mathbb{R}^N$ where F is locally Lipschitz near x. Then there exists $\lambda \in (0, 1)$ and $V \in \partial F(y + \lambda(z - y))$ such that $F(z) - F(y) =< V, y - z >$.*

In [16], the authors extend the concept of a generalized gradient to $F : \mathbb{R}^N \to \mathbb{R}^N$

Example: The absolute value function
To get a better understanding of the definition of the generalized Jacobian, consider the following example. Let $F : \mathbb{R} \to \mathbb{R}$ with $F(x) = |x|$.
F is Lipschitz continuous but not differentiable at $x = 0$. We have that

$$\partial F(x) = \begin{cases} 1 & x > 0 \\ \delta \in [-1, 1] & x = 0 \\ -1 & x < 0 \end{cases}$$

to define the generalized Jacobian,

$$\partial F(x) \equiv conv\{\lim_{n\to\infty} F'(x_n) : x_n \in D_F, x_n \to x\}. \tag{3.6}$$

Moreover, a mean-value theorem still holds. That is, if y and z are any points in $\mathbb{R}^N$ then

$$F(y) - F(z) \in conv\partial F([y, z])(y - z),$$

the right side as as the convex hull of points of the form $V(u)$ where $u \in [y, z]$ and $V \in \partial F(u)$.

In [34] the authors prove the following proposition about generalized Jacobians which is used to show convergence for nonsmooth Newton's method.

Proposition 3.1. *Suppose $F : \mathbb{R}^N \to \mathbb{R}^N$ is locally Lipschitz continuous. If all $V \in \partial F(x)$ are nonsingular, then there is a neighborhood $N(x)$ of x and a constant*

C such that for any $y \in N(x)$ and any $V \in \partial F(y)$, V is nonsingular and

$$||V^{-1}|| \leq C.$$

Semismoothness

In [30], the authors introduce the concept of semismoothness. $F : \mathbb{R}^N \to \mathbb{R}$ is called *semismooth* at x if

- F is locally Lipschitz near x, and
- $\forall w \in \mathbb{R}^N, \{t_n\} \downarrow 0, \{d_n\} \subset \mathbb{R}^N$, and $\{V_n\} \in \mathbb{R}^N$ such that $\{d_n/t_n\} \downarrow 0 \in \mathbb{R}^N$ and $\{V_n\} \in \partial F(x+t_n w+d_n)$ the sequence $\{< V_n, w >\}$ has exactly one accumulation point.

They go on to show that if F is semismooth at x then F is directionally differentiable with directional derivative $F'(x;w)$ and $F'(x;w) = \lim_{n\to\infty} < V_n, w >$ where V_n is as in the above definition.

In [34], the authors extend the concept of semismoothness to $F : \mathbb{R}^N \to \mathbb{R}^N$. Suppose F is locally Lipschitz near $x \in \mathbb{R}^N$, then if for any $w \in \mathbb{R}^n$, the limit

$$\lim_{V \in \partial F(x+tw'), w' \to w, t \downarrow 0} \{Vw'\} \tag{3.7}$$

exists, we say F is semismooth at x. It was shown that F is semismooth at x if all the functional components of F are semismooth [14].

Semismoothness implies that

$$\lim_{w' \to w, t \downarrow 0} \frac{F(x+tw') - F(x)}{t} = \lim_{V \in \partial F(x+tw'), w' \to w, t \downarrow 0} \{Vw'\}.$$

Moreover, in [30] the authors showed that the class of semismooth functions includes all smooth functions, convex functions, and piecewise smooth functions. In [38] the authors prove the following proposition, that is used to prove the convergence of the nonsmooth Newton-method.

Proposition 3.2. *Suppose $F : \mathbb{R}^N \to \mathbb{R}^N$ is locally Lipschitz continuous, then the following are equivalent:*

i.) F is semismooth at x

ii.) $\lim_{w \to 0, V \in \partial F(x+w)} \frac{||F(x+w) - F(x) - Vw||}{||w||} = 0.$

Nonsmooth Newton's Method

The above ideas can be extended to Newton's method by defining the Newton-like iteration,

$$x_+ = x_c - (V_c)^{-1} F(x_c), V_c \in \partial F(x_c). \tag{3.8}$$

It was shown in [38] that if F is semismooth at x^*, where $F(x^*) = 0$, and all elements in $\partial F(x^*)$ are nonsingular then this generalized Jacobian Newton method converges superlinearly. In [27], the authors extended the above theory for Newton's method to include inexact-Newton's method with generalized Jacobians. We present their theorem below.

In [27] the authors consider the $B - Differential$ of F defined by

$$\partial_B F(x) \equiv \{\lim_{n\to\infty} F'(x_n) : x_n \in D_F, x_n \to x\}, \tag{3.9}$$

which is smaller than (3.6). We state their theorem and provide a proof using the generalized Jacobian instead of the B-differential.

Theorem 3.2. *Let $F : \mathbb{R}^N \to \mathbb{R}^N$ be locally Lipschitz continuous in the open convex set $D \subseteq \mathbb{R}^N$. Assume that F is semismooth and all the elements of $\partial F(x^*)$ are nonsingular where $F(x^*) = 0, x^* \in D$. Consider the following inexact-Newton method:*

$$||V_n s_n + F(x_n)|| \leq \eta_n ||F(x_n)||, \tag{3.10}$$

where $V_n \in \partial F(x_n), s_n = x_{n+1} - x_n$, and $\{\eta_n\}$ is a bounded sequence of positive numbers. Then there exists a number η such that if $\{\eta_n\}$ is bounded by η, then there is a neighborhood of x^* such that any x_0 in this neighborhood, the inexact-Newton method (3.10) *is Q-linearly convergent to x^*. If $\eta_n \to 0$ then* (3.10) *is Q-superlinearly convergent to x^*.*

Proof. F locally Lipschitz and Proposition 3.1 imply there exist constants L, M, and $\delta_1 > 0$ such that if $||x_n - x^*|| < \delta_1$, V_n is nonsingular, $||V_n^{-1}|| \leq M$ and $||F(x_n)|| \leq L||x_n - x^*||$. By Proposition 3.2 in [38], for $\hat{\epsilon} > 0$ there is a $\delta \in (0, \delta_1)$ such that if

$x_n \in N(x^*) \equiv \{x : ||x - x^*|| \leq \delta\}$ and if $V_n \in \partial F(x)$ then

$$||F(x_n) - F(x^*) - V_n(x_n - x^*)|| \leq \hat{\epsilon}||x_n - x^*||. \tag{3.11}$$

We then have that if $x_n \in N(x^*)$ then

$$||x_{n+1} - x_n|| = ||x_n - x^* - (V_n)^{-1}F(x_n) + (V_n)^{-1}[V_n s_n + F(x_n)]||$$

$$\leq ||(V_n)^{-1}|| \, [||V_n(x_n - x^*) + F(x_n) - F(x^*) + [V_n s_n + F(x_n)]||]$$

$$\leq M \, [||F(x_n) + F(x^*) - V_n(x_n - x^*)|| + ||V_n s_n + F(x_n)||]$$

$$\leq M \, [||F(x_n) + F(x^*) - V_n(x_n - x^*)|| + ||V_n s_n + F(x_n)||]$$

$$\leq M \, [\hat{\epsilon}||x_n - x^*|| + \eta_n||F(x_n) - F(x^*)||]$$

$$\leq M(\hat{\epsilon} + \eta L)||x_n - x^*||.$$

So if $\eta_n \leq \eta < 1/LM$ and if $\hat{\epsilon} < 1/M - \eta L$ then $x_{n+1} \in N(x^*)$ and the inexact-Newton method converges Q-linearly to x^*. If $\eta_n \to 0$ then the convergence is superlinear. $\square$

3.3.2 Finite Difference Jacobians

We continue an analysis on the model problem given by (3.1) using the ideas presented in §3.3.1. We consider the centered–finite difference Jacobian for the reformulated locally Lipschitz problem, in which the non-Lipschitz nonlinearity was approximated with a piecewise smooth spline.

For the model problem, recall (3.3) and (3.4) given by

$$F_s(u^k) = u^k - u^{k-1} + \Delta t_k A u^k + \Delta t_k \lambda f_s(u^k) = 0,$$

and

$$F_s'(u^k) = I + \Delta t_k A + \Delta t_k \lambda L(u^k),$$

where L is a diagonal matrix with diagonal elements,

$$L_{ii}(u^k) = ((l+1)^p - l^p)\Delta^{p-1} \text{ for } (u^k)_i \in [u_l, u_{l+1}].$$

Here $(u^k)_i$ denotes the i^{th} coordinate of u^k. Let $\mu_l = ((l+1)^p - l^p)\Delta^{p-1}$ for $(u^k)_i \in [u_l, u_{l+1}]$.

We then have that

$$\begin{aligned}
\partial F_s(u^k) &= conv\{\lim_{n\to\infty} I + \Delta t_k A + \Delta t_k \lambda f_s'(u_n^k) : u_n^k \to u^k, u_n^k \in D_F\} \\
&= conv\{I + \Delta t_k A + \Delta t_k \lambda \lim_{n\to\infty} f_s'(u_n^k) : u_n^k \to u^k, u_n^k \in D_F\} \qquad (3.12) \\
&= \{I + \Delta t_k A + \Delta t_k \lambda \tilde{L}\}
\end{aligned}$$

where $\tilde{L}$ is a diagonal matrix with $\tilde{L}_{ii} \in [\mu_l, \mu_{l+1}]$ if $(u^k)_i \in [u_{l-1}, u_{l+1}]$.

Let

$$\nabla_\epsilon F(u)_j = \frac{F(u + \epsilon e_j) - F(u - \epsilon e_j)}{2\epsilon} \tag{3.13}$$

denote the j^{th} column of the centered difference approximation to the Jacobian of $F : \mathbb{R}^n \to \mathbb{R}^n$, e_j the j^{th} coordinate unit vector, and $\epsilon > 0 \in \mathbb{R}$. We show that if the finite difference stencil lies entirely on one side of a spline knot, then for piecewise-linear splines, $\nabla_\epsilon F_s(u) = F_s'(u)$ and if $(u)_i = u_l$ is a spline knot then $\nabla_\epsilon F_s(u) \in \partial F_s(\bar{u})$, where $(\bar{u})_i = u_l$. The interesting case is when the finite difference stencil crosses a spline knot u_l. In this case, we have that $\nabla_\epsilon F_s(u) \in \partial F_s(\bar{u})$. That is, the centered difference Jacobian is a Clarke-Generalized Jacobian for a nearby point. We present the above ideas in the following proposition.

Proposition 3.3. *Let $F_s(u)$ be given by* (3.3), *with corresponding Jacobian* (3.4) *then* $\nabla_\epsilon F_s(u) \in \partial F(\bar{u})$, *where* $||u - \bar{u}|| < \epsilon$.

Proof: Applying ∇_ϵ to $F_s(u^k)$ yields

$$\begin{aligned} \nabla_\epsilon F_s(u^k)_j &= (F(u^k + \epsilon e_j) - F(u^k - \epsilon e_j))/(2\epsilon) \\ &= e_j + \Delta t_k A e_j - \Delta t_k \lambda [f_s(u^k + \epsilon e_j) - f_s(u^k - \epsilon e_j)]/(2\epsilon). \end{aligned} \tag{3.14}$$

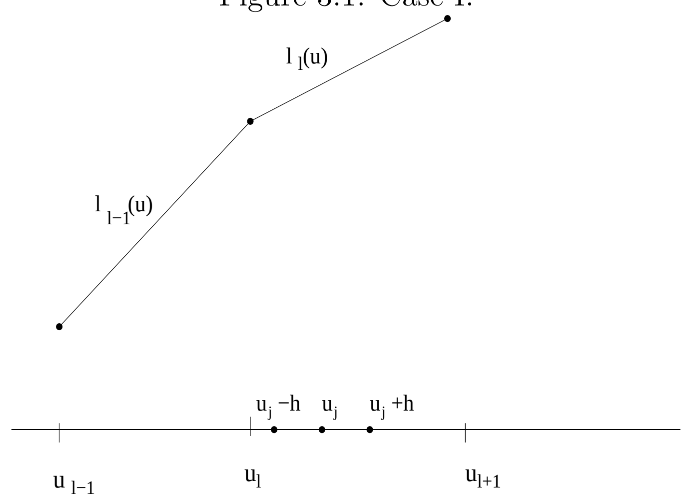
Note that the first 2 terms on the right side correspond exactly to those in (3.4) and the last term is the j^{th} column of a diagonal matrix. We consider the diagonal entries of this last term independently. Consider the piecewise linear spline approximation

$(f_s(u^k))_i$ component-wise over $[u_{l-1}, u_{l+1}]$.

We consider three cases: (I.) when the difference stencil lies completely on one side of the spline, (II.) when $(u)_i = u_l$ for some l, and (III.) when $(u)_i \neq u_l$ but $(u)_i \pm \epsilon$ crosses a spline knot.

Case I: Suppose that $(u^k)_i, (u^k)_i + \epsilon, (u^k)_i - \epsilon \in [u_l, u_{l+1}]$ and $l_l(u)$ is the line segment joining $f_s(u_l)$ to $f_s(u_{l+1})$ as in figure 3.1.

Figure 3.1: Case I.



We have

$$
\begin{aligned}
\nabla_\epsilon f_s(u)_{jj} &= ([\mu_l((u^k)_j + \epsilon - u_l) + (u_l)^p] - [\mu_l((u^k)_j - \epsilon - u_l) + (u_l)^p])/(2\epsilon) \\
&= (2\epsilon\mu_l)/(2\epsilon) \\
&= \mu_l.
\end{aligned}
$$

Thus $\nabla_\epsilon f_s(u)_{jj} = f_s'(u)_{jj}$ and $\nabla_\epsilon F_s(u^k) = F_s'(u^k)$ if the difference stencil lies entirely between two spline knots.

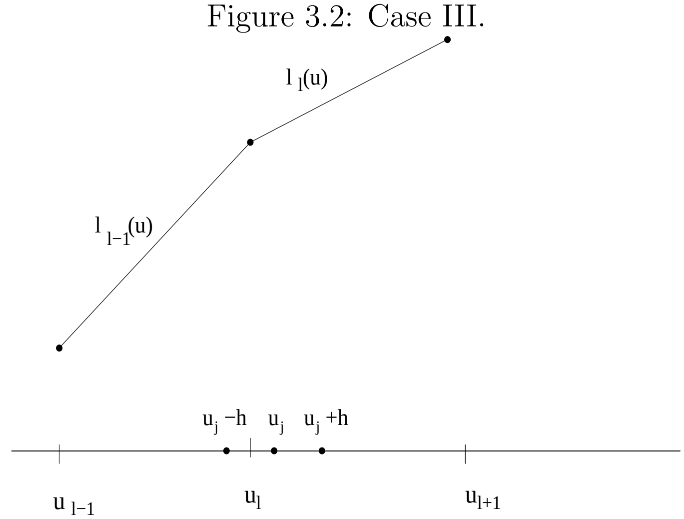
Case II. Next consider the case when $(u)_j = u_l$. We have that

$$\begin{aligned}
\nabla_\epsilon f_s(u)_{jj} &= ([\mu_l(u_l + \epsilon - u_l) + (u_l)^p] - [\mu_l(u^l - \epsilon - u_l) + (u_l)^p])/(2\epsilon) \\
&= \epsilon(\mu_l + \mu_{l+1})/(2\epsilon) \\
&= (\mu_l + \mu_{l-1})/2.
\end{aligned}$$

Clearly, $\nabla_\epsilon f_s(u)_{jj} \in [\mu_{l-1}, \mu_l]$ and so $\nabla_\epsilon f_s(u) \in \partial F_s(\bar{u})$.

Case III: Next consider the case where the centered difference stencil crosses over a spline knot as in figure 3.2. Note that, without loss of generality, we assume $\mu_{l-1} > \mu_l$.

Figure 3.2: Case III.



We have that

$$
\begin{aligned}
\nabla_\epsilon f_s(u)_{jj} &= ([\mu_l((u^k)_j + \epsilon - u_l) + (u_l)^p] - [\mu_{l-1}((u^k)_j - \epsilon - u_l) + (u_l)^p])/(2\epsilon) \\
&< \mu_{l-1}((u^k)_j + \epsilon - u_l) + (u_l)^p] - [\mu_{l-1}((u^k)_j - \epsilon - u_l) + (u_l)^p])/(2\epsilon) \\
&= (2\epsilon\mu_{l-1})/2\epsilon \\
&= \mu_{l-1}.
\end{aligned}
$$

We also have that

$$
\begin{aligned}
\nabla_\epsilon f_s(u)_{jj} &= ([\mu_l((u^k)_j + \epsilon - u_l) + (u_l)^p] - [\mu_{l-1}((u^k)_j - \epsilon - u_l) + (u_l)^p])/(2\epsilon) \\
&> \mu_l((u^k)_j + \epsilon - u_l) + (u_l)^p] - [\mu_l((u^k)_j - \epsilon - u_l) + (u_l)^p])/(2\epsilon) \\
&= (2\epsilon\mu_l)/2\epsilon \\
&= \mu_l.
\end{aligned}
$$

Thus $\nabla_\epsilon f_s(u)_{jj} \in [\mu_{l-1}, \mu_l]$ whenever the centered difference stencil crosses over a spline knot. We have that $\nabla_\epsilon F_s(u^k) \in \partial F_s(\bar{u})$.

Remark : Although we have shown that the centered difference approximation leads to a generalized–Jacobian at a nearby point with piecewise linear splines, a similar analysis will lead to a first order approximation to a nearby generalized Jacobian if any piecewise smooth spline is used or when a one sided difference is used. $\square$

Note that when we cross a spline knot, we actually obtain a generalized-Jacobian for a nearby point. For the model problem, the spline is approximating $f(u)$ given by (3.2), which is a smooth function except at $u = 0$. We should expect that when we cross a spline knot away from $u = 0$ that the slopes in the spline functions are not varying much and then $\nabla_\epsilon f_s(u)_{jj}$ would be close to $f_s'(u_j)$.

3.3.3 Convergence Theory

Theorem 3.2 states that using a generalized Jacobian leads to linear convergence of the inexact-Newton method. We have shown that using a difference operator to approximate the Jacobian leads to, at worst, a generalized-Jacobian at a nearby point. We can view the problem in the context of backward error analysis in that we consider the exact solution to a slightly nearby problem. Let $\bar{u}$ be defined by

$$(\bar{u})_i = \begin{cases} (u)_i & \text{if } |(u)_i - u_l| > \epsilon, \text{ for some } l \\ u_l & \text{if } |(u)_i - u_l| \leq \epsilon, \text{ for some } l \end{cases} \tag{3.15}$$

and suppose that at the $\nabla_\epsilon F_s(u)$ crosses over a spline knot m times. Note that $\bar{u} = u$ except at the m components where the stencil crosses a spline knot and so

$$||\bar{u} - u|| = \frac{\sqrt{m}}{\sqrt{N}}\epsilon, \tag{3.16}$$

where N is the number of spatial grid points. The inexact-Newton-like iteration that results when using a finite difference Jacobian is given by solving

$$||\bar{V}_n s_n + F(u_n)|| \leq \eta_n ||F(u_n)||, \tag{3.17}$$

where $\bar{V}_n \in \partial F(\bar{u}_n) + \mathcal{O}(\epsilon)$ and $u_{n+1} = u_n + s_n$.

Note that $\bar{V}_n$ is the generalized-Jacobian to the inexact–Newton iteration given by

$$||\bar{V}_n \bar{s}_n + F(\bar{u}_n)|| \leq \eta_n ||F(\bar{u}_n)||, \tag{3.18}$$

with $\bar{u}_{n+1} = \bar{u}_n + \bar{s}_n$. Note that $\bar{u}_n = u_n + \bar{\epsilon}_n$, for some $\bar{\epsilon}_n \in \mathbb{R}^N$ where $\max_i |(\bar{\epsilon}_n)_i| \leq \epsilon$. We have the following lemma to relate the solutions of (3.17) and (3.18).

Lemma 3.1. *Let $F : \mathbb{R}^N \to \mathbb{R}^N$ be locally Lipschitz continuous with Lipschitz constant L_F and suppose Proposition 3.1 holds at u^* where $F(u^*) = 0$. Let $\bar{u}_n$ given by* (3.15)*, $\bar{V}_n \in \partial F(\bar{u}_n)$, and u_n the current inexact–Newton iterate for* (3.17)*. If $\bar{u}_{n+1}$ is the next inexact–Newton iterate for* (3.18) *then there exists $\eta > 0$ and $K_\eta > 0$ such that*

$$||\bar{u}_{n+1} - u_{n+1}|| \leq K_\eta ||u_n - u^*|| + \mathcal{O}(\epsilon).$$

Proof. Let $r_n = -\bar{V}_n^{-1} s_n - F(u_n)$ and $\bar{r}_n = -\bar{V}_n^{-1} \bar{s}_n - F(\bar{u}_n) = -\bar{V}_n^{-1} \bar{s}_n - F(u_n + \bar{\epsilon}_n)$ be the linear residuals for (3.17) and (3.18), then

$$s_n = -\bar{V}_n^{-1}(r_n + F(u_n)) \text{ and } \bar{s}_n = -\bar{V}_n^{-1}(\bar{r}_n + F(u_n + \epsilon)).$$

Since Proposition 3.1 holds, there exists $M > 0$ such that $||\bar{V}_n^{-1}|| < M$.

$$||\bar{s}_n - s_n|| = || - \bar{V}_n^{-1}(\bar{r}_n - r_n + F(u_n + \bar{\epsilon}_n) - F(u_n))||$$

$$\leq ||\bar{V}_n^{-1}||(||\bar{r}_n - r_n|| + ||F(u_n + \bar{\epsilon}_n) - F(u_n)||)$$

$$\leq M(||\bar{r}_n|| + ||r_n|| + L_F||\bar{\epsilon}_n||$$

$$\leq M(\eta_n||F(u_n + \bar{\epsilon}_n)|| + \eta_n||F(u_n)|| + L_F\epsilon)$$

$$\leq M(\eta_n||F(u_n + \bar{\epsilon}_n) - F(u^*)|| + \eta_n||F(u_n) - F(u^*)|| + L_F\epsilon)$$

$$\leq M(\eta_n L_F(||u_n - u^*|| + ||\bar{\epsilon}_n||) + \eta_n L_F||u_n - u^*|| + L_F\epsilon)$$

$$\leq ML_F 2\eta_n||u_n - u^*|| + ML_F(\eta_n + 1)\epsilon$$

So if $\eta_n < \eta = 1/(2ML_F)$ then we have

$$||\bar{u}_{n+1} - u_{n+1}|| = ||\bar{s}_n + \bar{u}_n - s_n - u_n||$$

$$= ||\bar{s}_n - s_n + \bar{\epsilon}_n||$$

$$\leq ||\bar{s}_n - s_n|| + ||\bar{\epsilon}_n||$$

$$\leq K_\eta||u_n - u^*|| + \mathcal{O}(\epsilon),$$

where $K_\eta = 2ML_F\eta$. $\square$

We can now relate errors in (3.17) to (3.18) to analyze the convergence of (3.17). We have the following theorem.

Theorem 3.3. *Let $F : \mathbb{R}^N \to \mathbb{R}^N$ be locally Lipschitz continuous and semismooth with Lispchitz constant L_F. Suppose Proposition 3.1 holds at u^* where $F(u^*) = 0$. Let u_{n+1} be defined by the* (3.17) *then there exists $\eta > 0$ and $0 < \bar{K}_\eta < 1$ such that*

$$||u_{n+1} - u^*|| \leq \bar{K}||u_n - u^*|| + \mathcal{O}(\epsilon).$$

Proof. Let $\bar{u_n}$ be given by (3.15). Thereom 3.1 implies that the inexact–Newton iteration given by (3.18) converges linearly to u^*. Thus there exists a neighborhood $N(u^*)$ so that if $\bar{u}_n \in N(u^*)$ then there exists $\eta > 0$ and $C_\eta > 0$ such that

$$||\bar{u}_{n+1} - u^*|| \leq C_\eta||\bar{u}_n - u^*||.$$

Lemma 3.1 implies that there exists K_η, K such that

$$||u_{n+1} - \bar{u}_{n+1}|| \leq K_\eta||u_n - u^*|| + K\epsilon$$

and so we have

$$||u_{n+1} - u^*|| = ||u_{n+1} - \bar{u}_{n+1} + \bar{u}_{n+1} - u^*||$$

$$\leq ||\bar{u}_{n+1} - u^*|| + ||u_{n+1} - \bar{u}_{n+1}||$$

$$\leq C_\eta ||\bar{u}_n - u^*|| + K_\eta ||u_n - u^*|| + K\epsilon$$

$$\leq C_\eta ||u_n - u^*|| + C_\eta \epsilon + K_\eta ||u_n - u^*|| + K\epsilon$$

$$= (C_\eta + K_\eta)||u_n - u^*|| + \mathcal{O}(\epsilon)$$

$$= \bar{K}_\eta ||u_n - u^*|| + \mathcal{O}(\epsilon).$$

Here $\bar{K}_\eta = C_\eta + K_\eta$ and we choose $\eta_n < \eta < 1/(4ML_F)$ and $\hat{\epsilon} < 1/M - \eta L$ (from Theorem 3.2) so that $\bar{K}_\eta < 1$. $\square$

Theorem 3.3 implies that (3.17) will converge linearly until $||e_n|| = \mathcal{O}(\epsilon)$ and then the iteration will stagnate. Although this sounds pessimistic, $\bar{V}_n$ is only used when the difference increment crosses a spline knot. We would expect then that on fine grids or in cases where the nonsmooth nonlinearities have a large local Lipschitz constant in nontrivial regions of the domain that we would see stagnation if the nonlinear stopping tolerances are below ϵ. In the next section we provide some numerical results to verify the convergence analysis.

Numerical Results

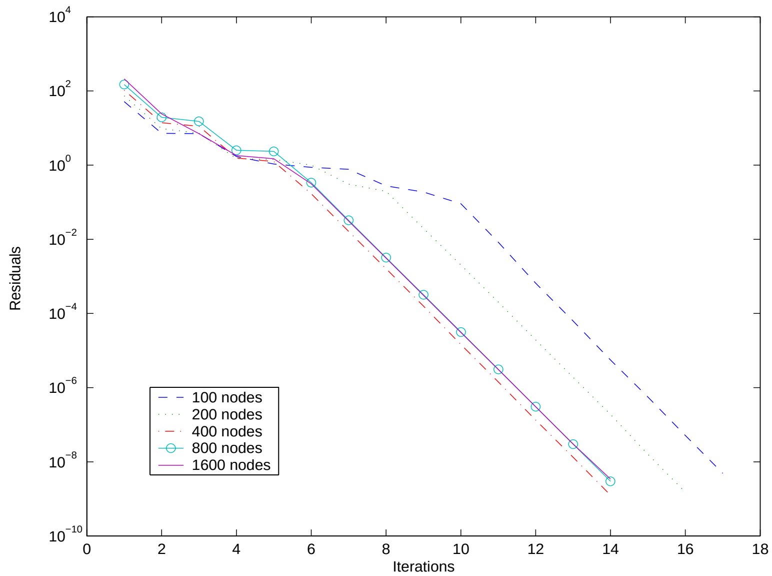
We performed a numerical study using Newton-GMRES on the steady state, one dimensional model problem for a series of spatial grids. We consider $u = x^4$ so that (3.1) implies $p = 1/2$ and $\lambda = 12$. We used 5000 spline knots for $f_s(u)$ and an initial iterate of $u_0 = x^2$. For this problem, the nonlinear iteration terminated when

$||F(u_+)||/||F(u_0)|| < 10^{-10}$. We used $\eta = 0.1$ as the forcing term for the GMRES iteration and used $\epsilon = 10^{-7}$ as the difference increment size. Table 3.1 shows the number of nonlinear iterations (NL) for Newton-GMRES to converge and the nonlinear residual for the last three iterations, NL-2, NL-1, and NL. Figure 3.3 shows the reduction in the residuals over the iterations for each grid. We see local linear convergence and for this problem we did not see stagnation of the iteration although the stopping criteria is well below the difference increment. Note however that for this problem the only place where (3.2) is nonsmooth is at $x = 0$.

Table 3.1: Iteration History

No. Nodes	NL	$\|\|F(u_{NL-2})\|\|$	$\|\|F(u_{NL-1})\|\|$	$\|\|F(u_{NL})\|\|$
100	17	5.5489e-07	5.1614e-08	4.9082e-09
200	16	1.8938e-07	1.5917e-08	1.5317e-09
400	14	1.3333e-07	1.3241e-08	1.2620e-09
800	14	3.0673e-07	2.9931e-08	2.9973e-09
1600	14	3.0499e-07	3.0229e-08	3.4375e-09

Figure 3.3: Reduction in Residuals



Increasing the value of λ yields a solution with more than one node having a value of zero, which in turn means (3.2) is nonsmooth for more values of u. Figure 3.4 shows a plot of u using $N = 50$ nodes with $\lambda = 120$ and $p = 0.5$. We compared the performance of Newton-GMRES and two values of ϵ. For both simulations we used a nonlinear tolerances of 10^{-9}. For one simulation we let $\epsilon = 10^{-7}$ and for the other we let $\epsilon = 10^{-10}$. We allowed for a maximum of 200 Newton iterations. Figure 3.5 shows the iteration history. For the larger value of ϵ the iteration stagnates when $||F(u_n)|| = \mathcal{O}(\epsilon)$ as the backward error analysis predicts.

Figure 3.4: u for $\lambda = 120$

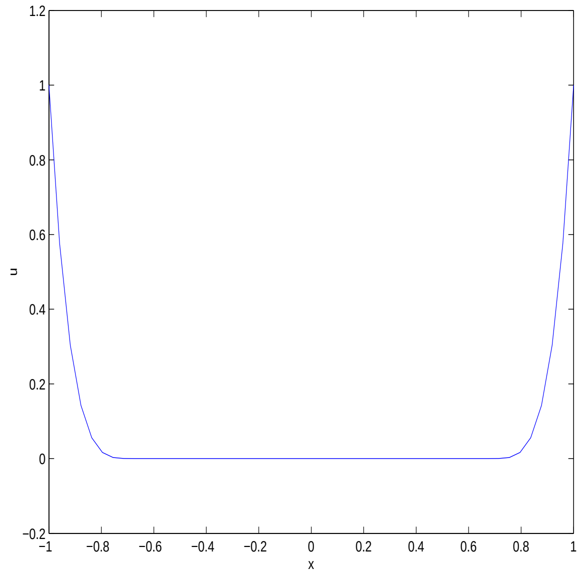
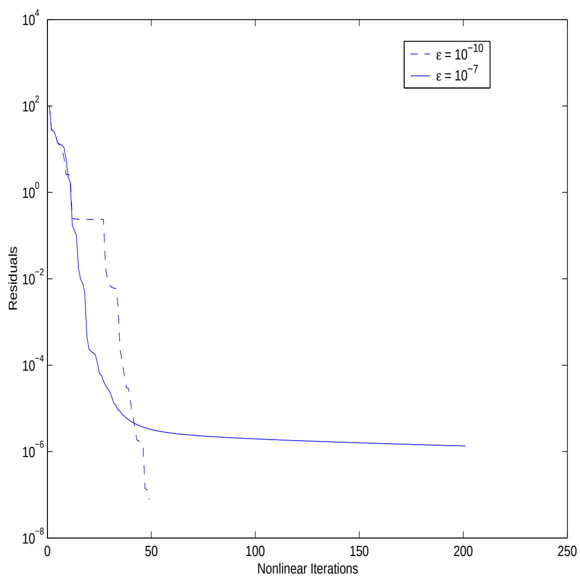


Figure 3.5: Comparing Values of ϵ



3.3.4 1-D Richards' Equation

We extend the ideas from §3.3.1 and results in §3.3.2 to study the one-dimensional Richards' Equation. Recall the one–dimension Richards' equation,(2.18) is given by

$$S_s S_a(\psi)\frac{\partial \psi}{\partial t} + \eta \frac{\partial S_a}{\partial t} = \frac{\partial}{\partial z}\left[K_s k_r(\psi)\left(1 + \frac{\partial \psi}{\partial z}\right)\right].$$

We denote the residual form of RE as

$$RE(\psi) = S_s S_a(\psi)\frac{\partial \psi}{\partial t} + \eta \frac{\partial S_a}{\partial t} - \frac{\partial}{\partial z}\left[K_s kr(\psi)\left(1 + \frac{\partial \psi}{\partial z}\right)\right] = 0 \tag{3.19}$$

Let D_F denote the set of points where $RE(\psi)$ is differentiable. We can show that using a centered difference operator on the locally Lipschitz reformulation yields the same results as Proposition 3.3. In the rest of this section, we will compute the Jacobian of $RE(\psi)$ for $\psi \in D_F$ and show that $\nabla_\epsilon RE(\psi)$ is a first order generalized-Jacobian of a nearby point if the difference stencil crosses a spline knot.

We disctretize (3.19) with finite elements and backward Euler as in Appendix B. We have that the discretized $RE(\psi)$ restricted to an arbitrary element, $\bar{\Omega} = [z_1, z_2]$ can be expressed as

$$RE(\psi)|_{\bar{\Omega}} = F_1(\psi) + F_2(\psi) + F_3(\psi)|_{\bar{\Omega}} \in \mathbb{R}^{2\times 1}$$

with

$$F_1(\psi^k) = \frac{\Delta z}{2} \begin{bmatrix} S_s S_a(\psi_1^k)(\psi_1^k - \psi_1^{k-1}) \\ S_s S_a(\psi_2^k)(\psi_2^k - \psi_2^{k-1}) \end{bmatrix},$$

$$F_2(\psi^k) = \frac{\Delta z}{2} \begin{bmatrix} S_s S_a(\psi_1^k) - S_s S_a(\psi_1^{k-1}) \\ S_s S_a(\psi_2^k) - S_s S_a(\psi_2^{k-1}) \end{bmatrix}$$

$$F_3(\psi^k) = \Delta t \begin{bmatrix} \frac{1}{2\Delta z}(\psi_1^k - \psi_2^k)(K_s k_r(\psi_1^k) + K_s k_r(\psi_2^k)) + \frac{1}{2}(K_s k_r(\psi_1^k) + K_s k_r(\psi_2^k)) \\ \frac{1}{2\Delta z}(\psi_2^k - \psi_1^k)(K_s k_r(\psi_1^k) + K_s k_r(\psi_2^k)) + \frac{1}{2}(K_s k_r(\psi_1^k) + K_s k_r(\psi_2^k)) \end{bmatrix},$$

where ψ_1^k denotes the approximation to $\psi(t^k, z_1)$ and similarly for ψ_2^k.

After assembling the 2×2 load vectors over the entire domain, we have the fully discretized nonlinear equation $RE(\psi)$ whose i^{th} component is given by

$$\begin{aligned} RE(\psi^k)_i \quad = \quad & \Delta z[S_s S_a(\psi_i^k)(\psi_i^k - \psi_i^{k-1})] \\ + \quad & \Delta z(S_s S_a(\psi_i^k) - S_s S_a(\psi_i^{k-1})) \\ + \quad & \tfrac{\Delta t}{2\Delta z}(\psi_i^k - \psi_{i-1}^k)[K_s k_r(\psi_i^k) + K_s k_r(\psi_{i-1}^k)] \\ + \quad & \tfrac{\Delta t}{2\Delta z}(\psi_i^k - \psi_{i+1}^k)[K_s k_r(\psi_i^k) + K_s k_r(\psi_{i+1}^k)] \\ + \quad & \tfrac{\Delta t}{2}[K_s k_r(\psi_i^k) + K_s k_r(\psi_{i-1}^k)] + \tfrac{\Delta t}{2}[K_s k_r(\psi_i^k) + K_s k_r(\psi_{i+1}^k)]. \end{aligned} \tag{3.20}$$

If $\psi \in D_F$ then $RE'(\psi)$ is the tridiagonal matrix defined component-wise by

$$
\begin{aligned}
RE'(\psi)_{i,i} \quad = \quad & \Delta z[S_s S_a'(\psi_i^k)(\psi_i^k - \psi_i^{k-1}) + S_s S_a(\psi_i^k)] \\
+ \quad & \Delta z S_s S_a'(\psi_i^k) \\
+ \quad & \tfrac{\Delta t}{2\Delta z}\{[K_s k_r(\psi_i^k) + K_s k_r(\psi_{i-1}^k)] + (\psi_i^k - \psi_{i-1}^k) K_s k_r'(\psi_i^k)\} \\
+ \quad & \tfrac{\Delta t}{2\Delta z}\{[K_s k_r(\psi_i^k) + K_s k_r(\psi_{i+1}^k)] + (\psi_i^k - \psi_{i+1}^k) K_s k_r'(\psi_i^k)\} \\
+ \quad & \tfrac{\Delta t}{2} K_s k_r'(\psi_i^k)
\end{aligned}
$$

$$
\begin{aligned}
RE'(\psi)_{i,i-1} \quad = \quad & \tfrac{\Delta t}{2\Delta z}\{-[K_s k_r(\psi_i^k) + K_s k_r(\psi_{i-1}^k)] + (\psi_i^k - \psi_{i-1}^k) g'(\psi_{i-1}^k)\} \\
+ \quad & \tfrac{\Delta t}{2} K_s k_r'(\psi_{i-1}^k)
\end{aligned}
$$

$$
\begin{aligned}
RE'(\psi)_{i,i+1} \quad = \quad & \tfrac{\Delta t}{2\Delta z}\{-[K_s k_r(\psi_i^k) + K_s k_r(\psi_{i+1}^k)] + (\psi_i^k - \psi_{i+1}^k) g'(\psi_{i+1}^k)\} \\
+ \quad & \tfrac{\Delta t}{2} K_s k_r'(\psi_{i+1}^k)
\end{aligned}
$$

We consider the centered finite difference operator given by (3.13) whose i^{th} column is given by

$$
\nabla_\epsilon RE(\psi)_i = \frac{S_s S_a(\psi + \epsilon e_i) - S_s S_a(\psi - \epsilon e_i)}{2\epsilon}
$$

We then have that the subdiagonal entries of $\nabla_\epsilon RE(\psi)$ are given by

$$
\begin{aligned}
\nabla_\epsilon RE(\psi)_{i,i-1} \quad = \quad & \tfrac{\Delta t}{4\Delta z\epsilon}\{(\psi_i^k - \psi_{i-1} - \epsilon)[K_s k_r(\psi_i^k) + K_s k_r(\psi_{i-1}^k)] \\
- \quad & (\psi_i^k - \psi_{i-1}^k + \epsilon)[K_s k_r(\psi_i^k) + K_s k_r(\psi_{i-1}^k)] \\
+ \quad & \tfrac{\Delta t}{4\epsilon}[K_s k_r(\psi_{i-1}^k + \epsilon) - K_s k_r(\psi_{i-1}^k)] \\
= \quad & \tfrac{\Delta t}{2\Delta z}\left\{(\psi_i^k - \psi_{i-1}^k)\tfrac{K_s k_r(\psi_{i-1}^k+\epsilon) - K_s k_r(\psi_{i-1}^k-\epsilon)}{2\epsilon}\right. \\
- \quad & \left.\tfrac{K_s k_r(\psi_{i-1}^k+\epsilon) + K_s k_r(\psi_{i-1}-\epsilon)}{2}\right\} \\
+ \quad & \tfrac{\Delta t}{2}\left[\tfrac{K_s k_r(\psi_{i-1}^k+\epsilon) - K_s k_r(\psi_{i-1}^k-\epsilon)}{2\epsilon}\right].
\end{aligned}
$$

The supdiagonal entries are similar to those of the subdiagonal with $i-1$ replaced by $i+1$.

The diagonal entries are given by

$$
\begin{aligned}
\nabla_\epsilon RE(\psi)_{i,i} \quad = \quad & \tfrac{\Delta z}{2\epsilon}\left[S_sS_a(\psi_i^k+\epsilon)(\psi_i^k+\epsilon-\psi_i^{k-1}) - S_sS_a(\psi_i^k-\epsilon)(\psi_i^k-\epsilon-\psi_i^{k-1})\right] \\
+ \quad & \tfrac{\Delta z}{2\epsilon}\left[S_sS_a(\psi_i^k+\epsilon) - S_sS_a(\psi_i^k-\epsilon)\right] \\
+ \quad & \tfrac{\Delta t}{4\Delta z\epsilon}\left\{(\psi_i^k+\epsilon-\psi_i^{k-1})[K_sk_r(\psi_i^k+\epsilon)+K_sk_r(\psi_{i-1}^k)] - \right. \\
& (\psi_i^k-\epsilon-\psi_i^{k-1})[K_sk_r(\psi_i^k-\epsilon)+K_sk_r(\psi_{i-1}^k)] + \\
& (\psi_i^k+\epsilon-\psi_i^{k+1})[K_sk_r(\psi_i^k+\epsilon)+K_sk_r(\psi_{i+1}^k)] - \\
& \left.(\psi_i^k-\epsilon-\psi_i^{k+1})[K_sk_r(\psi_i^k-\epsilon)+K_sk_r(\psi_{i+1}^k)]\right\} \\
+ \quad & \Delta t\tfrac{K_sk_r(\psi_i^k+\epsilon)+K_sk_r(\psi_i^k-\epsilon)}{2} \\
= \quad & \Delta z\left[\tfrac{S_sS_a(\psi_i^k+\epsilon)-S_sS_a(\psi_i^k-\epsilon)}{2\epsilon}(\psi_i^k-\psi_i^{k-1}) + \tfrac{S_sS_a(\psi_i^k+\epsilon)+S_sS_a(\psi_i^k-\epsilon)}{2}\right] \\
+ \quad & \tfrac{\Delta z}{2\epsilon}\left[S_sS_a(\psi_i^k+\epsilon) - S_sS_a(\psi_i^k-\epsilon)\right] \\
+ \quad & \tfrac{\Delta t}{2\Delta z}\left\{\left[\tfrac{K_sk_r(\psi_i+\epsilon)+K_sk_r(\psi_i-\epsilon)}{2} + K_sk_r(\psi_{i-1})\right] + (\psi_i-\psi_{i-1})\tfrac{K_sk_r(\psi_i+\epsilon)-K_sk_r(\psi_i-\epsilon)}{2\epsilon}\right. \\
+ \quad & \left.\left[\tfrac{K_sk_r(\psi_i+\epsilon)+K_sk_r(\psi_i-\epsilon)}{2} + K_sk_r(\psi_{i+1})\right] + (\psi_i-\psi_{i+1})\tfrac{K_sk_r(\psi_i+\epsilon)-K_sk_r(\psi_i-\epsilon)}{2\epsilon}\right\} \\
+ \quad & \Delta t\tfrac{K_sk_r(\psi_{i-1}^k+\epsilon)+K_sk_r(\psi_{i-1}^k-\epsilon)}{2}.
\end{aligned}
$$

If we replace $S_a(\psi)$ and $k_r(\psi)$ with a a spline approximation, we can view the above discretization in terms of Proposition 3.3 for $\psi \notin D_F$. If $\psi_i \pm \epsilon$ lies entirely on one side of a spline knot then $\nabla_\epsilon RE(\psi) = RE'(\psi)$ as was the case with the model problem. If $\psi_i \pm \epsilon$ crosses a spline knot then using Proposition 3.3 and the fact that

$$\frac{S_sS_a(\psi+\epsilon)+S_sS_a(\psi-\epsilon)}{2} = S_sS_a(\psi) + \mathcal{O}(\epsilon), \tag{3.21}$$

we have that $\nabla_\epsilon RE(\psi) \in \partial RE(\bar{\psi}) + \mathcal{O}(\epsilon)$. Here the constant in the $\mathcal{O}(\epsilon)$ term is the difference of the local Lipschitz constants of the spline across the knot.

Richards' equation differs from the model problem in that the region of nonsmoothness, which occurs in a neighborhood of $\psi = 0$, is small. In one spatial dimension, this would be small number of nodes near $\psi = 0$, meaning that the constant in (3.16) would be small. The constant in the (3.21) can still be quite large as well as the terms in Theorem 3.3 where ϵ is multiplied by the local Lipschitz constant. An advantage of solving Richards' equation with error control in place is that the spatial terms are multiplied by Δt. When the Lipschitz constant is large, the time step will be reduced based on the error estimate and this will improve the performace of the nonlinear solver. To study the effect of the error control mechanism on the performance of the nonlinear solver, we ran some numerical experiments for one-dimension Richards' equation.

Numerical Results

We consider the clay infiltration problem with 400 nodes as described in §2.9 which results in nonsmooth nonlinearites. We used a nonlinear tolerance of 10^{-6}, $\epsilon = 10^{-7}$, $TOL <= 0.01$ and allowed for 8 nonlinear iterations before reducing the time step. The error control mechanism will attempt to grow the time step to the maximum allowable step. We considered a maximum allowable time step of 10 days, 5 days, and 2 days. We observed that for $max\Delta t = 10$ that the nonlinear iteration would repeatedly stagnate with $||RE(\psi)|| = \mathcal{O}(10^{-5})$ and the time step would need to be cut. Reducing the maximum allowable time step resulted in less nonlinear iterations but slightly more time steps to reach the final solution however the run time was almost twice as fast. Table 3.2 shows the iteration history. We can conclude that the temporal adaption scheme not only provides more accurate solutions, but it also improves the performance of the nonlinear solver.

Table 3.2: Iteration History: 400 Nodes

$Max\Delta t$	NL	NL cuts	No. ts	run time (s)
10	887	68	166	155
5	475	23	180	85
2	370	2	335	93

3.4 Conclusions: Nonsmooth Analysis

In this chapter, we have analyzed the numerical approach taken in ADH to resolve the nonsmooth nonlinearities in Richards' Equation by considering a non–Lipschitz model problem from the reaction diffusion literature. The nonlinearities are replaced with a piecewise linear spline, making the problem locally Lipschitz continuous, but still nonsmooth. The Jacobian for the reformulated problem is approximated with centered differences. We showed that the centered difference Jacobian for the reformulated model problem is a generalized–Jacobian at a nearby point. Our analysis shows that convergence of the inexact–Newton method with the nearby generalized–Jacobian will be linear and will stagnate as the error approaches the size of the difference increment. Our numerical results for the model problem verify that Newton-GMRES converges linearly and we observe stagnation with an improper choice of tolerances and the difference increment.

We also showed that the same numerical approach when applied to Richards' equation yields a first order generalized–Jacobian at a nearby point. The solution to the nonlinear system for Richards' equation is the backward Euler approximation. Since the solution at each time step can be computed and the reformulated problem is locally Lipschitz, the method for temporal adaption based on error control is well-defined. Moreover, for a time dependent problem, such as RE, temporal adaption will improve the performance of the nonlinear solver since the terms where the Lipschitz constant can be large are multiplied by the time step.

Chapter 4

Optimization of Remediation Problems

4.1 Introduction

The focus of this chapter is optimal design for groundwater remediation problems. The simulation of groundwater flow and transport is combined with methods from optimization to design management strategies. Cleaning up a contaminated site involves numerous, expensive decisions that rely on accurate modeling of subsurface phenomena and accurate numerical methods. Solving such optimal control problems involve the following steps:

1. development of a conceptual model for the physical phenomena,

2. formulation of the governing equations and closure relations as a well-posed

problem,

3. using this formulation to develop an objective function and constraints,
4. realization of the objective function and constraints via simulation, and
5. solution of the optimization problem.

In this chapter we present a subset of test problems that were proposed in [28] to be used in the research community for benchmarking and comparing optimizers. The authors of [28] propose 1-3 from above. We describe one approach for steps 4-5 and present numerical results for a subset of the community problems using the implicit filtering algorithm.

We have two main objectives. One is to implement a subset of the problems given in [28] and provide the necessary codes, data files, and instructions to the public via a webpage. This is useful to the optimization community in that the objective function subroutine and corresponding simulator data will be readily available to try with different optimizers. This eliminates task of trying to duplicate previously published results or learning how to operate a simulator before implementing a different optimization technique. This project is beneficial to the groundwater community in that different simulators can be exchanged or the objective function reformulated to compare different flow and transport models. To meet this objective we need to explain how to implement an application, form and evaluate the corresponding objective function, and solve the optimal control problem.

The second objective of this work is to investigate the performance of deterministic sampling methods on optimal control problems in hydrology. Little is known about how to best formulate the constraints or which optimization techniques perform well. The objective functions require the solution to one or possibly two PDEs. The solutions are approximated by using a groundwater flow simulator, hence noise can be added to the objective function by error in the simulator. Moreover, the formulation of the objective function can include discontinuities. For example, the objective function can be formulated as a discontinuous function of the number of wells in the model. The optimizer should be able to handle nonsmoothness and discontinuities and sampling methods are appealing for these types of problems.

From [28], we consider two applications. The first application is is a well–field design (WD) problem and the second application is a hydraulic capture (HC) problem. The decision variables include the number of wells n, the locations of the wells $\mathbf{x}_i, i = 1...n(m)$, and the pumping rates $Q_i(m^3/s)$. We implement an objective function proposed in [28] that minimizes the cost of installing and operating wells. Constraints are imposed on pumping rates and hydraulic head.

In §4.2 we describe the models, physical domain, objective function, and constraints as presented in [28]. In §4.3 we describe the implicit filtering algorithm and the parallel implementation we used and in §4.4 we describe the groundwater flow and transport simulators used for this work. The simulators are part a graphical interface called the Groundwater Model System (GMS). In §4.5 we step through the implementation for the a well-field design application in the context of the software we used. In §4.6 and §4.7 we show numerical results for the well-field design application in two different hydrological settings and for the hydraulic capture application.

4.2 Community Problem Design

In this section we describe the models, domains, objective function, and constraints for the subset of the community problems proposed in [28] that we consider here. Although we only consider a subset of the community problems, the problems presented in [28] will be the subject of future work.

4.2.1 Domain

The spatial domain for both problems is $\Omega = [0, 1000] \times [0, 1000] \times [0, 30]$m. In [28], the authors consider five different physical domains that differ in the hydraulic conductivities, K. For this work, we consider the simple, homogeneous domain with $K = 5.01 \times 10^{-5} (m/s)$.

4.2.2 Models

We consider 2 different hydrological settings which we refer to as Model 1 and Model 2. Saturated flow in porous media can be described by

$$S_s \frac{\partial h}{\partial t} = \nabla \cdot (K \nabla h) + \bar{\mathcal{S}}, \tag{4.1}$$

where $S_s(1/m)$ is the specific storage and h is the hydraulic head. $\int_\Omega \bar{\mathcal{S}} d\Omega = \sum_{i=1}^n Q_i(\mathbf{x}_i, t)$ is a fluid source term and is one term where the decision variables enter the state equations.

For the hydraulic capture problem, a solution to the transport equation is needed to generate an initial contaminant plume. Transport can be modeled by

$$\frac{\partial(\theta^a C^\iota)}{\partial t} = \nabla \cdot (\theta^a \mathbf{D}^\iota \cdot \nabla C^\iota) - \nabla \cdot (\mathbf{q} C^\iota) + \mathcal{I}^\iota + \mathcal{R}^\iota + \mathcal{S}^\iota \tag{4.2}$$

Here C^ι is the concentration in the aqueous phase for species ι, $\mathbf{D}^\iota$ is a hydrodynamic dispersion tensor, $\mathbf{q}$ is the Darcy velocity, as defined in §3.3. $\mathcal{I}^\iota$, $\mathcal{R}^\iota$, and $\mathcal{S}^\iota$ represent interphase mass transfer, biochemical reactions, and sources of mass respectively. $\int_\Omega \mathcal{S}^\iota d\Omega = \sum_{i=1}^n Q_i(\mathbf{x}_i, t) C_i^\iota(\mathbf{x}_i, t)$ is the other term where the decision variables enter the state equations.

The dispersion tenor is expressed by

$$\mathbf{D}^\iota = D_{ij}^\iota = \delta_{ij} \alpha_t |\mathbf{v}| + (\alpha_l - \alpha_t) \frac{v_i v_j}{|\mathbf{v}|} + \delta_{ij} \tau D^{\iota *}, \tag{4.3}$$

where α_l and α_t are the longitudinal and transverse dispersivities, τ is the tortuosity, and $D^{\iota *}$ is the free liquid diffusivity of species ι.

Model 1 is for saturated, single phase flow in a confined aquifer. An aquifer is categorized as a confined aquifer if the top and bottom layers bounding the water bearing region have hydraulic conductivities of four or more orders of magnitude less than the inner layers [32]. Model 1 uses (4.1) with initial and boundary conditions given by

$$\left.\frac{\partial h}{\partial x}\right|_{x=0} = \left.\frac{\partial h}{\partial y}\right|_{y=0} = \left.\frac{\partial h}{\partial z}\right|_{z=0} = 0, t > 0 \tag{4.4}$$

$$q_z(x, y, 30, t > 0) = 1.903\text{x}10^{-8}(m/s) \tag{4.5}$$

$$h(1000, y, z, t > 0) = 50 - 0.001y(m) \tag{4.6}$$

$$h(x, 1000, z, t > 0) = 50 - 0.001x(m) \tag{4.7}$$

$$\bar{S}(x, y, z, t > 0) = 0.0(m^3/s) \tag{4.8}$$

$$h(x, y, z, 0) = h_s(m) \tag{4.9}$$

Here q_z is recharge, perhaps from rainfall, and is the Darcy flux into the domain. h_s is the steady state solution to the flow problem prior to the addition of wells.

Model 2 is similar to Model 1 but simulates saturated, single phase flow in an unconfined aquifer. An aquifer is considered unconfined if the ground surface is the upper bound on the water bearing region [32]. In this case the height of the water table can change. Model 2 uses (4.1),(4.4), (4.5),(4.8), and (4.9), however (4.5), (4.6) and (4.7) are replaced with

$$q_z(x, y, h, t > 0) = 1.903\text{x}10^{-8}(m/s) \tag{4.10}$$

$$h(1000, y, z, t > 0) = 20 - 0.001y(m) \text{ and} \tag{4.11}$$

$$h(x, 1000, z, t > 0) = 20 - 0.001x(m). \tag{4.12}$$

For model 1, the storage term is $S_s = 10^{-6}$ and for model 2, S_s represents the specific yield of the aquifer and is $S_s = 2 \times 10^{-1}$.

4.2.3 Objective Function

In [28], they consider an objective function f, that minimizes capital (installation) cost and operational cost for the wells. f depends on the number of wells (n), the location of the wells ($\mathbf{x}_i$), and the extraction/injection rates $\{Q_i\}_{i=1}^{n}(m^3/s)$. Note that

$Q_i > 0$ means water is being injected and $Q_i < 0$ means that water is being extracted. We assume that injection wells operate under gravity feed conditions. Let n^e be the number of extraction wells and n^i be the number of injection wells $n = n^e + n^i$. They define f by

$$f(n, \mathbf{x}_i, \{Q_i\}_{i=1}^n) = \underbrace{\sum_{i=1}^{n} c_0 d_i^{b0} + \sum_{i=1}^{n^e} c_1 |Q_i^m|^{b_1} (z_{gs} - h^{min})^{b_2}}_{\text{capital cost}} + \tag{4.13}$$

$$\underbrace{\int_0^{t_f} \left(\sum_{i=1}^{n^e} c_2 Q_i (h_i - z_{gs}) + \sum_{n^e+1}^{n} c_3 Q_i \right) dt}_{\text{operational cost}},$$

where b_j and c_j are exponents and coefficients of the cost model, d_i is the depth of the well, Q_i^m is the design pumping rate, z_{gs} is the ground surface elevation, h^{min} is the minimum allowable head, t_f is the final time for the operation, and h_i is the hydraulic head in well i. In the capital cost part, the first term denotes the cost to install all the wells, and the second term accounts for an additional cost for pumps for the extraction wells. In the operational cost, we have include a lift cost that applies to the extraction wells.

Table 4.1 shows the cost coefficents and exponents.

Table 4.1: Objective Function Data

data	value	units
c_0	5.5×10^3	$\$/m^{b_0}$
c_1	5.75×10^3	$\$/[(m^3/s)^{b_1} \cdot m^{b_2}]$
c_2	$2.90^{-}4$	$\$/[(m^3/s) \cdot m \cdot s]$
c_3	$1.45 \times 10^{-}4$	$\$/[(m^3/s) \cdot s$
b_0	0.3	-
b_1	0.45	$\$/[(m^3/s) \cdot m \cdot s]$
b_2	0.64	$\$/[(m^3/s) \cdot s$

4.2.4 Constraints

For the WD application we impose constraints on the well rates and head values in the wells. The constraints are given by constraints for as

$$Q_T = \sum_{i=1}^{n} Q_i \leq Q_T^{min}(m^3/s) \tag{4.14}$$

$$Q^{emax} \leq Q_i \leq Q^{imax}(m^3/s), i = 1, ..., n \tag{4.15}$$

$$h^{max} \geq h_i \geq h^{min}(m), i = 1, ..., n \tag{4.16}$$

where Q_T is the net pumping rate, Q_T^{min} is minimum allowable total extraction rate, Q^{emax} is the maximum extraction rate at any well, Q^{imax} is the maximum injection rate at any well, h^{max} is the maximum allowable head, and h^{min} is the minimum allowable head. Constraint (4.14) means that the overall effect of the wells must be extracting at least a certain amount. Also note that (4.15) depends on Q_i through the solution of the governing equations for h_i. Moreover, the values h_i are obtained by calling a groundwater flow simulator

The distance from the wells to the specified head boundary conditions is also constrained to limit the amount of water that can be drawn from outside the domain. We require the wells be at least 200 meters from the specified head boundaries. This constraint is given by

$$\mathbf{x}_i = (x_i, y_i) \in [(0, 800); (0, 800)](m) \tag{4.17}$$

We also do not allow two wells to converge to the same grid point since the simulator will implement one well pumping at the sum of the rates. This constraint enforces the minimum distance between two wells to be at least the spatial grid size.

For the HC problem, constraint (4.14) is replaced by a constraint on the net extraction rate given by

$$\sum_{i=1}^{n} Q_i \geq Q_T^{max}(m^3/s), \tag{4.18}$$

where Q_T^{max} is the maximum total extraction rate. For the hydraulic capture application, the objective is to use wells to contain a contaminant plume. In [28] the authors leave it to the user to decide on final boundary of the plume and the constraints for containment. We describe the capture constraints we implemented in §4.7. Table 4.2 contains the constrains data.

In §4.6 we provide numerical results for the well-design problem with Models 1

Table 4.2: Constraint Data

data	value	units
Q_T^{min} (4.14)	-3.2×10^{-2}	m^3/s
Q^{emax} (4.15)	-6.4×10^{-3}	m^3/s
Q^{imax} (4.15)	6.4×10^{-3}	m^3/s
h^{min} (4.16)	40 (Model 1)	m
h^{max} (4.16)	60 (Model 1)	m
h^{min} (4.16)	10 (Model 2)	m
h^{max} (4.16)	30 (Model 2)	m
Q_T^{max} (4.18)	-3.2×10^{-2}	m^3/s

and 2. In §4.7 we provide numerical results for the hydraulic capture problem which uses Model 2.

4.3 Implicit Filtering

In this section we pose the optimization problem and describe the implicit filtering algorithm. We demonstrate the algorithm on a simple one dimensional example. Finally, we describe a parallel, Fortran 77 implementation of the algorithm called IFFCO [15]. IFFCO stands for **I**mplicit **F**iltering for **C**onstrained **O**ptimization.

We pose our optimization problem as

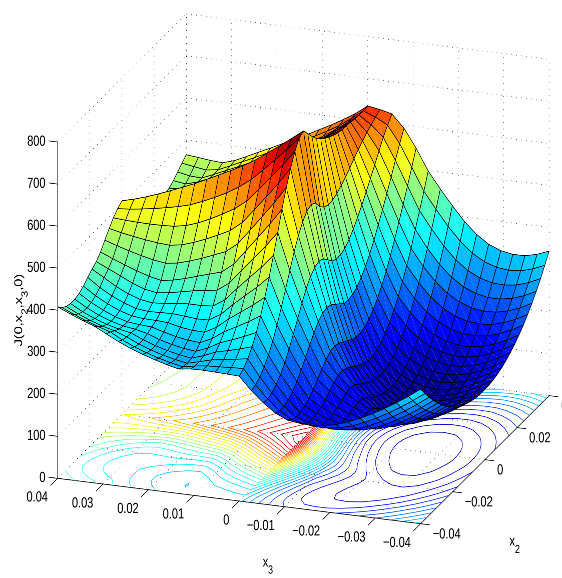
$$\min_{\Omega_{opt}} f(u) \tag{4.19}$$

where Ω_{opt} is a set of possible decision variables. For the community problems Ω_{opt}

would be a set of integers representing the possible number of wells, pumping rates, and possible well locations. To explain the theory behind the implicit filtering algorithm, we will assume that $\Omega_{opt} \subset \mathbb{R}^N$ where N is the number of decision variables. Keep in mind f would require a flow simulation and possibly a transport simulation for each function evaluation.

Since we do not have access to gradient information, a sampling method is most appealing for optimal control problems for groundwater remediation that include a call to a simulator. Recently, we applied the implicit filtering algorithm to an optimal control problem from subsurface heat transport [8] and obtained the following optimization landscape shown in Figure 4.1 for a simple quadratic objective function and box constraints on the pumping rates.

Figure 4.1: Optimization Landscape: Heat Transport Problem



Even if gradient information were available, it is clear that the optimization landscape includes several local minima and regions that are nonsmooth. Conventional gradient methods may get trapped in a local minimum or breakdown because the gradient does not exist.

4.3.1 The Implicit Filtering Algorithm

Implicit filtering is designed for functions that are small perturbations of a smooth function such as

$$f(u) = \bar{f}(u) + \phi(u), \tag{4.20}$$

where $\bar{f}(u)$ is smooth and $\phi(u)$ has small oscillations that could add several minima to $\bar{f}$. For example, ϕ could be the result of an experiment or simulation and we need not know ϕ explicitly.

In its simplest form, implicit filtering is a steepest decent method that uses a finite difference gradient and decreases the difference increment as the optimization progresses. We let $\delta u > 0$ be the difference gradient and $S(u, \delta u)$ denote our centered difference stencil. For example if $u \in \mathbb{R}^1$ then $S(u, \delta u) = \{u, u + \delta u, u - \delta u\}$. For sampling methods, we say *stencil failure* occurs if the center of the difference stencil has the smallest function value. For this example we would have $f(u) \leq f(u \pm \delta u)$. Let $\nabla_{\delta u} f(u)$ be the centered difference gradient of f. Recall that the finite difference steepest decent direction is $d = -\nabla_{\delta u} f(u)$ and the corresponding method of steepest decent updates the current iterate u_c by

$$u_+ = u_c - \lambda \nabla_{\delta u} f(u_c), \tag{4.21}$$

where λ is the steplength. For this work we use an Armijo line search for sufficient decrease given by

$$f(u + \lambda f(u)) - f(u) < -\alpha\lambda ||\nabla_{\delta u} f(u)||^2, \tag{4.22}$$

where $\alpha > 0$ is a parameter. We terminate the optimization when $||\nabla_{\delta u} f(u)|| = \mathcal{O}(\delta u)$, stencil failure occurs, when more than *pmax* iterations have taken place, or when the line search fails by taking more than *amax* backtracks. When one of these termination criterion occurs, we decrease the difference increment and start again. We express implicit filtering in algorithmic form below [26].

Algorithm imfil(u,f,$\{(\delta u)_n\}$,*amax*,*pmax*)

1. For n=0,...
2. for p=1:*pmax*

(a.) compute $f(u)$ and $\nabla_{\delta u} f(u)$, terminate if stencil faliure occurs or $||\nabla_{\delta u} f(u)|| = \mathcal{O}((\delta u)_n)$
(b.) find λ such that (4.22) holds or until *amax* backtracks have occured
(c) $u = u + \lambda \nabla_{\delta u} f(u)$

Some convergence theory has been developed for the implicit filtering algorithmand we include one such theorem.

Theorem [25] *Let f satisfy* (4.20) *and let $\nabla \bar{f}$ be Lipschitz continuous. Let* $(\delta u)_n \to 0$, $\{u_n\}$ *be the implicit filtering sequence, and* $S^n = S(u, (\delta u)_n)$. *Assume* (4.22) *holds for* *all but finitely many n. Then if*

$$\lim_{n\to\infty}((\delta u)_n + (\delta u)_n^{-1}||\phi||_{S_k}) = 0 \tag{4.23}$$

then any limit point of the sequence $\{u_n\}$ *is a critical point of* f_s.

4.3.2 Example

We demonstrate the implicit filtering algorithm on an a small perturbation of a quadratic function as in Figure 4.2. Suppose we begin with an initial iterate $u_0 = -1.25$ and $\delta u = 0.25$. The resulting centered finite difference stencil is given in Figure 4.3. The center of the stencil, $f(u_0)$ is denoted with an "x" and $f(u_0 \pm \delta u)$ is denoted with "o". Since the center of the stencil has the lowest function value the algorithm would proceed and take a steepest decent step. Suppose the next iterate is as in Figure 4.4. The lowest function value occurs on the stencil at $f(u_1 + \delta u)$ and thus stencil failure has occured. We would then reduce δu, in this case cut δu is half, and re–evaluate the stencil as in Figure 4.5. Since stencil failure would not occur in this case, the algorithm would proceed.

4.3.3 IFFCO

For this work we used a parallel, Fortran77 implementation of the implicit filtering algorithm called IFFCO. The parallel versions include MPI or PVM although a serial version of IFFCO is also available. Since optimization is governed by function eval-

Figure 4.2: Example: Implicit Filtering

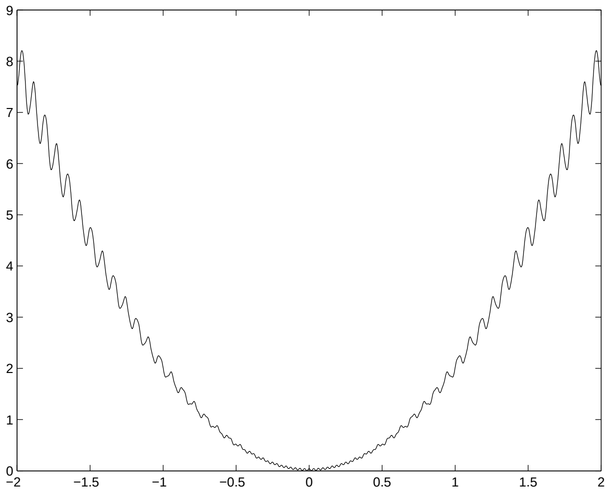


Figure 4.3: First Stencil

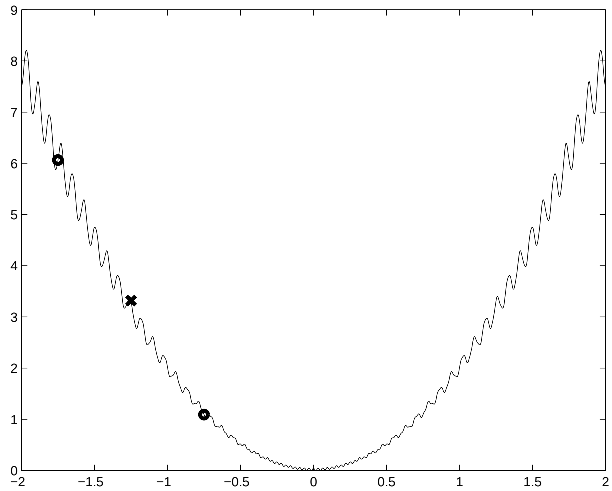


Figure 4.4: Stencil Failure

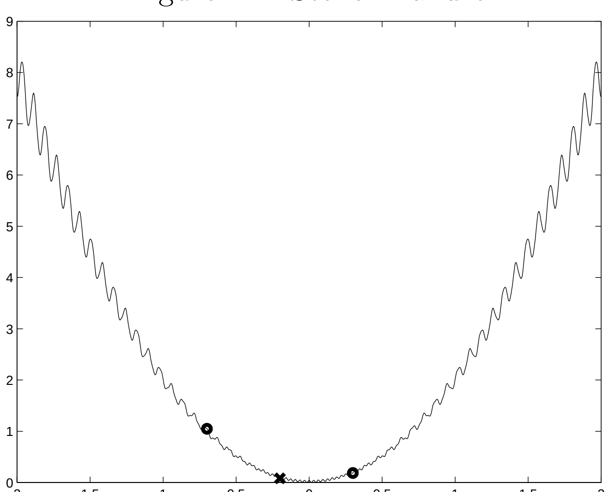
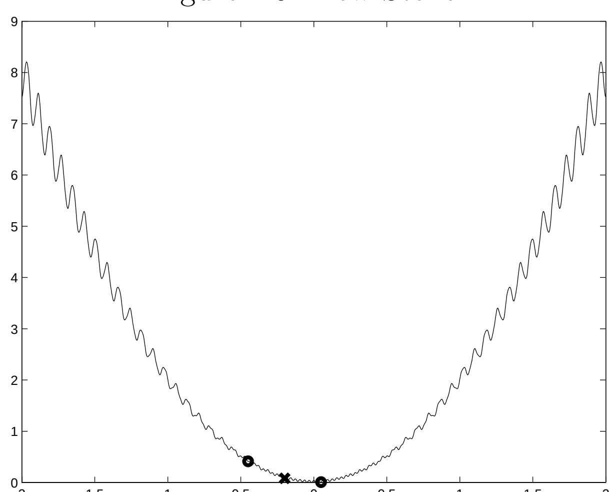


Figure 4.5: New Stencil



uations the parallel version exploits the fact that function evaluations can be done on seperate processors. For groundwater optimal control problems a function evaluation will probably involve a call to a flow and possibly a transport simulator. One simulation can take minutes to hours, hence parallelization improves the timings and is reccommended for these problems. IFFCO evaluates the function to compute the centered difference stencil and to perform the line search. A master–slave paradigm is used such that the master processor manages the algorithm and the slave processors only perform function evaluations. For IFFCO, no more than $\max(2N+1, m)$ processors whould be used, where N is the number of decision variables and m is the maximum number of allowable cutbacks.

IFFCO differs from the **imfil** algorithm above in that instead of the steepest decent algorithm, a projected quasi–Newton method is used. The hope is that if f is smooth near a minimum then convergence will be accelerated. IFFCO is designed for simple box constraints such as $\Omega_{opt} = \{u | L_i \leq u_i \leq U_i, i = 1...N\}$. In IFFCO the user has a choice of a either a BFGS or an SR1 update. The new algorithm is given below.

4.4 Software

To evaluate the objective function and constraints software is needed to simulate flow and transport. For this work we used off–the–shelf software as opposed to writing entire simulators from scratch. Since the objective of this work is to create problems

Algorithm iffco$(u, f, \{(\delta u)_n\}, amax, pmax)$

Compute $\nabla_{(\delta u)_0} f(u)$ and initialize the model Hessian B to the identity
1. for n=0...
2. for p=1...pmax
(a.) Compute $f(u)$ and terminate if stencil failure occurs or $||\nabla_{(\delta u)_n} f(u)|| = \mathcal{O}((\delta u)_n)$
(b.) Use $\nabla_{(\delta u)_n} f(u)$ and B to calculate a descent direction d
(c) Perform a line search and terminate if more than $amax$ backtracks are taken
(d.) Project the new solution onto the box constraints
(e.) $(\delta u)_n \leftarrow (\delta u)_{n+1}$
(f.) Calculate $\nabla_{(\delta u)_{n+1}} f$ and update B with either a rank–one SR1 update or a rank–two BFGS update

used for benchmarking, it makes sense to use simulators that are commonly used in the groundwater engineering community. We used MODFLOW and MT3D with the Groundwater Modeling Simulator (GMS) as an interface to generate the necessary data files. In this section we give a brief overview of the codes.

GMS

GMS was created by the Department of Defense in partnership with the Department of Energy, the U.S. EPA, Cray Research, and various academic partners and is maintained by the U.S. Army Corps of Engineers Coastal and Hydraulics lab at the Waterways Experimentation Station. GMS offers a modular interface as a computational tool for simulating groundwater and surface water flow and contaminant transport. The graphical environment allows the user to easily generate meshes and grids for site modeling, characterize the porous media, and apply geostatistics. In addition, GMS includes graphical visualization and supports a wide variety of software including MODFLOW and MT3D. For this work we used GMS 3.1 to generate the data files.

MODFLOW

MODFLOW is a modular three dimensional finite difference Fortran 77 code provided by the U.S. Geological Survey [29]. MODFLOW simulates steady state and transient saturated groundwater flow in irregularly shaped confined or unconfined aquifers. MODFLOW is able to use different aquifer layers so that physical parameters can vary in the vertical dimension as well as be anisotropic within each layer. The problem is disctretized using block centered finite differences. Medium properties within a block are considered uniform. MODFLOW can allow for flow to and from wells, areal recharge, evapotranspiration, flow to drains, and flow through riverbeds. MODFLOW solves (4.1) to compute the hydraulic head and then uses Darcy's law to compute the velocity field with the option of also computing drawdown. MODFLOW is one the the most widely used flow simulators and was chosen for this work because it is well-supported, is available through GMS, and the well-supported, transport simulator, MT3D, is designed with a MODFLOW interface. For this work we used MODFLOW 96 version 3.3. A full description the spatial and temporal discretization in MODFLOW is given in Appendix E. A call to MODFLOW is made to evaluate the objective function and constraints in both the well-field design problem and the hydraulic capture problem.

MT3DMS

MT3D is a three dimensional mass transport Fortran 90 code developed by the U.S. Environmental Protection Agency [45]. MT3D interfaces directly with MODFLOW to read in the hydraulic head and velocity field to solve (4.2) and uses the identical spatial disctretization and hydrological setting. MT3D is can implement a specified concentration of mass flux boundaries as well as contamination from wells. We used MT3D to generate the initial contaminant plume for the hydraulic capture problem, although MT3D was not used at all in the course of the optimization.

4.5 Implementation for Community Problems

In this section we describe how to fully implement the well-field design problem using GMS, MODFLOW, and IFFCO. We demonstrate the implementation process on WD, although the implementation technique is similar for HC. The hydrological settings for WD problem are a homogeneous, confined aquifer and a homogeneous unconfined aquifer, using Models 1 and 2. Keep in mind that optimization with implicit filtering is completely governed by objective function values. Moreover, to evaluate the objective function (4.13), a call to MODFLOW must be done to obtain the values of the hydraulic head h_i. Once the head values are extracted, the objective function can be evaluated.

We begin by describing how to formulate the objective function for IFFCO. We then describe how to use GMS to generate the necessary data files for MODFLOW. Finally, we describe the necessary I/O management to implement an application in parallel.

4.5.1 Forming the Objective Function

IFFCO requires an external subroutine that evaluates the objective function. The user must provide a subroutine of the form

```
feval(dim,x,f,flag)
```

where dim is the dimension of the problem, x is the point to be evaluated, f is the resulting function value, and $flag$ is set to 0 is the function evaluation was successful or 1 if it failed. In this example, if the constraints (4.14) and (4.16) are violated, we set the flag to 1 for failure. If the objective function fails to evaluate, IFFCO sets the function value to a user-specified maximum function value ($fscale$) if the failure occurs in the line search or to a value slightly higher than the largest stencil value if the failure occurs in forming the finite difference gradient.

For this particular application, five wells extracting at the maximum amount minimizes the installation cost and satisfies (4.14) with equality. We formulate this problem as to find the optimal locations of five wells, each operating at the maximum extraction rate. Hence, $x = \mathbf{x}_{i}{}_{i=1}^{5}$, where $\mathbf{x}_i = (x_i, y_i)$ is the coordinate location in meters of well i so that we have ten decision variables. We refer to the operation cost part of f as f^o and the capital cost portion of f_a as f_a^c. Note that since we are using a fixed number of wells operating at a constant pumping rate, we can ignore the capital cost. Hence, we only seek to minimize the operating cost, $f = f^o$

Each time the function is evaluated, a new flow simulation must be done. Once the flow simulation is complete, we must extract the values of the hydraulic head h_i to compute the operational cost. The user must provide two additional subroutines that are called by $feval.f$ which I refer to as *writewell.f* and *readhead.f*. *writewell.f* writes the new well locations, which have been rounded to grid points, to the input well file for the MODFLOW simulation. Care must be taken that the information is written to the MODFLOW well file in the proper format for MODFLOW to use. *readhead.f* is a subroutine that will read in the hydraulic head values from the MODFLOW output file and write them into an array that can be manipulated in $feval.f$. To avoid tedious programming setbacks, the user must know exactly how the output from MODFLOW is formatted. Moreover, *readhead.f* must be rewritten if the discretization changes at all.

4.5.2 Calling Sequence

The optimization algorithm requires a main routine which we refer to as *optmain.f* in which the input for IFFCO is set. $iffco.f$ is called within *optmain.f* to perform the optimization. The complete implementation has the following structure;

Optimization Calling Sequence
1. In **optmain.f** set the parameters for IFFCO
2. Call **iffco.f**: iffco.f will make repeated calls to **feval.f**
to evaluate the objective function for the difference gradient and line search
3. In feval.f:
(a.) Call **writewell.f** to create the well file for MODFLOW
(b.) Call the MODFLOW executable for the groundwater flow simulation
(b.) Call **readhead.f** to read in the hydraulic head values from the MODFLOW output file
(c.) Compute the cost and return the value

4.5.3 Generating the MODFLOW Data Files

We generate the appropriate data files for MODFLOW using GMS. Two approaches can be used to generate a flow model with GMS. One is the *grid approach* and one is the *conceptual model approach*. For the grid approach, the user can manipulate a 3-D grid by applying sources and sinks and other parameters directly to the grid cells. The grid approach is more simple than the conceptual model and useful for first time users of GMS. We use the conceptual approach for this work. For the conceptual approach, the user defines a conceptual site to be modeled. Actual maps may be imported and irregular boundaries or model parameters can be defined. The site is then copied to a 3-D grid and all the cell properties are defined automatically.

GMS consists of ten different *modules* and each module has tools the user uses to manipulate the object in the workspace. For this application, we use the map module to generate the conceptual site, the 3-D grid module to define the discretized problem, and the MODFLOW interface to run the flow simulation. The boundary conditions are defined on the grid and then once the MODFLOW interface is activated, the appropriate data files are automatically generated.

To run a MODFLOW simulation, we require several data files that correspond to MODFLOW packages. MODFLOW packages are subroutines pertaining to the simulation. There are optional packages for different physical phenomena such as drains, wells, rivers, or recharge. Each package requires an input data file, which is generated with GMS. Application 1 requires the following MODFLOW packages;

- The Basic Package is required for any MODFLOW simulation and is used to initialize MODFLOW. The corresponding .bas file contains the fundamental information such as what other packages will be used, the computational time, the units, and the starting head values.

- The Block Centered Flow Package is also required for any MODFLOW simulation and used to compute the flow between cells. The .bcf file contains layer information and material properties.

- The Well Package is an optional package and used to simulate wells. The .wel file contains the location and rates of the wells

- The Recharge Package is an optional package and is used to simulate infiltration perhaps due to rainfall. The .rch package contains the rate of infiltration and location of the recharge region.

- The Solver Package used to solve the system of equations. A solver package is required for any MODFLOW simulation to solve the resulting linear system. We choose a preconditioned conjugate gradient method for this work. The .pcg file contains the maximum number of iterations, stopping tolerance, and other iteration parameters.

4.5.4 Parallel Programming Issues

The parallel version of IFFCO performs function evaluations on each processor. Care must be taken in $feval.f$ so that each processor has a different .wel file and will in turn have a different .hed file. Otherwise, multiple processors could try to open and write to the same .wel file or read a .hed file that came from a different simulation. In addition, when running the MODFLOW executable, the user is prompted to enter a name file. A name file typically has the suffix .mfn and the file contains a list of the MODFLOW packages for the simulation, unit numbers for the package input files, and the names of the input files. I typical .mfn file looks like the following;

LIST 26 A1.out
BAS 1 A1.bas
BCF 11 A1.bcf
OC 10 A1.oc
DATA 30 A1.hed
SIP 12 A1.sip
WEL 13 A1.wel
RCH 20 A1.rch

where here “A1” is the name for this simulation and this file would be called A1.mfn. Each processor must have a different .mfn file containing the appropriate .wel and .hed file names. To do this, in $feval.f$ we must extract the processor number and append it filenames for the .mfn, .wel, and .hed files. Thus processor 2 would replace the A1.wel with A1.2.wel and A1.hed with A1.2.hed in the file called A1.2.mfn. Thus, each processor will have its own .mfn, .wel, and .hed files.

4.6 Numerical Results: Well-Field Design

4.6.1 Conceptual Model

In this section we consider the well-field design problem. A typical well-field design problem includes the cost to install the wells and operate them over a period time to supply a specified amount of water. Constraint (4.14) defines the amount of water the wells are required to supply. This constraint is satisfied exactly with five wells operating at the maximum allowable extraction rate $Q^{emax} = -0.0064(m^3/s)$, hence we do not include the installation cost and seek just to minimize the operational cost. Extraction wells will have a cheaper operating cost in regions where the hydraulic head is highest since less energy will be needed to lift the water to the ground surface elevation. The decision variables are the (x, y) locations of the five wells, as all the wells have a fixed depth.

4.6.2 Spatial Discretization

For the confined aquifer we discretize the domain $\Omega = [0, 1000] \times [0, 1000] \times [0, 30]$ (m) on an equally spaced $50 \times 50 \times 10$ grid. For the unconfined aquifer, we used MODFLOW to determine the saturated domain $\Omega_{unc} = [0, 1000] \times [0, 1000] \times [0, 27] \subset \Omega$ and then discretized Ω_{unc} on an equally spaced $50 \times 50 \times 10$ grid.

4.6.3 Initial Iterate

IFFCO requires a feasible initial iterate. Figure 4.6 shows the steady state flow field for the confined aquifer. Since the head value is high in the lower left corner we initially placed one well there. After the wells are activated, the constraint on the drawdown is violated if the wells are too close together. We looked at several different initial iterates until we found one that satisfied the drawdown constraint for both the confined and unconfined aquifer. We found that placing the remaining four wells close to the specified head boundaries and significantly apart from each other was feasible for both physical domains. Figure 4.7 shows the relative location of the wells and the hydraulic head contours for confined aquifer with the wells pumping at the initial iterate. Note the same initial well locations were used for both aquifers.

Figure 4.6: Steady State Head

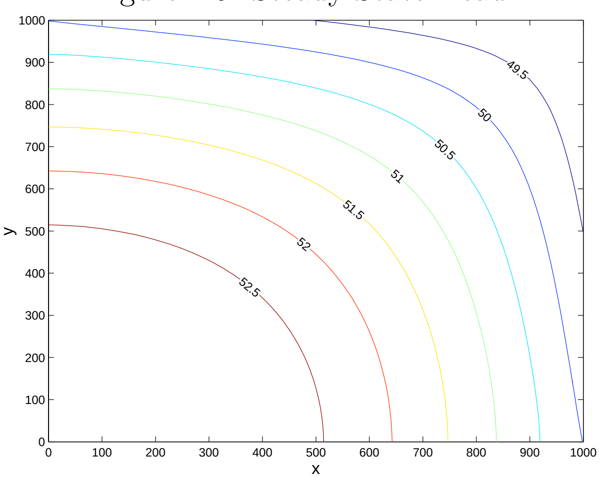
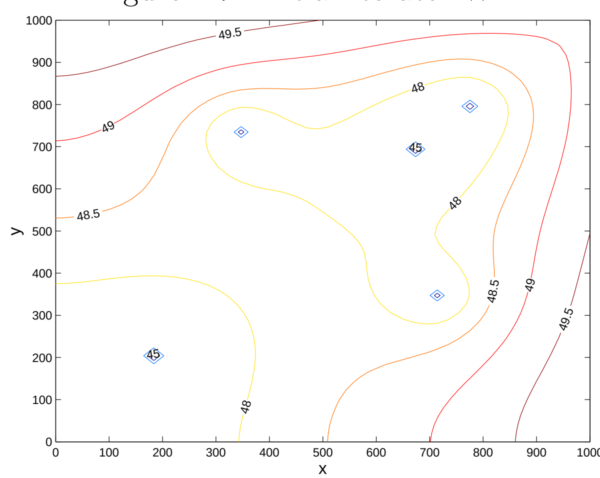


Figure 4.7: Initial Iterate: WD



4.6.4 Results

We ran IFFCO with the default settings, but allowing for one restart. We allowed a budget of 10,000 function evaluations and a sequence of scales given by $2^{-k-1}, k = 0, ..., 10$. We refer to the confined aquifer as CON and unconfined aquifer as UNC. Table 4.3 shows the function value obtained at the initial iterate, the minimum function value,the percent decrease in the objective function, the number of function evaluations needed for each aquifer. Table 4.4 shows the initial x-y coordinates for the 5 wells and the optimal locations for each aquifer. The well locations at the optimal point lie on the boundary constraint (4.17). This physically makes sense since the head values there are higher due to the Dirichlet boundary conditions.

Table 4.3: Decrease in Cost

Problem	f(init_co)	min f	% decrease	No. f'evals
CON	\$23,204	\$21,830	6%	254
UNC	\$26,958	\$23,930	11%	302

Figure 4.8 shows the decrease in the objective function value over the course of the optimization. IFFCO converged to an optimal point within 3% of the function evaluation budget. Instead, IFFCO terminated the optimization based on the budget of difference increment scales. Figure 4.8 shows that after roughly 100 function evaluations, the objective function does not decrease significantly.

Table 4.4: Optimal Locations

X/Y	Init_Co (m)	CON (m)	UNC (m)
X(1)	350.0	401.7	464.2
Y(1)	725.0	800.0	800.0
X(2)	775.0	800.0	800.0
Y(2)	775.0	800.0	800.0
X(3)	675.0	776.9	800.0
Y(3)	675.0	481.1	445.4
X(4)	200.0	138.2	138.2
Y(4)	200.0	800.0	800.0
X(5)	725.0	798.4	800.0
Y(5)	350.0	168.9	144.8

Figures 4.9 and 4.10 show the head contours in the layers containing the wells with the wells at the optimal locations.

Figure 4.8: Decrease in Function Values: WD

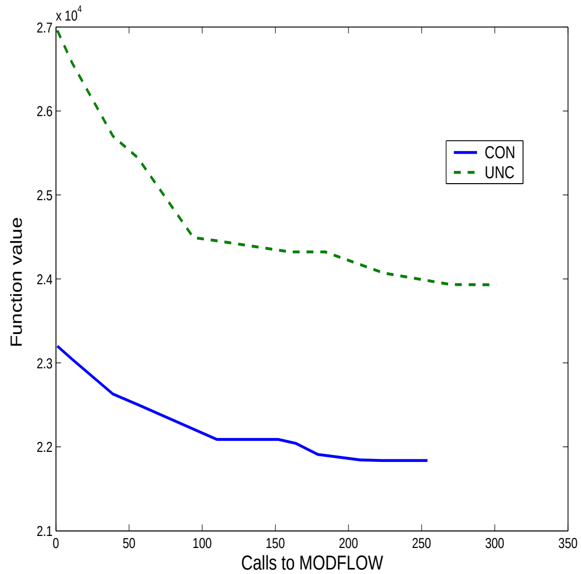


Figure 4.9: Optimal Point: CON

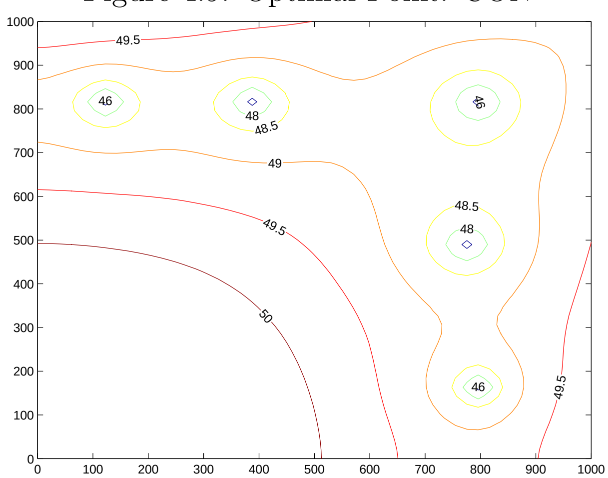
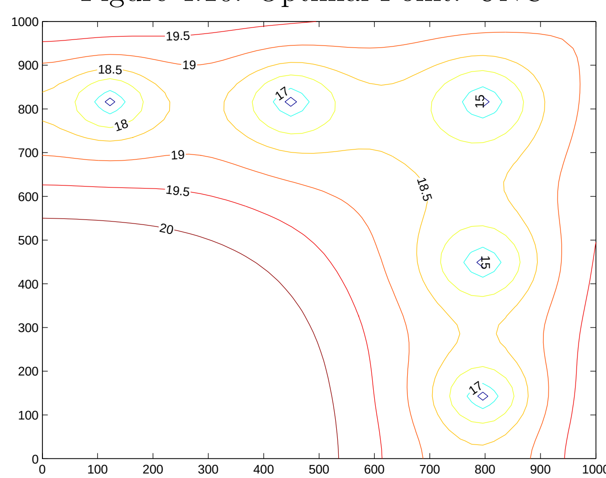


Figure 4.10: Optimal Point: UNC



4.6.5 Optimization Landscapes

To get a better understanding of the objective function, we fixed wells 2-5 and computed the cost be evaluating the objective function and letting the x and y coordinates for well 1 vary between 20 and 800 meters. Figure 4.11 shows contours for objective function values when wells 2-5 are set pumping at the initial locations as in column 1 of table 4.3 for the confined aquifer. The contours shows the small discontinuities in the objective function since the well locations are rounded to grid points for the call to MODFLOW.

Figure 4.11: Contours Near Initial Iterate: Confined Aquifer

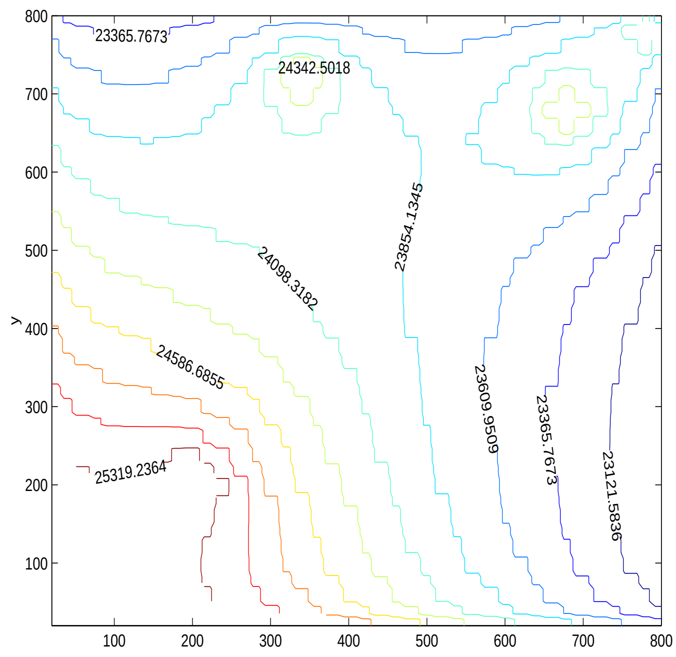


Figure 4.12 is the corresponding surface for the confined aquifer and Figure 4.13 is the landscape for the unconfined aquifer. The peaks in the landscape occur when two wells get close together, hence make the head values lower and the operational cost higher. Within each of the peaks is an infeasible region where the constraint on the head values, (4.16), is violated or because two wells cannot occupy the same grid point. These peaks also make the landscapes nonconvex and introduce local minima. Note that for the unconfined aquifer, most of the region is infeasible. This is due to numerous violations of the head constraint, (4.16), making the unconfined case much more challenging.

Figure 4.12: Landscape Near Initial Iterate: Confined Aquifer

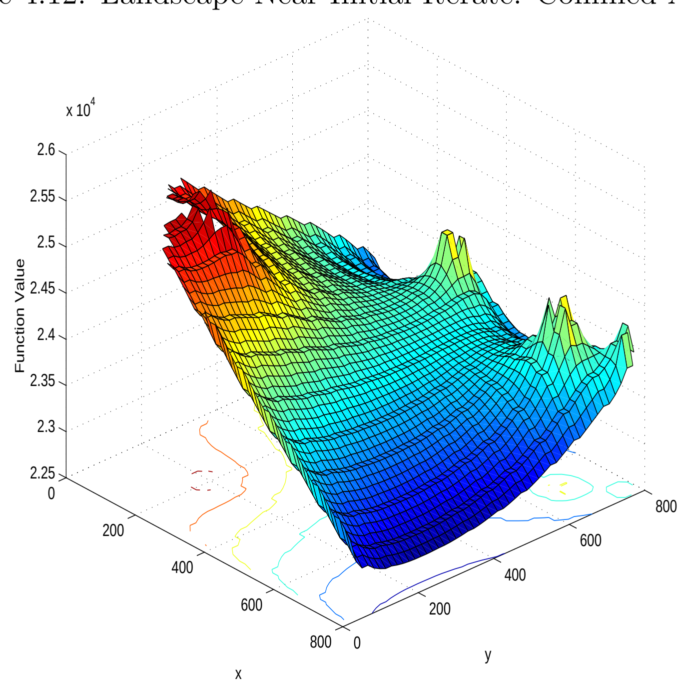


Figure 4.13: Landscape Near Initial Iterate: Unconfined Aquifer

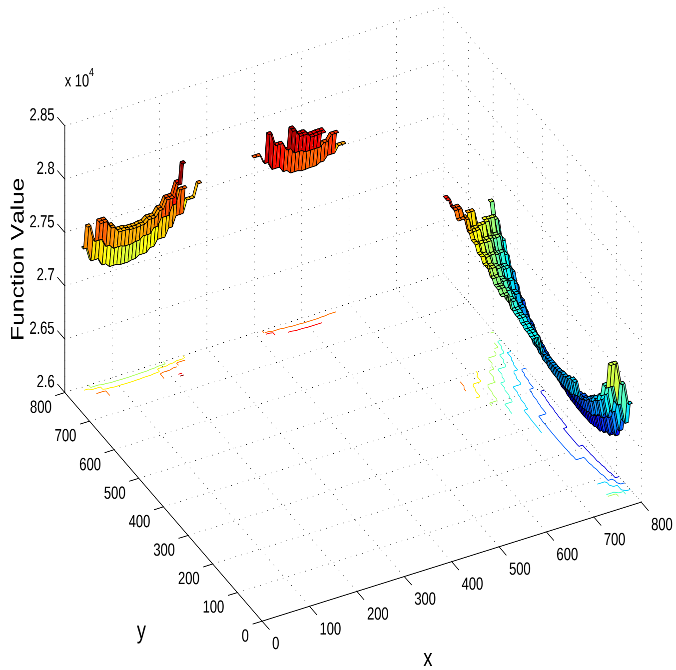


Figure 4.14 is the surface obtained when wells 2-5 are set at the optimal locations for the confined aquifer and 4.15 is the corresponding landscape for the unconfined aquifer.

Figure 4.14: Landscape Near Solution: Confined Aquifer

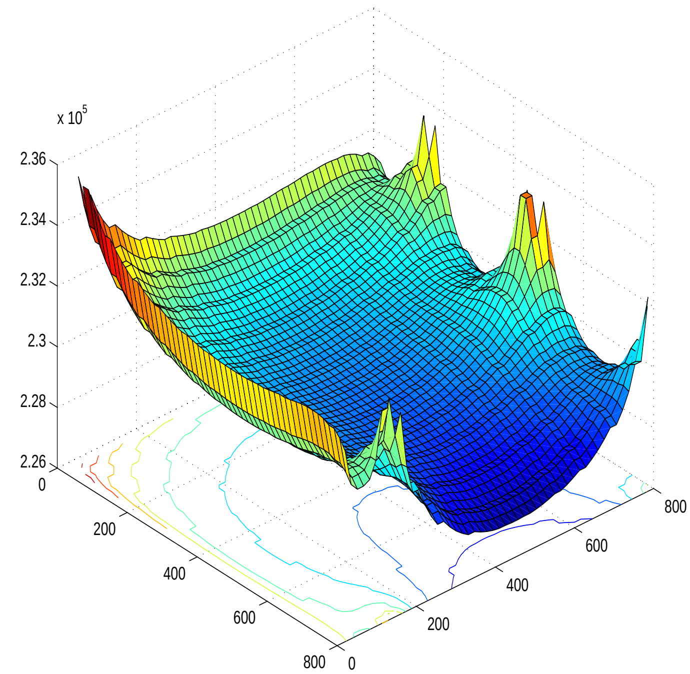
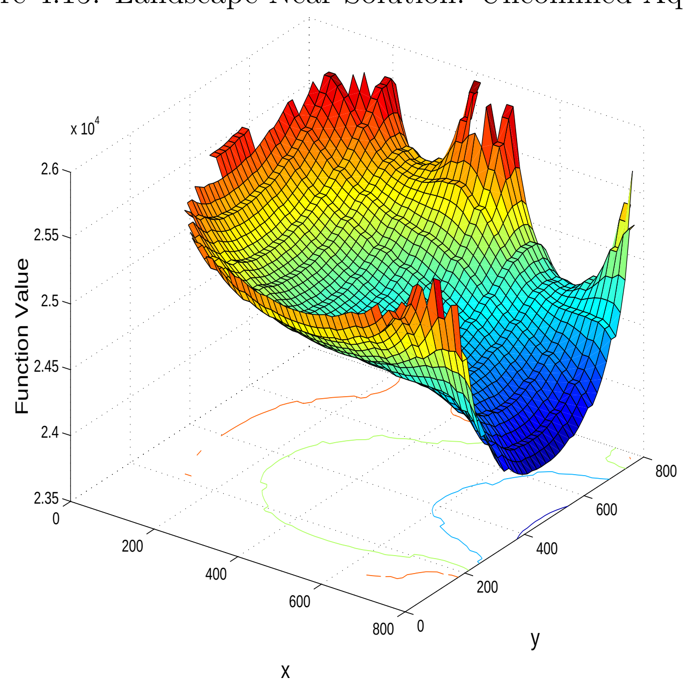


Figure 4.15: Landscape Near Solution: Unconfined Aquifer

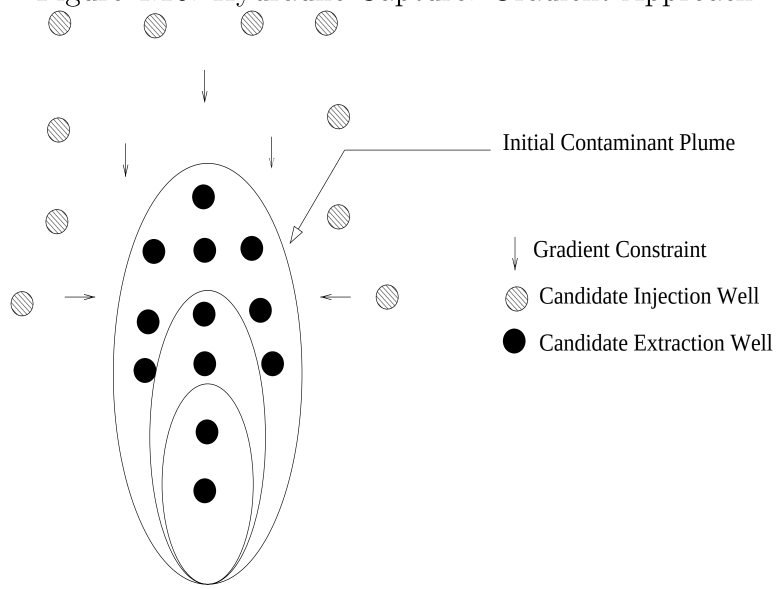


4.7 Numerical Results: Hydraulic Capture

4.7.1 Conceptual Model

In this section we consider the hydraulic capture problem. The hydrological setting is a homogeneous, unconfined aquifer and the application uses Model 2. An initial contaminant plume is situated in the aquifer and wells are installed and operate to keep the plume from spreading. Spreading is prevented by reversing the flow field around the perimeter of the plume. Several approaches exist to prevent the spreading of a contaminant plume [2],[1],[3], [4]. One approach to capturing the plume is to impose constraints on the gradient at certain points around the plume [2]. Candidate injection and extraction wells are used to reverse the flow field. The decision variables can be the pumping rates at the wells, the locations of the wells, or the number of wells. Figure 4.16 is a conceptual model for this type of approach, although it does not represent the actual plume or wells for this particular application.

Figure 4.16: Hydraulic Capture: Gradient Approach



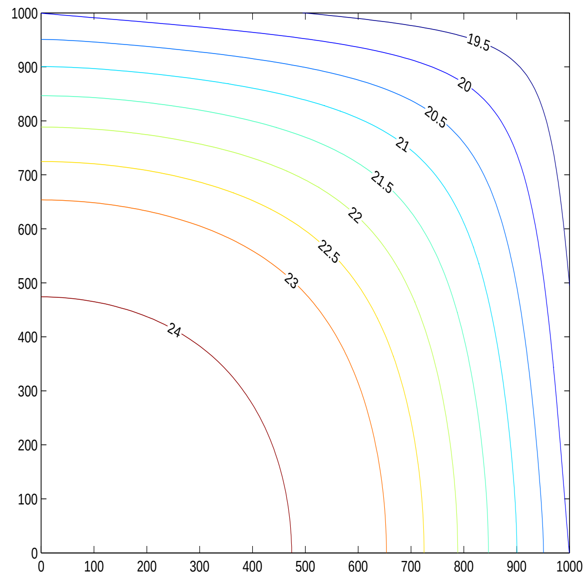
4.7.2 Spatial Discretization

We discretize the entire unconfined aquifer given by $\Omega = [0, 1000] \times [0, 1000] \times [0, 30](m)$ on a $100 \times 100 \times 10$ equally spaced grid. Figure 4.17 shows the steady state hydraulic head contours obtained with MODFLOW in layer 4. The top three layers are dry however we still include them in the model since an injection well could cause dry cells to become saturated in the course of the optimization.

4.7.3 Plume Initialization

To generate the initial plume, as described in [28], we simulated plume development from a finite source for $t \in [-t_s, 0], t_s = 1.58 \times 10^8 s$, with a source concentration of $1kg/m^3$ located physically in the region bounded by $[(200, 225); (475, 525); (h, h-2)]$. To simulate contaminant transport we used MT3D. Figure 4.18 shows the initial

Figure 4.17: Steady State Heads

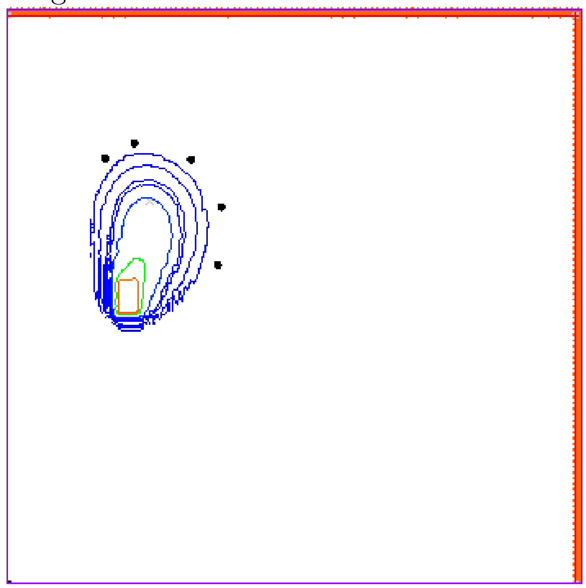


plume. The black circles located around the plume indicate the five locations where we impose the gradient constraint around the 5×10^{-5} contour line of the plume.

4.7.4 Constraint Formulation

In [28] the authors leave it to the reader to choose the concentration that defines the plume boundary and to choose the constraint to capture the plume. For this work, we chose the 5×10^{-5} concentration contour line as the boundary of the plume. Several capture approaches exist in the literature, including a gradient control approach and a particle tracking approach [2], [1], [3]. We chose to implement the gradient control approach for its simplicity. Although a transport simulator is used to create the initial plume and verify the effectiveness of the optimal point, only a flow simulator is required in the course of the optimization. A gradient constraint can be formulated

Figure 4.18: Initial Contaminant Plume



as a constraint on the difference in hydraulic heads values at specified locations, for example

$$h_j^m - h_{j+1}^m \geq d, \tag{4.24}$$

where h_j^m, h_{j+1}^m are hydraulic head values at adjacent nodes, $m = 1 : \bar{m}$ where $\bar{m}$ is the number of gradient constraints, and d is the bound on the difference. Dividing (4.24) by Δh and multiplying by the hydraulic conductivity K yields

$$K \left(\frac{h_j^m - h_{j+1}^m}{\Delta h} \right) \geq K \frac{d}{\Delta h}, \tag{4.25}$$

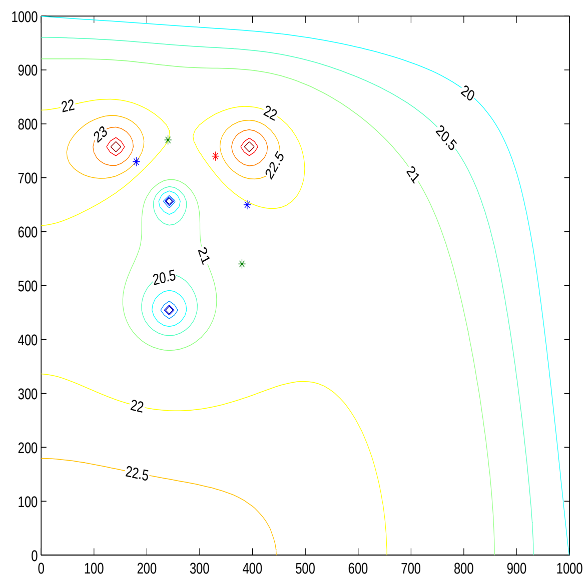
where the term on the right coincides with the velocity of the fluid. We imposed five head difference constraints around the perimeter of the plume in the fourth layer. Since the plume covers the most horizontal area in the fourth layer and the wells are in the bottom layer, the plume will have the largest horizontal area at the end of the remediation period in the fourth layer. If the plume is contained in the fourth layer then in should be contained in the lower layers as well. We used a value of $d = 10^{-4}$

We formulate this problem using candidate wells that may be removed from the design space in the course of the optimization. For this application, if a well rate satisfies $|Q_i| < 10^{-6}$ a well is not considered in the design space and is not included in the flow simulation or cost calculations. For this work the installation cost is much greater than the operational cost. Although removing a well results in a discontinous objective function, even if a well is not pumping at a significant rate, it is still included in the installation cost. Removing the well from the design space simply means you would not install a well that was not significantly affecting the flow.

4.7.5 Initial Iterate

We chose two injection and two extraction wells as candidate wells. The injection wells were initialized at $Q^{imax} = 0.0064 m^3/s$ and the extraction wells were initialized at $Q^{emax} = -0.0064 m^3/s$. Figure 4.19 shows the initial well locations, the hydraulic head contours (not the contaminant plume) and the locations of the head difference constraints, which are indicated with a *.

Figure 4.19: Initial Iterate: HC



4.7.6 Results

We ran IFFCO with the default settings, but allowing for one restart. We allowed a budget if 10,000 function evaluations and a sequence of scales given by $2^{-k-1}, k = 0, ..., 10$. The optimal point only included one extraction well, situated inside the perimeter of the plume. The function value at the initial iterate was \$ 80,211 and after 359 function evaluations the cost decreased by 70% giving a minimum function value of \$24,032. The initial iterate and optimal point are given in Table 4.5. The decrease in cost over the course of the optimization is shown in Figure 4.20. Figure 4.21 shows the hydraulic head contours at the optimal point along with the head difference constraint locations.

Table 4.5: Optimal Point: HC

X/Y/Rate	Init_Co (m), (m^3/s)	Opt (m), (m^3/s)
X(1)	150.0	165.0
Y(1)	750.0	750.0
Rate(1)	0.0064	0.0
X(2)	400.0	400.0
Y(2)	750.0	750.0
RATE(2)	0.0064	0.0
X(3)	250.0	250
Y(3)	650.0	673.0
RATE(3)	-0.0064	-0.00575
X(4)	250.0	250
Y(4)	450.0	450
RATE(4)	-0.0064	0.0

Figure 4.20: Decrease in Function Values: HC

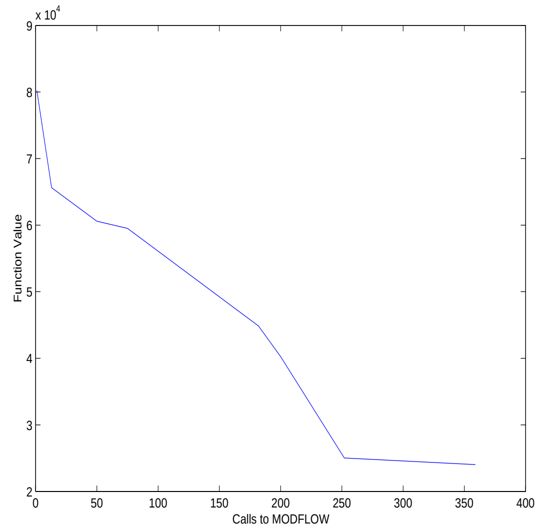
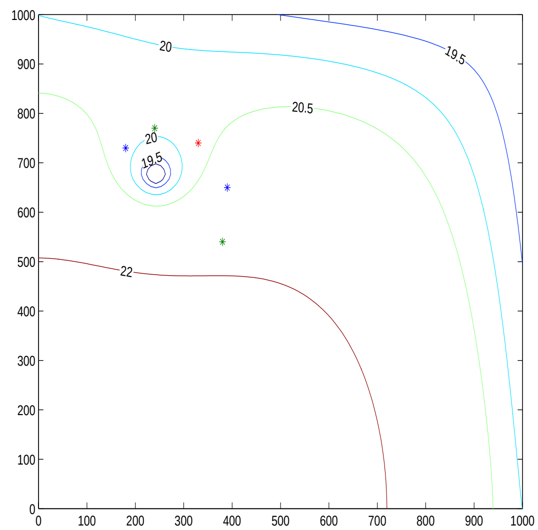
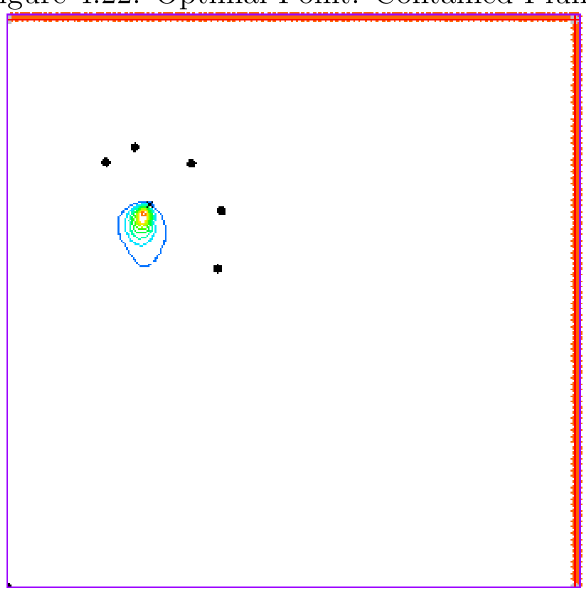


Figure 4.21: Optimal Point: HC



To check that the plume was contained, we simulated contaminant transport with MT3DMS with the initial plume as the initial concentration and the flow field generated with MODFLOW with the extraction well operating at the optimal point. We simulated flow for 5 years and obtained the concentration distribution as in Figure 4.22. The black dots indicate the location of the gradient constraints originally imposed around the perimeter of the plume.

Figure 4.22: Optimal Point: Contained Plume



4.7.7 Optimization Landscapes

The landscape in Figure 4.23 was obtained by the well locations and rates at the initial iterate and allowing the (x, y) coordinate of well 1 to vary over part of the domain containing the plume. The landscape in Figure 4.24 was obtained by fixing all the locations and rates at the initial iterate and letting the pumping rates at wells 1 and 4 vary over -0.0064 to 0.0064. Most of the region was infeasible so in Figure 4.24 we have zoomed in on the region with the minimum. Note that this minimum is obtained when both rates satisfy $|Q_i| < 10^{-6}$, which means they would not be installed. Figure 4.25 was obtained by fixing the locations and rates at the optimal point and letting the (x, y) location of the remaining well vary over a region containing the plume. To obtain these landscapes, we sampled the objective function at 10,201 points.

Figure 4.23: Landscape: Well 1 Locations Vary Near Initial Iterate

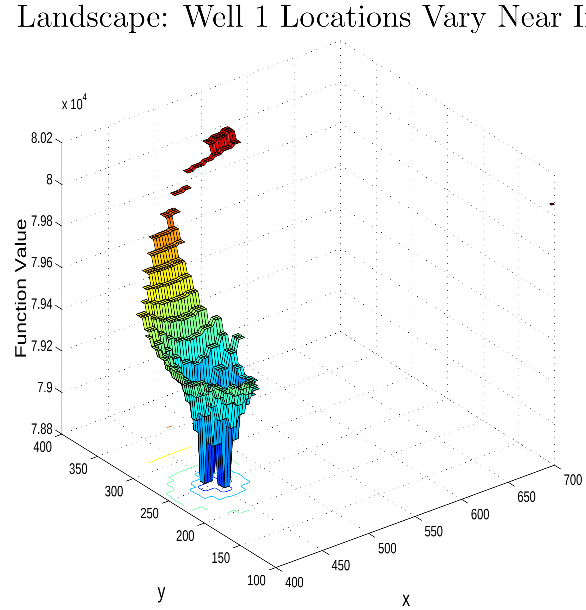


Figure 4.24: Landscape: Wells 1,4 with Varying Rates Near Initial Iterate

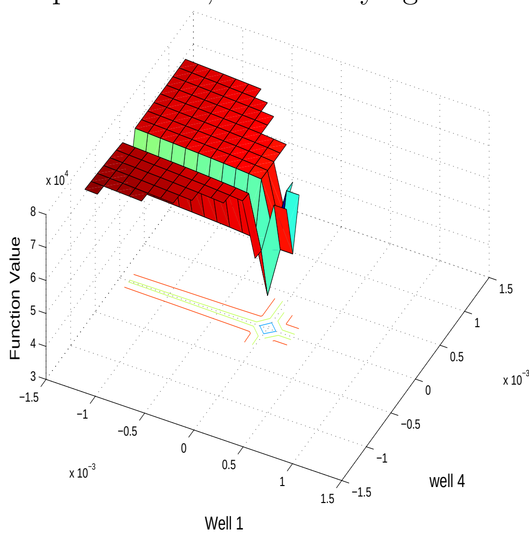
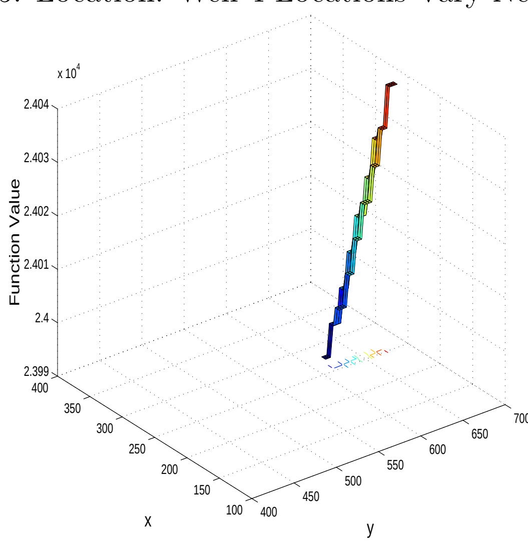


Figure 4.25: Location: Well 4 Locations Vary Near Solution



4.8 Conclusions: Optimization

We have implemented a subset of the community problems proposed in [28]. This was a first attempt to solve any of the applications proposed in [28]. The realization of the objective function and constraints requires a call to the groundwater simulator MODFLOW. The resulting objective functions are nonsmooth, discontinuous, and have local minima. We used a parallel implementation of the implicit filtering algorithm, IFFCO, for the optimization. IFFCO was able to decrease the function values significantly within the designated function evaluation budget proposed. Moreover, we have made the MODFLOW data files, objective function and constraint subroutines, and the links to MODFLOW available on a URL, `http://www4.ncsu.edu/~ctk/community.html` so that memembers of the optimization community can compare the performance of various optimization techniques. Currently, research groups at Sandia National Lab, The Boeing Company, Ecole Polytechnique de Montreal, Rice University, and University of North Carolina at Chapel Hill have tested optimization techniques on the WD and HC problems by way of the URL.

Chapter 5

Conclusions

• In the simulation part of this work, we have implemented a method for temporal adaption based on error estimation to obtain a solution to a nonsmooth, nonlinear PDE called Richards' equation. The method was implemented in a three-dimensional subsurface simulator, ADH. The simulator is now able to solve problems in the nonsmooth regime more accurately and efficiently. The method for error control is based on smooth functions yet ADH performs well on nonsmooth problems.

The nonlinear equations in ADH are solved with a Newton-Krylov-Schwarz method, using a finite difference Jacobian. Moreover, the nonsmooth nonlinearities in RE are replaced with spline approximations, making the reformulated problem locally Lipschitz. Although most convergence theory for Newton–like methods are based on smooth functions, the nonsmooth analysis community has done much work in the past several decades develop convergence theory for nonsmooth functions. We were able to fit the above numerical approach into this framework and show that the resulting finite difference Jacobian is a generalized–Jacobian at a nearby point. We were able to show that using the finite difference Jacobian within the inexact–Newton method will lead to linear convergence until the error approaches the size of the difference increment. Our numerical results for the nonsmooth model problem support the convergence theory.

• In the optimization part of this work, we consider optimal design for saturated groundwater flow and remediation problems. We have implemented a subset of the community problems proposed in [28] for benchmarking purposes. The objective function uses a black–box formulation and is nonsmooth and discontinuous. The goal for this work was to solve a subset of these problems with a deterministic sampling method and to provide full implementation instructions to the public for comparison of various optimization techniques. We used the implicit filtering algorithm for optimization. Implicit filtering uses finite differences to approximate the gradient and decreases the difference increment as the optimization progresses. We provided numerical results for a well-field design application and a hydraulic capture application. Implicit filtering performed well on these problems, despite the nonsmooth objective function. Complete serial and parallel packages to implement and solve these applications are available via a webpage.

Future Work

Nonsmooth Analysis and Richards' Equation

• We plan to continue the analysis on the model problem, as well as RE. In [13] the authors reformulate the model problem resulting in a locally-Lipschitz problem without using spline approximations. We will study the performance of a finite difference Jacobian on the new reformulated problem in [13]. Moreover, in ADH, the centered difference Jacobian is computed using coordinate directions and stored. Typically in Newton-Krylov methods, the columns of the Jacobian are built by taking directional derivatives in the Krylov directions. We plan to investigate the the use of finite differences for this approach and try to fit it into the framework of nonsmooth analysis. In addition, we will consider the use of higher–order splines for reformulating the model problem and RE to be locally Lipschitz continuous.

• **Optimization**

We are currently working on a manuscript in which we compare the performance of several sampling methods on the applications presented in chapter 4. This is joint work done with researches at Sandia National Lab, The Boeing Company, Ecole Polytechnique de Montreal, Rice University, and the University of North Carolina at Chapel Hill. We also plan to consider time-dependent pumping strategies for the hydraulic capture problem as well as investigate several ways to formulate the capture constraint. This will be joint work with researchers at the University of North Carolina at Chapel Hill.

References

[1] David P. Ahfeld and Ann E. Mulligan. Advective control of groundwater contaminant plumes: Model development and comparison to hydraulic control. *Water Resources Research*, 35:2285–2294, 1999.

[2] David P. Ahfeld and Ann E. Mulligan. *Optimal Design of Flow in Groundwater Systems*. Academic Press, San Diego, 2000.

[3] David P. Ahfeld, Roger H. Page, and George F. Pinder. Optimal ground-water remediation methods apllied to a superfund site: From formulation to implementation. *Ground Water*, 33:58–70, 1995.

[4] David P. Ahfeld, Antigoni Zafirakou, and R. Guy Riefler. Solution of a groundwater transport management problem by sequential relaxation. *Advances in Water Resources*, 21:591–604, 1998.

[5] U. M. Ascher and L. R. Petzold. *Computer Methods for Orindary Differential Equations and Differential-Algebraic Equations*. SIAM, Philadelphia, 1998.

[6] A.K. Aziz, A.B. Stephens, and Manil Suri. Numerical methods for reaction-

diffusion problems with non-differentiable kinetics. *Numerische Mathematik*, 53:1–11, 1988.

[7] John W. Barrett and Robert M. Shanahan. Finite element approximation of a model reaction-diffusion problem with a non-lipschitz nonlinearity. *Numerische Mathematik*, 59:217–242, 1991.

[8] A. Battermann, J. M. Gablonsky, A. Patrick, C. T. Kelley, K. R. Kavanagh, T. Coffey, and C. T. Miller. Solution of a groundwater flow problem with implicit filtering. *Optimization and Engineering*, 3:189–199, 2002.

[9] M. Brezina. *Robust Iterative Methods on Unstructured Meshes*. PhD thesis, University of Colorado at Denver, Denver, Colorado, 1997.

[10] R.H. Brooks and A.T. Corey. Properties of porous media affecting fluid flow. *J.Irrig.Drain.Div.Am.Soc.Civ.Eng*, 92:61–88, 1966.

[11] P. N. Brown, A. C. Hindmarsh, and L. R. Petzold. Using Krylov methods in the solution of large-scale differential-algebraic systems. *SIAM J. Sci. Comput.*, 15:1467–1488, 1994.

[12] R.F. Carsel and R. S. Parrish. Developing joint probability distributions of soil water retention characteristics. *Water Resources Research*, 24:755–769, 1988.

[13] Xiaojun Chen. A superlinearly and globally convergent method for reaction and diffusion problems with a non-lipschitz operator. *To Appear?*, 2000.

[14] Xiaojun Chen, Zuhair Nashed, and Liqun Qi. Smoothing methods and semismooth methods for nondifferentiable operator equations. *Siam Journal of Numerical Analysis*, 38:1200–1216, 2000.

[15] T.D. Choi, O.J. Eslinger, P.A. Gilmore, C.T. Kelley, and H.A. Patrick. *User's guide to IFFCO*, May 2001.

[16] Frank H. Clarke. *Optimization and Nonsmooth Analysis*. Wiley–Interscience, New York, 1983.

[17] Patrick A. Domenico and Franklin W. Schwartz. *Phsyical and Chemical Hydrology*. John Wiley and Sons, New York, second edition, 1998.

[18] Kenneth Eriksson and Claes Johnson. Adaptive finite element method for parabolic problems i: Linear model problem. *SIAM J. Numer. Anal.*, 28:43–77, 1991.

[19] A.C. Hindmarsh. *The PVODE and IDA Algorithms*, December 2000.

[20] T.J.R Hughes, L.P. Franca, , and M. Balestra. A new finite element formulation for computational fluid dynamics: V. circumventing the babuska—brezzi condition: A stable petrov-galerkin formulation of the stokes problem accomodating equal–order interpolations. *Computational Methods in Applied Mechanics and Engineering*, 59:85—99, 1986.

[21] E. W. Jenkins. *The Application of Two-Level Domain Decomposition Precondi-*

tioners to Problems in Hydrology. PhD thesis, North Carolina State University, Raleigh, North Carolina, 2000.

[22] Claes Johnson. *Numerical Methods for Partial Differential Equations by the Finite Element Method.* Cambridge University Press, New York, 1987.

[23] D. Kavetski, P. Binning, and S.W. Sloan. Adaptive backward euler time stepping with truncation error control for numerical modelling of unsaturated fluid flow. *International Journal of Numerical Methods in Engineering*, 53:1301–1322, 2001.

[24] D. Kavetski, P. Binning, and S.W. Sloan. Adaptive time stepping and error control in a mass conservative numerical solution of the mixed form of ricahrds' equation. *Advances in Water Resources*, 24:595–605, 2001.

[25] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*, volume 16 of *Frontiers in Applied Mathematics.* SIAM, Philadelphia, PA, first edition, 1995.

[26] C. T. Kelley. *Iterative Methods for Optimization*, volume 16 of *Frontiers in Applied Mathematics.* SIAM, Philadelphia, PA, first edition, 1999.

[27] Jose Mario Martinez and Liqun Qi. Inexact newton methods for solving nonsmooth equations. Technical Report 93/9, School of Mathematics, University of South Whales,Sydney, Australia, March 1999.

[28] A. S. Mayer, C. T. Kelley, and C. T. Miller. Optimal design for problems involving flow and transport phenomena in saturated subsurface systems. Technical

Report CRSC-TR01-33, North Carolina State University, Center for Research in Scientific Computation, December 2001. to appear in Advances in Water Resources.

[29] M.G. McDonald and A.W. Harbaugh. A modular three dimensional finite difference groundwater flow model. *U.S. Geological Survey Techniques of Water Resources Investigations*, 1988.

[30] Robert Mifflin. Semismooth and semiconvex functions in constrained optimization. *Siam Journal on Control and Optimization*, 15:959–972, 1977.

[31] C. T. Miller, G. A. Williams, C. T. Kelley, and M. D. Tocci. Robust solution of Richards' equation for non-uniform porous media. *Water Resources Research*, 34:2599–2610, 1998.

[32] C.T. Miller. *Supplementary Class Notes for ENVR176.* University of North Carolina at Chapel Hill, Chapel Hill, NC, 1999.

[33] Y. Mualem. A new model for predicting the hydraulic conductivity of unsaturated porous media. *Water Resources Research*, 12(3):513–522, 1976.

[34] Liqun Qi and Jie Sun. A nonsmooth version of newton's method. *Mathematical Programming*, 68:353–367, 1993.

[35] K. Radhakrishnan and A. C. Hindmarsh. Description and use of LSODE, the

Livermore solver for ordinary differential equations. Technical Report URCL-ID-113855, Lawrence Livermore National Laboratory, December 1993.

[36] K. Rathfelder and L. M. Abroia. Mass conservative numerical solutions of the head-based form of richards' equation. *Water Resources Research*, 30:267–270, 1994.

[37] L. A. Richards. Capillary conduction of liquids through porous media. *Physics*, 1:318–333, 1931.

[38] Jogn shi Pang and Liqun Qi. Nonsmooth equations:motivation and algorithms. *Siam Journal of Optmization*, 3:443–465, 1993.

[39] B. Smith, P. Bjorstad, and W. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, Cambridge, 1996.

[40] Kimberlie Staheli, Joseph H. Schmidt, and Spencer Swift. *Guidelines for Solving Groundwater Problems with ADH*, January 1998.

[41] M. D. Tocci, C. T. Kelley, and C. T. Miller. Accurate and economical solution of the pressure head form of Richards' equation by the method of lines. *Advances in Water Resources*, 20:1–14, 1997.

[42] Fred T. tracy. 1-d,2-d,and 3-d analytical solutions of unsaturated flow in groundwater. *Journal of Hydrology*, 170:199–214, 1995.

[43] M. T. van Genuchten. Predicting the hydraulic conductivity of unsaturated soils. *Soil Science Society of America Journal*, 44:892–898, 1980.

[44] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth orderelliptic problems. *Computing*, 56:179–196, 1996.

[45] Chunmiao Zheng and P. Patrick Wang. *MT3DMS: A Modular Three-Dimensional Multispecies Transport Model for Simulation of Advection, Dispersion, and Chemical Reactions of Contaminants in Groundwater Systems; Documentation and User's Guide*, December 1999.

Appendix A

Notation

Latin

A	Cross sectional area (§2.3.2)
A	Discrete Laplacian
A_C	Course mesh operator
$B(\delta)$	$\{x : \|\|e\|\| < \delta\}$
B	Right Preconditioner
B_i	i^{th} ubdomain preconditioner
c_i	Cost coefficient
C^ι	Concentration of species ι
C_k	Approximation to second derivative in error estimate

d_i	Depth of well i
d	Steepest decent direction
D^{ι}	Dispersion tensor
D_F	Set of points where F is differentiable
$D(\theta)$	Diffusivity
e_+	$x_+ - x^*$: Error in next Newton iteration
e_c	$x_c - x^*$: Error in current Newton iteration
e_j	j^{th} coordinate unit vector
E_k	Estimated temporal truncation error at k^{th} time step
$F_{p,i}^m$	i^{th} coordinate for the p^{th} term of RE of
	the elementary load vector on element m
$F'(x)$	Jacobian of $F(x)$
$f_s(u)$	Piecewise linear spline approximation to $f(u)$
g	Gravitational constant
h	Hydraulic head
h_i	Hydraulic head in well i
h^{max}	Maximum allowable head
h^{min}	Minimum allowable head
H_0^1	$\{v : v \in L_2, v = 0 on \Gamma\}$
$H^{1,h}(\Omega)$	FIX
H	Subdomain width

I	Identity operator
I_i	i^{th} partition
$\mathcal{I}_\alpha^\iota$	Interphase mass transfer in species ι in the α phase
$\mathbf{j}_\alpha^\iota$	Non-advective flux for species ι in the α phase
k	Intrinsic permeability
$k_r(\psi)$	Relative permeability
K	$k_r(\psi)K_s$
$K(\theta)$	Permeability
K_s	Water saturated hydraulic conductivity
L_i	Lower bound constraints
$\tilde{L}$	Diagonal matrix for $f_s'(u)$
L_f	Lipschitz constant of $f(y)$
L_y	Lipschitz constant of $y(t)$
(m)	Meters
M_α^ι	Mass of species ι in the α phase
n	Number of wells
n^e	Number of extraction wells
N_e	Number of elements
$N(x)$	Neighborhood of x
P_α	Pressure of the α phase
$\mathbf{q}_a$	Darcy flux of the aquesous phase
$\mathbf{q}_z$	Darcy flux in the z direction

$q_{i,j}^{\bar{S}}$	Discretized source/sink in MODFLOW
Q	Source term in RE
Q_i	Well rate in well i
Q_T	Net pumping rate: $\sum_{i=1}^{n} Q_i$
Q_T^{max}	maximum allowable total extraction rate
Q_T^{min}	minimum allowable total extraction rate
Q^{emax}	Maximum allowable extraction rate
Q^{emin}	Minimum allowable extraction rate
Q_i^m	Design pumping rate for well i
$\mathcal{R}_\alpha^\iota$	Reactions in species ι in the α phase
$r \in \mathbb{R}^N$	Residual: $b - Ax$
R_i	Restriction operator to subdomain i
R_i^T	Prolongation operator from subdomain i
s	Newton step, $x_+ - x_c$
(s)	Seconds
$\bar{\mathcal{S}}$	External fluid source/sink
$\mathcal{S}_\alpha^\iota$	External source/sink of species ι in the α phase
$S(u, \Delta u)$	Centered difference stencil about u

$S_a(\psi)$	Saturation
S_r	Residual saturation
S_s	Specific storage
t, T	Time
t_f	Final time
U_i	Upper bound constraint
u_1^k	Approximation to $u(z_1, t^k)$
$\mathbf{v}_\alpha$	Velocity of α phase
v_i	i^{th} component of $\mathbf{v}$
V	Volume of porous media
V_n	Element of $\partial F(x_n)$
x_+, x_c	Next Newton iterate and current Newton iterate
x^*	Solution to $F(x) = 0$
y_{BE}^k	Backward Euler approximation to $y(t^k)$
y_{PRE}^k	linear extrapolation predictor for $y(t^k)$
z_1, z_2	Endpoints of arbitrary element
z_{gs}	Ground surface elevation
z	Depth (vertical spatial dimension)

Greek

α	Measure of mean pore size
α_L, α_t	Longitudinal and transversal dispersivities
β_a	Compressibility of the aqueous phase
$\delta_{i,j}$	Kronecker delta
Δ	Spline increment
ϵ	Centered difference increment
η	porosity
η_n, η_c	Forcing terms in inexact Newton iteration
λ	Ratio of reaction to diffusion rates (§2.3.1)
	Armijo step-length (§4.3.1)
ω_α^ι	Mass fraction of species ι in the α phase
Ω	Spatial Domain
Ω_i	i^{th} subdomain
Ω_m	m^{th} element
Ω_{opt}	Set of feasible points
Ω_{unc}	Saturated part of Ω for unconfined aquifer
ϕ	Linear basis functions of V_h
ψ	Hydraulic head
$\hat{\psi}^k$	Approximation to $\psi(t^k)$
$\bar{\psi}$	Dirichlet boundary conditions for ψ

ρ_α	Density of the α phase
θ_α	Volume fraction of the α phase
μ_a	Viscosity of the aqueous phase
μ_l	Slope of piecewise linear spline on $[u_l, u_{l+1}]$
Γ	boundary of Ω
Γ_i	boundary of subdomain Ω_i
$\partial F(u)$	Clarke-generalized derivative of F at u
$\partial_B F(u)$	B-differential of F at u
∇_ϵ	Centered difference operator with increment ϵ

Subscripts

α	phase
a	aquesous phase
g	gas phase
n	sequence of numbers
k	temporal location
t	temporal derivative
h	Finite element discretization
x, y, z	Spatial derivatives

Superscripts

b_i	Cost exponent
k	Temperal location
m	$m = 1 - 1/n$, as in Van Genuchten and Mualem relations
n	Measure of pore size uniformity
p	$p \in (0, 1)$ exponent in model problem

Appendix B

Discretization: 1-D Richards' Equation

Our system for 1-D Richards' equation is formulated in a similar way as for the model problem (2.21), except the problem is nonlinear. In this Appendix we will compute a 2x1 element load vector for an arbitrary element $\Omega_m = [z_m, z_{m+1}]$. It should be understood that the actual element load vector is Nx1, however all entries are zero other than the the m^{th} and $m+1^{st}$ entries so to simplify notation, we express the element load vector as 2x1 [32]. To simplify the integration, we will partition (4) into four terms as follows,

$$\underbrace{\int\limits_{\Omega} S_s S_a(\hat{\psi}^{k+1})[\hat{\psi}^{k+1} - \hat{\psi}^k] v_i(z) dz}_{\text{Term 1}} + \underbrace{\int\limits_{\Omega} \eta[S_a(\hat{\psi}^{k+1}) - S_a(\hat{\psi}^k)] v_i(z) dz}_{\text{Term2}}$$

$$+\underbrace{\Delta t \int_{\Omega} [K_s k_r(\psi)(\psi_z + 1)] v_i'(z) dz}_{\text{Term 3}}$$

$$\underbrace{-\Delta t \int_{\Omega} Q^{k+1} v_i(z) dz}_{\text{Term 4}} = 0,$$

for $i = 1...N$

Let $F^m_{p,i}$ for $p = 1, 2, 3, 4$ and $i = 1, 2$ denote the ith component of the pth term for the vector formed in the formulation of RE. We use the trapezoid rule to approximate the integrals. Assume local coordinates so that $\Omega_m = [z_1, z_2]$ with $z_2 - z_1 = \Delta z$. We define our basis functions, as in ADH, as the piecewise linear functions $v_i(z,t)$ for $t \in [t^k, t^{k+1}]$ by

$$v_i(z,t) = \begin{cases} \frac{z - z_{i-1}}{h} & z \in [z_{i-1}, z_i] \\ \frac{z_{i+1} - z}{h} & z \in [z_i, z_{i+1}] \\ 0 & z \leq z_{i-1}, z \geq z_{i+1} \end{cases}$$

Term 1:

$$\int_{\Omega_m} S_s S_a(\hat{\psi}^{k+1})[\sum_{j=1}^{2}(\psi_j^{k+1} - \psi_j^k) v_j] v_i dz$$

Let i=1.

$$F^m_{1,1} = \int_{z_1}^{z_2} S_s S_a(\hat{\psi}^{k+1})[(\psi_1^{k+1} - \psi_1^k) v_1^2 + (\psi_2^{k+1} - \psi_2^k) v_1 v_2] dz =$$

$$\frac{\Delta z}{2}(S_s S_a(\hat{\psi}^{k+1}(z_1))[(\psi_1^{k+1} - \psi_1^k) v_1^2(z_1) + (\psi_2^{k+1} - \psi_2^k) v_1(z_1) v_2(z_1)]$$

$$+S_sS_a(\hat{\psi}^{k+1}(z_2))[(\psi_1^{k+1}-\psi_1^k)v_1^2(z_2)+(\psi_2^{k+1}+\psi_2^k)v_1(z_2)v_2(z_2)])$$

$$=S_sS_a(\hat{\psi}^{k+1}(z_1))(\psi_1^{k+1}-\psi_1^k).$$

Here we used the facts that $v_1(z_1)=1, v_2(z_1)=0, v_1(z_2)=0, v_2(z_2)=1$. Similarly one can compute $F_{1,2}^m$ to get the 2x1 element load vector associated with this term;

$$F_{1,2}^m=\frac{\Delta z}{2}\begin{bmatrix} S_sS_a(\hat{\psi}^{k+1}(z_1))(\psi_1^{k+1}-\psi_1^k) \\ S_sS_a(\hat{\psi}^{k+1}(z_2))(\psi_2^{k+1}-\psi_2^k) \end{bmatrix}$$

Term 2:

$$\int\limits_{\Omega_m}\eta[S_a(\hat{\psi}^{k+1})-S_a(\hat{\psi}^k)]v_idz$$

Let i=1.

$$F_{3,1}^m=\int_{z_1}^{z_2}\eta[S_a(\hat{\psi}^{k+1})-S_a(\hat{\psi}^k)]v_1d$$

$$=\frac{\Delta z\eta}{2}[(S_a(\hat{\psi}^{k+1}(z_1))-S_a(\hat{\psi}^k(z_1)))v_1(z_1)+(S_a(\hat{\psi}^{k+1}(z_2))-S_a(\hat{\psi}^k(z_2))v_1(z_2))]$$

$$=\frac{\Delta z\eta}{2}[S_a(\hat{\psi}^{k+1}(z_1))-S_a(\hat{\psi}^k(z_1))]$$

We can find $F_{4,2}^m$ to get

$$F_{3,2}^m=\begin{bmatrix} \frac{\Delta z\eta}{2}[S_a(\hat{\psi}^{k+1}(z_1))-S_a(\hat{\psi}^k(z_1))] \\ \frac{\Delta z\eta}{2}[S_a(\hat{\psi}^{k+1}(z_2))-S_a(\hat{\psi}^k(z_2))] \end{bmatrix}$$

Term 3:

$$\int_{\Omega_m} [K_s k_r(\hat{\psi}^{k+1}) \left(\sum_{j=1}^{2} \psi_j^{k+1} v_j' + 1 \right)] v_i' dz$$

Let i=1.

$$F_{2,1}^m = \int_{z_1}^{z_2} K_s k_r(\hat{\psi}^{k+1})[\psi_1^{k+1}(v_1')^2 + \psi_2^{k+1} v_2' v_1' + v_1'] dz$$

$$= \int_{z_1}^{z_2} K_s k_r(\hat{\psi}^{k+1})[\frac{1}{\Delta z^2}\psi_1^{k+1} - \frac{1}{\Delta z^2}\psi_2^{k+1} + \frac{1}{\Delta z}] dz$$

$$= \frac{K_s}{2\Delta z}(\psi_1^{k+1} - \psi_2^{k+1})[k_r(\hat{\psi}^{k+1}(z_1)) + k_r(\hat{\psi}^{k+1}(z_2))] + \frac{K_s}{2}[kr(\hat{\psi}^{k+1}(z_1)) + k_r(\hat{\psi}^{k+1}(z_2))]$$

Similar computations give

$$F_{2,2}^m = \begin{bmatrix} \frac{K_s}{2\Delta z}(\psi_1^{k+1} - \psi_2^{k+1})[k_r(\hat{\psi}^{k+1})(z_1)) + k_r(\hat{\psi}^{k+1}(z_2))] + \frac{K_s}{2}[kr(\hat{\psi}^{k+1}(z_1)) + k_r(\hat{\psi}^{k+1}(z_2))] \\ \frac{K_s}{2\Delta z}(\psi_2^{k+1} - \psi_1^{k+1})[k_r(\hat{\psi}^{k+1}(z_1)) + k_r(\hat{\psi}^{k+1}(z_2))] + \frac{K_s}{2}[kr(\hat{\psi}^{k+1}(z_1)) + k_r(\hat{\psi}^{k+1}(z_2))] \end{bmatrix}$$

Term4:

$$\int_{\Omega_m} Q^{k+1} v_1 dz$$

Let i=1, then

$$F_{4,1}^m = \int_{z_1}^{z_2} Q^{k+1} v1 dz$$

$$= \frac{\Delta z}{2}(Q^{k+1}(z_1)v_1(z_1) + Q^{k+1}(z_2)v_1(z_2))$$

$$= \frac{\Delta z}{2} Q^{k+1}(z_1).$$

Similar computations for $i = 2$ yield

$$F_{4,2}^m = \begin{bmatrix} \frac{\Delta z}{2} Q^{k+1}(z_1) \\ \\ \frac{\Delta z}{2} Q^{k+1}(z_2) \end{bmatrix}$$

Finally, our 2x1 element residual load vector on Ω_k is

$$\sum_{p=1}^{4} F_p^m.$$

Appendix C

Technical Changes to ADH

In this Appendix we note the technical changes made to the source code for ADH. The objectives of this work are (I.) to implement a temporal adaption stratedgy based on error estimation and control and (II.) implement a preconditioner update strategy for the coarse mesh components of the two level preconditioners based on the error control. Essentially, we need to compute an explicit approximation to the head, compute an estimated truncation error based on the explicit approximation and the current approximation, check to see if this error is less than a user specified tolerance, and compute the next step size in hope to control the error in the next step.

Additions to share_declare.h and share_extern.h

The following global variables were added:

doubles:

dt_old: the previous time step size

dt_update: the step size the last time the preconditioner was updated

Atol, Rtol: absolute and relative error tolerances for (**??**)

max_dt: maximum allowable step size

$*$**doubles:**

old_head2: solution from two previous time steps ago

explicit_head: predictor polynomial

ints:

refresh_flag: flag to update coarse mesh component of the preconditioners

p: counts the number of time steps since last preconditioner update

t: counts the time steps

lag: user specified integer to tell how many time steps until preconditioner is updated

Addition to fnctn.h

Added the subroutine tc_adapt.c with

```
double tc\_adapt(double)
```

The purpose of this subroutine is described in this document later.

Additions to init_bin_read_bc.c, init_binin.c, init_binout.c

Additions were made so that on the TC ADP card one must input Atol, Rtol, and max_dt. A typical line in the foo.bc file would be: TC ADP 0.1 0.1 0.5

Also, to the OP PRE card, one must specify a value for lag. Setting lag = 1 means the caorse mesh component is updated every time step. This value, lag, only needs to be set if PRE OP is 2 or 3. An example of this line in the foo.bc file is: OP PRE 2 1

Additions to adh.c

Set dt_update = dt

Set refresh_flag = 0;

Set t=0;

Additions to fe_main.c

Allocate space for old_head2 and explicit_head using the tl_realloc() subroutine.

Additions to solv_bcgstab.c

The while–do loop for the linear solve was changed to a do–while loop to ensure at least one pass through. For this to work, the call to the overland flow (in fe_main.c ?) solve needs to be commented out if a groundwater simulation is being done without overland flow, otherwise a core dump will occur since the overland flow matrix is singular.

If the linear solver fails to converge, set refresh_flag = -1.

Additions to fe_re_newton.c

The following declarations were added:

ints:

ierr_code: used for the MPI call

total_nodes: used to sum the nodes over all processors

double:

dt_adapt: time step computed for error control

check=0.0, value = 0.0,check_tot=0.0: used in computing the norm

temp_error: temp used in computing the norm

time_error: estimated truncation error

First, we compute explicit_head using old_head and old_head2.

Next, we loop around the two–level precondioners so that the coarse mesh component is updated if refresh_flag=-1, p=lag, or $|dt/dt_{update} - 1| > 0.3$.

If the nonlinear solver fails, the refresh_flag is set to -1.

If the temporal adaption flag is on, there is a loop at the end of fe_newton.c that deals with the error control. Temporal error control begins for $t > 0$ since the explicit_head depends on old_head and old_head2. Within this section of fe_newton.c, we compute $temp\_error = ||explicit\_head - head||$. If this is an MPI job, we must communicate across all processors to be sure everyone has the same value for the norm. We then set dt_adapt = tc_adapt(temp_error). The subroutine tc_adapt computes the next step size for error control.

We then compute the approximated truncation error, time_error, and check to see if this meets the error tolerance–recall the absolute and relative tolerances are taken into consideration when computing the norm. If we fail, we rescale the time step and start over otherwise we move on in time.

Additions to tc_end.c

If the temporal adaption is on, there is a loop so that the time step computed by the error indicator or the time step computed from the time series is always taken. We do not grow the time step based on the performance of the nonlinear solver if temporal adaption is on. However note that the time step will be reduced if the nonlinear or linear solvers fail.

Additional subroutine: tc_adapt.c This subroutine computes the next step size, h_+, based on the approximated error. If $h_+ > max\_dt$, then $h_+ = max\_dt$. If the $h_+/h_c > 10.0$ then $h_+ = 1.1 * DT\_ENLARGE\_FACTOR * h_c$. Here the 1.1 is to ensure that $h_+ \neq 2h_c$ otherwise a divide by zero will occur in the computation of the step size.

Appendix D

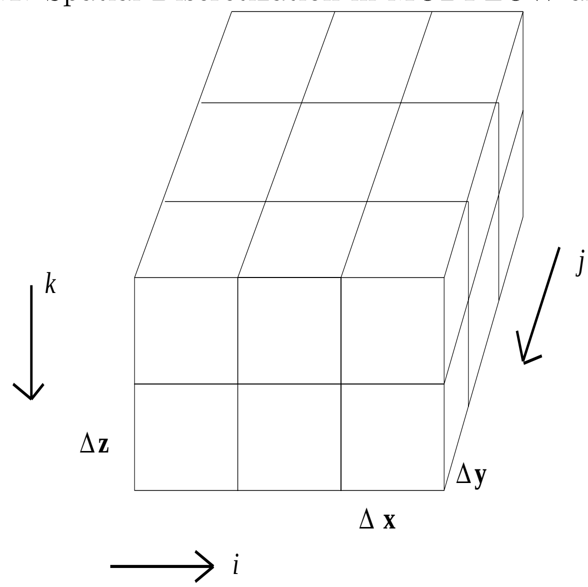
Discretization and Solvers in MODFLOW

D.1 discretization

MODFLOW is a three dimensional groundwater flow code. The three dimensional region is split into layers that are stacked on top of each other. The layers can be confined or unconfined and the media properties should be fairly uniform within each layer. The aquifer is split into cells as in figure D.1, although the spacing need not be uniform as in the figure.

MODFLOW uses block centered finite difference such that the value of the hydraulic head is computed at a node in center of each cell. Physical parameters, such as the hydraulic conductivity, are considered uniform throughout a cell.

Figure D.1: Spatial Discretization in MODFLOW and MT3D



We now describe the disctretization in MODFLOW. Recall the groundwater flow equation,

$$S_s \frac{\partial h}{\partial t} = \nabla \cdot K \nabla h + \bar{\mathcal{S}}, \tag{D.1}$$

where the hydraulic conductivity K can be spatial dependent. Equation (D.1) is solved using a finite difference approximation. To simplify the derivations, assume we have a uniform two dimensional discretization with grid spacing Δx and Δy. Recall that (D.1) was derived from the continuity equation and Darcy's law, (2.7) and we made the assumption that changes in density were negligible. We begin with the discretized version of Darcy's law and use the continuity equation to arrive at the discretized version of (D.1). The finite difference discretizations of Darcy's law for flow in the x and y directions respectively are

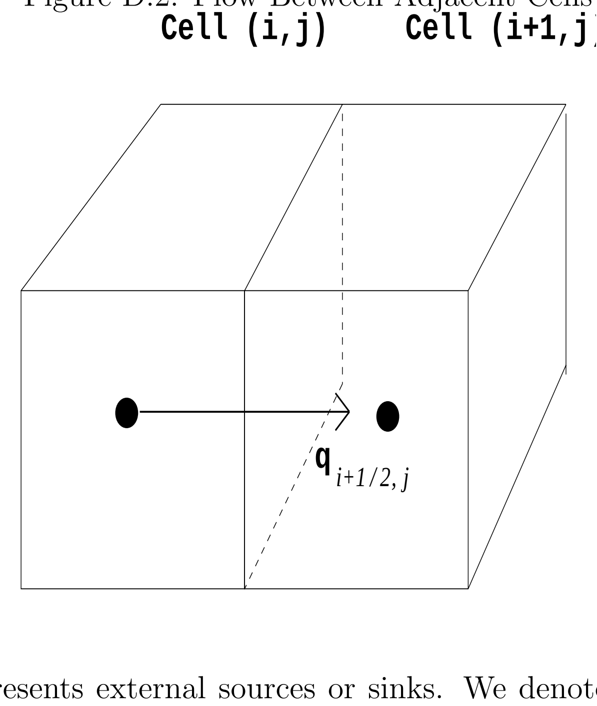
$$q_{i+1/2,j} = -K_{i+1/2,j} \frac{h_{i+1,j} - h_{i,j}}{\Delta x} \Delta yz, \tag{D.2}$$

and

$$q_{i,j+1/2} = -K_{i,j+1/2} \frac{h_{i,j+1} - h_{i,j}}{\Delta y} \Delta xz. \tag{D.3}$$

Here (D.2) corresponds to flow between the adjacent nodes (i, j) and $(i + 1, j)$ as in figure D.2. Similarly, (D.3) corresponds to flow between nodes (i, j) and $(i, j + 1)$. In (D.2) and (D.2), z is the vertical length of the cell and so Δxz and Δyz are the corresponding areas perpendicular to the flow.

Figure D.2: Flow Between Adjacent Cells



In (D.1), $\bar{\mathcal{S}}$ represents external sources or sinks. We denote a discretized external flux as $q^{\bar{\mathcal{S}}}_{i,j}$, and then the discretized form of $\bar{\mathcal{S}}$ is $q^{\bar{\mathcal{S}}_{i,j}} \Delta x \Delta yz$. The discretized continuity equation is

$$(S_S)_{i,j}\frac{\Delta h_{i,j}}{\Delta t}\Delta x\Delta yz = -(q_{i,j+1/2} + q_{i,j-1/2} + q_{i+1/2,j} + q_{i-1/2,j}) + q^{\bar{S}}_{i,j}\Delta x\Delta yz. \quad \text{(D.4)}$$

Here $(S_s)_{i,j}$ is the specific storage that is constant on cell (i,j) and $\frac{\Delta h_{i,j}}{\Delta t}$ is the discretized form of $\frac{\partial h}{\partial t}$. Substituting(D.2) and (D.3) into (D.4) we get the spatially discertized version of (D.1),

$$(S_S)_{i,j}\frac{\Delta h_{i,j}}{\Delta t} = K_{i+1/2,j}\frac{h_{i+1,j}-h_{i,j}}{\Delta x^2} + K_{i-1/2,j}\frac{h_{i-1,j}-h_{i,j}}{\Delta x^2} \quad \text{(D.5)}$$

$$+K_{i,j+1/2}\frac{h_{i,j+1}-h_{i,j}}{\Delta y^2} + K_{i,j-1/2}\frac{h_{i,j-1}-h_{i,j}}{\Delta y^2} + q^{\bar{S}}.$$

We use implicit temporal integration with backward Euler to define $\frac{\Delta h_{i,j}}{\Delta t}$ by

$$\frac{h^k_{i,j} - h^{k-1}_{i,j}}{\Delta t}. \quad \text{(D.6)}$$

Substituting (D.6) into (D.5) we get the fully discretized flow equation,

$$(S_S)_{i,j}\frac{h^k_{i,j} - h^{k-1}_{i,j}}{\Delta t} = K_{i+1/2,j}\frac{h_{i+1,j}-h_{i,j}}{\Delta x^2} + K_{i-1/2,j}\frac{h_{i-1,j}-h_{i,j}}{\Delta x^2} \quad \text{(D.7)}$$

$$+K_{i,j+1/2}\frac{h_{i,j+1}-h_{i,j}}{\Delta y^2} + K_{i,j-1/2}\frac{h_{i,j-1}-h_{i,j}}{\Delta y^2} + q^{\bar{S}}.$$

If we assume $\Delta x = \Delta y$ then the above method is $\mathcal{O}(\Delta x^2)$ in space and $\mathcal{O}(\Delta t)$ in time. A linear system must be solved at each time step. We use the conjugate gradient method with an incomplete Cholesky preconditioner (PCG).