ABSTRACT

WINTON, COREY W. Parameter Estimation in Groundwater Models Using Proper Orthogonal Decomposition. (Under the direction of Dr. Carl T. Kelley.)

We develop a new Proper Orthogonal Decomposition (POD) reduced order model for saturated groundwater flow, and apply that model to an inverse problem for the hydraulic conductivity field. We use sensitivities as the POD basis. We compare the reduced order model results to results obtained with a full finite element model. The solutions generated using the POD reduced model are comparable in residual norm to the solutions formed using only the full-scale model. The material parameters are similarly comparable. The time to solution when using the reduced model is cut in half and the number of calls to the full model are reduced by at least an order of magnitude. The following thesis will give an overview of groundwater modeling, construction and usage of POD, and demonstrate the results of our implementation. We also examine how the inexactness of a reduced order model affects the convergence of the Levenberg-Marquardt optimizer.

Parameter Estimation in Groundwater Models Using Proper Orthogonal Decomposition

by
Corey W. Winton

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

Raleigh, North Carolina

2012

APPROVED BY:

DEDICATION

To my parents and my wife. Thank you so much.

BIOGRAPHY

Corey Winton was born September 19, 1981, in Titusville, PA, to Dennis and Rosemary Winton. He graduated from Titusville High School in 2000 and went to Asbury College for his undergraduate degree in Applied Mathematics. After four years in Wilmore, KY, he left the bluegrass for Raleigh, NC, and a Ph.D. program in Applied Mathematics at North Carolina State University.

ACKNOWLEDGEMENTS

I cannot hope to thank everyone who has helped me achieve my degree. The following will attempt to thank, in some sort of chronological order, some of those who have made this possible.

- Obviously, I thank my parents Dennis and Rosemary Winton first. Without them none of this would be possible.

- Mrs. Fitzgerald and those involved with the SEEK program. Thank you for giving me an opportunity to learn and be challenged at an early age.

- Mr. and Mrs. Carrell and Betty Rainey and the many other teachers of Titusville area schools who challenged generations of students to reach for their goals and never settle for less than their absolute best. Thank you for preparing me so well for college, graduate school, and beyond.

- I cannot hope to list everyone from Asbury College who I want to thank, but want to mention Dr. Coulliette, Dr. Rietz, Dr. Charalambakis, Dr. Lee, and the rest of the Asbury mathematics department for encouraging me to pursue a career in mathematics.

- My fellow Asbury mathematicians, especially Skyler Speakman, Corey Robertson, Ben Flannery, Joel Kilty, and my math weekend teammates. I'm so thankful you all helped me finish my math degree from Asbury and inspired me to continue to graduate school.

- My fellow graduate students, especially Brendan O'Connor, Steve and Lindsay May, Ryan Siskind, and Teresa Selee. Thank you for helping me study, learn, and pass graduate exams, not to mention survive and enjoy graduate school to the fullest.

- My team members and supervisors at ERDC in Vicksburg, especially Rob Wallace, Dave Richards, Owen Eslinger, Amanda Hines, and Jeff Hensley. Thank you for helping me balance the pursuit of this degree and my job.

- My thesis committee. Thank you for your insightful feedback and enduring what turned out to be a much longer process than any of us anticipated.

- Tim Kelley. It's been an honor to learn from you and work with you the past years. I look forward to many productive years of collaboration ahead.
- My wife, Christy Paine Winton. Your encouragement and endurance through this process has been invaluable. Thank you for standing by me and encouraging me all the way through.

This research was funded in part by the Army Research Office grants W911NF-06-1-0412 and W911NF-11-1-0367 and US Army Corps of Engineers contracts W912HZ-10-P-0221 and W912HZ-10-P-0256.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

We describe an algorithm for model calibration of saturated flow codes using a reduced order model, specifically Proper Orthogonal Decomposition (POD). We will recover the hydraulic conductivity of materials in a domain from measurements of the hydraulic head and the pumping rates of any wells. We will use a nonlinear least squares approach to measure the quality of our approximation. We seek to minimize the sum of squared differences between data points and our simulated results by adjusting the values of hydraulic conductivity. We will develop models in both 2-D and 3-D and will demonstrate results for both. We measure the quality of our process by accuracy of the solution and computation time required to obtain that solution.

We begin our discussion with a brief outline of groundwater physics and equations. Our research extends only to saturated groundwater models and our discussion will be similarly limited. We will discuss the discretization of the groundwater equations for implementation in a finite element mesh. In Chapter 5, we discuss the implementation of Proper Orthogonal Decomposition (POD) in the optimization process. Our goal with POD is to reduce the number of full, expensive model calls necessary for the optimizer to find a minimizing solution. Chapter 6 describes the use of the Levenberg-Marquardt code PEST to find a suitable suite of parameters to match data. Chapter 7 will detail the results we have found by using POD in saturated groundwater models. In Chapter 8, we explore how inaccuracies in the linear solver affect the solution speed and quality.

We also include several discussions for the sake of clarity in our investigations. Chapter 4 describes the construction and implementation of the “Total Flux” boundary condition. This boundary condition ensures a unique solution for saturated domains in the absence of a well. Appendix A proves a discussion of finite element methods. Appendix B should be read in conjunction with Appendix A as it details convergence properties of finite element methods. Appendix C details exactly how we used the 3-D FEM code `ADH` , including what routines had to be modified to extract the information used in POD. Appendix D gives a brief overview of our use of `PEST` and the parameters we changed in the process of optimization. Finally, Appendix E describes `ScaLAPACK` and how we used it to solve for the SVD of a matrix in parallel.

Chapter 2

Saturated Flow

2.1 Model Development

2.1.1 Introduction

Our research focuses on parameter estimation in porous media with a single liquid phase. In this chapter, we will derive the equations that govern steady-state, saturated flow through an immobile (or non-deformable) porous medium. We are careful to note the methods described later in this document hold only under these conditions. For the sake of clarity, vectors will be represented in italicized boldface ($\boldsymbol{v}$) and matrices or tensors will be represented in bold ($\mathbf{M}$).

2.1.2 Derivation via Mass Conservation

In our derivation, we begin with conservation of mass in a Representative Elementary Volume (REV). A Representative Elementary Volume (REV) defines the scale at which a given property is insensitive to changes in the length scale associated with an observation [46], [69]. Figure 2.1 gives a visual description of how material properties can be misleading without a proper REV.

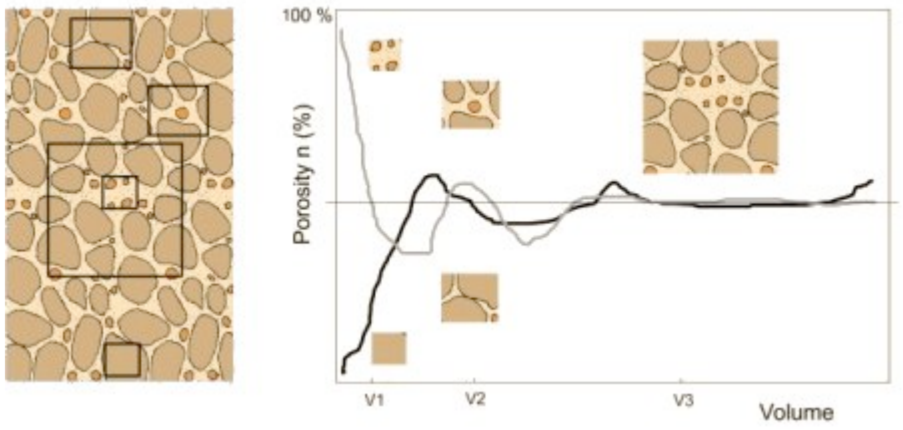


Figure 2.1: Representative Elementary Volume

In this work, we deal strictly with single phase flow in a saturated medium but for the sake of completeness in the derivation we will note other phases with the notation α, where $\alpha = s, l$, or g for solid, liquid, or gas, respectively. The void space (also called the pore space) is represented with $\alpha = p$. Distinct species present in each phase–for instance multiple gases–are represented with the superscript i.

Table 2.1 describes several characteristics for our REV.

We can balance several of these properties over the volume. We see

$$\sum_{\alpha} \varepsilon^{\alpha} = 1 : \text{ Volume Fractions sum to 1,}$$

$$\sum_{i} w^{\alpha,i} = 1 : \text{ Mass Fractions within each phase sum to 1,}$$

$$\sum_{\alpha \neq s} s^{\alpha} = 1 : \text{ Saturation of non-solid phases sum to 1.}$$

We use mass-averaged velocity in our calculations so that macroscale properties are directly related to conservation principles. That is,

$$v^{\overline{\alpha}} = \frac{\int_{\Omega} v_{\alpha} \rho_{\alpha} \, dr}{\int_{\Omega} \rho_{\alpha} \, dr}, \tag{2.1}$$

Table 2.1: Key REV Characteristics

Symbol	Name	Definition	Units
n	Porosity	$\frac{V^p}{V}$	-
ε^α	Volume Fraction	$\frac{V^\alpha}{V}$	-
s^α	Saturation	$\frac{V^\alpha}{V^p}$	-
ρ^α	Density	$\frac{M^\alpha}{V^\alpha}$	$\frac{M}{L^3}$
$w^{\alpha,i}$	Mass Fraction	$\frac{M^{\alpha,i}}{M^\alpha}$	-

and we see

$$\rho^\alpha v^{\overline{\alpha}} = \frac{\int_\Omega \rho_\alpha \, dr}{\int_\Omega \, dr} \times \frac{\int_\Omega v_\alpha \rho_\alpha \, dr}{\int_\Omega \rho_\alpha \, dr} = \frac{\int_\Omega v_\alpha \rho_\alpha \, dr}{\int_\Omega \, dr}.$$

2.1.3 General Mass-Flux Equation

We have established our properties across the REV and turn to conservation of mass to develop our model [77], [51]:

$$\text{[Change In Mass Rate]} = \text{[Mass Inflow Rate] - [Mass Outflow Rate]} \; , \tag{2.2}$$

or

$$\begin{aligned} \text{[Accumulation of Mass]} \quad &= \quad \text{[Net Advective Transport (Flow)]} \\ &+ \quad \text{[Non-Advective Transport]} \\ &+ \quad \text{[Reactions]} + \text{[Interphase Mass Exchange]} \\ &+ \quad \text{[Sources].} \end{aligned} \tag{2.3}$$

From this general statement of mass conservation, we will derive our elliptic boundary value problem, following the derivation in [70]. We first describe accumulation of mass, the change over time of a species i in phase α over volume V:

$$\text{Accumulation of Mass } = V\frac{\partial}{\partial t}\left(\varepsilon^{\alpha}\rho^{\alpha}w^{\alpha,i}\right). \tag{2.4}$$

We next turn our attention to Net Advective Transport (Flow). We examine flow through a sample cubic volume with vertices at $x, x+\Delta x, y, y+\Delta y, z, z+\Delta z$. We define $v_x^{\overline{\alpha}}$ as the mass-averaged velocity of the α-phase in the positive x-direction and $A_{yz} = \Delta y\Delta z$ as the area of the face where x is fixed. Total mass of the i species in the α phase passing through that face into the volume V is given by

$$M_x^{\alpha,i} = A_{yz}\varepsilon^{\alpha}\ \rho^{\alpha}\ v_x^{\overline{\alpha}}\ w^{\alpha,i}\Big|_x. \tag{2.5}$$

Similarly, total mass passing through the opposite face, out of the volume is given by

$$M_{x+\Delta x}^{\alpha,i} = A_{yz}\varepsilon^{\alpha}\ \rho^{\alpha}\ v_x^{\overline{\alpha}}\ w^{\alpha,i}\Big|_{x+\Delta x}. \tag{2.6}$$

We expand (2.6) with a Taylor's series at the initial face and keep only the first order terms to see

$$M_{x+\Delta x}^{\alpha,i} = A_{yz}\varepsilon^{\alpha}\ \rho^{\alpha}\ v_x^{\overline{\alpha}}\ w^{\alpha,i} + \varepsilon^{\alpha}\rho^{\alpha}w^{\alpha,i}\Delta x\frac{\partial v_x^{\overline{\alpha}}}{\partial x}\Bigg|_x. \tag{2.7}$$

Over our REV, we assume porosity (ε), density (ρ), and mass fraction (w) are constant and need only expand velocity in (2.7). Thus, net advective transport of the i species in the α phase in the x-direction is given by

$$M_x^{\alpha\, i} - M_{x+\Delta x}^{\alpha\, i} = -\varepsilon^{\alpha}\rho^{\alpha}w^{\alpha\, i}\Delta x\frac{\partial v_x^{\overline{\alpha}}}{\partial x}\Bigg|_x. \tag{2.8}$$

We apply (2.8) to the x, y and z directions, recall that $\Delta x \times \Delta y \times \Delta z = V$ and see

$$\text{Net Advective Transport } = -V\nabla\cdot\left(\varepsilon^{\alpha}\rho^{\alpha}w^{\alpha\, i}\boldsymbol{v}^{\overline{\alpha}}\right). \tag{2.9}$$

We will characterize non-advective transport of the i species in the α phase for a volume V with $-V\nabla\cdot\boldsymbol{j}^{\alpha,i}$. $V\psi^{\alpha,i}, VR^{\alpha,i}$, and $VS^{\alpha,i}$ represent change of mass due to phase change, reactions, and sources for a volume V, respectively. As the generic volume coefficient is present in all terms, we factor it away and have the general mass balance equation:

$$\frac{\partial}{\partial t}\left(\varepsilon^\alpha \rho^\alpha w^{\alpha,i}\right) = -\nabla \cdot \left(\varepsilon^\alpha \rho^\alpha w^{\alpha,i} \boldsymbol{v}^{\overline{\alpha}} + \boldsymbol{j}^{\alpha,i}\right) + (\psi^{\alpha,i} + R^{\alpha,i} + S^{\alpha,i}). \tag{2.10}$$

We sum (2.10) over all species and note several terms can be simplified:

- Change in mass due to phase change across each species i is the total change in mass for each phase α:
$$\sum_i \psi^{\alpha,i} = \psi^\alpha.$$
- Change in mass due to sources/sinks across each species i is the total change in mass for each phase α:
$$\sum_i S^{\alpha,i} = S^\alpha.$$
- Mass-fractions ($w^{\alpha,i}$) for each species i in phase α sum to 1:
$$\sum_i w^{\alpha,i} = 1.$$

Conservation of mass allows us to eliminate some terms:

- Total change of mass across all species due to reactions is zero:
$$\sum_i R^{\alpha,i} = 0.$$
- Sum of non-advective flux across all species within a single phase is zero:
$$\sum_i \boldsymbol{j}^{\alpha,i} = 0.$$

We only investigate movement of a single fluid phase in the following research, so we eliminate the phase-change term ψ^α and drop the superscript α for all other terms. This leaves our general equation for accumulation of mass of a single phase:

$$\frac{\partial}{\partial t}(\varepsilon\rho) = -\nabla \cdot (\varepsilon\rho\boldsymbol{v}) + S. \tag{2.11}$$

We turn to Darcy's Law to describe the velocity of the liquid in our domain. As we describe the fluid moving through the solid domain, we assume the solid is immovable and has no velocity component.

2.1.4 Darcy's Law

Henry Darcy, a civil engineer in Dijon, France, motivated by the concern over public water supply was driven to investigate better designs for water purification through filtered sands. This led him "to determine the laws of flow of water through sand" [32] and his initial results were published in 1856. Darcy constructed an apparatus similar to the one shown in Figure 2.2 [12]. The volumetric flow rate Q (units: L^3/T) was

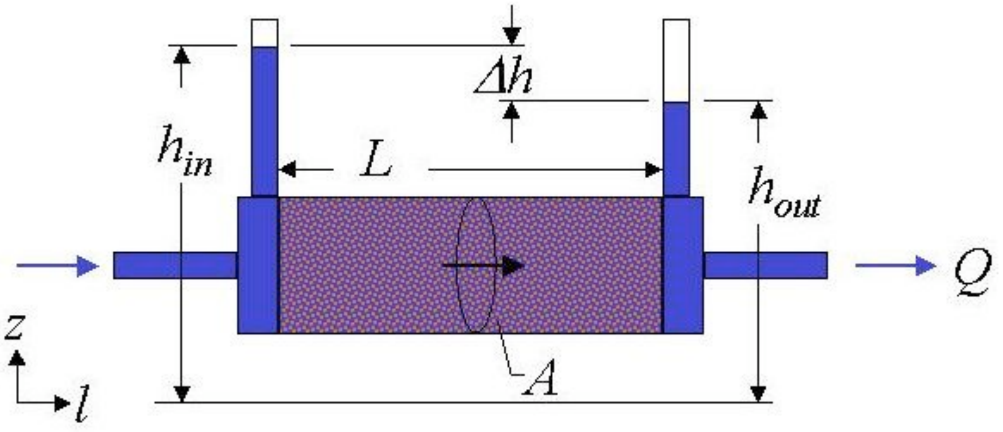


Figure 2.2: Explanation of Darcy's Experiment

controlled by Darcy. The cross sectional area A (L^2) was known, as was the length of the sand column between sensors l (L). The readings h_1, h_2 (L) were taken from manometers and represented the height to which water would rise in the absence of restrictive media (ie – a length-scale representation of hydraulic head). From these measurements, Darcy calculated the flow of water through the apparatus $q = Q/A$ (L/T) which is called "specific discharge" and a dimensionless quantity termed "hydraulic gradient" $(h_1 - h_2)/l$, which is the change in hydraulic head over distance. This dimensionless quantity is also expressed $\partial h/\partial l$. His calculations gave rise to Darcy's Law:

$$\frac{Q}{A} = q = \frac{-k(h_1 - h_2)}{l} = -k\frac{\partial h}{\partial l}. \tag{2.12}$$

The proportionality constant k (L/T) represents the hydraulic conductivity of the medium. This value represents the ease or difficulty a fluid has passing through a porous medium. The minus sign conveys that flow occurs in the direction of decreased head in the manometers. For clarification, (2.12) can be stated as:

Rate of flow through a cross sectional area is proportional to hydraulic gradient. (2.13)

This result is only valid for 1-D flow. To generalize the results for 3-D flow, we describe an anisotropic material with the conductivity tensor $\mathbf{K}$. The i, j coefficient entry of $\mathbf{K}$ describes flow in the i direction in response to a unit gradient in the $-j$ direction.

$$\begin{aligned} q_x &= -K_{xx}\frac{\partial h}{\partial x} - K_{xy}\frac{\partial h}{\partial y} - K_{xz}\frac{\partial h}{\partial z}, \\ q_y &= -K_{yx}\frac{\partial h}{\partial x} - K_{yy}\frac{\partial h}{\partial y} - K_{yz}\frac{\partial h}{\partial z}, \\ q_z &= -K_{zx}\frac{\partial h}{\partial x} - K_{zy}\frac{\partial h}{\partial y} - K_{zz}\frac{\partial h}{\partial z} \end{aligned}$$

and Darcy's Law for 3-dimensions is given by:

$$\boldsymbol{q} = -\mathbf{K} \cdot \nabla \boldsymbol{h}.$$

In our examples, we do not have anisotropic materials. Instead, the flow response for any given gradient is only along the lines of that gradient and is uniform in all directions. Therefore, for any material the conductivity tensor can be replaced with a single coefficient κ. We note the flux of a liquid is equivalent to the volume fraction (ε) of that phase times the velocity. We see the Darcy flux can be represented

$$\boldsymbol{q} = -\kappa \nabla \boldsymbol{h} = \varepsilon \boldsymbol{v}. \tag{2.14}$$

In later discussions, this isotropy is very important as it allows us to characterize the hydraulic conductivity throughout the domain as a piecewise constant function where each material has a single scalar conductivity value, not a tensor. This, along with the linearity of the differential equation, allows us to recombine the matrix and vector as shown in (3.13).

2.1.5 Complete Single Phase Flow Equation for Steady State Domains

We substitute (2.14) in (2.11) and have:

$$\frac{\partial}{\partial t}(\varepsilon\rho) = -\nabla \cdot (-\kappa\ \rho\ \nabla \boldsymbol{h}) + S. \tag{2.15}$$

In the following examples, we will only examine steady state problems ($\frac{\partial}{\partial t}(\varepsilon\rho) \equiv 0$). While field systems are rarely at steady state, for conditions in which boundary conditions and sources and sinks are invariant with time, heads will tend toward a steady value with increasing time. We will assume steady state conditions in this work, but the methods to be explored can be expended to the transient case.

These assumptions give us the following equation for the hydraulic head ($\boldsymbol{h}$) of a fluid in an isotropic, single fluid domain given a source term S, hydraulic conductivity κ, and density ρ:

$$-\nabla \cdot (\kappa\rho\nabla \boldsymbol{h}) = S.$$

We assume that spatial gradients in density of our fluid are small in comparison to gradients in head of the fluid and conductivity of the materials. Therefore, we assume ρ is constant and our final equation is:

$$-\nabla \cdot (\kappa\nabla \boldsymbol{h}) = S. \tag{2.16}$$

Chapter 3

Single-Phase Groundwater Model

We apply our method to both 2-D and 3-D domains. The 2-D domain serves only as a proof-of-concept and is on a small scale to be run easily within `Matlab`. The parameters in the 2-D example serve only to test our algorithm; they are not representative of actual data or scenarios.

Our 3-D examples are a mix of simulations and actual data. When we use simulated domains, the scale of the domain is intended to represent possible "real-world" scenarios and all values approximate those we would experience (with the assumptions previously described) in field tests.

3.1 2-D Model

In our 2-D example, we investigate a square domain with one kilometer length on each side. We directly assign hydraulic head values with a Dirichlet condition on the left and right sides of the domain. The top and bottom of the domain are restricted with homogenous Neumann ("no flow") conditions. The pumping wells in the domain are represented with the term $g(\overline{x}, \overline{y})$, where $(\overline{x}, \overline{y})$ is the location of the well.

$$\begin{aligned} -\nabla\cdot(\kappa(x,y)\nabla h(x,y)) &= g(x,y) \qquad (3.1)\\ h(0,y) &= h_0 \qquad (3.2)\\ h(1,y) &= h_1 \qquad (3.3)\\ \frac{\partial h}{\partial n}(x,0) &= 0 \qquad (3.4)\\ \frac{\partial h}{\partial n}(x,1) &= 0. \qquad (3.5)\end{aligned}$$

We let $h_0 = .4\ km,\ h_1 = 1km$. The conductivities of the sixteen equally spaced, isotropic materials are in the range $\kappa(x,y) \in [10^{-10}, 10^{-2}]$. We place two wells $g(x,y)$ that pump at a rate of $\approx 6\times 10^{-2}\ m^3/s$. These parameters are designed solely as a proof-of-concept for our method and do not represent any real-world example.

To generate the hydraulic head $h(x,y)$, we use the finite element implementation coded by [38] to discretize and solve (3.1)-(3.5) with

$$\mathbf{A}\boldsymbol{h} = \boldsymbol{f}. \qquad (3.6)$$

3.2 3-D Model

Our 3-D examples are also saturated and in steady-state. The materials are all isotropic and we still have only a single fluid phase. Unlike the 2-D example, we do not include any wells. That is, $g(\boldsymbol{x}) \equiv 0$ from (3.1). To ensure that the hydraulic head $\boldsymbol{h}$ are uniquely defined by material conductivities κ, we must introduce a flow boundary condition. The total flux boundary condition is stated here in (3.7c). The derivation and need for this boundary condition is given in (§4). Our system is

$$\begin{aligned} -\nabla\cdot(\kappa(\boldsymbol{x})\nabla h(\boldsymbol{x})) &= 0 \qquad (3.7a)\\ h(\boldsymbol{x}) &= \alpha \text{ on } \Gamma_D \qquad (3.7b)\\ \int_{\Gamma_Q}(\kappa(\boldsymbol{x})\nabla h(\boldsymbol{x}))\cdot n\, dS &= q, \qquad (3.7c)\end{aligned}$$

where

$$\boldsymbol{x} \in \Omega \subset \mathbb{R}^3, \tag{3.8a}$$

$$\boldsymbol{h} \in \mathbb{R}^N, \tag{3.8b}$$

$$\partial\Omega = \Gamma_D \cup \Gamma_Q, \tag{3.8c}$$

$$\Gamma_D \cap \Gamma_Q = \emptyset. \tag{3.8d}$$

To discretize and solve our 3-D model, we use `ADH`, developed at the United States Army Corps of Engineers (USACE) lab in Vicksburg, MS. In Appendix C, we discuss what steps were taken to extract the necessary information and the relevant model settings we use. Just as in the 2-D example (albeit with significantly more complexity in 3-D), we build the matrix $\mathbf{A}$ and vector $\boldsymbol{f}$ to solve

$$\mathbf{A}\boldsymbol{h} = \boldsymbol{f}.$$

3.3 Approximating the Hydraulic Conductivities

We represent the conductivity field as a piecewise constant function. In our discussion of the optimization process (§6) we investigate how this representation of the conductivities affects our solution. We let $\chi_i(\boldsymbol{x})$ be a characteristic function that indicates where material i (and therefore hydraulic conductivity κ_i) appears in the domain and depict the hydraulic conductivity in the domain as

$$\kappa(\boldsymbol{x}) = \sum_i \kappa_i \chi_i(\boldsymbol{x}) \tag{3.9a}$$

$$\sum_i \chi_i(\boldsymbol{x}) = \boldsymbol{1}. \tag{3.9b}$$

In both the 2-D and 3-D examples, we use finite elements to discretize the weak form of the system of equations. Appendix A describes how the finite element method achieves this task. In general, we multiply the flow equation by a test function v, integrate, and, using Green's identity, reach the weak form of the flow equation:

$$\int_\Omega \kappa(\boldsymbol{x}) \nabla h(\boldsymbol{x}) \nabla v(\boldsymbol{x}) d\boldsymbol{x} = \int_\Omega f(\boldsymbol{x}) v(\boldsymbol{x}) d\boldsymbol{x}. \tag{3.10}$$

The discretized form of (3.10) is represented as a linear system

$$\mathbf{A}\boldsymbol{h} = \boldsymbol{f}.$$

The isotropic, piecewise constant conductivity function allows us to separate the single operator $\mathbf{A}$, $\boldsymbol{f}$ into an operator on each material, seen by substituting (3.9) into (3.10):

$$\sum_i \int_\Omega \chi_i(\boldsymbol{x})\kappa_i \nabla h(\boldsymbol{x}) \nabla v(\boldsymbol{x}) d\boldsymbol{x} = \int_\Omega f(\boldsymbol{x}) v(\boldsymbol{x}) d\boldsymbol{x}. \tag{3.11}$$

We let

$$\mathbf{A}_i = \int_\Omega \chi_i(\boldsymbol{x}) \nabla h(\boldsymbol{x}) \nabla v(\boldsymbol{x}) d\boldsymbol{x} \tag{3.12}$$

Dirichlet boundary conditions are implemented by directly changing the stiffness matrix $\mathbf{A}$. Those changes are contained in $\mathbf{A}_0$. In Appendix C, we describe the construction of each sub vector $\boldsymbol{f}_i$. So, our matrix and vector can be described:

$$\mathbf{A} = \mathbf{A}_0 + \sum_i \mathbf{A}_i \kappa_i \tag{3.13a}$$

$$\boldsymbol{f} = \boldsymbol{f}_0 + \sum_i \boldsymbol{f}_i \kappa_i. \tag{3.13b}$$

3.4 Comparison to Data

The purpose of the methods described in later chapters of this document is to find a set of hydraulic conductivities that best fit provided hydraulic head data. The residual vector $\mathcal{R}$ contains the difference between the model approximation of hydraulic head $\boldsymbol{h}$ given hydraulic conductivity κ and data $\boldsymbol{d}$ for M points is

$$\mathcal{R}(\kappa)_i = h_i - d_i, \ i = 1, ..., M. \tag{3.14}$$

Our methods will minimize the least-squares error:

$$\frac{1}{2} min ||\mathcal{R}(\kappa)||_2^2 = \frac{1}{2} min \left(\mathcal{R}(\kappa)^T \mathcal{R}(\kappa) \right). \tag{3.15}$$

Chapter 4

Total Flux Boundary Condition

In this chapter, we discuss the formulation, construction, and solution of the “Total Flux” boundary used in 3.7c. The experimental data that we wish to compare to consists of a set of heads over a heterogeneous domain in which a known quantity of fluid flows across one boundary and a Dirichlet condition is maintained at the opposite face with no flow conditions on the other four boundaries. The ideas are similar in two dimensions with two no flow boundaries. The flux boundary is specified in integral form as the total flux over the face, or edge in two dimensions, of the domain. An additional constraint is that the head is constant, but unknown, across the flux boundary.

Because this flux boundary condition is not known everywhere on the domain as with typical second-kind boundary conditions, it is necessary to derive the appropriate boundary conditions in a non-standard manner. Conceptually, we seek a constant Dirichlet boundary condition such that the specified integral flux condition is met. The linearity of the problem makes this a straightforward calculation.

4.1 Problem Statement

4.1.1 Formulation

We develop a strategy to solve the BVP in a domain that has mixed boundary conditions. We consider

$$\nabla \cdot (\kappa(\boldsymbol{x}) \nabla h(\boldsymbol{x})) = f(\boldsymbol{x}), \tag{4.1}$$

where $\boldsymbol{x}$ is considered in the domain Ω with boundary $\partial\Omega$. The boundary has a mix of three non overlapping boundary conditions, Dirichlet (D), Neumann (N), and Flux (Q). So

$$\boldsymbol{x} \in \Omega \subset \mathbb{R}^3, \tag{4.2}$$

$$\boldsymbol{h} \in \mathbb{R}^N, \tag{4.3}$$

$$\partial\Omega = \Gamma_D \cup \Gamma_N \cup \Gamma_Q, \tag{4.4}$$

$$\Gamma_D \cap \Gamma_N \cap \Gamma_Q = \emptyset. \tag{4.5}$$

In general, there may be several boundary conditions of each type

$$\Gamma_D = \quad \Gamma_{d_1} \cup ... \cup \Gamma_{d_{Nd}}, \tag{4.6a}$$

$$\Gamma_N = \quad \Gamma_{n_1} \cup ... \cup \Gamma_{n_{Nn}}, \tag{4.6b}$$

$$\Gamma_Q = \quad \Gamma_{q_1} \cup ... \cup \Gamma_{q_{Nq}}. \tag{4.6c}$$

Each type of boundary condition will be represented as shown below:

$$\Gamma_{d_i} \Rightarrow \qquad h(\boldsymbol{x}) = \alpha_{d_i}, \; \boldsymbol{x} \in \Gamma_{d_i}, \tag{4.7a}$$

$$\Gamma_{n_i} \Rightarrow \qquad \tfrac{\partial h(\boldsymbol{x})}{\partial n} = \beta_{n_i}, \; \boldsymbol{x} \in \Gamma_{n_i}, \tag{4.7b}$$

$$\Gamma_{q_i} \Rightarrow \quad \textstyle\int_{\Gamma_{q_i}} (\kappa(\boldsymbol{x})\nabla h(\boldsymbol{x})) \cdot n \, dS = \phi_{q_i}, \; \boldsymbol{x} \in \Gamma_{q_i}. \tag{4.7c}$$

The solution to (4.1) is given by

$$\boldsymbol{h} = \boldsymbol{h}_{DN} + \sum_{Nq} \gamma_i \boldsymbol{h}_{q_i} \in \mathbb{R}^N, \tag{4.8}$$

where $\boldsymbol{h}_{DN}$ and $\boldsymbol{h}_{q_i}$ are solutions to (4.1) with boundary conditions modified as follows:

- To solve for $\boldsymbol{h}_{DN}$, we leave Γ_D and Γ_N as defined in (4.6), (4.7), but force $\boldsymbol{h} = 0$

on Γ_Q:

$$\begin{aligned}\nabla \cdot (\kappa(\boldsymbol{x})\nabla h(\boldsymbol{x})) = & \quad f(\boldsymbol{x}) \\ \Gamma_{d_i} \Rightarrow & \quad h(\boldsymbol{x}) = \alpha_{d_i},\ \boldsymbol{x} \in \Gamma_{d_i}, \\ \Gamma_{n_i} \Rightarrow & \quad \tfrac{\partial h(\boldsymbol{x})}{\partial n} = \beta_{n_i},\ \boldsymbol{x} \in \Gamma_{n_i}, \\ \Gamma_{q_i} \Rightarrow & \quad h(\boldsymbol{x}) = 0\end{aligned}$$

- To compute $\boldsymbol{h}_{q_i}$, we set $\boldsymbol{h} = 0$ on Γ_D, $\dfrac{\partial \boldsymbol{h}}{\partial n} = 0$ on Γ_N, and $\boldsymbol{h} = 1$ on Γ_Q:

$$\begin{aligned}\nabla \cdot (\kappa(\boldsymbol{x})\nabla h(\boldsymbol{x})) = & \quad f(\boldsymbol{x}) \\ \Gamma_{d_i} \Rightarrow & \quad h(\boldsymbol{x}) = 0, \\ \Gamma_{n_i} \Rightarrow & \quad \tfrac{\partial h(\boldsymbol{x})}{\partial n} = 0, \\ \Gamma_{q_i} \Rightarrow & \quad h(\boldsymbol{x}) = 1\end{aligned}$$

We fix γ_i to satisfy the flux boundary condition on each relevant boundary. The flux through each boundary Γ_{q_j} is given by

$$\int_{\Gamma_{q_j}} (\kappa(\boldsymbol{x})\nabla h_{DN}(\boldsymbol{x})) \cdot n\, dS + \sum_{i=1}^{N_q} \gamma_i \int_{\Gamma_{q_j}} (\kappa(\boldsymbol{x})\nabla h_{q_i}(\boldsymbol{x})) \cdot n\, dS = \phi_{q_j},\ j = 1...N_q, \qquad (4.9)$$

and these N_q equations provide the relationship for γ_i.

4.1.2 Boundary Value Map

The following discusses how we compute the flux for the boundary condition in (4.7c). For sake of clarity, the examples will be done in 1-D but can be expanded to 2-D and 3-D domains.

To calculate the boundary flux terms $\left(\int_{\Gamma_{q_j}} \kappa(\boldsymbol{x})\nabla h_*(\boldsymbol{x})\ dS\right)$, we turn to the derivation in [4]. For all points $x \in [0, \ell]$, we represent the flux at that point with

$$\sigma(x) = -\kappa(x)\frac{dh(x)}{dx}. \qquad (4.10)$$

In the absence of change in mass due to reactions and phase changes, flux must be conserved through the boundaries. That is, all mass passes through a boundary or is accounted for by an internal source of flux $g(x)$. Over a 1-D domain $x \in [0, \ell]$, the conservation of flux demands

$$\sigma(\ell) + \sigma(0) = \int_0^\ell g(x)\, dx. \tag{4.11}$$

At the boundaries of the domain, we denote the flux σ_0, σ_ℓ. If we take any arbitrarily small slice of the domain $x \in [0, a]$, flux is conserved on that segment if

$$\sigma(a) + \sigma_0 = \int_0^a g(x)\, dx.$$

As we take $a \to 0$, we can see that $\sigma(0) = \sigma_0$. Similarly, we can show $\sigma(\ell) = \sigma_\ell$, where σ_ℓ is the prescribed flux through the right hand boundary. Thus,

$$\sigma_\ell + \sigma_0 = \int_0^\ell g(x)\, dx.$$

Construction of Map

We construct a method that will explicitly compute the flux $\sigma_0,\ \sigma_\ell$ through each Dirichlet boundary using information constructed by the full model simulator. This can be done with the information normally discarded in the finite element formulation of the load matrix.

For the sake of clarity, we return to a simple 1-D domain and use linear shape functions u. Each element e is bounded by two nodes and, locally, we can represent the equations as

$$\begin{aligned} a_{11}^e u_1^e + a_{12}^e u_2^e &= f_1^e + \sigma(s_1^e), \\ a_{21}^e u_1^e + a_{22}^e u_2^e &= f_2^e - \sigma(s_2^e), \end{aligned}$$

where $\sigma(s_i^e)$ is the flux at node s_i^e and a_{ij}^e are the contributions from element e to the (local) matrix at location i, j. Extending this investigation to two adjacent nodes, one of which is on the boundary $x = 0$, we see the following system of equations develop

$$\begin{bmatrix} a^1_{11}u_1 & + & a^1_{12}u_2 & & & = & f^1_1 & + & \sigma(0) \\ a^1_{21}u_1 & + & (a^1_{22}+a^2_{11})u_2 & + & a^2_{12}u_3 & = & f^1_2+f^2_1 & - & \sigma(x_1^-)+\sigma(x_1^+) \\ & & a^2_{21}u_2 & + & a^2_{22}u_3 & = & f^2_2 & - & \sigma(x_2^-) \end{bmatrix}$$

We represent the jump in flux at each node with $[[\sigma(x_i)]] = \hat{f}_i$, where $\hat{f}_i$ is zero except in the case of a prescribed source. We see that the vector contains all of the boundary flux information, as well as any interior sources.

For Dirichlet nodes, the information in the matrix is replaced by a one on the diagonal and zero elsewhere in the corresponding row. Similarly, the vector entry that corresponds to a Dirichlet node is replaced with the prescribed Dirichlet boundary value. This discarded information provides the necessary equations to ascertain the flux through a given boundary.

For example, in the 1-D case, the entries in the matrix and vector for a Dirichlet node would be

$$A_{11}u_1 + A_{12}u_2 = F_1 + \sigma(0), \tag{4.12a}$$

$$A_{N,N_1}u_{N-1} + A_{NN}u_N = F_N - \sigma(\ell). \tag{4.12b}$$

We store this information separately. After we have solved the system of equations to yield u_1, u_2, u_{N-1}, u_N, we return to (4.12) to obtain $\sigma(0), \sigma(\ell)$. We note (4.12) can be extended to the 2-D and 3-D cases by maintaining the corresponding matrix / vector structure and equations for the boundary nodes.

Once we have solved for each of the sub-solutions in (4.8), the flux value on any boundary Γ_{q_j} is given by

$$\left(\int_{\Gamma_{q_j}} \kappa(x)\nabla h_*(x)\; dS\right) = \sum \sigma_{h_*}(x_k),\; x_k \in \Gamma_{q_j}, \tag{4.13}$$

where $\sigma_{h_*}(x_k)$ represents the flux at x_k calculated from solution h_{q_i}.

Chapter 5

Proper Orthogonal Decomposition

Proper Orthogonal Decomposition (POD) is one way to construct a reduced order model. The purpose of such models is to replace an expensive simulation within an optimization loop with a surrogate that is inexpensive to evaluate but still accurate enough to improve the fit to data. A typical approach [11] is to perform an optimization based on the surrogate and then evaluate the expensive simulation one or more times, both to decide whether or not to terminate the optimization and to update the surrogate for the next pass through the optimization loop.

5.1 POD Background

POD is a classical approach with roots in the control of fluids [50, 60, 63, 41]. POD has been developed using several different techniques. The earliest development of POD was under the name "Principal Component Analysis" (PCA) by Karl Pearson in 1901[74]. Using a different technique, Kari Karhunen and Michel Loéve developed the Karhunen-Loéve Decomposition (KLD) in 1955 [61]. Another method of performing POD is through use of the Singular Value Decomposition (SVD). The SVD was first introduced in 1873 and 1874 by Eugenio Beltrami [6, 10] and Camille Jordan [48, 49]. An excellent review of the early history of the SVD is given by [82]. The current method of calculating the SVD was presented by Gene Golub and William Kahan [39]. The three techniques for deriving the POD are shown to be equivalent in [59].

There has been other work using POD in hydrology. In [86] the authors build a two-dimensional POD model based on a time-dependent finite-difference simulator, and, assuming knowledge of the conductivity tensor, use that model as a predictive tool. One can also use the POD projection to construct useful preconditioners [64]. Control applications have been reported in [85, 14, 63, 62, 3, 57]. POD has frequently been used as a reduced order model in turbulent flows [18, 7].

POD is a surrogate for the simulator that uses the simulator's own discretizations and physics models. The advantages of POD over approaches that model the objective function ϕ

$$\phi(\kappa) = \frac{1}{2}\|\mathcal{R}(\kappa)\|^2, \tag{5.1}$$

such as neural networks or interpolatory models [76, 20, 78, 43], is that POD directly uses the finite element discretization in hand and can exploit the least-squares structure via simple projections. The disadvantage is that, as with the sensitivity equations approach to computing the Jacobian, one must modify the finite element code to extract information. We have done that with the `ADH` simulator.

5.2 Classic Proper Orthogonal Decomposition

In the applications of POD listed above, one begins with a set of snapshots. Snapshots are solutions of the PDE corresponding to an array of parameter values and evaluated at various times of the solution evolution [13]. There is currently no rigorous method for constructing the set of snapshots [42]. However, it is very important that the full solution can be well approximated in the span of the snapshot set [13]. If an aspect of the true solution is not contained in the snapshot set, it cannot be approximated by the POD model.

There will be much redundant information contained in the snapshot set. It is typical to use the SVD [68, 27, 59] to reduce the basis and obtain a set of basis vectors that contain most of the relevant information in the snapshot set. The following shows how to approximate the error present in a truncated POD basis.

Error in POD Projection We start with the set of n snapshot vectors $\boldsymbol{s}_i,\ i = 1...n$ where each $\boldsymbol{s}_i \in \mathbb{R}^N$ and form the matrix

$$\mathbf{S} = [\boldsymbol{s}_1, ..., \boldsymbol{s}_n] \in \mathbb{R}^{N\times n}.$$

The SVD of that matrix is

$$\mathbf{S} = \mathbf{U}\Sigma\mathbf{V}^T,$$

where Σ is a diagonal matrix with entries $\Sigma = diag(\sigma_1, ..., \sigma_n)$, $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$. $\mathbf{U}$, the left hand singular matrix, is an orthogonal matrix with the same range as $\mathbf{S}$. $\mathbf{V}$ holds the right hand singular vectors.

The POD basis is formed from the d eigenvectors in $\mathbf{U}$ corresponding to the largest d eigenvalues. To determine d, we note the following error approximation algorithm.

Let $\mathbf{P}_U \boldsymbol{s}_i$ be the projection of the snapshot $\mathbf{s}_i$ onto the span of the POD basis from $\mathbf{U}$. The error of this approximation can be given by

$$\varepsilon_i = |s_i - P_U s_i|^2 .$$

The total error for the approximations of each snapshot is given by the sum of the discarded eigenvalues

$$\varepsilon = \sum_{i=1}^{n} \varepsilon_i = \sum_{i=1}^{n} |\boldsymbol{s}_i - \mathbf{P}_U \boldsymbol{s}_i|^2 .$$

As shown in [42, 59, 56] and others, this error can also be represented

$$\varepsilon = \sum_{i=d+1}^{n} \sigma_i^2 .$$

Therefore, to find a basis with approximation error less than some prescribed error δ, we must find the smallest d such that

$$\frac{\sum_{i=1}^{d} \sigma_i^2}{\sum_{i=1}^{n} \sigma_i^2} \geq 1 - \delta.$$

As shown in [42, 13, 17] and others, one normally chooses $\delta \geq .9$, signifying the reduced basis is able to capture 90% of the physics contained in the snapshot set.

5.3 POD For Saturated Groundwater Models

We investigate saturated groundwater models at steady state, so there is no time progression in the simulation. Rather than run the full-model with several different parameters to generate a snapshot set, we will use a single full-model solution and the sensitivity

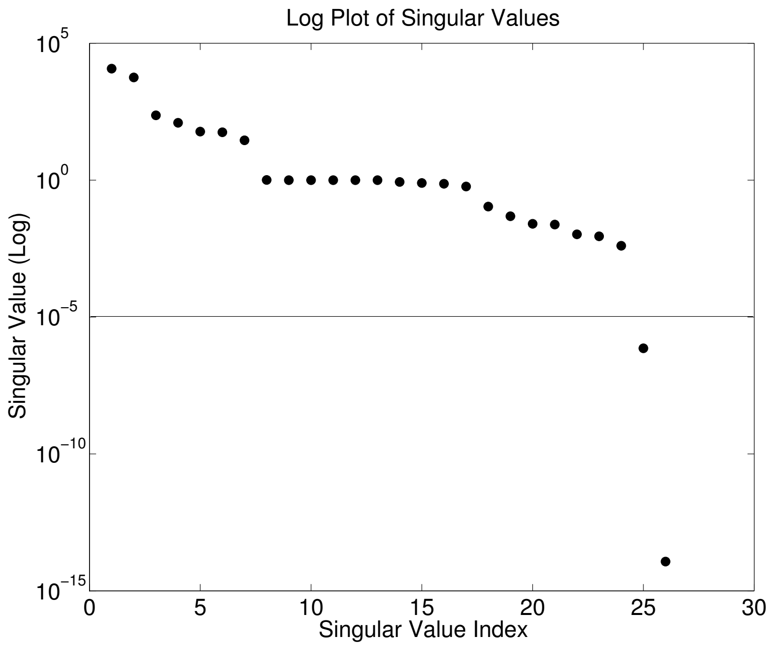


Figure 5.1: Distribution of Singular Values

vectors of that solution to the parameters as a basis. We scale the basis by solving the solution and sensitivities with log transfomed conductivities [15]. For our application, we utilize the total flux boundary condition described in §4. As such, we will have at least two sub-solutions and the sensitivities of those sub solutions in the basis.

To prune the size of our basis, we still utilize information provided by the SVD. We do not analyze the projection error detailed in §5.2, but instead note there is a natural gap in the magnitude of eigenvalues and discard those vectors corresponding to eigenvalues smaller than 10^{-5}. Figure 5.1 gives an example of the distribution of magnitudes for singular values for the application in §7.2.2 after two iterations. This distribution is typical of any moment in the optimization process. The implementation of the `SCALAPACK` SVD is discussed in Appendix E.

5.3.1 The POD basis

The following will discuss the calculation of the $2N + 2$ POD basis. We use the sub-solutions of (4.8) and their sensitivities to the K material parameters (5.5) to form

$$\mathcal{B} = span\left(\boldsymbol{h}_{DN}, \{\partial \boldsymbol{h}_{DN}/\partial \kappa_k\}_{k=1}^{K}, \boldsymbol{h}_{Q}, \{\partial \boldsymbol{h}_{Q}/\partial \kappa_k\}_{k=1}^{K}\right). \tag{5.2}$$

Unlike the use of POD for time-dependent problems, we need not create a basis by applying a singular value decomposition to a sequence of "snapshots". Instead, we apply the singular value decomposition to this basis $\mathcal{B}$ and keep only those singular vectors that correspond to singular values greater than 10^{-5}. The orthonormal basis for the significant singular vectors of $\mathcal{B}$ are stored in the columns of $\mathbf{U}$.

Basis Expansion

As we perform the optimization algorithm described in §8, we do not discard the basis information from one iteration to the next. Instead, the new vectors of (5.2) are added to the previous basis and the entire system is orthogonalized. Vectors corresponding to small singular vectors are discarded and the new, expanded basis is used in the optimization routine.

By reusing the previous basis information, the reduced order model becomes more accurate as we iterate and the basis becomes more robust. We note that additional basis vectors increase the dimension of our reduced order model. Our parallel implementation of the SVD (Appendix E) ensures that even for bases with ≈ 100 vectors, the reduced order model is significantly faster than the full model.

Sensitivity Scaling

A basis consisting of solutions and sensitivities must be appropriately scaled, otherwise, a singular value decomposition will arbitrarily return small singular values [15]. Our calculation of the sensitivities is scaled by solving against the log-transformed conductivities. That is,

$$\kappa_k = e^{p_k},$$
$$\frac{\partial \kappa_k}{\partial p_k} = e^{p_k},$$

and our basis (5.2) is

$$\mathcal{B} = span\left(\boldsymbol{h}_{DN}, \{\partial \boldsymbol{h}_{DN}/\partial p_k\}_{k=1}^{K}, \boldsymbol{h}_Q, \{\partial \boldsymbol{h}_Q/\partial p_k\}_{k=1}^{K}\right). \tag{5.3}$$

We note the vector norm for the log-transformed sensitivities are within an order of magnitude of one another. Also, each sensitivity vector is within two orders of magnitude of the solution vector. Due to the relative similarity in norm, we are assured the SVD will not discard information from poor scaling.

5.3.2 Sensitivity Calculation

We are able to compute the sensitivity of the solution to the K material parameters without significant additional computational effort. We recall the solution to our BVP is given by (4.8) and shown again here:

$$\boldsymbol{h} = \boldsymbol{h}_{DN} + \sum_{i=1}^{N_q} \gamma_i \boldsymbol{h}_{q_i} \in \mathbb{R}^N, \tag{5.4}$$

We differentiate with respect to the log-transformed parameters p_k to see the sensitivities are

$$\frac{\partial \boldsymbol{h}}{\partial p_k} = \frac{\partial \boldsymbol{h}_{DN}}{\partial \kappa_k}\frac{\partial \kappa_k}{\partial p_k} + \sum_{i=1}^{N_q}\left(\gamma_i \frac{\partial \boldsymbol{h}_{q_i}}{\partial \kappa_k}\frac{\partial \kappa_k}{\partial p_k} + \frac{\partial \gamma_i}{\partial \kappa_k}\frac{\partial \kappa_k}{\partial p_k}\boldsymbol{h}_{q_i}\right), \tag{5.5}$$

or

$$\frac{\partial \boldsymbol{h}}{\partial p_k} = e^{p_k}\cdot\frac{\partial \boldsymbol{h}_{DN}}{\partial \kappa_k} + \sum_{i=1}^{N_q}\left(\gamma_i e^{p_k}\cdot \frac{\partial \boldsymbol{h}_{q_i}}{\partial \kappa_k} + e^{p_k}\cdot\frac{\partial \gamma_i}{\partial \kappa_k}\boldsymbol{h}_{q_i}\right). \tag{5.6}$$

Flux Coefficient Sensitivities

To compute $\dfrac{\partial \gamma_j}{\partial \kappa_k}$, we analyze the response each of the N_q equations that satisfy the flux boundary condition (4.9) (shown again here) against the K material properties κ_k:

$$\int_{\Gamma_{q_j}} (\kappa(\boldsymbol{x})\nabla \boldsymbol{h}_{DN}(\boldsymbol{x}))\cdot n\, dS + \sum_{i=1}^{N_q}\gamma_i \int_{\Gamma_{q_j}} (\kappa(\boldsymbol{x})\nabla h_{q_i}(\boldsymbol{x}))\cdot n\, dS = \phi_{q_j},\ j = 1...N_q, \tag{5.7}$$

to yield (5.8). Recall from §2 that we use zonation to represent the material conductivity field. Thus,

$$\begin{aligned} \kappa(\boldsymbol{x}) &= \textstyle\sum_k \kappa_k \chi_k(\boldsymbol{x}) \\ \frac{\partial \kappa(\boldsymbol{x})}{\partial \kappa_k} &= \chi_k(\boldsymbol{x}) \end{aligned}$$

Also, the flux through the boundary is set as a boundary condition and does not respond to perturbations in the material parameters:

$$\frac{\partial \phi_{q_j}}{\partial \kappa_k} \equiv 0.$$

Finally, we note the flux map in §4 can be applied to any vector in the solution space. That is, we can apply our technique to

$$\int_{\Gamma_{q_j}} (\kappa(\boldsymbol{x}) \nabla h_*(\boldsymbol{x})) \cdot n \, dS$$

as well as to

$$\int_{\Gamma_{q_j}} \left(\kappa(\boldsymbol{x}) \nabla \frac{\partial h_*(\boldsymbol{x})}{\partial \kappa_k} \right) \cdot n \, dS.$$

The system of equations that must be solved for $\dfrac{\partial \gamma_j}{\partial p_k}$ is given by:

$$\begin{aligned} & \tfrac{\partial}{\partial p_k} \left(\textstyle\int_{\Gamma_{q_j}} (\kappa(\boldsymbol{x}) \nabla h_{DN}(\boldsymbol{x})) \cdot n \, dS \right) \\ + \; & \textstyle\sum_{i=1}^{N_q} \left(\gamma_i \tfrac{\partial}{\partial p_k} \left(\textstyle\int_{\Gamma_{q_j}} (\kappa(\boldsymbol{x}) \nabla h_{q_i}(\boldsymbol{x})) \cdot n \, dS \right) \right. \\ + \; & \left. \tfrac{\partial \gamma}{\partial p_k} \left(\textstyle\int_{\Gamma_{q_j}} (\kappa(\boldsymbol{x}) \nabla h_{q_i}(\boldsymbol{x})) \cdot n \, dS \right) \right) = 0, \\ & \text{for } j = 1 \ldots N_q. \end{aligned}$$

We solve this system for $\dfrac{\partial \gamma_j}{\partial p_k}$ for each $j = 1 \ldots N_q$. It is noted that N_q is very small and solving for these coefficient sensitivities does not significantly affect the time to solution of the method.

Solution Sensitivities

We compute $\dfrac{\partial \boldsymbol{h}_*}{\partial p_k}$ analytically (where $\boldsymbol{h}_*$ represents both $\boldsymbol{h}_{DN}$ and all solutions $\boldsymbol{h}_{q_j}$). We do this by exploiting zonation and the linearity of the problem. We note that all work done to precondition and factor the matrix in the original solve need not be redone for the sensitivity computation. We reuse the same matrix factorization, thus adding very little computational effort to the process.

The finite element discretization of (4.1) for any of the boundary conditions in (4.7) will yield

$$\mathbf{A}\boldsymbol{h}_* = \boldsymbol{f}_*. \tag{5.8}$$

We note the structure and values in $A(\kappa)$ are not changed, regardless of the values on the boundary. We are not manipulating the structure of the problem when we solve $\boldsymbol{h}_*$, only the value of the boundary conditions. The boundary condition information is contained in $\boldsymbol{f}_*$. Appendix C describes the derivation of the terms in (5.8) and how the boundary information affects $\boldsymbol{f}_*$.

As previously described, we can represent the matrix $\mathbf{A}$ and each vector $\boldsymbol{f}_*$ as a linear combination of sub matrices and vectors:

$$\mathbf{A} = \mathbf{A}_0 + \textstyle\sum_k \mathbf{A}_k \kappa_k \tag{5.9a}$$

$$\boldsymbol{f}_*(\kappa) = \boldsymbol{f}_{*0} + \textstyle\sum_k \boldsymbol{f}_k \kappa_k. \tag{5.9b}$$

Again, we note that the information for the boundaries is contained in $\boldsymbol{f}_{*0}$, each $\boldsymbol{f}_k$ is constant. $\mathbf{A}_0$ contains the information for the Dirichlet nodes and contains only the entries 1 and 0 on the diagonal, depending on whether the corresponding node is constrained by a Dirichlet boundary condition. Similarly, the entries in $\boldsymbol{f}_{*0}$ hold the values at those nodes.

We calculate the sensitivity of each solution $\boldsymbol{h}_*$ to the parameters κ_k by differentiating (5.8) and substituting the information from (5.9).

$$\mathbf{A}\frac{\partial \boldsymbol{h}_*}{\partial \kappa_k} = \boldsymbol{f}_k - \mathbf{A}_k \boldsymbol{h}_*. \tag{5.10}$$

It is important to note that (5.10) can be readily solved as $\mathbf{A}$ has not been changed. The work done in assembly and factorization to solve (5.8) is not repeated in the multiple solves required in (5.10).

5.3.3 POD Reduced Model

With coefficient and sub-solution sensitivities, we form the POD basis $\mathcal{B}$. We orthogonalize the basis with an SVD and keep only those d vectors corresponding to singular values of magnitude greater than 10^{-5}. Those orthogonal vectors are stored as the new basis $\mathcal{U} \in \mathbb{R}^{N\times d}$. We let $\mathcal{W} \in \mathbb{R}^{N\times d}$ be an orthonormal basis for $\mathbf{A}\mathcal{U}$. The reduced order model for each sub-solution is

$$\mathcal{W}^T\mathbf{A}_0\mathcal{U}\,\overline{\boldsymbol{u}}_\ell + \sum_i \kappa_i \mathcal{W}^T\mathbf{A}_i\mathcal{U}\,\overline{\boldsymbol{u}}_\ell = \mathcal{W}^T\boldsymbol{f}_0^{\{\ell\}} + \sum_i \kappa_i \mathcal{W}^T\boldsymbol{f}_i,\ \ell = DN, q_i$$

or

$$\mathcal{W}^T\mathbf{A}\mathcal{U}\overline{\boldsymbol{u}}_\ell = \mathcal{W}^T\boldsymbol{f}^{\{\ell\}}, \tag{5.11}$$

where

$$\begin{aligned}\mathcal{W}^T\mathbf{A}\mathcal{U} &\in \mathbb{R}^{d\times d},\\ \mathcal{W}^T\boldsymbol{f}^{\{\ell\}} &\in \mathbb{R}^{d},\end{aligned}$$

and the sub-matrices $\mathbf{A}_i$ are calculated as previously discussed. We note $d << N$ and the effort to solve this $d \times d$ system is significantly less than required to solve the full model.

The reduced model solution for hydraulic head is compared to data via

$$\overline{\boldsymbol{h}}_\ell = \mathcal{U}\overline{\boldsymbol{u}}_\ell,\ \ell = DN, q_i$$

and

$$\overline{\boldsymbol{h}} = \mathcal{U}\left(\overline{\boldsymbol{u}}_{DN} + \sum_i^{N_q} \gamma_i \overline{\boldsymbol{u}}_{q_i}\right). \tag{5.12}$$

The residual that we will minimize is our approximation of hydraulic head $\overline{\boldsymbol{h}}$ against the M data points d_i

$$\mathcal{R}(\kappa)_i = \overline{h}(\kappa)_i - d_i,\ i = 1, ..., M,\ h \in \mathbb{R}^N. \tag{5.13}$$

Chapter 6

Parameter Estimation

6.1 Inverse Problems in Hydrogeology

We will approximate the material parameters by applying a Levenberg Marquardt optimization code to a reduced order model generated through Proper Orthogonal Decomposition. Ours is not the only approach to solve nor model the inverse problem for groundwater problems. The survey of [67] categorizes methods by how the authors parameterize the domain, how they model the forward problem, and which optimization scheme is used to fit the parameters. Their conclusion is that the various inverse methods — including those similar to our approach — all have merit but some are more appropriate than others for a sample problem. They conclude a blocked approach to parameterization such as ours is convenient for domains in which the boundaries are distinct. In the following, we survey other approaches to construction and optimization of the minimization problem.

6.1.1 Approximating the Hydraulic Conductivities

It is challenging to accurately model hydraulic conductivity in a domain, even for saturated, single phase, steady examples [16]. Rather than attempt to construct a fine-scale conductivity field, one approach is to generate an effective "block conductivity" through upscaling [80, 47] or a stochastic approach [79]. The block homogeneity is often sufficient to characterize general hydraulic properties such as flow or transport in a large domain. We must ensure the blocks are significantly smaller than the entire domain else the model will not faithfully represent flux. The authors in [5] recommend blocks at least one tenth the size of the full domain.

Alternatively, one could parameterize the model with many small homogenous zones. If more zones than data points are used, the system is under-determined and no unique solution will exist. One can employ a regularization technique to move toward convergence to a unique solution. In [84], the authors apply both Tikhonov regularization [83, 87] and a singular value decomposition to approximate the material properties in an under-determined system. In [71], an analysis of cost of these regularization schemes is performed.

Both of the previous approaches create or assume some zones of uniform conductivity. In [30] the authors describe the construction and limitations of the “zonation” approach. They note that pilot points allow the modeler to avoid constructing the zones independent of the optimization process. Instead, as part of the parameter estimation process the location of the materials changes. In [55], the authors seek to discern which not only best fit of parameters, but also how many parameters to optimize. They provide an algorithm by which the complexity of the model is considered along with the fit to data of the end result.

Our depiction of the conductivity field is in the form of a small number of homogenous material zones. While we do not adjust the zone locations in the course of our optimization nor do we adjust the number of zones present, our method could be included during the parameter fitting phase of both of those techniques.

6.1.2 Optimization Methods

We construct a reduced order model with Proper Orthogonal Decomposition, then use a Levenberg-Marquardt [58, 65, 72, 28, 52] (LM) code `PEST` [31] to optimize parameters in that model. As noted in [45] and others, LM is widely used, but requires a large number of function calls to be effective. They use the adjoint state method to compute the derivatives of the objective function. Their approach allows them to compute the jacobian matrix for an LM method at a cost independent of the number of parameters. The construction of our reduced order model allows us to compute an analytic jacobian with very little computational effort. As noted in [91], the computation and accuracy of the jacobian vectors drive the success of the inverse problem.

Our method minimizes an objective function in order to calibrate parameters. Others have approached the problem with stochastically and sample posterior distributions of parameters [40, 90]. Markov chain Monte Carlo (McMC) methods are a popular stochastic approach to the optimization method [36]. McMC methods determine the next iteration based on a random step from the current iterate. Stochastic approaches demand significantly more function evaluations than inverse modeling and [45] shows the final results are often similar. Ensemble Kalman Filtering [35] has been suggested to improve the sampling of Monte Carlo methods in groundwater optimization. When analyzing results of Monte Carlo approximations, one must take care the optimization has converged to a solution [2].

We use a Levenberg-Marquadt optimizer in our parameter estimation. Our discretization permits us to compute analytic sensitivity vectors and we are thus able to take advantage of gradient based methods without having to compute numerical approximations of the jacobian matrix. However, many use derivative-free techniques to explore the parameter space. [34] compares derivative-free optimization techniques including genetic algorithms (GA) [26] and stencil based methods such as implicit filtering [37]. [73] benchmarks several derivative-free options with a variety of convergence tolerances for noisy problems. Our construction of the gradient does not require additional full function evaluations and we will sample the full model significantly fewer times than these derivative free options.

6.2 Levenberg-Marquardt

We minimize the error in model output versus gathered data (5.13) with `PEST` [31], a nonlinear least squares solver that utilizes the Levenberg-Marquardt method.

The Levenberg-Marquardt method [58, 65, 72, 28, 52] is a standard iterative method for nonlinear least squares problems. In the context of our applications, the iteration is

$$\vec{\kappa}_+ = \vec{\kappa}_c - \left(\nu \mathbf{I} + \mathcal{R}'(\vec{\kappa}_c)^T \mathcal{R}'(\vec{\kappa}_c)\right)^{-1} \mathcal{R}'(\vec{\kappa}_c)^T \mathcal{R}(\vec{\kappa}_c), \tag{6.1}$$

where $\vec{\kappa}_c$ is the current point, $\vec{\kappa}_+$ the next iteration, $\mathbf{I}$ is the identity matrix, $\mathcal{R}$ is the error in model output against collected data, $\mathcal{R}'$ is the response of that error to parameter perturbation (derived in (6.3)) and ν is a parameter that is adjusted as the iteration progresses [28, 52, 31]. In the small residual case, the Levenberg-Marquardt iteration is locally rapidly convergent. We have used the `PEST` [31] nonlinear least squares solver for the computations in our applications, and it has performed well.

Recall the difference $\mathcal{R}$ between our model $\boldsymbol{h}$ with parameters κ and the data $\boldsymbol{d}$ at M points is represented in (5.13):

$$\mathcal{R}(\kappa)_i = h(\kappa)_i - d_i,\ i = 1, ..., M,\ h \in \mathbb{R}^N. \tag{6.2}$$

If we differentiate (6.2) with respect to the log transformed parameters p_i we obtain

$$\frac{\partial \mathcal{R}_j}{\partial p_i} = \frac{\partial \mathcal{R}_j}{\partial \kappa_i}\frac{\partial \kappa_i}{\partial p_i} = e^{p_i}\frac{\partial h(\boldsymbol{x}_j)}{\partial p_i}. \tag{6.3}$$

Hence the columns of the Jacobian are the scaled sensitivities $\partial \boldsymbol{h}/\partial p_i$ evaluated at the M sampling points $\{\boldsymbol{x}_j\}$. We show in §5 that these sensitivities can be computed without significant additional computational effort.

6.3 Optimization Algorithm

6.3.1 Codes

1. `Mat_FEM`: A 2-D finite element model in Matlab. In the algorithm detailed below, this code is substituted for `ADH` on the 2-D example. It is not used at all in the 3-D examples.

2. `ADH`: This is the full, high-resolution 3-D finite element model. Given a set of material parameters, `ADH` returns the hydraulic head for a discretized domain.

   We have modified `ADH` to also compute the matrices and vectors needed to run the reduced POD model. `ADH` also generates, reduces, and orthogonalizes the basis each time it is run.

3. `SCALAPACK`: We use `SCALAPACK` to compute the SVD inside of `ADH` . Also, the dense system solves inside of `POD_ROM` are computed with `SCALAPACK` routines.

4. `POD_ROM`: The reduced order model. This model can only estimate behavior on the basis produced by the full resolution finite element models. Given a set of material properties, `POD_ROM` will approximate the hydraulic head on the space spanned by the basis.

The `POD_ROM` code must be initialized each time there is new information from `ADH` available. As the basis grows, this step becomes increasingly cumbersome as it is accomplished via file I/O. We use a semaphore system to eliminate the need to do this step more than once per `ADH` run. The user (or `PEST`) saves the material properties to a file, then flips the semaphore to trigger `POD_ROM` to access that file and produce an output for those properties. This process eliminates the need to continually read in static basis and matrix information during the optimization process.

5. `PEST`: The Levenberg-Marquardt least squares optimization code. `PEST` queries the model with a set of parameters, computes the model residual against the data, and updates the parameters based on (6.1). We discuss termination criteria and parameter settings for `PEST` in Appendix D.

6.3.2 Algorithm

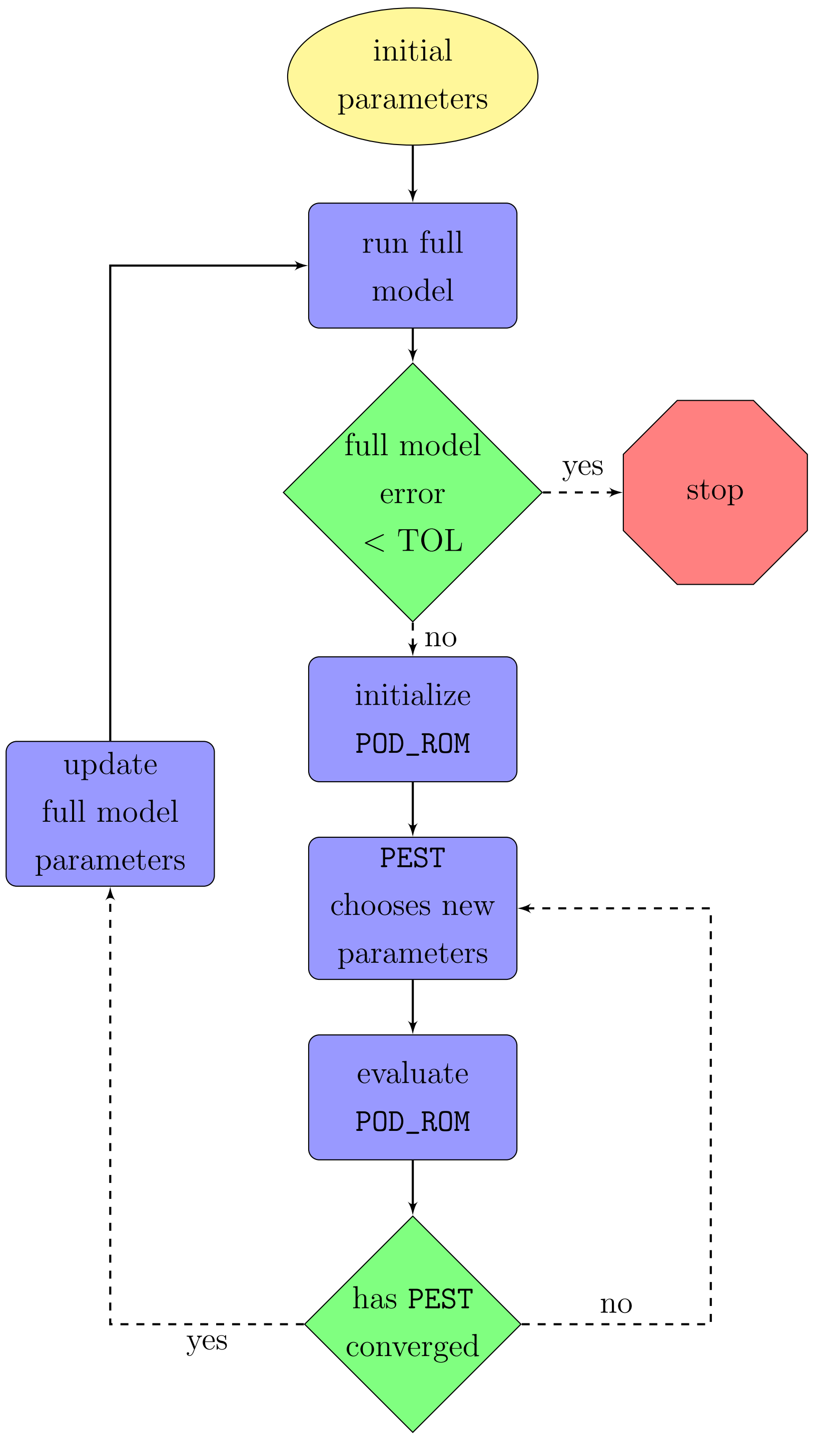


Figure 6.1: Optimization Algorithm for Full Model - Reduced Model - PEST Interaction

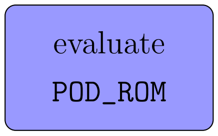
1. initial parameters : The user must select the initial iterate. Our method has been shown to work even for very poor initial iterates. There is a danger of the Levenberg-Marquardt parameter estimation process finding a local minimum, but in our results that has not been the case.

2. run full model : We always run `ADH` or `Mat_FEM` first to generate the POD basis and gather the finite element matrix information. When the `PEST` optimization process has completed, we must rerun the full model to update the basis with the new information.

3. full model error $<$ TOL : We stop the parameter estimation process when the error between the full model and the given data ($\mathcal{R}$ from (6.2)) has converged or is less than some tolerance.

4. initialize `POD_ROM` : We only need to read in the information from the full model once for each `PEST` estimation loop. We use the same parallel partitioning in both `ADH` and `POD_ROM`. This allows each processor to read in information simultaneously, pseudo-parallelizing the I/O from the full model to the reduced model.

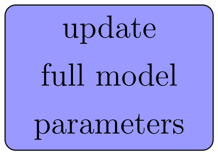
5. `PEST` chooses new parameters : Based on the residual from the output of `POD_ROM`, `PEST` selects a new set of parameter values with the Levenberg-Marquardt algorithm.

6. 

: We emphasize that PEST uses only the output from POD_ROM to perform the optimization process. As such, a reduction in the POD_ROM residual does not necessarily translate to a reduction in the residual from the full model. In our results, the first solution from PEST often increases the residual from the full model. When this new information is included in the POD basis, however, PEST quickly converges to the correct solution.

7.

: We discuss in Appendix D exactly what termination criteria we use in PEST. It is a mixture of iterations without sufficient improvement and absolute tolerance on the residual. If PEST has not yet found a solution, the parameters are updated and the reduced order model is again executed.

8. 

: Once the PEST algorithm has found a set of parameters, the full model is run with the new conductivities. Only when the output from the full model is within an acceptable tolerance of the data do we terminate the process. Otherwise, the information from the full model is used to update the basis and we reenter the PEST optimization loop.

The results [88] of this process are detailed in the following chapter.

Chapter 7

Results

7.1 2-D Results

Description of Problem

Our first implementation of POD on a groundwater system used a 2-D domain coded in `Matlab`. We use a 2-D finite element code distributed by Gockenbach [38]. The domain is a square kilometer partitioned into sixteen equal sized material zones. We placed two pumping wells in the domain with a pumping rate set by the user. We implemented a hydraulic gradient along the x-axis using dirichlet boundary conditions at the $x = 0$ and $x = 1km$ boundaries and no flow conditions in the y-direction at the $y = 0$ and $y = 1km$ boundaries. We constructed a data set using a specific conductivity field, then set about attempting to recover that data from a homogeneous initial guess. The data was sampled at M points, with those points being allocated to ensure each unique material zone would contain a minimum of three data points. Additional data points were included near the wells. An illustration of the domain with sampling points in shown in Figure 7.1.

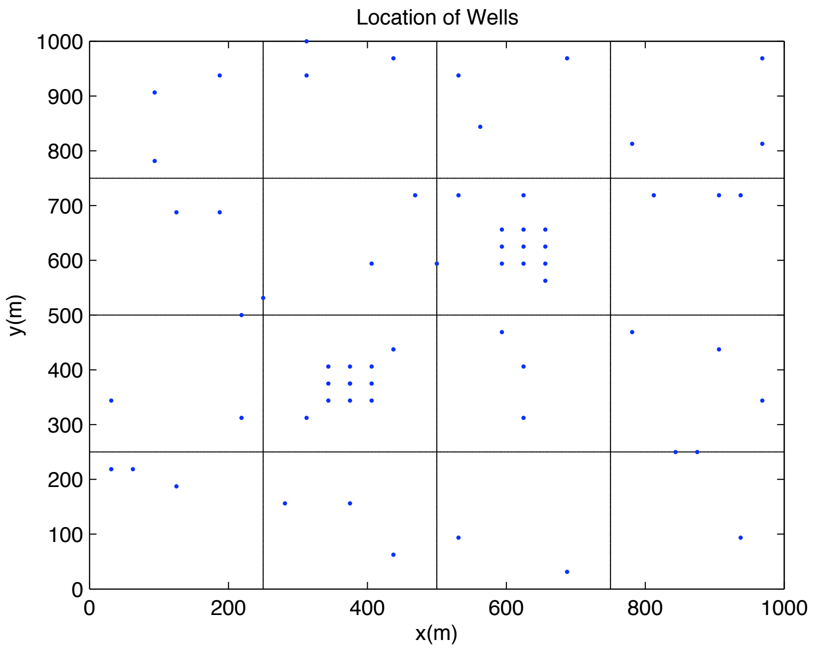


Figure 7.1: 2-D Domain with Sensors

A sample data set is shown in Figure 7.2. In this solution, we can clearly see the boundaries between the conductivity zones. Also, we can see drawdown surrounding the two pumping wells. Drawdown is a phenomenon where the hydraulic pressure decreases in the area immediately surrounding pumping wells.

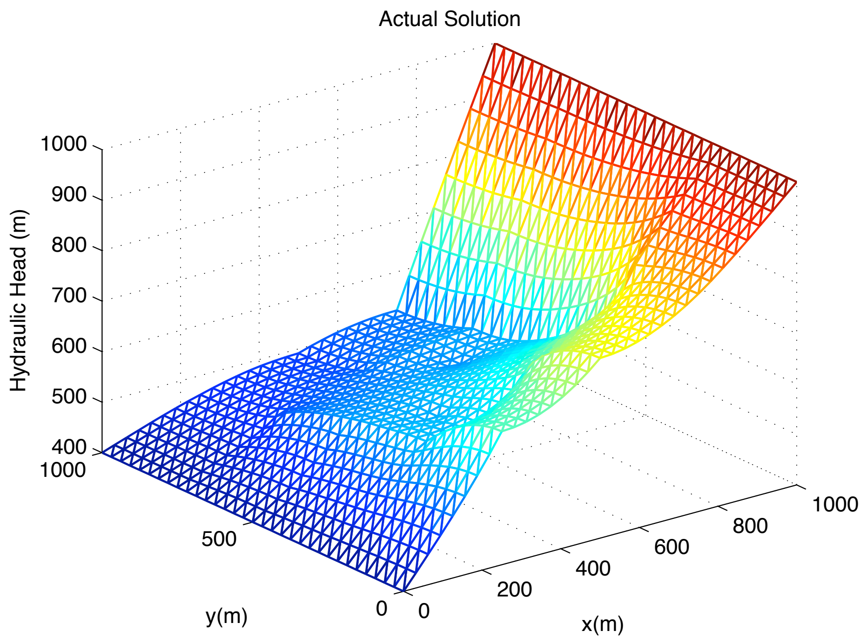


Figure 7.2: "Data" Solution with 4x4 Conductivity Grid

POD Solution

The purpose of the 2D investigation was as a "proof of concept" rather than a rigorous trial of POD performance. To that end, our results are limited in scope as they were only intended to demonstrate the potential of POD implementation on a saturated groundwater problem. There are no timings associated with this domain nor was it run on the HPC machines.

Our goal in this investigation is to recreate the synthetic data set from a homogeneous initial iterate. We access the full-model simulation once with a homogeneous conductivity field to create the POD basis. The homogeneous solution is shown in Figure 7.3. In this solution, we can see the drawdown around the two wells distinctly and the hydraulic gradient is nearly linear between the dirichlet boundary conditions. The gradient would be completely linear if not for the influence of the wells.

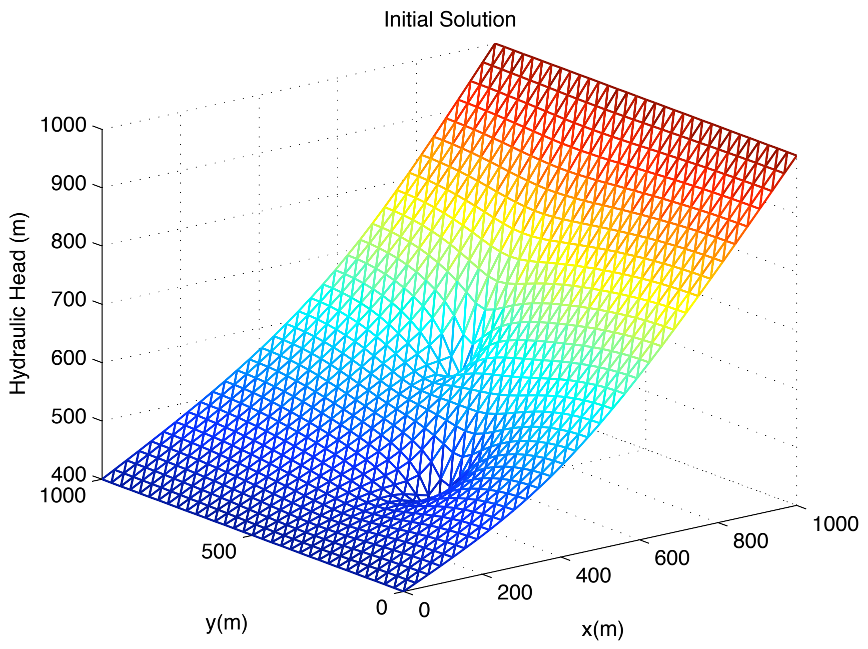


Figure 7.3: Initial Solution (Homogeneous Conductivity)

From this homogeneous solution, we extract the sensitivities $\dfrac{\partial h}{\partial \kappa_i}$ as previously discussed and create the POD basis $U = \left[h, \dfrac{\partial h}{\partial \kappa_i}\right]$. The wells negate any need for the total flux boundary condition previously discussed. We project the dynamics of the full solution onto that basis and feed the reduced order problem to our least squares algorithm. For these problems, we tested both a Gauss-Newton (GN) and Levenberg-Marquardt (LM) code. We see the success of GN in Figure 7.4 and LM in Figure 7.5. In both cases, the final solution is visually indistinguishable from the data solution as shown in Figure 7.6.

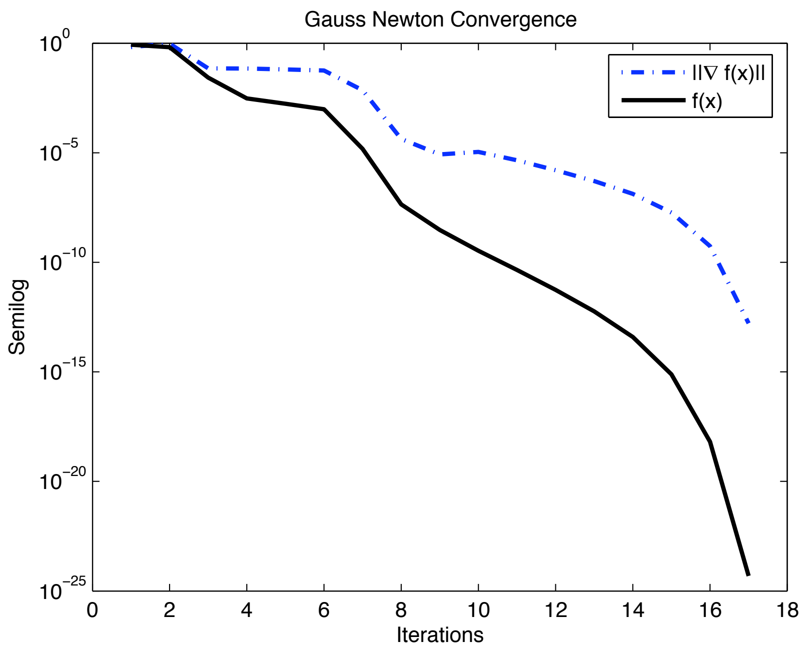


Figure 7.4: Gauss-Newton Convergence from Homogeneous Initial Iterate

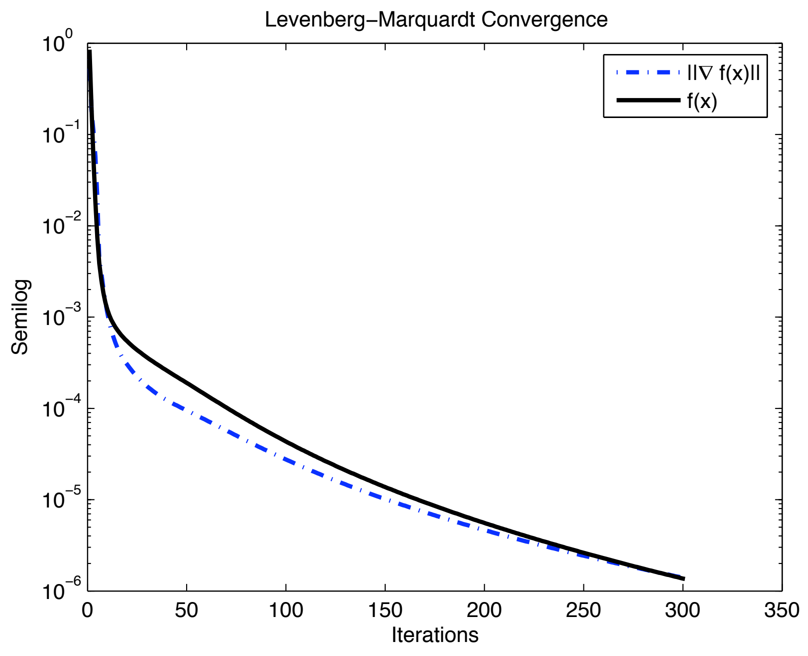


Figure 7.5: Levenberg-Marquardt Iteration History

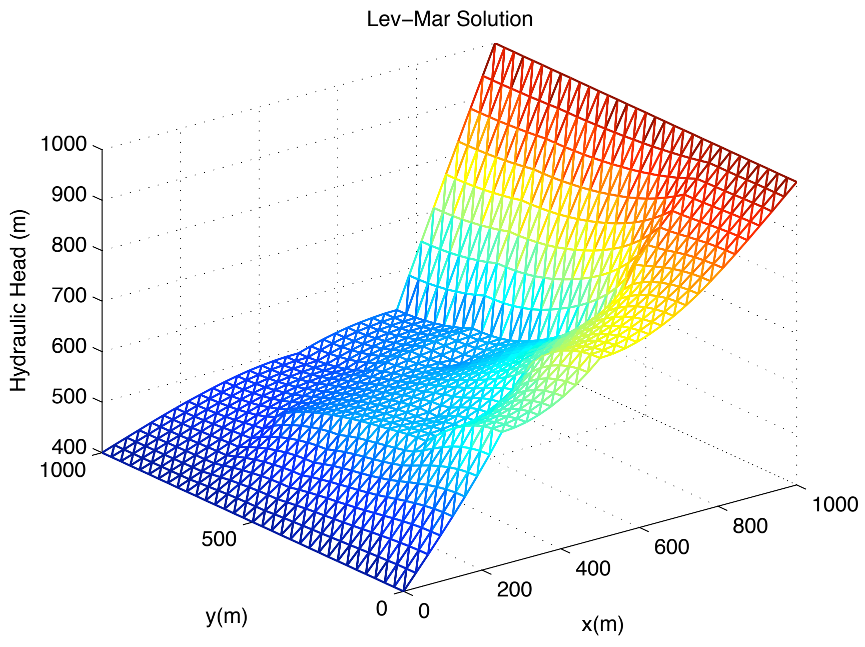


Figure 7.6: Levenberg-Marquardt Solution

Upon examining the iteration histories, we notice the LM code takes significantly more calls to the reduced function than GN. We are not concerned about this, however, as the reduced model function calls are so inexpensive it is a small computational investment to compute even several hundred iterations. What is significant is that both methods, GN and LM, were able to successfully reduce the error between the approximated model and the "data" solution to near zero.

From these results, we felt confident POD could be applied to more complex, 3-D domains. The following section demonstrates how POD has performed with such models.

7.2 3-D Results

7.2.1 Synthetic Column

Our first application of POD with `ADH` is a ten meter high column containing three separate materials. The materials are distributed unevenly throughout the column with the third material (κ_3) only located as a "lens" near the bottom of the column. The location of the three materials in the column is shown in Figure 7.7. The column is discretized with a finite element mesh with 5,881 nodes and 30,720 tetrahedral elements.

We create a hydraulic gradient with a Dirichlet condition at the bottom of the column and a flux at the top of the column. The vertical sides of the column have no-flow Neumann boundary conditions. These boundary conditions represent a thin column where hydraulic head is driven by gravity and an influx from the top. We are careful to ensure the flux through the boundary does not cause the column to become unsaturated.

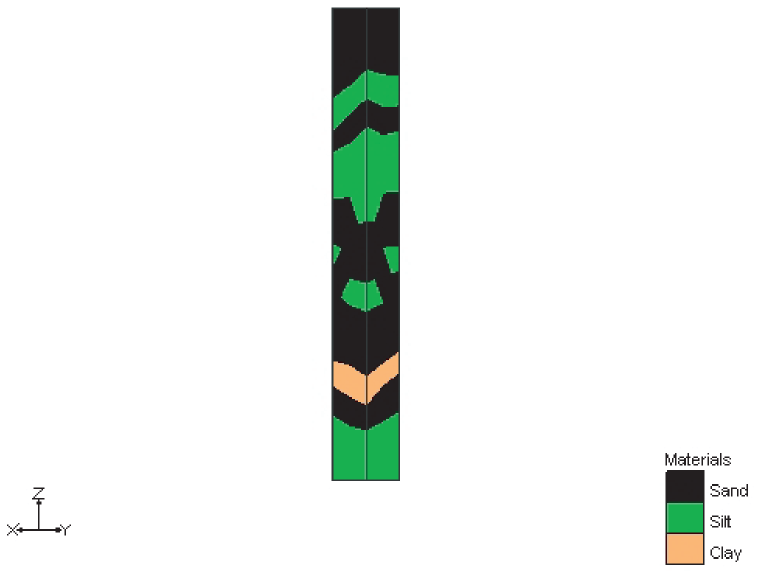


Figure 7.7: Composition of Column

We synthesize the data by assigning conductivities and using ADH to create a solution. The data set is created with conductivities

$$\vec{\kappa} = [1.1808e^{-1}, 1.1808e^{-3}, 1.1808e^{-5}](m/s).$$

The third material ($\kappa_3 = 1.1808e^{-5}$) forms a "lens" and effectively controls the rate at which the column drains. The steady-state solution to this column is shown in Figure 7.8. From this solution, we sample the data at $M = 18$ points, located randomly throughout the column as shown in Figure 7.9.

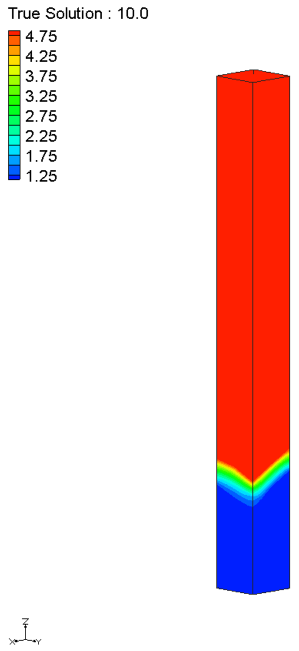


Figure 7.8: Hydraulic Head of Column

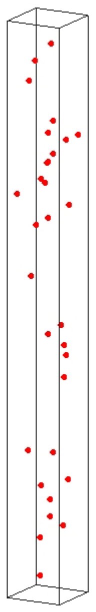


Figure 7.9: Location of Sensors

Our initial iterate is a homogeneous column, with $\vec{\kappa} = [1e^{-2}, 1e^{-2}, 1e^{-2}]$. Other initial guesses were chosen with no significant change in the behavior of the optimization process. The error between this iterate and the data is labelled as "`Initial Residual`" in Table 7.1. From this iterate, we generate the initial POD basis and start the Levenberg-Marquardt optimization process.

We must re-initialize the basis several times during the optimization process. That is, once the parameter estimation software returns a result using the reduced POD model, we re-run the full model using the optimized parameters and generate a new POD basis.

This increases the computational cost of POD as each full model access is expensive. Our goal is to recover the data solution using fewer full-model calls than would be required if the optimizer did not use the POD basis. When we display our results, we will show how many times the POD basis had to be reinitialized – the number of full-model calls – as well as how many times the optimizer used the reduced POD model. It should be noted that the POD model has a negligible contribution to computational effort in comparison to the cost of a full-model solve.

After we successfully recovered the solution for this synthetic column with zero-residual data, we increased the complexity of the problem by adding random "noise" to the data. We perturbed the data by a relative norm of $1\%, 5\%$, and 10%. As shown in Table 7.1, `PEST` reduced the objective function to a norm of similar magnitude when it queried only the POD model and when it used the full model. However, the optimization process with the POD model required significantly fewer full-model calls to obtain that solution. We see in Table 7.2 that the optimization process was able to recover the exact parameters when fed accurate data. With very noisy data, we see that `PEST` with both the reduced model and the full model was only able to recover two to three digits of the parameters. We also see in these results the clay "lens" (κ_3) dominates the behavior of the column and is most accurately recovered with respect to the other parameters even with noisy data.

Table 7.1: Analysis for Column

Noise	Model	Full Calls	POD Calls	Initial Residual	Final Residual	Time (s)
0%	POD	9	249	3.47E+02	6.00E-08	28
	Full	170	-		4.37E-10	286
1%	POD	7	268	3.49E+02	1.54E-02	31
	Full	186	-		1.54E-02	352
5%	POD	4	153	3.48E+02	4.24E-01	15
	Full	152	-		4.22E-01	289
10%	POD	6	187	3.61E+02	2.02	15
	Full	178	-		2.03	337

Table 7.2: Relative Log-Transformed Parameter Error for Column

Noise	Model	Relative Log-Parameter Error $\frac{\lvert ln(\kappa_i)-ln(\kappa_{est})\rvert}{\lvert ln(\kappa_i)\rvert}$		
		κ_1	κ_2	κ_3
0%	POD	2.37E-04	4.60E-07	2.93E-06
	Full	2.46E-04	2.51E-07	3.73E-07
1%	POD	1.00E+00	8.61E-03	6.40E-04
	Full	1.00E+00	8.61E-03	6.41E-04
5%	POD	1.00E+00	5.55E-02	1.62E-04
	Full	9.78E-01	5.43E-02	7.80E-05
10%	POD	2.40E-01	5.67E-02	5.93E-04
	Full	1.08E+00	3.32E-02	1.08E-03
Actual $ln(\kappa_i)$**:**		-2.14E+00	-6.74E+00	-1.13E+01

Figure 7.10 shows the performance of the optimizer when used in conjunction with just the full model and with both the full model and the POD model. The $x-$axis is iterations of the *full model* only, as the reduced model calls are computationally inexpensive in comparison. We note these results use UMFPACK to perform an exact matrix inverse for all solves. This is contrasted against the examples in Chapter 8 that approximate a solution using an iterative solver.

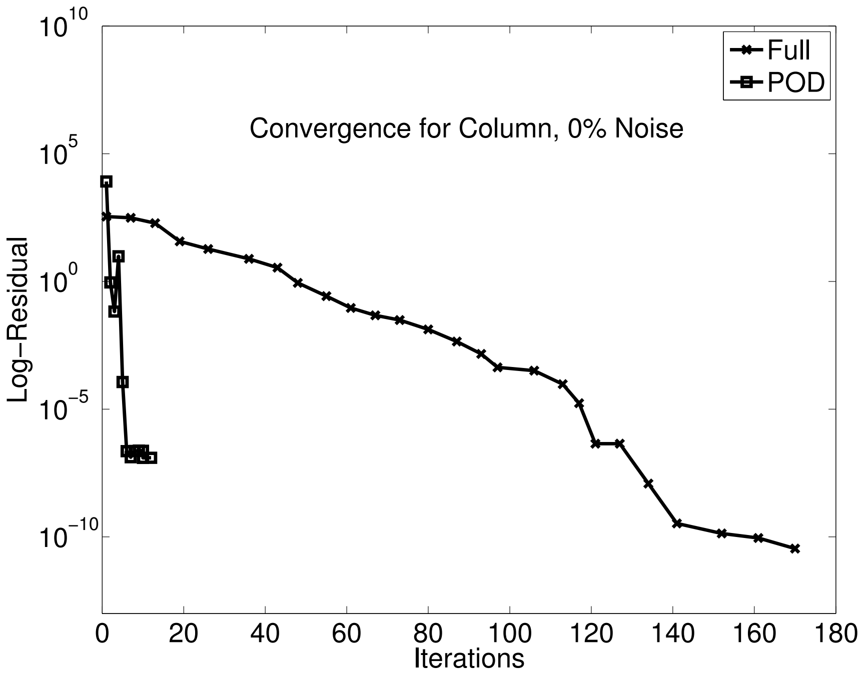
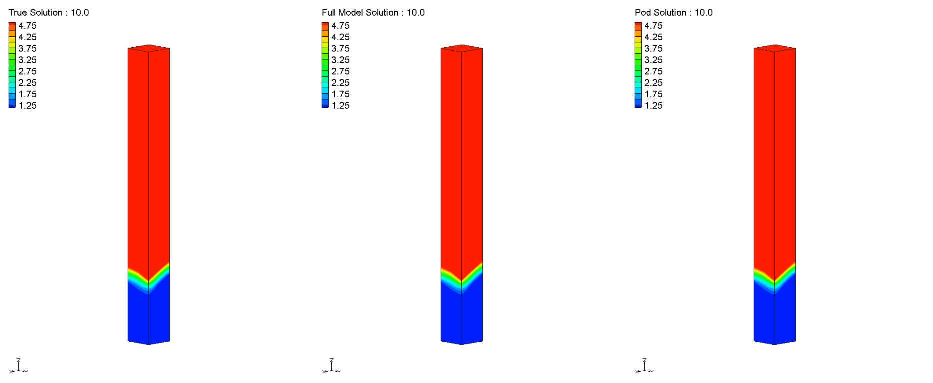


Figure 7.10: Convergence of Optimizer for 0% noise

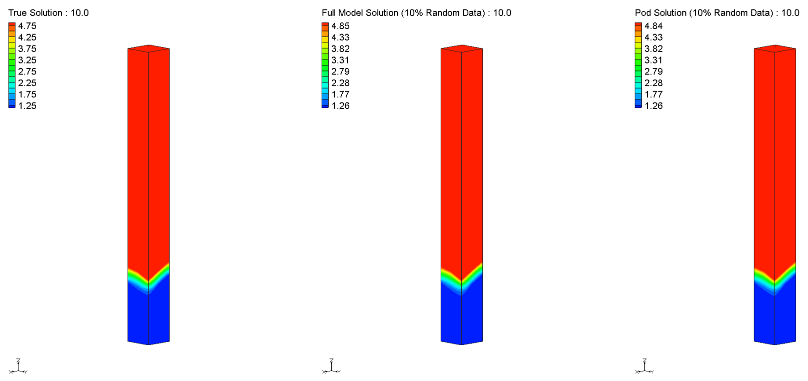
The solutions from the POD approximation are visually indistinguishable from the generated data as shown in Figure 7.11.



(a) True Solution (b) Full Model Solution (c) POD Model Solution

Figure 7.11: Solutions of Column

The solution against noisy data shown in Figure 7.12 is very similar in behavior to the zero-residual case. We notice the scale of the solution has changed from the non-perturbed case.



(a) True Solution (b) Full Model, 10% Noise (c) POD Model, 10% Noise

Figure 7.12: Solutions of Column With 10% Randomized Data

7.2.2 Laboratory Scale Synthetic Aquifer

Our second application is a tank packed with five materials by Tissa H. Illangasekare and his team at the Colorado School of Mines (CSM) [81]. The container, pictured in Figure 7.13, is 208cm x 117cm x 57cm and divided into Cartesian blocks with 28,290 cells of material. To fill the tank, a mesh was placed at each level and each hole was filled with material. The mesh was then removed leaving a precisely packed domain. We can see the packing method in Figure 7.14 and the allocation of materials in Figure 7.15.

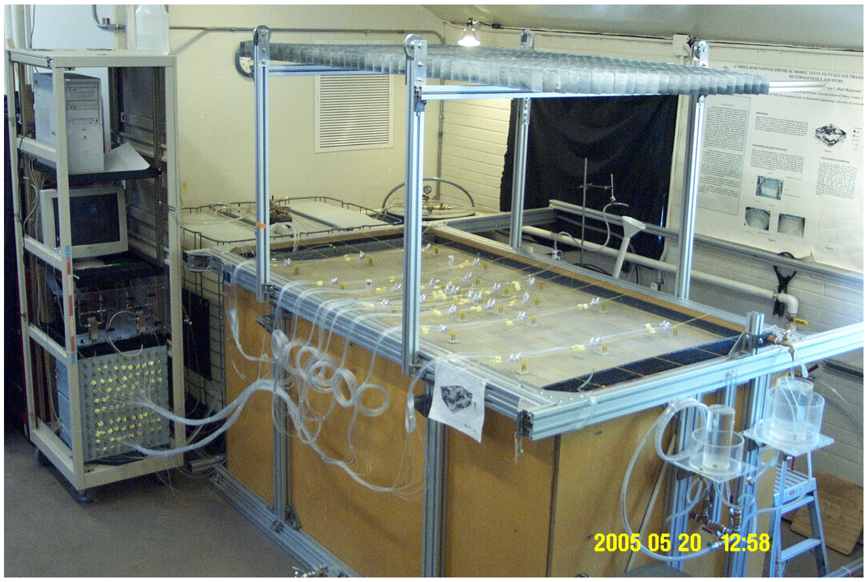


Figure 7.13: Colorado School of Mines Tank

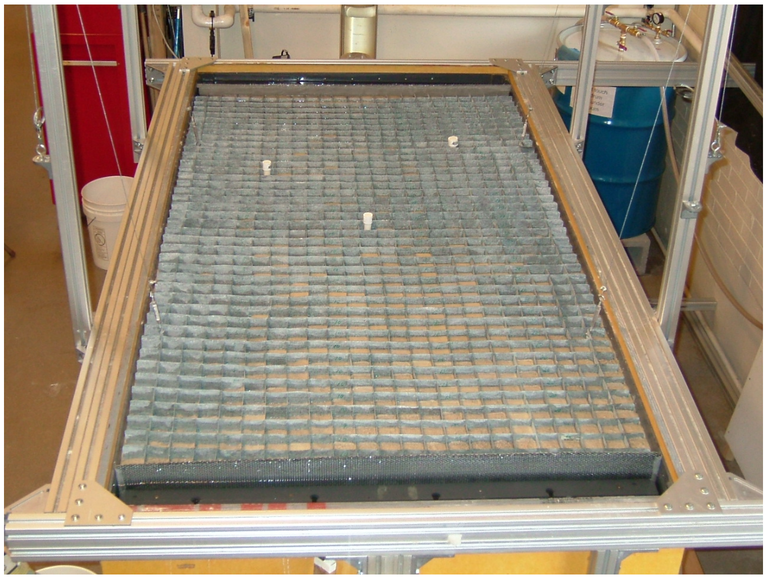


Figure 7.14: Packing Method

Flow through the domain was generated with Dirichlet boundaries on the two "short" sides of the tank. Figure 7.13 shows the gap on the near edge where the water height is maintained.

To model this tank, we created a mesh with 59,538 nodes and 339,480 tetrahedral elements and assigned material ids to each element according to the packing information.

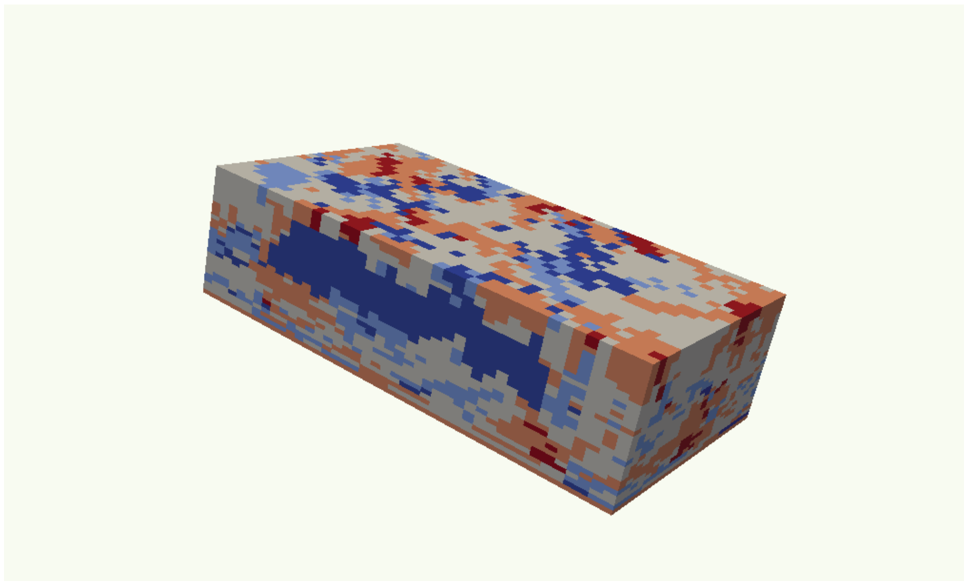


Figure 7.15: Allocation of Materials

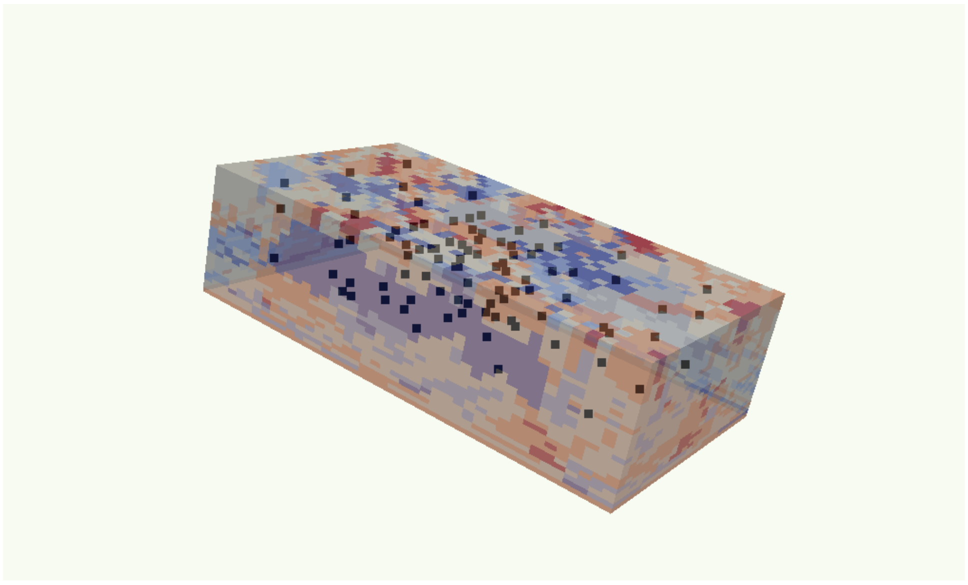


Figure 7.16: Location of Sensors in Tank

The team at CSM measured the pressure at 98 points throughout the domain. Those sample points are shown in Figure 7.16. Our comparison against their measurements is labelled "`Data`" in Tables 7.3 - 7.4. We also synthesized data from the virtual representation of the tank to generate a zero residual problem. That zero residual solution was perturbed by 1%, 5%, and 10% in order to test convergence with noisy data.

We see in Table 7.3 that `PEST` was able to recover a similar solution with both the reduced model and the full model. When it used the POD model, the number of full model calls was reduced by two orders of magnitude and the time to solution was reduced by an order of magnitude. As the noise was increased in the problem, the quality of fit deteriorated significantly. However, the reduced model provided a fit no worse in norm or parameter fit than the fit found with the full model. Table 7.4 shows the parameter fit was similar when the optimizer used either the reduced or full model for all noise levels. For this example, unlike the column, there is no single material that dominates the behavior of the domain. Figure 7.17 shows the optimizer behaves similarly for the tank as it did for the column.

Table 7.3: Results for CSM Tank with Direct Solver

Noise	Model	Full Calls	POD Calls	Initial Residual	Final Residual	Time (s)
0%	POD	7	230	3.73E+03	1.34E-09	5.52E+03
	Full	121	-		2.26E-05	7.92E+04
1%	POD	3	80	3.74E+03	9.63	2.20E+03
	Full	98	-		9.63	6.48E+04
5%	POD	3	69	4.29E+03	2.55E+02	2.21E+03
	Full	179	-		2.50E+02	1.18E+05
10%	POD	3	49	4.62E+03	1.16E+03	1.42E+03
	Full	103	-		1.14E+03	6.95E+04
Data	POD	4	103	3.63E+03	2.67E-02	2.92E+03
	Full	145	-		2.67E-02	9.28E+04

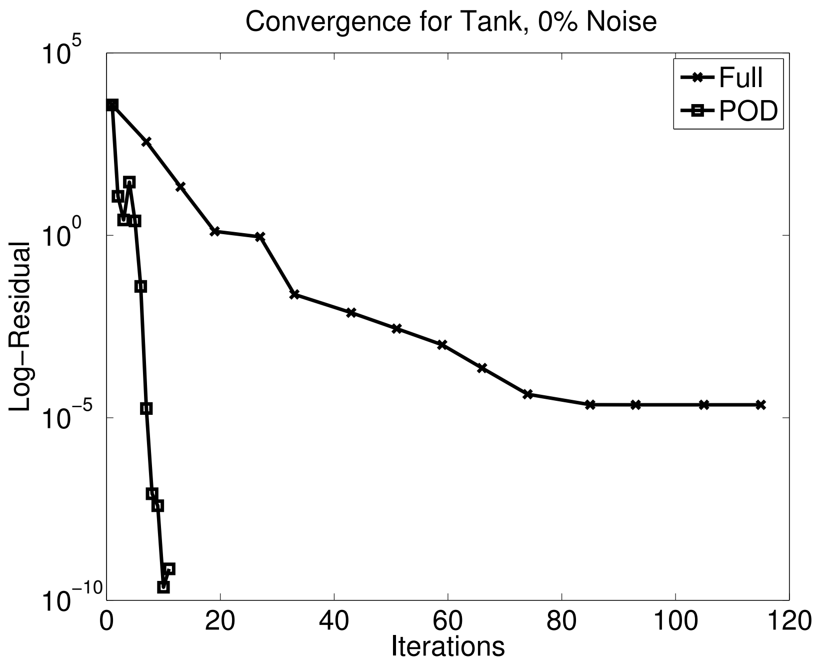


Figure 7.17: Convergence of Optimizer for 0% noise on CSM Tank

Table 7.4: Relative Log-Transformed Parameter Error for Tank

Noise	Model	Relative Log-Parameter Error $\frac{\lvert ln(\kappa_i)-ln(\kappa_{est})\rvert}{\lvert ln(\kappa_i)\rvert}$				
		κ_1	κ_2	κ_3	κ_4	κ_5
0%	POD	3.55E-06	7.47E-05	1.63E-06	6.27E-05	3.90E-04
	Full	5.97E-04	6.27E-04	3.20E-03	1.67E-03	7.16E-03
1%	POD	6.38E-02	4.64E-01	3.08E-01	2.23E-01	4.42E+00
	Full	6.05E-02	4.37E-01	2.94E-01	1.80E-01	5.11E+00
5%	POD	2.99E-01	6.72E-01	1.35E+00	7.97E-01	9.98E-01
	Full	7.12E-01	2.38E+00	3.75E+00	8.54E-01	1.00E+00
10%	POD	2.99E-01	2.36E-01	2.50E-01	6.76E-01	1.16E+01
	Full	1.32E+00	4.97E-03	5.75E-01	3.51E+00	1.43E+01
Data	POD	4.57E-01	5.06E-01	1.14E-01	8.50E-01	6.15E+00
	Full	4.57E-01	5.06E-01	1.14E-01	8.50E-01	6.18E+00
Actual $ln(\kappa_i)$**:**		-5.01E+00	-3.89E+00	-2.76E+00	-1.64E+00	-5.16E-01

The following figures illustrate the hydraulic head profiles for the tank. Figure 7.18 shows the solution when the exact parameters are used in the model. Figure 7.19 demonstrates the results of the optimization when compared against that zero residual data. We see that PEST is able to recover the solution with both the full and POD reduced model. Figure 7.20 displays the solutions when the optimizer minimizes against the data measured in the lab.

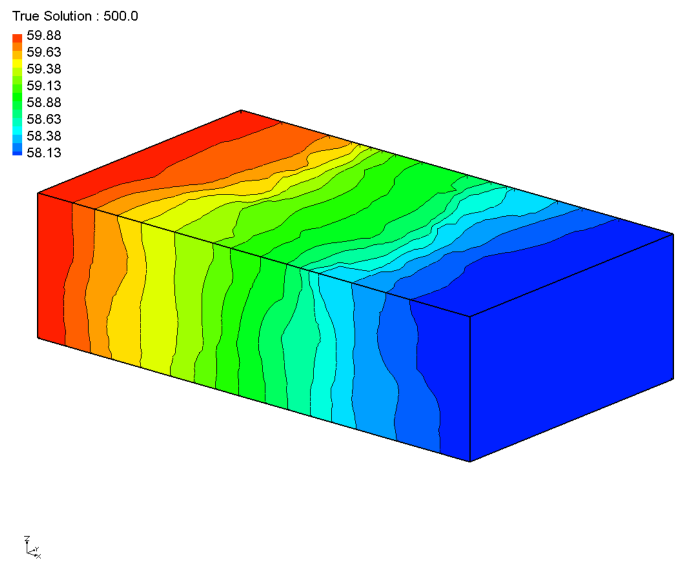
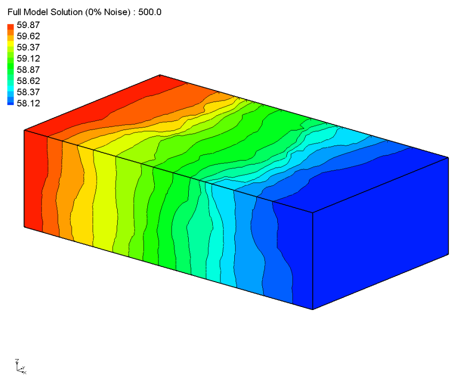
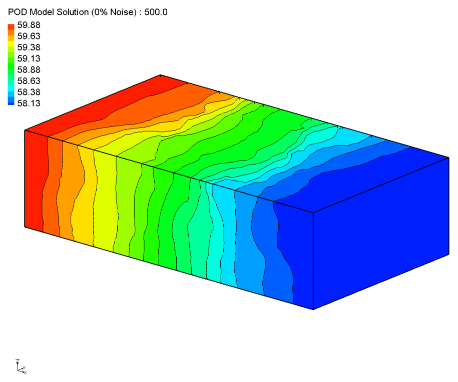


Figure 7.18: Exact Solution of Tank

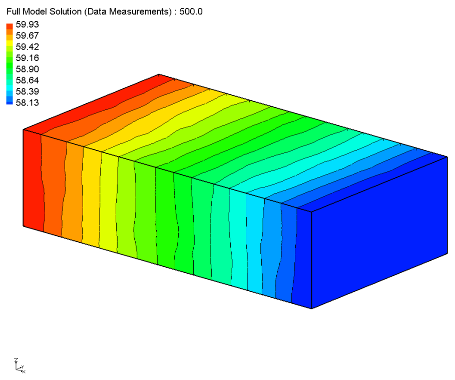


(a) Full Model, 0% Noise

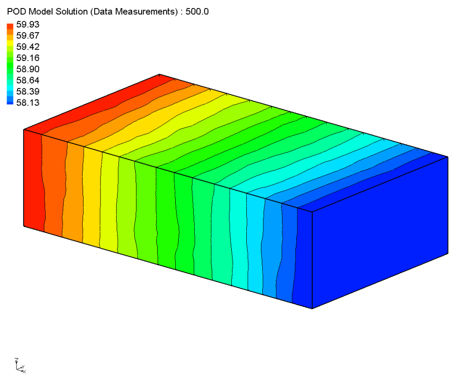


(b) POD Model, 0% Noise

Figure 7.19: Solutions of Tank With 0% Noise



(a) Full Model, Measured Data



(b) POD Model, Measured Data

Figure 7.20: Solutions of Tank With Measured Data

7.2.3 SPE10

Our final example is from the Society of Petroleum Engineers Tenth Comparative Solution Project (SPE10) [19]. The model is specifically designed to test the limits of any methods attempting to use a fine grid implementaton due to its size and complexity. The domain is $1200 \times 2200 \times 170$ feet and is split into 85 layers with five materials. Our mesh of the domain has 1.1 million nodes and 4.5 million elements. As shown in Table 8.1, it takes a desktop computer nearly 45 minutes for one full-model solve of SPE10. This in comparison to 40 seconds for the tank and just under two seconds for the column. The material allocation is shown in Figure 7.21.

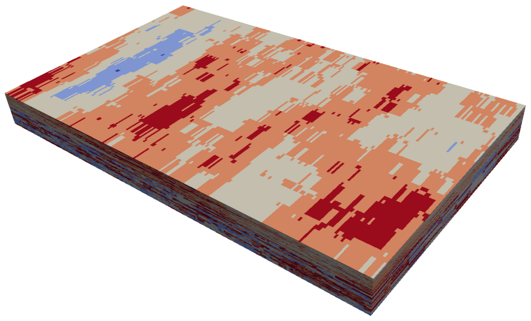


Figure 7.21: Material Allocation for SPE10

The size of SPE10 forces us to use an iterative solver instead of a direct solver as we did with the previous examples. The effects of an iterative solver are explored in §8. We use a L2 norm tolerance of $1e-6$ to achieve the following results.

Table 7.5: Relative Log-Transformed Parameter Error for SPE10

Solver Accuracy	Model	Relative Log-Parameter Error $\frac{\lvert ln(\kappa_i)-ln(\kappa_{est})\rvert}{\lvert ln(\kappa_i)\rvert}$				
		κ_1	κ_2	κ_3	κ_4	κ_5
1e-6	POD	3.07e-01	2.57e-03	1.20e-03	1.60e-03	7.62e-03
	Full	3.61e-01	1.25e-03	1.40e-02	2.21e-02	1.08e-01
Actual $ln(\kappa_i)$**:**		-1.58e+01	-1.32e+01	-8.68e+00	-5.75e+00	-1.14e+00

We see this table is consistent with the results of previous investigations; PEST with POD is able to achieve similar fits to parameters as PEST with the full ADH model. We see in Table 8.36 that the number of full model calls have been decreased by an order of magnitude and the time to solution has been cut in half if PEST queries the POD model. However, the final residual and parameter error are similar.

Table 7.6: SPE10 Results for $1e-6$ Solver Tolerance

	POD	Full
Time (s)	2.51e+04	5.26e+04
Final Residual	9.49e-04	9.31e-05
Parameter Error	2.10e-01	2.47e-01
ADH Calls	13	187
POD Calls	340	-

As in previous results, the solutions from PEST with POD and PEST with ADH are nearly indistinguishable visually. Figure 7.22 displays the "true solution" of SPE10, generated with the correct conductivity values. Figure 7.23 shows the POD solution and the ADH solution. Due to the extreme size of SPE10, we used a different visualization software to display the results. The column and tank results are displayed with GMS [1] and SPE10 is visualized with ParaView [44].

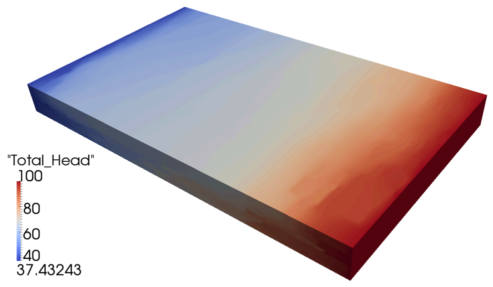
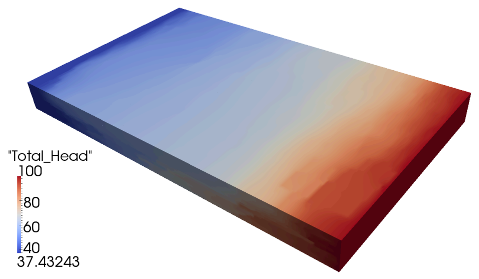
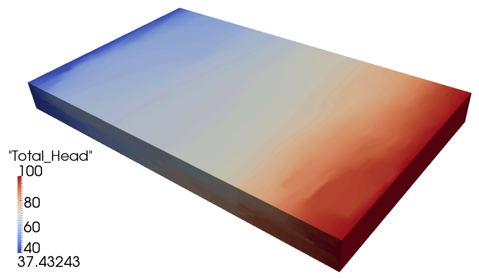


Figure 7.22: Exact Solution of SPE10



(a) SPE10 Solution with `ADH` as Model



(b) SPE10 Solution with POD as Model

Figure 7.23: SPE10 Solutions

Chapter 8

Inexact Levenberg Marquardt with Reduced Order Models

8.1 Solving the Linear System

In this chapter, we examine how inaccuracy in the computation of the sensitivity vectors (5.10) and the solution itself (5.8) affects the performance of the optimization process. In our first two 3-D examples (7.2.1, 7.2.2), we are able to use a direct solver to solve the linear system by computing the matrix inverse. `ADH` uses `UMFPACK` [22, 23, 24, 25] as its sparse direct solver. However, as shown in Table 8.1, the time and memory investment for larger problems can make `UMFPACK` prohibitive to use.

Table 8.1: Time (sec) for each `ADH` solution (serial)

Example	Iterative Solver	Direct Solver
column	1.98e+00	1.90e+00
tank	3.88e+01	4.64e+02
spe10	2.59e+03	–

8.1.1 Iterative Linear Methods

Instead of directly computing the matrix factorization, we can approximate a solution to $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ with an iterative process. There are several methods available, but within `ADH`, we use the Bi-conjugate gradient stabilized method (Bi-CGSTAB) [53] and precondition the system with Jacobi's method.

Jacobi splits the matrix $\mathbf{A}$ into

$$\begin{aligned} \mathbf{A} = & \quad \mathbf{A}_1 + \mathbf{A}_2, \\ \mathbf{A}_1 = & \qquad \mathbf{D}, \\ \mathbf{A}_2 = & \quad \mathbf{L} + \mathbf{U}. \end{aligned}$$

We precondition the system with the square root of the inverse diagonal $\mathbf{D}$

$$\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\boldsymbol{y} = \mathbf{D}^{-1/2}\boldsymbol{b}, \tag{8.1}$$

where $\boldsymbol{y} = \mathbf{D}^{1/2}\boldsymbol{x}$. If any diagonal entry is less than a user-defined tolerance, it is replaced by that tolerance to prevent division errors.

We solve this preconditioned system with Bi-CGSTAB. The algorithm in `ADH` is from [53] and is copied below:

1. $\boldsymbol{r} = \boldsymbol{b} - \mathbf{A}\boldsymbol{x},\ \hat{\boldsymbol{r}}_0 = \hat{\boldsymbol{r}} = \boldsymbol{r},\ \rho_0 = \alpha = \omega = 1,\ v = p = 0,\ k = 0,\ \rho_1 = \hat{r}_0^T r$
2. Do While $||\boldsymbol{r}||_2 > \varepsilon||\boldsymbol{b}||_2$ and $k < k_{max}$
    (a) $k = k + 1$
    (b) $\beta = (\rho_k/\rho_{k-1})(\alpha/\omega)$
    (c) $p = r + \beta(p - \omega v)$
    (d) $\boldsymbol{v} = \mathbf{A}\boldsymbol{p}$
    (e) $\alpha = \rho_k/(\hat{\boldsymbol{r}}_0^T\boldsymbol{v})$
    (f) $\boldsymbol{s} = \boldsymbol{r} - \alpha\boldsymbol{v},\ \boldsymbol{t} = \mathbf{A}\boldsymbol{s}$
    (g) $\omega = \boldsymbol{t}^T\boldsymbol{s}/||\boldsymbol{t}||_2^2,\ \rho_{k+1} = -\omega\hat{\boldsymbol{r}}_0^T\boldsymbol{t}$
    (h) $\boldsymbol{x} = \boldsymbol{x} + \alpha\boldsymbol{p} + \omega\boldsymbol{s}$
    (i) $\boldsymbol{r} = \boldsymbol{s} - \omega\boldsymbol{t}$

In our implementation, we allow Bi-CGSTAB a large number of linear iterations ($\approx 0.01 * degrees\_of\_freedom$) to reach the defined tolerance. If it has not converged at that point, we accept the solution regardless of quality. It is important to note in the following suite of results, the tolerance listed is the "best case" tolerance, but may not always be achieved at all points in the process.

8.2 Modifications to Optimization Algorithm

We test how low accuracy solutions and sensitivities affect the optimization process. With poor approximations to the solution, we expect the error between model output and data to be large even for the correct parameters. For some low accuracy solves, an improvement in the parameter set may not correlate with a reduction in error. Also, with poor estimates for the sensitivities, we expect the optimizer to have difficulty resolving the correct search direction. As shown in [21], we cannot expect Levenberg Marquardt to reliably move in the correct direction when provided with poor approximations to the jacobian calculations.

The low accuracy in the full model and sensitivities causes us to adjust the optimization process described in Figure 6.1. The optimizer exhibits two behaviors against which we guard with additional stopping criteria [29, 89]:

-

: If a user-defined number of iterations have passed without any reduction in the full model residual, we terminate the optimization process.

-

: If a user-defined number of iterations have passed without reducing the full-model residual from one iteration to the next by a certain tolerance, we terminate the process.

These two criteria are similar, but guard against two different behaviors. The first ensures that, over a period of several iterations, the full model residual has decreased. That is, if each residual at $\{\boldsymbol{x}_n, \boldsymbol{x}_{n-1}, ..., \boldsymbol{x}_{n-(N-1)}\}$ is greater than the residual at $\boldsymbol{x}_{n-N}$, we terminate the optimization process. The second criterion will terminate the process if N iterations have passed without a reduction in residual between $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n-1}$.

It is important to note the Levenberg Marquardt process in the `PEST` optimizer only reduces the error of the POD reduced model after each full `ADH` solve. `PEST` is unaware of how each iteration's error compares to the previous `ADH` error. For that reason, it is not uncommon for `PEST` to find a set of parameters that decrease the error in the POD model but are significantly worse in the `ADH` model. When a new basis is formed with the poor parameters, the POD model residual includes an accurate portrayal of those parameters. `PEST` is able to use the more accurate information to find a significantly improved parameter set. As the POD basis expands over several iterations, this tendency is reduced and the POD model becomes more accurate for all sets of parameters.

8.3 Inexact Levenberg Marquardt with Reduced Order Models

We provide a tolerance for the solution (5.8) and the sensitivities (5.10) independent of one another in order to gauge how each affects the behavior of the optimizer. We note that the tolerance is not always achieved throughout the process, however. There are times when the solver is not able to reach the prescribed tolerance; in those cases we still accept the solution and continue the process. We know from [52] that the accuracy of both the solution and gradient dictate the accuracy of the minimizer. As shown in [54, 33], we expect our process to converge even with poor approximations to the jacobian.

To test this hypothesis, we examine the two smaller 3-D domains (7.2.1, 7.2.2). In the results below, *ADH Tol* refers to the tolerance applied to the solution from `ADH` (5.8). *Sensitivity Tolerance* is the tolerance applied to the solution of the sensitivity vectors (5.10). We display four tables for each test:

1. `Time`: A simple measurement of time-to-solution as measured by the HPC machines. This is a comprehensive measure from when the process initiates to when it terminates.

2. `Final Residual`: The error between the final `ADH` run and the actual data. For the reduced order model, this means `ADH` must be run one additional time when the process has completed to measure the final error. That run is included in the above time estimates.

3. `Parameter Error`: The relative error in the **log-transformed** parameters. As the entire process is completed with the log-conds, we measure our final accuracy in those as well.

4. `Model Calls`: The total number of calls to the full model `ADH` . This includes the final call necessary to compute the accuracy.

We apply our technique to data with various noise levels. For the synthetic column, the "Data" represent a zero-residual problem. Random perturbations in that data are added to achieve the 1%, 5%, and 10% noise level results. For the tank, the "Data" represent actual laboratory measurements and are not a zero residual problem. Random perturbations in that measured data are added for the 1, 5, and 10% examples. The additional tables are not constructed for SPE10. Instead, a single solve with $1e-6$ as both the solution and sensitivity tolerance is performed. The results of that run are given in 8.4.13.

8.4 Tolerance Results

8.4.1 Synthetic Column

The following tables describe the results for the 3-D column (7.2.1). In all examples, the runs were performed with 8 processors on the `Garnet` HPC machines at ERDC. These are Cray XE6 machines and have 64-bit AMD Opteron chipsets running at 2.4 GHz. Each node has 16 cores and 32 gigabytes of memory. The login nodes run SUSE Linux.

8.4.2 Noise: 0%

Table 8.2: Analysis of **Time (s)** for Domain: **HET**, Noise: **Data**

Time (s)						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	8.20e+01	4.90e+01	9.80e+01	7.80e+01	1.49e+02
	1.0e-4	7.30e+01	7.50e+01	5.70e+01	6.30e+01	1.68e+02
	1.0e-6	1.96e+02	2.11e+02	2.62e+02	1.77e+02	1.67e+02
	1.0e-8	1.13e+02	1.06e+02	1.10e+02	1.21e+02	5.01e+02

Table 8.3: Analysis of **Final Residual** for Domain: **HET**, Noise: **Data**

Final Residual						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	5.58e+02	5.77e+02	5.58e+02	5.56e+02	5.76e+02
	1.0e-4	2.78e+02	2.78e+02	3.55e+02	3.55e+02	2.93e+02
	1.0e-6	9.59e-01	9.59e-01	6.82e-04	2.57e+01	2.59e+01
	1.0e-8	4.95e-02	4.95e-02	8.80e-02	8.07e-02	5.50e-09

Table 8.4: Analysis of **Parameter Error** for Domain: **HET**, Noise: **Data**

Parameter Error						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	9.98e-01	1.14e+00	1.00e+00	1.00e+00	4.06e-01
	1.0e-4	5.27e-01	5.27e-01	5.60e-01	5.60e-01	6.37e-01
	1.0e-6	6.45e-01	6.45e-01	5.22e-02	5.89e-01	5.69e-01
	1.0e-8	8.34e-01	8.34e-01	5.03e-01	3.38e-01	1.60e-01

Table 8.5: Analysis of **Model Calls** for Domain: **HET**, Noise: **Data**

Model Calls						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	5	3	6	5	59
	1.0e-4	4	4	3	3	72
	1.0e-6	14	14	15	11	70
	1.0e-8	8	8	7	7	245

8.4.3 Noise: 1%

Table 8.6: Analysis of **Time (s)** for Domain: **HET**, Noise: **1 %**

Time (s)						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	6.60e+01	1.05e+02	8.20e+01	3.70e+01	1.56e+02
	1.0e-4	1.04e+02	1.02e+02	7.00e+01	7.00e+01	1.32e+02
	1.0e-6	1.22e+02	1.22e+02	3.97e+02	1.18e+02	2.01e+02
	1.0e-8	1.11e+02	1.08e+02	1.32e+02	1.50e+02	3.83e+02

Table 8.7: Analysis of **Final Residual** for Domain: **HET**, Noise: **1 %**

Final Residual						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	5.59e+02	5.21e+02	5.59e+02	5.59e+02	5.78e+02
	1.0e-4	2.56e+02	2.56e+02	3.56e+02	3.56e+02	3.50e+02
	1.0e-6	3.34e+01	3.34e+01	2.83e-02	5.17e+00	1.31e+01
	1.0e-8	3.07e+01	3.07e+01	9.01e-02	8.94e-02	1.54e-02

Table 8.8: Analysis of **Parameter Error** for Domain: **HET**, Noise: **1 %**

Parameter Error						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	9.91e-01	1.21e+00	1.00e+00	5.60e-01	4.07e-01
	1.0e-4	5.60e-01	5.60e-01	5.60e-01	5.60e-01	5.18e-01
	1.0e-6	8.92e-01	8.92e-01	1.55e-01	2.71e-01	5.09e-01
	1.0e-8	9.88e-01	9.88e-01	2.96e-01	3.17e-01	1.60e-01

Table 8.9: Analysis of **Model Calls** for Domain: **HET**, Noise: **1 %**

Model Calls						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	4	7	5	2	59
	1.0e-4	6	6	3	3	57
	1.0e-6	8	8	19	6	84
	1.0e-8	6	6	8	10	161

8.4.4 Noise: 5%

Table 8.10: Analysis of **Time (s)** for Domain: **HET**, Noise: **5** %

Time (s)						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	5.00e+01	8.00e+01	9.10e+01	5.50e+01	1.53e+02
	1.0e-4	5.80e+01	5.80e+01	5.50e+01	5.20e+01	1.08e+02
	1.0e-6	1.29e+02	1.32e+02	2.02e+02	1.37e+02	2.55e+02
	1.0e-8	8.30e+01	7.40e+01	1.13e+02	1.29e+02	2.60e+02

Table 8.11: Analysis of **Final Residual** for Domain: **HET**, Noise: **5** %

Final Residual						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	5.59e+02	5.17e+02	5.58e+02	5.82e+02	5.77e+02
	1.0e-4	3.55e+02	3.55e+02	3.55e+02	3.55e+02	3.55e+02
	1.0e-6	6.04e+01	6.04e+01	4.23e-01	4.66e-01	1.80e+01
	1.0e-8	6.63e+01	6.63e+01	4.38e-01	4.44e-01	4.22e-01

Table 8.12: Analysis of **Parameter Error** for Domain: **HET**, Noise: **5 %**

Parameter Error						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	1.00e+00	1.14e+00	1.00e+00	5.60e-01	4.02e-01
	1.0e-4	5.60e-01	5.60e-01	5.60e-01	5.60e-01	7.32e-01
	1.0e-6	6.10e-01	6.10e-01	5.39e-01	3.72e-02	5.31e-01
	1.0e-8	5.09e-02	5.09e-02	5.60e-01	4.80e-02	5.23e-02

Table 8.13: Analysis of **Model Calls** for Domain: **HET**, Noise: **5 %**

Model Calls						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	3	5	6	3	58
	1.0e-4	3	3	3	3	44
	1.0e-6	8	8	14	9	106
	1.0e-8	4	4	8	9	105

8.4.5 Noise: 10%

Table 8.14: Analysis of **Time (s)** for Domain: **HET**, Noise: **10 %**

Time (s)						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	6.80e+01	4.80e+01	7.70e+01	4.70e+01	1.28e+02
	1.0e-4	5.70e+01	5.60e+01	5.10e+01	5.90e+01	3.49e+02
	1.0e-6	1.61e+02	1.53e+02	6.90e+01	2.60e+02	3.11e+02
	1.0e-8	3.59e+02	3.44e+02	5.70e+01	1.69e+02	2.46e+02

Table 8.15: Analysis of **Final Residual** for Domain: **HET**, Noise: **10 %**

Final Residual						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	5.75e+02	5.94e+02	5.75e+02	5.99e+02	5.94e+02
	1.0e-4	3.69e+02	3.69e+02	3.69e+02	3.69e+02	5.28e+01
	1.0e-6	3.90e+01	3.90e+01	7.47e+01	2.01e+00	7.85e+00
	1.0e-8	2.02e+00	2.02e+00	3.61e+02	2.02e+00	2.02e+00

Table 8.16: Analysis of **Parameter Error** for Domain: **HET**, Noise: **10 %**

Parameter Error						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	1.00e+00	1.17e+00	1.00e+00	9.98e-01	4.05e-01
	1.0e-4	4.39e-01	4.39e-01	5.60e-01	5.60e-01	4.92e-01
	1.0e-6	5.44e-01	5.44e-01	9.84e-02	1.13e+00	6.84e-01
	1.0e-8	2.93e-01	2.93e-01	5.12e-01	5.02e-01	2.90e-04

Table 8.17: Analysis of **Model Calls** for Domain: **HET**, Noise: **10 %**

Model Calls						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	5	3	6	3	59
	1.0e-4	3	3	3	3	164
	1.0e-6	9	9	4	15	139
	1.0e-8	24	24	3	12	107

8.4.6 Parameter Values

Finally, we list the actual conductivity values found by our method for each noise level in Table 8.18. We only list the conductivities for the instance when both solver and sensitivity tolerance were set to $1e-8$. We note the optimizer often matches the value of κ_3 with high precision, but not κ_1, κ_2. Materials 1 and 2 are spread throughout the column, but the third material is only located in the lens. This lens dominates the behavior of the entire column.

Table 8.18: Conductivity Values for Column

Noise	Model	Conductivity Values for Column		
		κ_1	κ_2	κ_3
0%	POD	1.7556e-03	2.1913e-01	1.1826e-05
	Full	1.1762e-01	1.1808e-03	1.1804e-05
1%	POD	2.8266e-03	1.5001e-02	1.1773e-05
	Full	9.9999e-01	1.2503e-03	1.1717e-05
5%	POD	2.5831e-03	7.4080e-03	1.1896e-05
	Full	1.4284e-02	1.7128e-03	1.1800e-05
10%	POD	7.0478e-02	8.0651e-04	1.1888e-05
	Full	6.5855e-02	8.0429e-04	1.1896e-05
Actual κ_i:		1.1808e-01	1.1808e-03	1.1808e-05

8.4.7 Laboratory Scale Synthetic Aquifer

For the laboratory scale aquifer (7.2.2), we used 16 processors for each full `ADH` and reduced model run. The runs were still performed on the `Garnet` machines.

8.4.8 Noise: 0%

Table 8.19: Analysis of **Time (s)** for Domain: **ALL**, Noise: **Data**

Time (s)						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	1.86e+02	7.85e+02	4.79e+02	4.98e+02	4.01e+02
	1.0e-4	3.05e+02	5.27e+02	4.40e+02	5.59e+02	3.51e+03
	1.0e-6	2.94e+02	6.15e+02	5.78e+02	5.32e+02	6.20e+02
	1.0e-8	1.11e+03	5.75e+02	4.16e+02	4.19e+02	1.44e+03

Table 8.20: Analysis of **Final Residual** for Domain: **ALL**, Noise: **Data**

Final Residual						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	2.71e+02	3.82e+00	3.30e+00	2.32e+01	4.30e-01
	1.0e-4	5.69e-02	4.67e-02	3.99e-02	2.63e-02	3.11e-02
	1.0e-6	5.07e-02	2.67e-02	2.66e-02	3.02e-02	2.67e-02
	1.0e-8	2.69e-02	2.67e-02	2.67e-02	2.67e-02	2.67e-02

Table 8.21: Analysis of **Parameter Error** for Domain: **ALL**, Noise: **Data**

Parameter Error						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	7.63e-01	1.26e+00	7.39e-01	8.83e-01	7.63e-01
	1.0e-4	6.77e-01	5.27e-01	6.98e-01	6.52e-01	8.21e-01
	1.0e-6	6.78e-01	6.38e-01	6.50e-01	5.66e-01	6.51e-01
	1.0e-8	6.43e-01	6.56e-01	6.55e-01	6.52e-01	6.53e-01

Table 8.22: Analysis of **Model Calls** for Domain: **ALL**, Noise: **Data**

Model Calls						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	3	8	6	6	52
	1.0e-4	5	7	7	9	391
	1.0e-6	5	9	8	7	67
	1.0e-8	15	9	7	7	136

8.4.9 Noise: 1%

Table 8.23: Analysis of **Time (s)** for Domain: **ALL**, Noise: **1 %**

Time (s)						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	3.00e+02	8.88e+02	1.03e+03	5.08e+02	1.10e+03
	1.0e-4	3.22e+02	4.13e+02	4.38e+02	2.78e+02	1.23e+03
	1.0e-6	2.63e+02	4.26e+02	3.66e+02	4.06e+02	1.00e+03
	1.0e-8	2.14e+02	4.34e+02	3.85e+02	3.88e+02	7.72e+02

Table 8.24: Analysis of **Final Residual** for Domain: **ALL**, Noise: **1 %**

Final Residual						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	1.80e+02	2.68e+01	1.94e+01	2.56e+01	1.08e+01
	1.0e-4	9.62e+00	9.62e+00	9.63e+00	9.62e+00	9.68e+00
	1.0e-6	9.71e+00	9.65e+00	9.63e+00	9.63e+00	9.63e+00
	1.0e-8	9.73e+00	9.63e+00	9.63e+00	9.76e+00	9.64e+00

Table 8.25: Analysis of **Parameter Error** for Domain: **ALL**, Noise: **1 %**

Parameter Error						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	1.09e+00	1.22e+00	3.43e-01	1.25e+00	7.95e-01
	1.0e-4	4.14e-01	4.50e-01	3.99e-01	4.25e-01	9.86e-01
	1.0e-6	5.08e-01	4.92e-01	4.74e-01	4.81e-01	4.44e-01
	1.0e-8	6.02e-01	4.15e-01	4.14e-01	6.95e-01	5.23e-01

Table 8.26: Analysis of **Model Calls** for Domain: **ALL**, Noise: **1 %**

Model Calls						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	4	10	10	6	139
	1.0e-4	6	7	7	5	144
	1.0e-6	5	7	6	6	106
	1.0e-8	4	7	6	6	71

8.4.10 Noise: 5%

Table 8.27: Analysis of **Time (s)** for Domain: **ALL**, Noise: **5 %**

Time (s)						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	3.17e+02	4.43e+02	6.01e+02	2.67e+02	6.79e+02
	1.0e-4	3.27e+02	5.67e+02	7.31e+02	2.66e+02	1.10e+03
	1.0e-6	7.80e+01	4.47e+02	5.45e+02	1.04e+02	6.73e+03
	1.0e-8	4.67e+02	6.45e+02	1.37e+02	5.08e+02	2.28e+03

Table 8.28: Analysis of **Final Residual** for Domain: **ALL**, Noise: **5 %**

Final Residual						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	2.62e+02	2.68e+02	2.67e+02	2.84e+02	2.69e+02
	1.0e-4	2.55e+02	2.55e+02	2.54e+02	2.55e+02	2.63e+02
	1.0e-6	2.71e+02	2.55e+02	2.55e+02	2.67e+02	2.52e+02
	1.0e-8	2.55e+02	2.55e+02	2.68e+02	2.55e+02	2.56e+02

Table 8.29: Analysis of **Parameter Error** for Domain: **ALL**, Noise: **5 %**

Parameter Error						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	8.11e-01	1.19e+00	6.17e-01	6.10e-01	7.70e-01
	1.0e-4	6.97e-01	7.02e-01	7.02e-01	6.99e-01	8.70e-01
	1.0e-6	8.56e-01	6.35e-01	6.98e-01	1.19e+00	1.74e+00
	1.0e-8	7.02e-01	6.96e-01	7.28e-01	6.55e-01	1.47e+00

Table 8.30: Analysis of **Model Calls** for Domain: **ALL**, Noise: **5 %**

Model Calls						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	5	7	8	4	91
	1.0e-4	6	9	10	4	108
	1.0e-6	1	7	8	1	177
	1.0e-8	7	9	2	7	99

8.4.11 Noise: 10%

Table 8.31: Analysis of **Time (s)** for Domain: **ALL**, Noise: **10 %**

Time (s)						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	2.70e+02	1.21e+02	3.59e+02	6.50e+02	5.49e+02
	1.0e-4	1.67e+02	2.22e+02	1.40e+02	1.98e+02	1.06e+03
	1.0e-6	2.23e+02	1.22e+02	3.09e+02	2.70e+02	6.13e+02
	1.0e-8	1.83e+02	2.40e+02	1.05e+02	2.85e+02	9.11e+02

Table 8.32: Analysis of **Final Residual** for Domain: **ALL**, Noise: **10 %**

Final Residual						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	1.18e+03	1.23e+03	1.16e+03	1.15e+03	1.17e+03
	1.0e-4	1.16e+03	1.16e+03	1.18e+03	1.15e+03	1.16e+03
	1.0e-6	1.15e+03	1.16e+03	1.16e+03	1.15e+03	1.15e+03
	1.0e-8	1.16e+03	1.16e+03	1.16e+03	1.16e+03	1.15e+03

Table 8.33: Analysis of **Parameter Error** for Domain: **ALL**, Noise: **10 %**

Parameter Error						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	7.75e-01	1.06e+00	1.13e+00	5.18e-01	7.42e-01
	1.0e-4	7.97e-01	5.77e-01	9.43e-01	1.06e+00	1.51e+00
	1.0e-6	1.02e+00	5.44e-01	6.25e-01	9.28e-01	1.54e+00
	1.0e-8	7.85e-01	5.87e-01	7.69e-01	6.33e-01	1.85e+00

Table 8.34: Analysis of **Model Calls** for Domain: **ALL**, Noise: **10 %**

Model Calls						
		Sensitivity Tolerance				
		1.0e-2	**1.0e-4**	**1.0e-6**	**1.0e-8**	**Full**
ADH Tol	**1.0e-2**	5	2	6	9	73
	1.0e-4	3	4	2	3	114
	1.0e-6	4	2	5	4	56
	1.0e-8	3	4	1	4	69

8.4.12 Parameter Values

In Table 8.35 we see the conductivity values found by the optimizer for the tank. We again only list the solutions when tolerances for both the solution and sensitivities were set to $1e - 8$. Unlike in the column, there is no single material that dominates the behavior of the solution.

Table 8.35: Conductivity Values for Tank

Noise	Model	Conductivity Values for Tank				
		κ_1	κ_2	κ_3	κ_4	κ_5
Data	POD	6.6061e-02	1.4687e-01	4.6006e-02	4.84085e-02	2.3676e-02
	Full	5.9279e-02	1.3104e-01	4.1250e-02	4.3270e-02	2.1569e-02
1%	POD	1.6098e-02	2.4898e-03	2.8257e-01	6.9233e-03	5.1852e-02
	Full	8.1799e-03	3.4399e-03	1.3249e-01	1.2409e-01	2.5559e-02
5%	POD	1.503e-03	3.0878e-03	1.5034e-03	7.3391e-01	9.7235e-01
	Full	2.4810e-2	8.8000e-6	6.600e-5	5.2483e-01	1.0000e+00
10%	POD	8.0311e-03	1.5034e-03	1.7177e-03	3.4979e-01	8.9090e-01
	Full	1.9999e-06	1.5049e-02	2.9987e-01	8.3000e-04	9.5000e-05
Actual κ_i:		6.7e-02	2.05e-02	6.31e-02	1.94e-01	5.97e-01

8.4.13 SPE10

This domain is significantly larger than our previous examples. We use 32 processors for both the full and reduced model runs. While more processors would have accelerated the process, the ratios between the full and reduced model timings remain similar. In Table 8.36, we see the results with the reduced order model were very similar to the results when the optimizer queried the full model. As in previous results, we took just under half the time and an order of magnitude fewer calls to the full model.

Table 8.36: SPE10 Results

	POD	Full
Time (s)	2.51e+04	5.26e+04
Final Residual	9.49e-04	9.31e-05
Parameter Error	2.10e-01	2.47e-01
Model Calls	13	187

In our comparison of the actual parametersin Table 8.37, we the optimizer performed extremely well with both the full and reduced order model.

Table 8.37: Conductivity Values for SPE10

Solver Tolerance	Model	Conductivity Values for SPE10				
		κ_1	κ_2	κ_3	κ_4	κ_5
Data	POD	1.7498e-05	1.8931e-06	1.7116e-04	3.1979e-03	3.2342e-01
	Full	4.1000e-05	1.7999e-06	1.4999e-04	2.7899e-03	2.8349e-01
Actual κ_i:		1.352e-07	1.8300e-06	1.6939e-04	3.1686e-03	3.2063e-01

8.5 Parameter Fit

8.6 Results Discussion

Several conclusions can be extracted from these results:

- All results were consistent between the tank and the column. That is, we realized similar speed up and accuracy regardless of problem size.
- The reduced order model with few exceptions reduces the time-to-solution by at least 50% versus `PEST` with `ADH` without sacrificing quality of result in either parameter error or model error.
- `PEST` with POD reduces the number of full model calls by approximately an order of magnitude.
- For most cases, tolerances of $1e-6$ and $1e-8$ in the `ADH` Tolerance achieve a similar final residual and parameter fit.
- An extremely precise sensitivity tolerance is never able to overcome low precision in the `ADH` Tolerance. However, some instances of low precision sensitivities alongside high precision `ADH` Tolerances were able to converge to a good solution.

- With few exceptions, more accuracy in the sensitivity vectors resulted in a faster time to solution.
- Our stopping criteria were too conservative for small residual cases. The optimizer with the reduced order model frequently exited before converging completely. In our final example of SPE10, we allowed the full model residual to increase for 5 iterations instead of the 3 used in the tank and column. With additional leeway, the optimizer with a reduced model was able to resolve the small residual problem to a similar final accuracy as with the full model. We do not allow 5 iterations for our investigations of low accuracy solves as the additional function evaluations do not provide a better solution in those cases.

For the reasons listed above, we chose $1e{-}6$ as the both the `ADH` Tolerance and Sensitivity Tolerance to solve SPE10. The results with those tolerances are typical of what we achieved with the smaller two domains.

From this investigation, we see the reduced order model generated by POD is viable to be used in the Levenberg Marquardt optimization process. Despite poor accuracy in both the jacobian evaluation and the actual function evaluation, the optimizer was able to find similar solutions with both the reduced order model and the full model solve. Our technique of evaluating termination criteria was shown to be viable in all three domains and for several noise levels in the data. In all cases, it was more computationally efficient to use the reduced order model in place of the full model in the optimization process. The POD model reduced the time to solution without significantly reducing the accuracy of the final parameters.

Chapter 9

Conclusions and Future Work

The previous results have clearly demonstrated the success of our method. Our goal is to demonstrate the effectiveness of a reduced model in the process to approximate the hydraulic conductivity of materials in a saturated domain. Our method reduces the time and full model calls necessary to achieve that goal without sacrificing accuracy in the parameter or model fit.

We explored four domains to test our model: a simple 2-D model with wells, a generated column, a laboratory scale aquifer, and a domain specifically designed to overwhelm the capacity of a high fidelity model. For the column and tank, we examined the behavior of our method with extremely noisy (up to 10% error) data.

We were able to test the limits of convergence for Inexact Levenberg Marquardt by manually controlling the accuracy of the solution and the jacobian throughout the optimization process.

In all of the previous scenarios, we reduced the time to solution by over 50% and reduced the number of calls to the high-fidelity simulator by an order of magnitude. In all cases, the solution from the reduced model is as accurate in both residual norm and parameter fit to the solution from the full model.

For future work, we look to include the material boundaries in the optimization process. We can generate several possibilities for the subsurface zonation scheme and test which realizations best match the data. Our reduced order model allows us to explore many more domains given the reduced time to solution.

We also seek to transfer the technology to thermal models. Several current projects at ERDC focus on modeling surface temperatures for a domain. Our reduced model allows us to test a wide range of subsurface parameters to match thermal imagery.

REFERENCES

[1] LLC Aquaveo. Gms 8.0 tutorials. *Retrieved from Aquaveo Website (URL: http://www. aquaveo. com/gms-learning)*, 2011.

[2] F. Ballio and A. Guadagnini. Convergence assessment of numerical monte carlo simulations in groundwater hydrology. *Water resources research*, 40(4):W04603, 2004.

[3] H.T. Banks, R.C.H. del Rosario, and R.C. Smith. Reduced-order model feedback control design: numerical implementation in a thin shell model. *Automatic Control, IEEE Transactions on*, 45(7):1312 –1324, jul 2000.

[4] E. B. Becker, G. F. Carey, and J. T. Oden. *Finite Elements: An Introduction*, volume 1. Prentice Hall, 1981.

[5] R. Beckie, A.A. Aldama, and E.F. Wood. The universal structure of the groundwater flow equations. *Water resources research*, 30(5):1407–1419, 1994.

[6] Eugenio Beltrami. Sulle funzioni bilineari. *Giornale di Matematiche ad Uso degli Studenti Delle Universita*, pages 98–106, 1873. English translation by Daniel Boley available from 3rd Int'l Workshop on SVD and Signal Processing, pp. 5-18, 1995.

[7] Gal Berkooz, Philip Holmes, and John L. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, 25:539–75, 1993.

[8] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK Users' Guide.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.

[9] Susan Blackford. Scalapack tutorial. `http://www.netlib.org/scalapack/tutorial`, 1998.

[10] Daniel Boley. On bilinear functions. *3rd International Workshop on SVD and signal processing*, pages 5–18, 1995.

[11] A. J. Booker, J. E. Dennis, P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset. A rigorous framework for optimization of expensive function by surrogates. *Structural Optimization*, 17:1–13, 1999.

[12] Glenn Brown. Darcy's law basics and more. `http://biosystems.okstate.edu/darcy/LaLoi/basics.htm`, 2005.

[13] John Burkardt, Max Gunzburger, and Hyung-Chun Lee. Pod and cvt-based reduced-order modeling of navier-stokes flows. *Computer Methods in Applied Mechanics and Engineering*, 196(1-3):337 – 355, 2006.

[14] M. A. Cardoso, L. J. Durlofsky, and P. Sarma. Development and application of reduced-order modeling procedures for subsurface flow simulation. *International Journal for Numerical Methods in Engineering*, 77(9):1322–1350, 2009.

[15] Kevin Carlberg and Charbel Farhat. A compact proper orthogonal decomposition basis for optimization-oriented reduced-order models. *12th AIAAISSMO Multidisciplinary Analysis and Optimization Conference*, (September), 2008.

[16] J. Carrera and S.P. Neuman. Estimation of aquifer parameters under transient and steady state conditions: 3. application to synthetic and field data. *Water Resour. Res*, 22(2):211–242, 1986.

[17] D. H. Chambers, R. J. Adrian, P. Moin, D. S. Stewart, and H. J. Sung. Karhunen–loéve expansion of burgers' model of turbulence. *Physics of Fluids*, 31(9):2573–2582, 1988.

[18] DH Chambers, RJ Adrian, P Moin, DS Stewart, and HJ Sung. Karhunen-loéve expansion of burgers' model of turbulence. *Phys. Fluids*, 31:2573–2582, 1988.

[19] M.A. Christie and M.J. Blunt. Tenth SPE comparative solution project: A comparison of upscaling techniques. *SPE Reservoir Evaluation & Engineering*, 4, 2001.

[20] Andrew R. Conn, Katya Scheinberg, and Luis N. Vicente. *Introduction to Derivative-Free Optimization.* Society for Industrial and Applied Mathematics, Philadephia, PA, 2009.

[21] Hiroshige Dan, Nobuo Yamashita, and Masao Fukushima. Convergence properties of the inexact levenberg-marquardt method under local error bound conditions. *Optimization Methods and Software*, 17(4):605–626, 2002.

[22] T.A. Davis. Algorithm 832: UMFPACK, an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30(2):196–199, June 2004.

[23] T.A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30(2):165–195, June 2004.

[24] T.A. Davis and I.S. Duff. An unsymmetric-pattern multifrontal method for sparse lu factorization. *SIAM Journal on Matrix Analysis and Applications*, 18(1):140–158, January 1997.

[25] T.A. Davis and I.S. Duff. A combined unifrontal/multifrontal method for unsymmetric sparse matrices. *ACM Transactions on Mathematical Software*, 25(1):1–19, March 1999.

[26] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. *Lecture notes in computer science*, 1917:849–858, 2000.

[27] James W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[28] J.E. Dennis. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Number 16 in Classics in Applied Mathematics. SIAM, Philadelphia, 1996.

[29] John E. Dennis, Jr., David M. Gay, and Roy E. Walsh. An adaptive nonlinear least-squares algorithm. *ACM Trans. Math. Softw.*, 7(3):348–368, September 1981.

[30] J. Doherty. Ground water model calibration using pilot points and regularization. *Ground Water*, 41(2):170–177, 2003.

[31] J. Doherty. *PEST: Model-Independent Parameter Estimation User Manual*. Watermark Numerical Computing, 5 edition, 2004.

[32] Patrick A. Domenico and Franklin W. Schwartz. *Physical and Chemical Hydrogeology*. John Wiley & Sons, Inc, 1990.

[33] Stanley C. Eisenstat and Homer F. Walker. Globally convergent inexact newton methods. *SIAM Journal on Optimization*, 4(2):393–422, 1994.

[34] K.R. Fowler, J.P. Reese, C.E. Kees, J.E. Dennis, C.T. Kelley, C.T. Miller, C. Audet, A.J. Booker, G. Couture, R.W. Darwin, et al. Comparison of derivative-free optimization methods for groundwater supply and hydraulic capture community problems. *Advances in Water Resources*, 31(5):743–757, 2008.

[35] HJ Franssen and W. Kinzelbach. Ensemble kalman filtering versus sequential self-calibration for inverse modelling of dynamic groundwater flow systems. *Journal of Hydrology*, 365(3-4):261–274, 2009.

[36] J. Fu and J. Jaime Gómez-Hernández. Uncertainty assessment and data worth in groundwater flow and mass transport modeling using a blocking markov chain monte carlo method. *Journal of Hydrology*, 364(3-4):328–341, 2009.

[37] P. Gilmore and C.T. Kelley. An implicit filtering algorithm for optimization of functions with many local minima. *SIAM Journal on Optimization*, 5:269, 1995.

[38] Mark S. Gockenbach. *Understanding and implementing the finite element method.* Society for Industrial and Applied Mathematics, Philadelphia, 2006.

[39] Gene H. Golub and William Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics*, 2(2):205–224, 1965.

[40] J.J. Gómez-Hernánez, A. Sahuquillo, and J.E. Capilla. Stochastic simulation of transmissivity fields conditional to both transmissivity and piezometric data–1. theory. *Journal of Hydrology(Amsterdam)*, 203(1):167–174, 1997.

[41] Max Gunzburger and Karen Wilcox. Reduced-order models of large scale computational systems. *SIAM News*, 38:11, 2005.

[42] Max D. Gunzburger, Janet S. Peterson, and John N. Shadid. Reduced-order modeling of time-dependent pdes with multiple parameters in the boundary data. *Computer Methods in Applied Mechanics and Engineering*, 196(4-6):1030 – 1047, 2007.

[43] Hans-Martin Gutmann. A radial basis function method for global optimization. *Journal of Global Optimization*, 19:201–227, 2001.

[44] A. Henderson, J. Ahrens, and C. Law. *The Paraview Guide.* Kitware Clifton Park, NY, 2004.

[45] HJ Hendricks Franssen, A. Alcolea, M. Riva, M. Bakr, N. Van Der Wiel, F. Stauffer, and A. Guadagnini. A comparison of seven methods for the inverse modelling of groundwater flow. application to the characterisation of well catchments. *Advances in Water Resources*, 32(6):851–872, 2009.

[46] Edward C. Heyse. Envr 640 - groundwater hydrology and contaminant transport. Class Notes, 1995. Florida State University.

[47] P. Indelman and G. Dagan. Upscaling of permeability of anisotropic heterogeneous formations: 1. the general framework. *Water Resources Research*, 29(4):917–923, 1993.

[48] Camille Jordan. Mémoire sur les formes bilinéaires. *Journal de Mathématiques Pures et Appliquées*, page 19:35, 1874.

[49] Camille Jordan. Sur la réduction des formes bilinéaires. *Comptes Rendus de l'Académie des Sciences*, page 78:614, 1874.

[50] K. Karhunen. Zur Spektraltheorie stochastischer Prozesse. *Annales Academiae Scientiarum Fennicae*, 37, 1946.

[51] Kathleen R. Kavanagh. *Nonsmooth Nonlinearities in Applications from Hydrology.* Phd thesis, North Carolina State University, July 2003.

[52] C. T. Kelley. *Iterative Methods for Optimization.* Number 18 in Frontiers in Applied Mathematics. SIAM, Philadelphia, 1999.

[53] C.T. Kelley. *Iterative Methods for Linear and Nonlinear Equations.* Number 16 in Frontiers in Applied Mathematics. SIAM, Philadelphia, 1995.

[54] C.T. Kelley and E.W. Sachs. Truncated newton methods for optimization with inaccurate functions and gradients. *Journal of Optimization Theory and Applications*, 116:83–98, 2003. 10.1023/A:1022110219090.

[55] G. Kourakos and A. Mantoglou. Inverse groundwater modeling with emphasis on model parameterization. *Water Resources Research*, 48(5):W05540, 2012.

[56] K. Kunisch and S. Volkwein. Galerkin proper orthogonal decomposition methods for parabolic problems. *Numerische Mathematik*, 90(1):117–148, 2001.

[57] K. Kunisch and S. Volkwein. Control of the burgers equation by a reduced-order approach using proper orthogonal decomposition. *Journal of Optimization Theory and Applications*, 102:345–371, August 1999.

[58] K. Levenberg. A method for the solution of certain nonlinear problems in least squares. *Quart. Appl. Math.*, 4:164–168, 1944.

[59] Y. C. Liang, H. P. Lee, S. P. Lim, W. Z. Lin, K. H. Lee, and C. G. Wu. Proper orthogonal decomposition and its applications–part i: Theory. *Journal of Sound and Vibration*, 252:527–544, May 2002.

[60] M. Loève. Functions aléatoire de second order. *Compte Rend. Acad. Sci.*, 83:297–310, 1945.

[61] Michel Loéve. *Probability Theory.* Von Nostrand, Princeton, NJ, 1955.

[62] H. Ly and H.T. Tran. Modeling and control of physical processes using proper orthogonal decomposition. *Computers and Mathematics with Applications*, 33(1-3):223–236, 2001.

[63] H. V. Ly and H. T. Tran. Proper orthogonal decomposition for flow calculation and optimal control in a horizontal cvd reactor. *Quart. J. Appl. Math.*, 60:631–656, 2002.

[64] R. Markovinović and J. D. Jansen. Accelerating iterative solution methods using reduced-order models as solution predictors. *International Journal for Numerical Methods in Engineering*, 68(5):525–541, 2006.

[65] D. W. Marquardt. An algorithm for least squares estimation of nonlinear parameters. *SIAM J.*, 11:431–441, 1963.

[66] Jerrold E. Marsden and Michael J. Hoffman. *Elementary Classical Analysis.* W.H. Freeman and Company, 1993.

[67] D. McLaughlin and L.R. Townley. A reassessment of the groundwater inverse problem. *Water Resources Research*, 32(5):1131–1161, 1996.

[68] Carl D. Meyer, editor. *Matrix analysis and applied linear algebra.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[69] Cass T. Miller. Envr 453 - groundwater hydrology. Class Notes University of North Carolina at Chapel Hill, 2006.

[70] Cass T. Miller, George Christakos, Paul T. Imhoff, John F. McBride, Joseph A. Pedit, and John A. Trangenstein. Multiphase flow and transport modeling in heterogeneous porous media: challenges and approaches. *Advances in Water Resources*, 21(2):77 – 120, 1998.

[71] C. Moore and J. Doherty. The cost of uniqueness in groundwater model calibration. *Advances in Water Resources*, 29(4):605–623, 2006.

[72] J. More. The levenberg-marquardt algorithm: implementation and theory. *Numerical analysis*, pages 105–116, 1978.

[73] J.J. Moré, S.M. Wild, et al. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1):172, 2010.

[74] Karl Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, pages 559–572, 1901.

[75] J. S. Pettway, J. H. Schmidt, and A. K. Stagg. Adaptive meshing in a mixed regime hydrologic simulation model. *Computational Geosciences*, 14(6):665–674, March 2010.

[76] M. J. D. Powell. UOBYQA: unconstrained optimization by quadratic approximation. *Math. Programming*, 92:555–582, 2002.

[77] Jill Reese. *Examining the Significance of Advective Acceleration to Single-Phase Flow Through Heterogeneous Porous Media.* Phd thesis, North Carolina State University, November 2006.

[78] R. G. Regis and C. A. Shoemaker. Local function approximation in evolutionary algorithms for the optimization of costly functions. *IEEE Transactions on Evolutionary Computation*, 8:490–505, 2004.

[79] Y. Rubin and J.J. Gómez-Hernández. A stochastic approach to the problem of upscaling of conductivity in disordered media: Theory and unconditional numerical simulations. *Water Resources Research*, 26(4):691–701, 1990.

[80] D. Russo. Upscaling of hydraulic conductivity in partially saturated heterogeneous porous formation. *Water Resour. Res*, 28(2):397–409, 1992.

[81] Toshihiro Sakaki, Christophe C. Frippiat, Mitsuru Komatsu, and Tissa H. Illangasekare. On the value of lithofacies data for improving groundwater flow model accuracy in a three-dimensional laboratory-scale synthetic aquifer. *Water Resour. Res.*, 45(11):18, 2009.

[82] G. W. Stewart. On the Early History of the Singular Value Decomposition. *SIAM Review*, 35:551–566, December 1993.

[83] A.N. Tikhonov, V.I.A. Arsenin, and F. John. Solutions of ill-posed problems. page 258, 1977.

[84] M.J. Tonkin and J. Doherty. A hybrid regularized inversion methodology for highly parameterized environmental models. *Water Resources Research*, 41(10):W10412, 2005.

[85] Jorn van Doren, R. Markovinović, and Jan-Dirk Jansen. Reduced-order optimal control of water flooding using proper orthogonal decomposition. *Computational Geosciences*, 10(1):137–158, 2006.

[86] P.T.M. Vermeulen, A.W. Heemink, and C.B.M. Te Stroet. Reduced models for linear groundwater flow models using empirical orthogonal functions. *Advances in Water Resources*, 27(1):57 – 69, 2004.

[87] C.R. Vogel. *Computational methods for inverse problems*, volume 23. Society for Industrial Mathematics, Philadelphia, PA, 2002.

[88] Corey Winton, Jackie Pettway, C.T. Kelley, Stacy Howington, and Owen J. Eslinger. Application of proper orthogonal decomposition (pod) to inverse problems in saturated groundwater flow. *Advances in Water Resources*, 34(12):1519 – 1526, 2011.

[89] S. J. Wright and J. N. Holt. An inexact levenberg-marquardt method for large sparse nonlinear least squres. *The ANZIAM Journal*, 26(04):387–403, 1985.

[90] T.C.J. Yeh, L.W. Gelhar, and A.L. Gutjahr. Stochastic analysis of unsaturated flow in heterogeneous soils: 1. statistically isotropic media. *Water Resour. Res*, 21(4):447–456, 1985.

[91] WG Yeh. Review of parameter identification procedures in groundwater hydrology. *Water Resour. Res*, 22(2):95–108, 1986.

APPENDICES

Appendix A

Finite Element Method

A.1 General Dirichlet BVP

We will begin with a Dirichlet problem:

$$\begin{aligned} -\nabla \cdot (\kappa \nabla u) &= f \text{ in } \Omega, & \text{(A.1a)} \\ u &= 0 \text{ on } \partial\Omega. & \text{(A.1b)} \end{aligned}$$

Following the text in [38], we will derive the weak form of the BVP (A.1) and show the solution of the weak form is equivalent. We first define the following Sobolev spaces (see § B for further information):

$$\begin{aligned} H_0^1(\Omega) &= \left\{ v \in H^1(\Omega) \; : \; v = 0 \text{ on } \partial\Omega \right\}. & \text{(A.2a)} \\ H^1(\Omega) &= \left\{ v \in L^2(\Omega) \; : \; \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y} \in L^2(\Omega) \right\}. & \text{(A.2b)} \\ L^2(\Omega) &= \left\{ v : \Omega \to \mathbb{R} \; : \; \int_\Omega v^2 < \infty \right\}. & \text{(A.2c)} \end{aligned}$$

We assume u is a solution of (A.1) and multiply both sides of the equation by a test function $v \in H_0^1(\Omega)$ and integrate over Ω to see

$$-\int_\Omega \nabla \cdot (\kappa \nabla u) v = \int_\Omega f v \text{ in } \Omega. \qquad \text{(A.3)}$$

We then use Green's first identity

$$-\int_\Omega v\Delta u = \int_\Omega \nabla v \cdot \nabla u - \int_{\partial\Omega} v\frac{\partial u}{\partial n}, \tag{A.4}$$

to the left side of (A.3) to see

$$-\int_\Omega \nabla\cdot(\kappa\nabla u)v = \int_\Omega \kappa\nabla u\cdot\nabla v - \int_{\partial\Omega}\kappa v\frac{\partial u}{\partial n}. \tag{A.5}$$

However, $v \equiv 0$ on $\partial\Omega$, so $\int_{\partial\Omega}\kappa v\frac{\partial u}{\partial n} = 0$. This leaves the weak formulation of (A.1): Find $u \in H_0^1(\Omega)$ such that for all $v \in H_0^1(\Omega)$,

$$\int_\Omega \kappa\nabla u\cdot\nabla v = \int_\Omega fv. \tag{A.6}$$

A.1.1 Equivalence of Forms

To see that (A.6) and (A.1) are indeed equivalent, we once again use the analysis found in [38]. We denote the ball of radius δ centered at (x_0, y_0) as $B_\delta(x_0, y_0)$ and is defined by

$$B_\delta(x_0,y_0) = \{(x,y)\in\mathbb{R}^2 \ : \ ||(x,y)-(x_0,y_0)|| < \delta\}. \tag{A.7}$$

We then suppose $(x_0, y_0) \in \Omega$ and $\delta > 0$ small enough to ensure $B_\delta(x_0, y_0)$ is entirely within Ω. We construct $v \in C_D^2(\overline{\Omega})$ with the following requirements:

1. $v(x,y) > 0 \ \forall \ (x,y) \in B_\delta(x_0,y_0)$,
2. $v(x,y) = 0 \ \forall \ (x,y) \notin B_\delta(x_0,y_0)$,
3. $\int_\Omega v = \int_{B_\delta(x_0,y_0)} v = 1.$

Therefore,

$$\int_\Omega fv = \int_{B_\delta(x_0,y_0)} fv, \tag{A.8}$$

$$-\int_\Omega \nabla\cdot(\kappa\nabla u)v = -\int_{B_\delta(x_0,y_0)}\nabla\cdot(\kappa\nabla u)v, \tag{A.9}$$

are both weighted averages of the integrands over $B_\delta(x_0, y_0)$. If we take the limit as $\delta \to 0$, then we see

$$\begin{aligned} \int_{B_\delta(x_0,y_0)} fv &= f(x_0, y_0), && \text{(A.10)} \\ -\int_{B_\delta(x_0,y_0)} \nabla \cdot (\kappa \nabla u) v &= -\nabla \cdot (\kappa(x_0, y_0) \nabla u(x_0, y_0)). && \text{(A.11)} \end{aligned}$$

Thus, if the space of test functions encompasses all functions previously defined and u satisfies (A.6), then the strong form of the BVP (A.1) must hold at every $(x_0, y_0) \in \Omega$.

A.2 General Neumann BVP

We next derive the weak formulation of a general Neumann BVP:

$$\begin{aligned} -\nabla \cdot (\kappa \nabla u) &= f \text{ in } \Omega, && \text{(A.12a)} \\ \kappa \frac{\partial u}{\partial n} &= 0 \text{ on } \partial\Omega. && \text{(A.12b)} \end{aligned}$$

We define the test function $v \in H^1(\Omega)$. We once again multiply (A.12) by our test function v and integrate over Ω.

$$-\int_\Omega \nabla \cdot (\kappa \nabla u) v = \int_\Omega fv \text{ for all } v \in H^1(\Omega) \qquad \text{(A.13)}$$

We use Green's identity and recall (A.12b) to zero the boundary integral. This leads us to the weak formulation of the Neumann boundary problem: Find $u \in H^1(\Omega)$ such that

$$\int_\Omega \kappa \nabla u \cdot \nabla v = \int_\Omega fv \qquad \text{(A.14)}$$

for all $v \in H^1(\Omega)$.
Similar analysis can be done to show the weak form for the Neumann problem is equivalent to the strong form.

A.3 Mixed and Inhomogeneous Boundary Conditions

A.3.1 Mixed Boundary Conditions

The mixed boundary value problem is stated:

$$-\nabla \cdot (\kappa \nabla u) = f \text{ in } \Omega, \tag{A.15a}$$

$$u = 0 \text{ on } \Gamma_1, \tag{A.15b}$$

$$\kappa \frac{\partial u}{\partial n} = 0 \text{ on } \partial\Gamma_2, \tag{A.15c}$$

where $\partial\Omega = \Gamma_1 \cup \Gamma_2$ and $\Gamma_1 \cap \Gamma_2 = \emptyset$. For this problem, we define the space of test functions to be

$$V = \{v \in H^1(\Omega) \;:\; v = 0 \text{ on } \Gamma_1\}. \tag{A.16}$$

Using a progression similar to that of the Dirichlet and Neumann problems, only this time splitting the boundary integral into two separate integrals

$$\int_{\partial\Omega} \kappa v \frac{\partial u}{\partial n} = \int_{\Gamma_1} \kappa v \frac{\partial u}{\partial n} + \int_{\Gamma_2} \kappa v \frac{\partial u}{\partial n}, \tag{A.17}$$

we have similar results.

A.3.2 Inhomogeneous Boundary Conditions

The weak form of the inhomogeneous Neumann boundary value problem is straightforward to derive. We begin with

$$-\nabla \cdot (\kappa \nabla u) = f \text{ in } \Omega, \tag{A.18a}$$

$$\kappa \frac{\partial u}{\partial n} = h \text{ on } \partial\Omega. \tag{A.18b}$$

The derivation is the same as previously, except now the boundary integral remains a part of the problem statement. That is:

$$\int_{\partial\Omega} v\kappa \frac{\partial u}{\partial n} = \int_{\partial\Omega} vh. \tag{A.19}$$

Thus, the weak statement of the problem is: Find $u \in H^1(\Omega)$ such that

$$\int_\Omega \kappa \nabla u \cdot \nabla v = \int_\Omega fv + \int_{\partial\Omega} vh \tag{A.20}$$

holds for all $v \in H^1(\Omega)$.

The Inhomogeneous Dirichlet problem is slightly different. We pose the problem

$$\begin{aligned} -\nabla \cdot (\kappa \nabla u) &= f \text{ in } \Omega, && \text{(A.21a)} \\ u &= g \text{ on } \partial\Omega. && \text{(A.21b)} \end{aligned}$$

To construct the weak form of this equation, we assume there is a function $G \in H^1(\Omega)$ such that $G = g$ on $\partial\Omega$. Our space of test functions is $H_0^1(\Omega)$ from (A.2). However, $u = g \neq 0$ on $\partial\Omega$, so $u \notin H_0^1$. We declare $w = u - G = 0$ on $\partial\Omega$, so the solution to (A.21) is $u = w + G$, where G is defined by (A.21b). We follow a similar progression to the previous problems:

$$\begin{aligned} -\nabla \cdot (\kappa(\nabla w + \nabla G)) &= f \in \Omega && \text{(A.22a)} \\ -\int_\Omega \nabla \cdot (\kappa(\nabla w + \nabla G))v &= \int_\Omega fv \; \forall v \in H_0^1(\Omega) && \text{(A.22b)} \\ \int_\Omega \kappa(\nabla w + \nabla G) \cdot \nabla v - \int_{\partial\Omega} \kappa v \left(\frac{\partial w}{\partial n} + \frac{\partial G}{\partial n} \right) &= \int_\Omega fv \; \forall v \in H_0^1(\Omega) && \text{(A.22c)} \\ \int_\Omega \kappa(\nabla w + \nabla G) \cdot \nabla v &= \int_\Omega fv \; \forall v \in H_0^1(\Omega). && \text{(A.22d)} \end{aligned}$$

The boundary integral is zero since $v \equiv 0$ on $\partial\Omega$. The weak formulation is thus: Find $u = w + G,\ w \in H_0^1(\Omega)$ such that

$$\int_\Omega \kappa \nabla w \cdot \nabla v = \int_\Omega fv - \int_\Omega \kappa \nabla G \cdot \nabla v \tag{A.23}$$

for all $v \in H_0^1(\Omega)$.

The last BVP we examine has inhomogeneous, mixed boundary conditions:

$$\begin{aligned} -\nabla \cdot (\kappa \nabla u) &= f \text{ in } \Omega, && \text{(A.24a)} \\ u &= g \text{ on } \Gamma_1, && \text{(A.24b)} \\ \kappa \frac{\partial u}{\partial n} &= h \text{ on } \Gamma_2. && \text{(A.24c)} \end{aligned}$$

We define $G = g$ on Γ_1 and the space

$$V = \{v \in H^1(\Omega) \;:\; v = 0 \text{ on } \Gamma_1\}. \tag{A.25}$$

Our solution will once again be of the form $u = w + G$, where $w \in V$. We follow the process in (A.22) with $v \in V$, but now the boundary integral will not be zero. Instead, we see

$$\begin{aligned} \int_{\partial\Omega} \kappa v \left(\frac{\partial w}{\partial n} + \frac{\partial G}{\partial n} \right) &= \int_{\Gamma_1} \kappa v \left(\frac{\partial w}{\partial n} + \frac{\partial G}{\partial n} \right) + \int_{\Gamma_2} \kappa v \left(\frac{\partial w}{\partial n} + \frac{\partial G}{\partial n} \right), \\ \int_{\Gamma_1} \kappa v \left(\frac{\partial w}{\partial n} + \frac{\partial G}{\partial n} \right) &\equiv 0, \\ \int_{\Gamma_2} \kappa v \left(\frac{\partial w}{\partial n} + \frac{\partial G}{\partial n} \right) &= \int_{\Gamma_2} vh. \end{aligned}$$

The weak formulation for a mixed, inhomogeneous boundary problem is: Find $u = w + G,\ w \in V$ such that

$$\int_\Omega \kappa \nabla w \cdot \nabla v = \int_\Omega fv - \int_\Omega \kappa \nabla G \cdot \nabla v + \int_{\partial\Gamma_2} vh \tag{A.26}$$

for all $v \in V$.

A.4 Discretization

In this section, we will outline how to discretize the weak formulation of a BVP into a matrix equation using the Galerkin method. Again, this follows the discussion in [38].

A.4.1 Linear Functionals

We define a linear functional:

Definition A.4.1 *In a normed linear space V, a linear functional ℓ is a real-valued function that is linear:*

$$\ell(\alpha u + \beta v) = \alpha\ell(u) + \beta\ell(v)\ \forall u, v \in V,\ \alpha, \beta \in \mathbb{R}.$$

We define continuity for a linear functional:

Definition A.4.2 *ℓ is continuous at $u \in V$ if*

$$\lim_{v \to u} \ell(v) = \ell(u).$$

Equivalently,

$$|\ell(u) - \ell(v)| \to 0 \text{ as } ||u - v|| \to 0.$$

We define the norm of the linear functional as the smallest nonnegative constant M such that

$$|\ell(u)| \leq M||u||\ \forall u \in V.$$

The *dual space* of V, denoted V^* is the set of all continuous linear functionals defined on V. The norm for the dual space is given by

$$||\ell||_{V^*} = \sup\{|\ell(u)|\ :\ u \in V,\ ||u|| \leq 1\}.$$

A.4.2 Existence and Uniqueness

Now that we have defined linear functionals and the dual space of V, we can turn to existence and uniqueness theory for our BVP. We begin with the Riesz representation theorem:

Theorem A.4.3 (The Riesz Representation Theorem) *Suppose V is a Hilbert space. Then V^* can be identified with V in the following sense:*

1. *For each $u \in V$, the linear functional ℓ defined by $\ell(v) =< u|v >$ belongs to V^* and*

$$||\ell||_{V^*} = ||u||_V.$$

2. *For each $\ell \in V^*$, there exists a unique $u \in V$ such that*

$$||\ell||_{V^*} = ||u||_V$$

and

$$\ell(v) =< u|v > \ \forall v \in V.$$

Examining some of the problems stated previously, we see the weak forms can be stated:

$$u \in H_0^1(\Omega),\ \int_\Omega \kappa \nabla u \cdot \nabla v = \int_\Omega fv \ \forall v \in H_0^1(\Omega),$$
$$u \in H^1(\Omega),\ \int_\Omega \kappa \nabla u \cdot \nabla v = \int_\Omega fv \ \forall v \in H^1(\Omega),$$
$$u \in H^1(\Omega),\ \int_\Omega \kappa \nabla u \cdot \nabla v = \int_\Omega fv + \int_{\partial\Omega} vh \ \forall v \in H^1(\Omega).$$

We could rewrite these problems in the form:

$$u \in V,\ a(u,v) = \ell(v) \ \forall v \in V, \tag{A.27}$$

where V is a Hilbert space. $a(\cdot,\cdot)$ is a symmetric bilinear form and satisfies the following properties of an inner product:

1. $a(u,v) = a(v,u) \ \forall u, v \in V.$
2. $a(\alpha u + \beta v, w) = \alpha a(u,w) + \beta(v,w) \ \forall u, v, w \in V,\ \forall \alpha, \beta \in \mathbb{R}.$
3. $a(u,u) \geq 0,\ u = 0 \Rightarrow a(u,u) = 0.$

We also note:

4. If there exists $\alpha > 0$ such that $a(u,u) \geq \alpha ||u||^2 \ \forall u \in V$, $a(\cdot,\cdot)$ is elliptic.
5. If there exists $\beta > 0$ such that $a(u,v) \leq \beta ||u|| ||v|| \ \forall u, v \in V$, $a(\cdot,\cdot)$ is bounded.

To demonstrate existence and uniqueness of our solution, we must establish V is complete with the inner product defined by $a(\cdot,\cdot)$ (§B shows why a solution exists and unique if V is complete). To accomplish this, we show the norm generated by $a(\cdot,\cdot)$ is equivalent to the standard norm on V. We define the energy norm:

$$||v||_E = \sqrt{a(v,v)}.$$

If ℓ is a continuous linear functional, then there is a constant $M \geq 0$ such that

$$|\ell(v)| \leq M||v|| \; \forall v \in V.$$

Since our BVP's are elliptic, we see

$$||v|| \leq \alpha^{-1/2}||v||_E,$$

and therefore ℓ is bounded with respect to the energy norm by

$$|\ell(v)| \leq \left(\alpha^{-1/2}M\right) ||v||_E \; \forall v \in V.$$

Similarly, if $a(\cdot,\cdot)$ is bounded, we can see

$$a(v,v) \leq \beta||v||^2 \Rightarrow ||v||_E \leq \sqrt{\beta}||v|| \; \forall v \in V.$$

Thus, the energy norm is equivalent to the standard norm on V.

A.4.3 Elliptic and Bounded

As the previous section demonstrated, if our problem is elliptic and bounded then the space V is still complete under the norm generated by $a(\cdot,\cdot)$ and thus the solution exists and is unique. The next sections will discuss how to obtain that solution. In this section, we show our problems are indeed elliptic and bounded. For sake of clarity, we will examine the inhomogeneous dirichlet problem:

$$-\nabla \cdot (\kappa \nabla u) = f \text{ in } \Omega \tag{A.28a}$$

$$u = g \text{ on } \partial\Omega. \tag{A.28b}$$

The weak form, as derived earlier, is

$$u = w + G, \; w \in V, \; a(w,v) = \ell(v) \; \forall v \in V, \tag{A.28c}$$

where

$$\begin{aligned} V &= H^1_0(\Omega), & \text{(A.28d)}\\ a(w,v) &= \int_\Omega \kappa \nabla w \cdot \nabla v, & \text{(A.28e)}\\ \ell(v) &= \int_\Omega fv - \int_\Omega \kappa \nabla G \cdot \nabla v. & \text{(A.28f)} \end{aligned}$$

We recall the definition of an inner product on V and $L^2(\Omega)$:

$$< u|v >_V = \int_\Omega (uv + \nabla u \cdot \nabla v)$$

$$< u|v >_{L^2(\Omega)} = \int_\Omega uv,$$

and notice

$$||\nabla u||^2_{L^2(\Omega)} = \int_\Omega \nabla u \cdot \nabla u \leq \int_\Omega (u^2 + \nabla u \cdot \nabla u) = ||u||^2_V.$$

Using the Cauchy-Schwartz inequality

$$| < u|v > | \leq ||u|| ||v|| \; \forall u, v \in V, \tag{A.29}$$

and assuming there are constants k_0, k_1 such that

$$k_0 \leq \kappa \leq k_1,$$

we can show our problem is bounded:

$$\begin{aligned} a(u,v) &= \int_\Omega \kappa \nabla u \cdot \nabla v \\ &\leq k_1 \int_\Omega |\nabla u \cdot \nabla v| \\ &\leq k_1 \int_\Omega ||\nabla u|| \; ||\nabla v|| \text{ (Cauchy-Schwartz)} \\ &\leq k_1 ||\nabla u||_{L^2(\Omega)} ||\nabla v||_{L^2(\Omega)} \text{ (Cauchy-Schwartz for} L^2(\Omega)) \\ &\leq k_1 ||u||_V ||v||_V. \end{aligned}$$

Now we show that $a(\cdot,\cdot)$ is $H^1_0(\Omega)$-elliptic. We will use *Poincare's Inequality* [38]:

Definition A.4.4 (Poincare's Inequality) *There exists a positive constant C, depending only on the domain Ω, such that*

$$\sqrt{\int_\Omega \nabla u \cdot \nabla u} \geq C||u||_{H^1(\Omega)} \; \forall u \in H_0^1(\Omega)$$

Therefore, given any $u \in H_0^1(\Omega)$,

$$\begin{aligned} a(u,u) &= \int_\Omega \kappa \nabla u \cdot \nabla u \\ &\geq k_0 \int_\Omega \nabla u \cdot \nabla u \\ &\geq k_0 C^2 ||u||^2_{H^1(\Omega)} \end{aligned}$$

and we see $a(\cdot,\cdot)$ is elliptic. Since $a(\cdot,\cdot)$ is both elliptic and bounded, we have shown it generates an equivalent norm to the standard norm on V. Thus, V is complete with the energy norm and the solution exists and unique.

It should be noted that Definition A.4.4 only holds for H_0^1. For the case of mixed boundary conditions, we have to use a slightly different definition [38]:

Definition A.4.5

$$\sqrt{\int_\Omega \nabla u \cdot \nabla u} \geq C||u||_{H^1(\Omega)} \; \forall u \in V.$$

Other specific cases will require further definitions. However, the procedure is similar throughout for showing a problem is elliptic in the space.

A.4.4 Projection Theory

We begin with the the projection theorem:

Theorem A.4.6 *Suppose V is an inner product space, W is a finite-dimensional subspace of V, and $u \in V$. Then*

1. *there is a unique vector $w \in W$ satisfying*

$$||u - w|| < ||u - z|| \; \forall z \in W, \; z \neq w.$$

The vector w is called the "best approximation to u from W" or the "projection of u onto W" and is denoted $proj_W u$.

2. *a vector w is the best approximation to u from W if and only if it satisfies the following orthogonality condition:*

$$w \in W, \ < u - w|z >= 0 \ \forall z \in W.$$

Since W is finite-dimensional, it has a basis $\{w_1, ..., w_n\}$ and any element $w \in W$ can be represented as a linear combination of the basis elements:

$$w = \sum_{j=1}^{n} \alpha_j w_j.$$

Thus, if we assume w is the projection of u onto W (that is, $w = proj_W u = \sum_{j=1}^{n} \alpha_j w_j$), then we see the following hold for any $z \in W$ and specifically for w_i:

$$\left(u - \sum_{j=1}^{n} \alpha_j w_j, w_i\right) = 0, \ i = 1, ...n$$
$$\Rightarrow < u|w_i > - \sum_{j=1}^{n} \alpha_j < w_j|w_i >= 0, \ i = 1, ...n$$
$$\Rightarrow \sum_{j=1}^{n} \alpha_j < w_j|w_i >=< u|w_i >, \ i = 1, ...n.$$

We can see that $w = proj_W u$ is determined by a system of linear equations with unknown $\alpha = \alpha_1, ..., \alpha_n$, where $D_{ij} =< w_j|w_i >, \ b_i =< u|w_i >$ and $D\alpha = b$.

A.4.5 The Galerkin Method

As shown previously, we can represent a BVP in the form:

$$\text{find } u \in V, \text{ such that } a(u, v) = \ell(v) \ \forall v \in V,$$

where $a(\cdot,\cdot)$ is symmetric bilinear on the Hilbert space V and ℓ is a continuous linear functional. We also showed in a finite dimensional subspace W of V, the best approximation to the true solution u is given by the solution of $D\alpha = b$, where

$$D_{ij} =< w_j|w_i >,\ i,j = 1,...,n$$
$$b_i =< u|w_i >,\ i = 1,...n.$$

The matrix D is able to be constructed as we have the basis vectors w_i. However, without the exact solution u, we cannot construct b_i. We see that if we are dealing with an elliptic BVP, then $a(\cdot,\cdot)$ defines an inner product on V since

$$a(v,v) \geq \alpha||v||^2\ \forall v \in V\ (\alpha > 0)$$
$$a(w,v) \leq \beta||w||\ ||v||\ \forall w,v \in V\ (\beta > 0).$$

The Galerkin method computes the best approximation to u using the inner product defined by the bilinear form $a(\cdot,\cdot)$. To find the best approximation to u, we solve $KU = F$, where the stiffness matrix K is defined:

$$K_{ij} = a(w_j, w_i),\ i,j = 1,...,n$$

and the load vector F is defined

$$F_i = a(u, w_i) = \ell(w_i),\ i = 1,...,n.$$

The vector U defines the approximate solution:

$$w = \sum_{i=1}^{n} U_i w_i.$$

Appendix B

Functional Analysis

B.1 Essential Functional Analysis Definitions

We first define what it is to be a vector space [68]:

Definition B.1.1 *The set V is called a* vector space over F *when the vector addition and scalar multiplication operations satisfy the following properties:*

1. $\boldsymbol{x} + \boldsymbol{y} \in V$ *for all* $\boldsymbol{x}, \boldsymbol{y} \in V$.
2. $(\boldsymbol{x} + \boldsymbol{y}) + \boldsymbol{z} = \boldsymbol{x} + (\boldsymbol{y} + \boldsymbol{z})\ \forall \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in V$.
3. $\boldsymbol{x} + \boldsymbol{y} = \boldsymbol{y} + \boldsymbol{x}\ \forall \boldsymbol{x}, \boldsymbol{y} \in V$.
4. *There is an element* $\boldsymbol{0} \in V$ *such that* $\boldsymbol{x} + \boldsymbol{0} = \boldsymbol{x}\ \forall \boldsymbol{x} \in V$.
5. *For each* $\boldsymbol{x} \in V$, *there is an element* $(-\boldsymbol{x}) \in V$ *such that* $\boldsymbol{x} + (-\boldsymbol{x}) = \boldsymbol{0}$.
6. $\alpha \boldsymbol{x} \in V$ *for all* $\alpha \in F$ *and* $\boldsymbol{x} \in V$.
7. $(\alpha\beta)\boldsymbol{x} = \alpha(\beta\boldsymbol{x})\ \forall \alpha, \beta \in F$ *and* $\forall \boldsymbol{x} \in V$.
8. $\alpha(\boldsymbol{x} + \boldsymbol{y}) = \alpha\boldsymbol{x} + \alpha\boldsymbol{y}\ \forall \alpha \in F$ *and* $\forall \boldsymbol{x}, \boldsymbol{y} \in V$.
9. $(\alpha + \beta)\boldsymbol{x} = \alpha\boldsymbol{x} + \beta\boldsymbol{x}\ \forall \alpha, \beta \in F$ *and* $\forall \boldsymbol{x} \in V$.
10. $1\boldsymbol{x} = \boldsymbol{x}\ \forall \boldsymbol{x} \in V$.

We also define a general inner product [68]:

Definition B.1.2 *An* inner product *on a vector space V is a function that maps each ordered pair of vectors x, y to a real or complex scalar $< x|y >$ such that the following hold:*

1. $< \boldsymbol{x}|\boldsymbol{x} >$ *is real with* $< \boldsymbol{x}|\boldsymbol{x} > \geq 0$,
2. $< \boldsymbol{x}|\boldsymbol{x} >= 0 \iff \boldsymbol{x} = 0$,
3. $< \boldsymbol{x}|\alpha\boldsymbol{y} >= \alpha < \boldsymbol{x}|\boldsymbol{y} >$ *for all scalars* α,
4. $< \boldsymbol{x}|\boldsymbol{y} + \boldsymbol{z} >=< \boldsymbol{x}|\boldsymbol{y} > + < \boldsymbol{x}|\boldsymbol{z} >$,
5. $< \boldsymbol{x}|\boldsymbol{y} >= \overline{< \boldsymbol{y}|\boldsymbol{x} >}$

Any real or complex space where Definition B.1.2 holds is called an *inner-product space.* We define a general vector norm as well [68]:

Definition B.1.3 *A* norm *for a vector space V is a function $||\cdot||$ mapping V into $\mathbb{R}$ that satisfies the following:*

1. $||\boldsymbol{x}|| \geq 0$,
2. $||\boldsymbol{x}|| = 0 \iff \boldsymbol{x} = 0$,
3. $||\alpha\boldsymbol{x}|| = |\alpha|\ ||\boldsymbol{x}||$ *for all scalars* α,
4. $||\boldsymbol{x} + \boldsymbol{y}|| \leq ||\boldsymbol{x}|| + ||\boldsymbol{y}||$.

There may be many norms defined on a vector space. We define if norms are equivalent by [27]:

Definition B.1.4 *Let $||\cdot||_\alpha$ and $||\cdot||_\beta$ be two norms. There are constants $c_1, c_2 > 0$ such that, for all x,*

$$c_1||x||_\alpha \leq ||x||_\beta \leq c_2||x||_\alpha.$$

We say that norms $||\cdot||_\alpha$ and $||\cdot||_\beta$ are equivalent *with respect to constants c_1 and c_2.*

We can see that if a sequence converges with respect to a specific norm, then it will also converge with respect to any equivalent norm. We can now define a Cauchy sequence [66]:

Definition B.1.5 *A sequence x_n is called a* Cauchy sequence *if for every $\varepsilon > 0$, there exists and integer N such that $||x_n - x_m|| < \varepsilon$ if $n \geq N, m \geq N$.*

Finally, we define completeness by [66]:

Definition B.1.6 *A normed vector space V is said to be* complete *if every Cauchy sequence in V converges to an element of V.*

A good example of a space that is not complete is the rational numbers. We can use the example of the real number π to demonstrate. There is a cauchy sequence of rational numbers that will converge to π:

$$3,\ 3.1,\ 3.14,\ 3.141,\ 3.1415, ...$$

however π is not in the rational numbers so this cauchy sequence of rational numbers will not converge to an element in the space. Therefore, we see the rational numbers are not complete. This leads to the concept of a dense subspace:

Definition B.1.7 *Suppose V is a normed vector space. A subset W is said to be* dense *in V if, given any $v \in V$ and any $\varepsilon > 0$, there exists $w \in W$ with $||v - w|| < \varepsilon$.*

We can see that the rational numbers are dense in the reals because any real number can be approximated by a rational. This brings us to the discussion of Sobolev spaces and why they are used in § A [38].

Definition B.1.8 *The following are some of the dense subspaces of the Sobolov spaces:*

1. *$C(\overline{\Omega})$ is dense in $L^2(\Omega)$ (that is, $L^2(\Omega)$ is the completion of $C(\overline{\Omega})$ under the L^2 norm).*
2. *$C_0^\infty(\Omega)$ is dense in $L^2(\Omega)$.*
3. *$C^1(\overline{\Omega})$ is dense in $H^1(\Omega)$.*
4. *$C_0^1(\Omega)$ is dense in $H_0^1(\Omega)$.*

5. *$C_0^\infty(\Omega)$ is dense in $H_0^1(\Omega)$.*

So, from this we can see why Sobolov spaces are used as our test-spaces. First, the bounded integrals ensure that the weak formulation of the problem is well defined. Second, we can see that any convergent sequence of solutions in the Sobolov space will converge to a unique solution in that space.

Appendix C

ADH

The following discussion is broken into two parts. In § C.1, we will discuss the methods used to use POD with `ADH`. in § C.2, we will discuss the specific subroutines altered in the `ADH` code to achieve that purpose.

C.1 POD with ADH

We must alter the original `ADH` code as it does not generate the "classic" finite element matrix and load vector analogous to those used in (3.6). Instead, `ADH` calculates the residual R in the Richards equation

$$R = \frac{\partial \theta}{\partial t} - \frac{\partial}{\partial x}\left[k(h)\left(\frac{\partial h}{\partial x} + 1\right)\right], \tag{C.1}$$

where

$$\begin{aligned} \theta &= \text{water content}, \\ k(h) &= \text{hydraulic conductivity}, \\ h &= \text{hydraulic head}. \end{aligned}$$

So, for a given value of h, `ADH` computes the necessary parameters in (C.1) and then finds the residual R that will result if h does not satisfy the conditions set forth by the conductivity k and the boundary conditions. Using R, `ADH` then builds the matrix

$$A = R' = \frac{\partial R}{\partial h} = \frac{R(h+\epsilon) - R(h-\epsilon)}{2\epsilon}. \tag{C.2}$$

`ADH` then solves the update equation

$$\frac{\partial R}{\partial h}\partial h = R, \tag{C.3a}$$
$$h = h_0 + \partial h. \tag{C.3b}$$

From the matrix A and vector R used in `ADH`, we wish to extract the “classic” finite element matrix and vector. Fortunately, we are able to still use zonation and create the submatrices

$$\begin{aligned} A &= A_0 + \sum_i A_i \kappa_i, \\ R &= R_0 + \sum_i R_i \kappa_i. \end{aligned}$$

We note that, since we are solving a saturated flow problem, R will be linear and can be represented

$$R(h) = f - Ah. \tag{C.4}$$

As we know $R(h_0)$, we will evaluate (C.2) at h_0 to find A. Furthermore, we see that if we can find A, f then solving the equation

$$Ah = f, \tag{C.5}$$

will set the residual R to zero and thus the matrix equation we are developing will yield the same solution as `ADH`.
We can extract the vector f easily as we know all the components in (C.4) except f at our initial head value h_0 (recall R, A both are based on the head value h):

$$f = R(h_0) + Ah_0. \tag{C.6}$$

We still must generate the f_i in order to solve for the sensitivities. We do this by noting h_0 is the initial guess and therefore unaffected by the conductivity $k(x)$. Thus, we can perturb $k(x)$ in each zone individually

$$\widetilde{k(x)}_\ell = \sum_{i\neq\ell}^{M} \kappa_i \chi_i + (\kappa_\ell + \epsilon)\chi_\ell. \tag{C.7}$$

We assume

$$f = \sum_{i}^{M} f_i \kappa_i, \tag{C.8}$$

and thus see

$$\begin{aligned}
\widetilde{R(h_0)}_\ell &= \sum_{i\neq\ell}^{M} \kappa_i f_i + (\kappa_\ell + \epsilon) f_\ell \; - \sum_{i\neq\ell}^{M} \kappa_i A_i h_0 - (\kappa_\ell + \epsilon) A_\ell h_0 \\
&= f + \epsilon f_\ell - A h_0 - \epsilon A_\ell h_0.
\end{aligned}$$

Therefore,

$$f_\ell = \frac{\widetilde{R(h_0)}_\ell - R(h_0) + \epsilon A_\ell h_0}{\epsilon}. \tag{C.9}$$

Because h_0 is unaffected by the conductivity, this does not require any more iterations of `ADH`, all we must do is generate an additional residual R for the M different zones in the region. By doing this, we extract the data from `ADH` to solve the matrix-vector equation. We recall (C.4) and see that

$$\left[A_0 + \sum_i A_i \kappa_i \right] h = f_0 + \sum_i f_i \kappa_i - R. \tag{C.10}$$

It should be noted the magnitude of R is a user-defined parameter in `ADH`. We normally set $R \approx 1e^{-6}$, but because $R \neq 0$, we must include R in (C.10).

C.2 Relevant Codes

We must make several adjustments to the `ADH` code in order to use our POD technique:

- `fe_gw_resid.c`: This code generates the residual R from (C.1). We must modify this to break the residual into material-specific vectors.

- `fe_gw_elem_resid.c`: This code goes element by element and forms the residual that will be combined in `fe_gw_resid.c`. We must remove the conductivity from the calculation and instead substitute a 1. We must do this so we can recombine the matrix as shown in (C.10).

- `fe_gw_load.c`: This generates the matrix A from the previously computed R. The routine computes the contribution to a reference element, then adds that reference element to the full matrix structure. We make two modifications:

  1. Create a reference element for each material. If the element is not the active material, a set of zeros are added to the non-active material matrices.
  2. If the element has boundary information, the corresponding information in the non-active material matrices must be eliminated. All Dirichlet information is stored in one non-material-specific matrix.

- `fe_assmb_matrix.c`: Typically this routine takes the reference element and includes that information in the full matrix. As we have a matrix specific to each material, we alter the routine to accommodate several matrices and several incoming reference elements.

Appendix D

PEST

In this section, we will discus the use of PEST and which variables we altered from the default settings to improve convergence of the optimizer.

D.1 PEST - POD Interaction

PEST interacts with the POD reduced model through simple file I/O. It writes the current set of parameters to a text file, then executes a simple bash script. This script flips a semaphore that triggers a POD run, then waits for POD to flip the semaphore back indicating completion. When the semaphore has been flipped, the bash script exits and PEST evaluates the error and adjusts the parameters. This process repeats until PEST has found a satisfactory solution. At this point, the full model is run with the new parameters, POD updates the basis and other information, and PEST starts anew.

D.2 Variables

We list each of the variables in the PEST header, the value we assign to it, and a brief explanation of what the variable indicates. If the parameters change for the synthetic column vs. the tank, they are listed as an ordered pair with the column first. All descriptions are referenced from the PEST user manual [31].

D.2.1 General `PEST` Settings

- `RSTFILE`: *restart* – Allows `PEST` to restart in case of a crash.
- `PESTMODE`: *estimation* – Set `PEST` to parameter estimation mode.
- `NPAR`: *(3, 5)* – Number of parameters (material conductivities) to be optimized
- `NOBS`: *(32, 92)* – Number of observation points
- `NPARGP`: *1* – Number of parameter groups.
- `NPRIOR`: *0* – Number of articles of prior information.
- `NOBSGP`: *1* – Number of observation groups.
- `NTPLFLE`: *1* – Number of template files.
- `NINSFLE`: *1* – Number of instruction files.
- `PRECIS`: *single* – Single precision output
- `DPOINT`: *point* – Include the decimal point in output
- `NUMCOM`: *1* – Number of command lines used to run the model.
- `JACFILE`: *1* – A *1* denotes we supply the jacobian information. When using `PEST` with the full model only, `JACFILE` is set to *0* and a difference jacobian must be used.
- `MESSFILE`: *0* – Tells `PEST` it need not right a message file to the model.

D.2.2 Levenberg-Marquardt Settings

- `RLAMBDA1`: *10.0* – Initial Marquardt λ value. The default range is *1.0 - 10.0*. High values for `RLAMBDA1` force the algorithm to approximate steepest descent, necessary for very poor initial iterates.
- `RLAMFAC`: *2.0* – The factor by which lambda is adjusted. *2.0* is a recommended value and works well for our examples.

- `PHIRATSUF`: *0.3* – `PEST` uses the variable Φ for the error in the model versus data. `PHIRATSUF` is the value such that $\Phi_i/\Phi_{i-1} \leq$ `PHIRATSUF` will terminate that Marquardt iteration and select a new value for λ. Low values for `PHIRATSUF` will force more iterations but perhaps waste function calls in search of a minimum that is unobtainable. High values will cause the iteration to cease before a reachable minimum is found. The manual suggests a value of *0.3*, which we found has worked well in our examples.

- `PHIREDLAM`: *0.003* – The ratio of Φ for successive λ values. This allows `PEST` to evaluate the current error based on the error for the previous λ value. The manual suggests a value of *0.01*, however we found this value too large and iterations would terminate before finding an available minimum value.

- `NUMLAM`: *10* – The upper limit on number of λ test during an optimization iteration. Values of *5-10* are suggested and we chose a higher value to more thoroughly test the problem space.

- `RELPARMAX`: *2* – The maximum relative change any parameter is allowed to undergo between optimization iterations. This value is not used in our examples as we use the next parameter (only one can be used for each variable).

- `FACPARMAX`: *2* – The maximum factor change that a parameter is allowed to undergo. So, a parameter can only change by 2 times its original value. Values lower than *5* are recommended for nonlinear cases.

- `FACORIG`: *.001* – If the parameter falls below a factor of this value times the original value, `FACORIG` is used as the denominator when determining the relative change in parameter. As we use a factored comparison, this variable is not used.

- `PHIREDSWH`: *0.1* – `PEST` will switch to a centered difference differentiation scheme when the relative reduction in Φ drops below this value. We provide a full jacobian and do not use numerical differentiation, so this variable is not used.

- `NOPTMAX`: *30* – Maximum number of optimization iterations. This is the upper bound on the suggested range. Our function evaluations are very inexpensive with the POD reduced order model, so we allow `PEST` more iterations to find a minimum value.

- `PHIREDSTP`: *0.01* – This variable and `NPHISTP` work together. This is the ratio of the current residual Φ and the lowest Φ achieved during the optimization process. The recommended value is *0.005*, but we found this to be too low as many of our parameters lie near the defined bounds.
- `NPHISTP`: *7* – The number of iterations since the lowest optimization has been achieved. If there have been `NPHISTP` iterations for which $(\Phi_i - \Phi_{min})/\Phi_i \leq$ `PHIREDSTP`, the process will terminate. A value of *4* is recommended, but we needed more optimizations to avoid settling on an extremely local minimum.
- `NPHINORED`: *7* – The number of iterations since `PEST` has reduced Φ. While *3-4* is recommended, we need more iterations to avoid a local minimum.
- `RELPARSTP`: *0.01* – The maximum relative parameter change between iterations. This works in conjunction with the next parameter. The default value of *0.01* is used.
- `NRELPAR`: *3* – Number of iterations to gauge maximum relative parameter change. The default value of *3* is used.

D.2.3 Output Settings

- `ICOV`: *1* – Output the covariance matrix for the parameter values.
- `ICOR`: *1* – Output a correlation coefficient matrix for the parameter values.
- `IEIG`: *1* – Output the eigenvectors and eigenvalues of the covariance matrix.

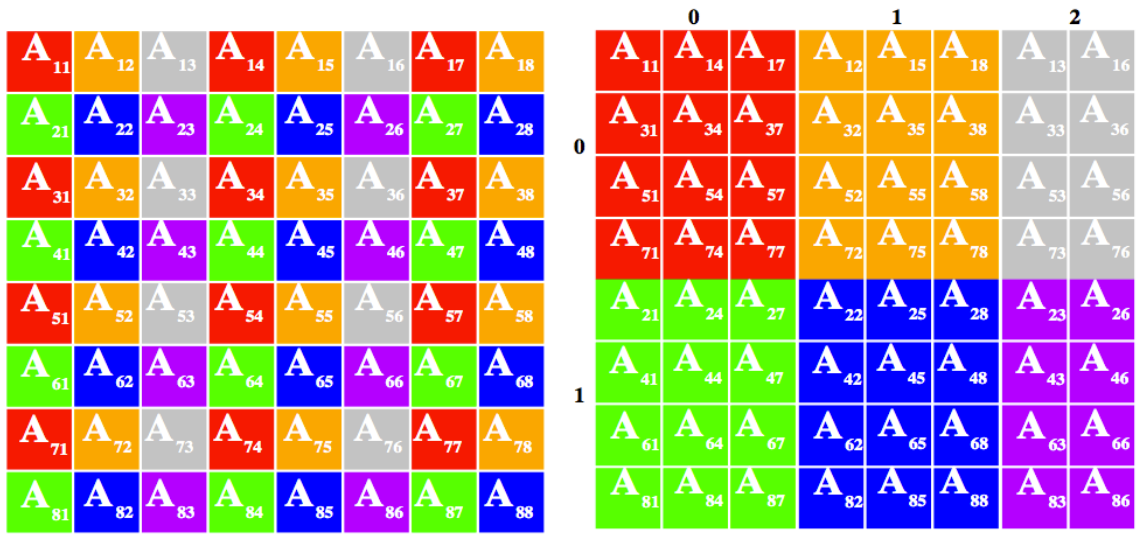
Appendix E

SCALAPACK

To perform any dense-matrix computations, including the SVD of the basis (5.2), we use the "Scalable Linear Algebra Package" `ScaLAPACK` [8, 9]. `ScaLAPACK` is essentially `LAPACK` implemented in parallel. To use `ScaLAPACK` , we had to manage two distinct partitioning schemes. `ADH` partitions the mesh by geospatial properties and attempts to keep nodes near each other in the mesh on the same processor [75]. `ScaLAPACK` , as discussed in the following sections, partitions the information in an effort to distribute work equally among the processors.

E.1 Mesh Partition

`ScaLAPACK` uses a block cyclic distribution of a matrix. The user defines a maximum block size (in nodes) per processor and `ScaLAPACK` partitions the matrix as shown in Figure E.1. To utilize the SVD tools in `ScaLAPACK` , the blocks must be square. To construct this partitioning, `ADH` must transmit a non-trivial amount of data between processors as the `ADH` and `ScaLAPACK` partitioning of the matrix do not overlap. Fortunately, this communication only need happen once per full-model run. On exit, each processor writes their own suite of data to a unique text file. The POD reduced model then reads each processor's data individually and does not do any re-distribution of information inside the reduced model. This effectively parallelizes the I/O stream and removes a key bottleneck in the optimization process.



Global (left) and distributed (right) views of matrix

Figure E.1: Partition of ScaLAPACK Matrix

The following table describes how `ScaLAPACK` partitions an array or vector on each processor. This nine-entry integer array is referenced by each subroutine in `ScaLAPACK` [8]. With this information, our POD code can read in the data from `ADH` without any repartitioning or reallocation of the matrix. We note `LOCr( K )` denotes the number of elements of `K` that a process would receive if `K` were distributed over the `p` processes of its process column.

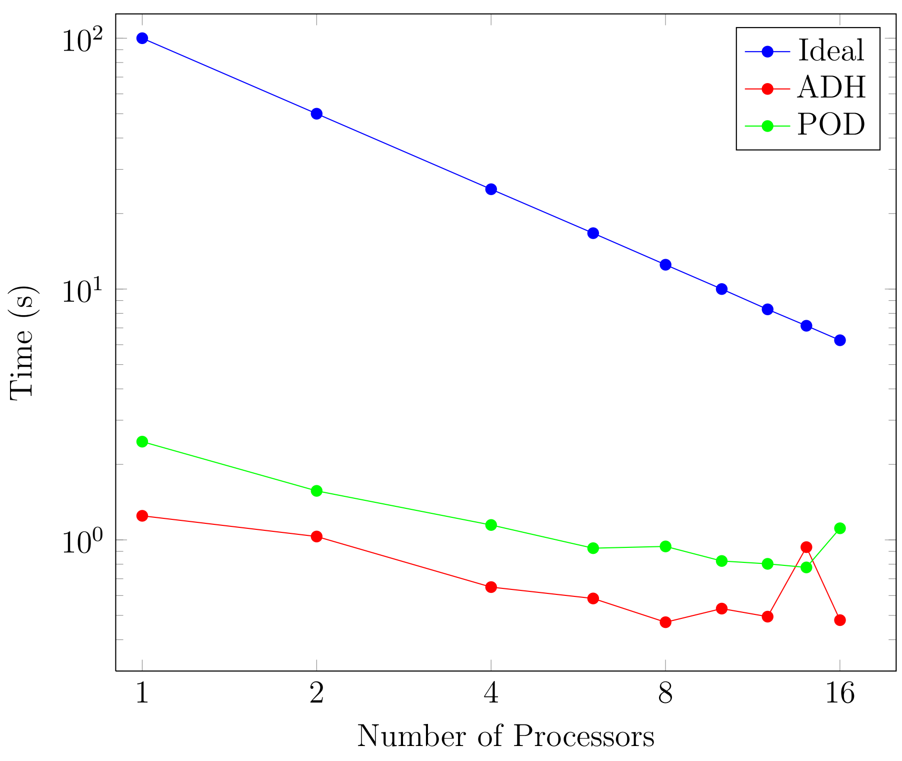
Table E.1: Description of an array or vector in ScaLAPACK

DTYPE_A(global)	The descriptor type. In this case, DTYPE_A = 1.
CTXT_A (global)	The BLACS context handle, indicating the BLACS process grid A is distributed over. The context itself is global, but the handle (the integer value) may vary.
M_A (global)	The number of rows in the global array A.
N_A (global)	The number of columns in the global array A.
MB_A (global)	The blocking factor used to distribute the rows of the array.
NB_A (global)	The blocking factor used to distribute the columns of the array.
RSRC_A (global)	The process row over which the first row of the array A is distributed.
CSRC_A (global)	The process column over which the first column of the array A is distributed.
LLD_A (local)	The leading dimension of the local array. LLD_A $\geq$ MAX(1,LOCr(M_A)).

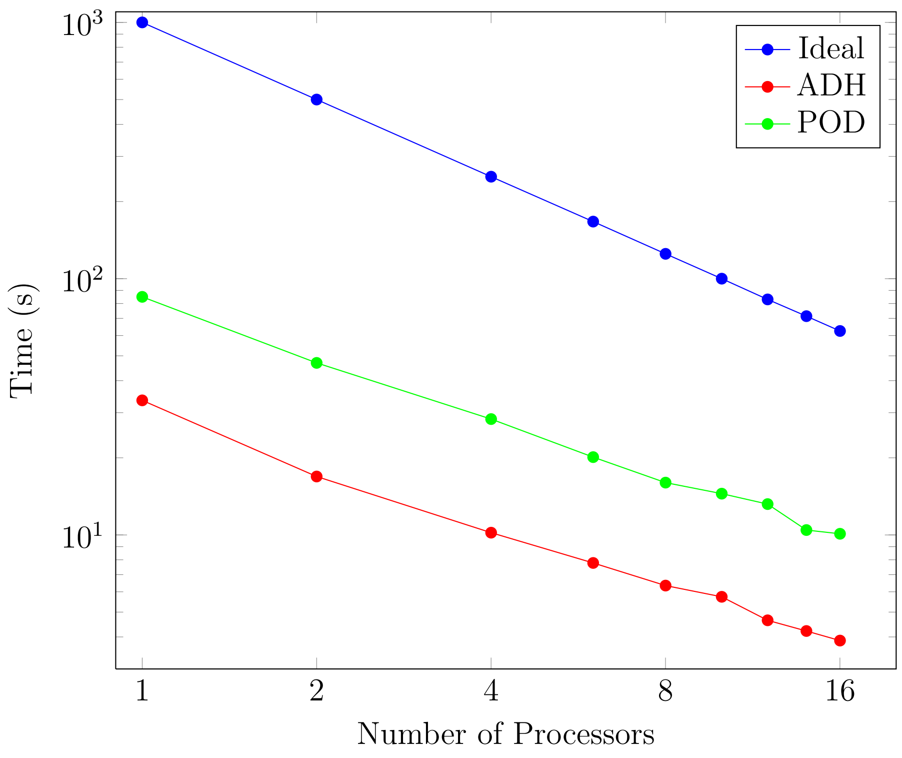
E.2 Speedup Analysis

We perform a brief analysis of how the time-to-solution for both the full and reduced model are affected by increased processor counts. We should expect that as the processor count doubles the time is cut in half. In the plots below, we see that both the full model (ADH) and the reduced POD model achieve excellent speedup for both domains as the processor count grows. It should be noted that the timings for the reduced POD model include the I/O step as well as the solve step. The I/O must only be done once for each full-model solve, while the reduced order solve is done tens or hundreds of times. While the POD run may appear to take more time than an individual ADH run, that is not the case. In Figure E.2, the “Ideal” curve demonstrates the slope of an ideal speedup curve. If the code were perfectly optimal, doubling the number of processors would halve the solution time. Unfortunately, communication between processors limits the ability of any code to achieve this standard. For the synthetic column, the problem is too small to truly exploit full parallelism. Therefore, the speedup curves for both `ADH` and POD are less steeply sloped than the "Ideal" curve. For the larger tank domain, however, the codes scale very well and are similar in slope to the "Ideal" curve.

Figure E.2: Speed Up Analysis



(a) Speed Up for Synthetic Column



(b) Speed Up for Tank

We note that the speed up is much closer to ideal for the larger domain. Sixteen processors are far too many for the trivial column. The processors do not have significant amounts of data and therefore excessive communication is required. For the tank, however, we see the speed up curves are nearly linear as we add processors.

From this analysis, we run the column with four (4) processors and the tank with sixteen (16). We did not perform a speed up analysis for SPE10 but decided to run with thirty-two (32) processors for all runs.