A framework for geophysical inversions

with application to vadose zone parameter estimation

by

Archa Rowan B. Cockett

B.Sc. Applied and Environmental Geology, University of Calgary, 2010

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Geophysics)

The University of British Columbia
(Vancouver)

December 2017

Abstract

Inverse modeling is a powerful tool for extracting information about the subsurface from geophysical and hydrologic data. Geophysical inverse problems are inherently multidisciplinary, requiring elements from the relevant physics, numerical simulation, and optimization, as well as knowledge of the geologic setting, hydrologic processes, and a comprehension of the interplay between all of these elements. Increasingly geoscientists are tackling complex problems that require integration of multiple types of information in order to better characterize the subsurface. However, many of the subfields of geophysics are developing simulation and inversion approaches, algorithms, and supporting software in isolation. This isolation is a barrier to quantitative integration and leads to inefficiencies in advancing interdisciplinary research. Greater efficiencies, and higher quality outcomes, could be achieved if (hydro)geophysicists had a common framework to accelerate an integrated approach. The main goal of my thesis is to organize the components of (hydro)geophysical simulations and inverse problems, and synthesize these into a comprehensive, modular framework.

The development of a geophysical framework requires considering a number of disciplines and geophysical problems (e.g. electromagnetics and potential fields) to ensure generality as well as extensibility. However, the goal is also to have the framework work outside of geophysics and most notably in hydrogeology; vadose zone fluid flow

is used as a model problem. Fluid flow in the vadose zone is governed by the Richards equation; it is parameterized by hydraulic conductivity, which is a nonlinear function of pressure head. The computational scalability of the Richards equation inversion is a significant challenge for three dimensional inversions in hydrogeophysics. Existing work explicitly calculates the sensitivity matrix using finite difference or automatic differentiation, however, for large-scale problems these methods are constrained by computation and memory. This dissertation provides an implicit sensitivity algorithm that enables large-scale inversion problems for distributed parameters in the Richards equation to become tractable on modest computational resources.

Lay Summary

Geophysical methods gather data remotely to enable insights into subsurface structure and processes (e.g. locating economic resources or monitoring environmental changes). The information derived from geophysical methods is of crucial importance in resource exploration, environmental remediation, and the study of deep-earth processes. Interpretation of geophysical data requires a combination of numerical simulation and inversion. Inversion is a procedure for using data to estimate an image or model of the earth (this is similar to medical imaging). Increasingly, geoscientists are tackling complex problems that require integration of multiple types of information in order to better characterize the subsurface. In hydrogeology and geophysics, this quantitative integration requires advances in both disciplines, as well as a framework for this collaboration. The objective of this dissertation is to identify and refine a computational framework that enables and encourages sustained cross-disciplinary communication, which is a necessary step in integrated geophysical simulation research.

Preface

The research for this dissertation was completed while studying at the University of British Columbia. This research has resulted in three peer reviewed publications, three expanded conference abstracts, and several auxiliary works. The main focus of my thesis is on a framework for geophysical simulations and inversions that increases quantitative geoscience communication. In 2016, Dr. Oldenburg, Dr. Pidlisecky, Lindsey Heagy and I organized an international conference around this work that was sponsored by the Banff International Research Station; excerpts from the introduction of my thesis were used in the conference proposal.

Chapter 2 presents a framework for simulation and parameter estimation for geophysical applications. An earlier version of which was published in Cockett et al. (2015c), and ideas from this chapter also have been presented at several international conferences (cf. Cockett et al. (2014b, 2015b,a)).

Chapter 3 presents a computationally scalable algorithm for solving inverse problems for hydraulic parameters in vadose zone flow using the Richards equation. This work has been submitted for peer review and the preprint is available on *arXiv* (Cockett et al., 2017); preliminary versions of this research were presented at two conferences (Cockett and Haber, 2013b,a).

Chapter 4 involves several numerical examples, which were inspired by work from

my undergraduate thesis, of which two papers were published during the course of my graduate research (Pidlisecky et al., 2013; Cockett and Pidlisecky, 2014). The forward simulation framework for multi-parameter simulations and inversions in time-domain physical problems used in this chapter was derived from collaborative work between electromagnetics and vadose zone flow (Heagy et al., 2016). One of the numerical examples in Chapter 4 has previously been published in (Cockett et al., 2017).

Two of the appendices contain supporting materials on finite volume and several numerical examples and case studies. Appendix B on finite volume contains work and figures that have been published in a computational tutorial on finite volume (Cockett et al., 2016a). Additionally, much of this work is supported by course material and instruction from Dr. Eldad Haber, Dr. Uri Ascher, and Dr. Chen Grief (Haber, 2015; Ascher and Greif, 2011). Appendix C presents an adaptation of the forward simulation framework published in Heagy et al. (2016) for the Richards equation. This appendix also summarizes conclusions and insights from three extended conference abstracts on electromagnetics and a publication on parametric geologic modelling (Heagy et al., 2014; Kang et al., 2015a; Heagy et al., 2015c; Cockett et al., 2016b).

Throughout the course of my graduate research, I have started and contributed to several open source software projects to support, test, and validate the geophysical simulation and inversion framework that is the main focus of my thesis. My main focus with this software was on inheritance, composition, terminology, and the interfaces between simulation and inversion components – the elements that define the framework. This is demonstrated by my personal contribution of 267,614 lines of code over the last five years, which have been reduced over 4.5 fold to 59,111 lines of code while increasing possibilities and geophysical applications. For an up-to-date, detailed analysis on code contribution and attribution over time, please see:

https://www.openhub.net/p/simpeg-geophysics. This is perhaps the most salient distinguishment between the focus on framework development as opposed to a script or executable that is aimed at a specific type of geophysical inversion. The major software packages that have been created are: (a) `SimPEG`, a framework for simulation and parameter estimation in geophysics (https://github.com/simpeg/simpeg); (b) `discretize`, a finite volume package for simulation in the context of inverse problems (https://github.com/simpeg/discretize); and (c) `pymatsolver`, a common interface to several matrix solvers and packages (https://github.com/rowanc1/pymatsolver). These projects have seen significant investment from my colleagues in testing, applying, and expanding the capabilities of the framework to other geophysical applications. This open, collaborative work has involved colleagues across industry, government, and six universities. Currently SIMPEG includes methods for: vadose zone flow (Cockett et al., 2017); direct current resistivity and induced polarization (Kang and Oldenburg, 2016); time-domain and frequency-domain electromagnetics (Heagy et al., 2016, 2017); magnetotellurics (Rosenkjaer et al., 2016); magnetics and gravity (Miller et al., 2017); and several examples of other inverse problems (Kang et al., 2017b). All software has been released under the permissive MIT license, to encourage reuse, adaptation, and sustained contribution to these ideas.

Table of Contents

List of Tables

List of Figures

List of Programs

Acknowledgments

The interdisciplinary work that I have started at the University of British Columbia would not of been possible without the enthusiastic support from a tremendous and diverse group of individuals. First and foremost, Dr. Eldad Haber, who is a trailblazer in the computational geosciences and whose guidance and expertise has not only been invaluable, but is foundational to almost every paragraph of this thesis. Thank you for all your investment in me. Next, Dr. Doug Oldenburg, whose infectious enthusiasm and curiosity is able to both motivate and mobilize a team of passionate students around him that are able to tackle grand goals that we wholeheartedly believe will change our field of study. Next, Dr. Adam Pidlisecky, who introduced me to scientific programming and hydrogeophysics and who is able to weave scientific narratives that inspire and broker innovation. Next, my parents for, among so many other things, perspective in this process. Next, my committee and examiners who improved the quality and focus of this dissertation. Thank you all for your unwavering support and encouragement.

The funding for this interdisciplinary work was provided from several sources. A special thanks to the University of British Columbia, the Natural Sciences and Engineering Research Council of Canada, the Vanier Graduate Canada Scholarship program, the Gilles Brassard Doctoral Prize for Interdisciplinary Research, the Killam

Scholarship program, the University of British Columbia Library, and the Banff International Research Station for their generous support.

A special thanks to my colleagues, friends and family who are always willing to talk through ideas and have supported me throughout this process. To name a few of these wonderful people: Dom Fournier, Gudni Rosenkjaer, Michael Mitchell, Thibaut Astic, Brendan Smithyman, Dave Marchant, Julie Nutini, Michael Wathen, Franklin Koch, Teddi Herring, Michael Firmin, Lars Ruthotto, Jenn Fohring, Luz Angelica Caudillo-Mata, Klara Steklova, Pieter Aukes, Kristyn Adams, Leonardo Uieda, Matt Hall, and many others. Also, a special thanks to the current and future users and contributors of SIMPEG, and to those who are pioneering connected and open geoscience communities.

Finally, this dissertation would not have been possible without the continued emotional and educational support and encouragement from Lindsey Heagy and Seogi Kang. You have been partners in thought and have been integral in nurturing many of the ideas that fill the following pages.

To giants and their shoulders.

If we then ask ourselves where that intelligence is embodied, we are forced to concede that it is elusively distributed throughout a hierarchy of functional processes – a hierarchy whose foundation extends down into natural processes below the depth of our comprehension. If there is any one thing upon which this 'intelligence depends' it would seem to be organization.

— Douglas C. Engelbart (1962)
Augmenting human intellect: a conceptual framework

Chapter 1

Introduction

1.1 Research context

One of the goals of the applied geosciences is to gain an understanding of subsurface structures and processes. These understandings are often used to make predictions and decisions associated with commercial and environmental challenges, including contaminant delineation, resource exploration, reservoir optimization, and watershed characterization. The accuracy of these predictions can have far-reaching economic and environmental implications. There are many disciplines and skills that are involved in providing predictions, and increasingly these disciplines must collaborate and integrate their domain-specific knowledge. In a managed aquifer recharge project, for example, the goal is to infiltrate water into the subsurface for storage and subsequent recovery. Throughout the lifetime of the project, monitoring and management of the infiltration site is necessary (e.g. Racz et al. (2011); Daily et al. (1992); Park (1998)). Such projects require input from geology, hydrology, and geophysics in order to map the hydrostratigraphy, collect and interpret time-lapse geophysical measurements, and

integrate all results to make predictions and decisions about fluid movement at the site. The quantitative integration of the geosciences is far from trivial as each discipline has differing descriptive terminology, as well as software tools that are domain specific with limited interoperability.

In the following two subsections, I will independently introduce two disciplines within applied geoscience: (a) geophysical inversions, and (b) hydrogeophysics in the vadose zone. The current state of these disciplines gives context to the work that follows and motivates research into a computational framework that improves the quantitative communication between methodologies and researchers. The subsequent section expands on these ideas and identifies a significant computational challenge of hydrogeophysics inversions in the vadose zone.

1.1.1 Geophysical inverse problems

Geophysical methods involve making measurements at or above the earth's surface, or in boreholes. The data acquired with these methods are then used to create models of the subsurface; this is similar to non-invasive medical imaging, but the spatial and temporal scales are typically much larger. The models, which can be 1D, 2D, or 3D distributions of various physical properties, are used for monitoring and extracting information about fluid flow and subsurface structures. The physical properties are linked to the data through various partial differential equations. The task of generating a quantitative understanding of the data requires the ability to carry out forward simulations of these equations and, in many situations, inverting the data to estimate a static or time-lapse model of the subsurface. Forward simulations use the physics of the underlying measurement approach to simulate the response of a given distribution of physical properties. Inversion is a mathematical, algorithmic, and occasionally

heuristic process that constructs a model consistent with the field measurements and *a priori* geologic, geophysical, and hydrologic information.

Many of these geophysical methods (e.g. electromagnetics, magnetotellurics, gravity, direct current resistivity) have mature solutions for both simulation and inversion in three dimensions and through time (Oldenburg, 2016). There is, of course, continued research into improving computational efficiency for large-scale geophysical surveys (cf. Haber and Schwarzbach (2014); Yang et al. (2014); Haber and Heldmann (2007)). In parallel to this effort, there is ongoing work to *integrate* these geophysical methodologies to create more informed interpretations of the subsurface from multiple data types and surveys (e.g. Devriese et al. (2017); Kang et al. (2017a); Fournier et al. (2017)). This research trend is true in exploration geophysics as well as in cross-disciplinary fields such as hydrogeophysics where hydrologic simulations and geophysical simulations can be combined to better inform predictions about groundwater flow.

The development of new methodologies to address the evolving challenges in quantitative geoscience integration will build upon and extend standard practices. These extensions and integrations will require experimentation with, and recombination of, existing techniques. This presupposes that researchers have access to consistent, well-tested tools that can be extended, adapted, and combined. One of the main goals of my thesis is to organize the components of geophysical simulations and inverse problems into a comprehensive, modular framework in order to support this *combinatorial* experimentation and exploration.

1.1.2 Hydrogeophysics in the vadose zone

The majority of groundwater recharge is derived through water that percolates through the vadose zone, the region between the earth's surface and the fully saturated zone. As such, studying the processes that occur in the vadose zone is of critical importance for understanding our groundwater resources. Much attention has been given to monitoring, describing and predicting processes that occur in this region of the earth. Traditionally, monitoring has been conducted by taking point-measurements of saturation or pressure, or laboratory measurements of soil hydraulic properties. More recently, geophysical methods are being used in conjunction with hydrologic data to create more informed models of and predictions about the subsurface (Linde and Doetsch, 2016). The advantages of employing geophysics to hydrogeology problems are numerous; geophysical methods allow data to be gathered remotely, and the data can then be used to create an image of a distributed physical property of interest (e.g. electrical conductivity) in the subsurface. However, the geophysical problem is inherently non-unique and when viewed independently, images produced by a geophysical inversion often lack the detail necessary to make informed hydrogeologic predictions. Some level of prediction can be offered by hydrogeologic simulations within the structural geologic context; however, these simulations are difficult to verify due to lack of constraining hydrologic data. Taken separately, each methodology involved in this monitoring challenge yields distinct interpretations and predictions that are often dissonant or actually conflicting.

Fluid flow in the vadose zone is described by the Richards equation and is parameterized by hydraulic conductivity, which is a nonlinear function of pressure head. Hydraulic conductivity defines how fluids move in the subsurface, and is an impor-

tant physical property to estimate for accurate predictions (Pollock and Cirpka, 2012; Šimunek et al., 2012). It is not possible to directly image hydraulic conductivity with geophysical data, however, geophysical electromagnetic methods are sensitive to bulk electrical conductivity, which changes significantly depending on the saturating fluid (e.g. gas or water) (Archie, 1942; Liang et al., 2012; Mendelson and Cohen, 1982). Changes in saturation over time, as fluids move, can be related to changes in electrical properties, and can be observed by electromagnetic geophysical methods. Knowing where and how the fluids move can subsequently be related to hydraulic conductivity (or other hydraulic properties). This technique has been used to estimate hydraulic properties directly from geophysical data. For example, in Binley et al. (2002), a cross-well tomography experiment was conducted using radar and direct current resistivity methods. The movement of a vadose zone tracer was tracked and a single parameter was estimated using the Richards equation, through trial and error, for homogeneous hydraulic conductivity. Both, the quality, and the spatial and temporal density of geophysical data available for monitoring vadose zone processes will continue to proliferate (e.g. Pidlisecky et al. (2013)). The increased data density and quality opens up the possibility to estimate many more distributed hydraulic parameters.

Time-lapse estimation problem presents a significant conceptual and computational challenge (Pollock and Cirpka, 2012; Haber and Gazit, 2013; Towara et al., 2015; Linde and Doetsch, 2016). It requires large-scale, time-lapse hydrogeologic simulations that must be efficiently solved and then integrated with geophysical methods. This multiphysics integration of geophysical and hydrologic simulation can be completed in a variety of ways. For example, this integration can be through direct coupling of the simulations or through qualitative observations and uncoupled workflows. Hinnell et al. (2010) presents uncoupled integrations as: (a) using the geophys-

ical data to estimate a physical property, such as electrical conductivity; (b) using an empirical relation, such as Archie's equation (eq. 4.1), to transform the geophysical estimate into a hydrological parameter, such as water content; and (c) using hydrological estimates to help inform or test a hydrogeologic simulation. A coupled inversion formulates the entire process as a single forward model and uses stochastic or deterministic parameter estimation to directly update the hydrogeologic and empirical parameters (Finsterle and Kowalsky, 2008; Ferré et al., 2009). Increasingly, there are instances of these sorts of collaborations and studies in near surface hydrogeophysics (cf. Linde and Doetsch (2016) and references within). The integration of geophysical and hydrologic data increases the scale of simulations and inversions that must be considered – hundreds of thousands to millions of hydrologic parameters must be estimated. Currently, this is not computationally feasible for large 3D inversions of vadose zone parameters using the Richards equation. For example, the relatively few parameters that can be estimated by stochastic inversions may not be sufficient for 3D inversions (Linde and Doetsch, 2016). Alternatively, deterministic inversions can be used, but will need to draw on improvements across the field of geophysical inverse problems. For example, regularization techniques developed in other areas of exploration geophysics (e.g. Paasche and Tronicke (2007); Sun et al. (2012)), have potential to be helpful in introducing known parameter distributions into a vadose zone inversion. In Hinnell et al. (2010), the authors conclude that, "the coupled approach [for hydrogeophysics] requires that the hydrologic and geophysical models be merged, [which] forces the hydrologist and the geophysicist to formulate a consistent framework." This consistent framework was identified as "*the primary limit* to the routine implementation of coupled inversion[s]. The formulation of common solution grids, time steps, and simulation accuracies requires an uncommon level of collaboration

during scientific analysis."

1.1.3 Research motivation

The quantitative integration of hydrology, geophysics, and geology remains an open problem (Liu and Gupta, 2007; Ferré et al., 2009; Pollock and Cirpka, 2012; Knight et al., 2013; Linde and Doetsch, 2016). This task is being worked on from many different perspectives in various research communities, and much progress has been made in case studies, new algorithms, and novel integrations. The complexity of this integration "intertwines various disciplines/subjects including geophysics, hydrology, petrophysics, geostatistics, [and] inverse theory" (Knight et al., 2013). Although each subdiscipline (e.g. flow modelling, electromagnetic simulation) invokes many of the same concepts and numerical pieces for solving simulation and inversion problems, the approaches developed and applied are not easily shared between subdisciplines. This is due to differing terminology, organization of methodologies, differing data densities and sensitivities, model conceptualizations, as well as software implementations. For example, in geophysics a *model* is often taken to be a volumetric distribution of physical properties (e.g. Oldenburg and Li (2005)); in geology a *model* is often more qualitative, represented by a sketch, description, 3D surfaces, or a cross section that opaquely embeds knowledge about geologic processes (e.g. Harder et al. (2009); Porwal and Kreuzer (2010)); in hydrology a *model* often refers to the representation of a physical simulation or empirical equation, frequently containing simplifying assumptions of homogeneity or dimensionality (cf. Devi et al. (2015)). As another example, in hydrogeology data is often collected as high precision point measurements with low spatial and/or temporal coverage; in electromagnetic geophysics, however, physical property recoveries are often less precise and are averaged over a larger spatial scale.

The inclusion of relevant information from one subdiscipline into another is difficult due to these differences in terminology, knowledge representation (e.g. quantitative or qualitative), knowledge mapping (e.g. through empirical or structural relations), model conceptualization (e.g. volumetric or surfaces or parametric), data sensitivities (e.g. point or bulk measurements), and simplifying or implicit assumptions (e.g. one dimensional or homogeneous). These disconnects are exceptionally apparent in software implementations, even though software is precisely where quantitative integration must occur! Software is often developed *ad hoc* for specific outcomes, and the algorithmic components, which are conceptually generic and could be shared with others, are deeply embedded and not easily transferred to other applications. Within a given subdiscipline this can create challenges, as the system under consideration can potentially embed hard-coded, tacit assumptions. Furthermore, this lack of transportability and interoperability severely hinders the advancement of novel geophysical *applications* since geoscientists in different subgroups often find themselves having to develop a complete software solution from scratch prior to investigating their scientific questions of interest. Overcoming these bottlenecks, and establishing a simulation and inversion framework that works across many subdisciplines of (hydro)geophysics, is the overarching goal of this thesis.

Based on the current state of the geoscience inversion and hydrogeophysics community and the observations outlined above, I have arranged this thesis to address two research topics:

1. the development of an extensible framework for geophysical inversions, and
2. formulation of the three dimensional Richards equation inversion for computational scalability.

The overarching goal is to promote both quantitative integration and collaboration between geoscience disciplines and communities. Interdisciplinary integration requires dissemination, reproducibility, accessibility, and collaboration; as such these are crucial to my work and demonstrated throughout the following thesis.

1.2 A framework for geophysical inversions

Geophysical inversions are the mathematical process of creating subsurface models to fit measured data and geophysical simulations. The language, workflows, and resulting software implementations of geoscience inversions vary across disciplines. These inconsistencies are among the large barriers to sustained cross-disciplinary integrations. One research approach to addressing these interdisciplinary barriers is the development of a framework that organizes, synthesizes and abstracts diverse methodologies. A framework should (a) serve as a means of organizing an approach to simulation and inverse problems, (b) facilitate quantitative communication between researchers and geophysics methodologies, and (c) act as a blueprint for both ideation and software implementations. The disciplines and methodologies that have been used to inform the research of this framework include: vadose zone flow using the Richards equation, direct current resistivity, time and frequency domain electromagnetics, magnetotellurics, potential fields including gravity and magnetics, and using geologic parameterizations to inform model conceptualization. Oldenburg (2016) noted that many of the geophysical inversion techniques can now be completed in three dimensions using computationally efficient inversion algorithms. This is significant as geological structures and processes such as electromagnetics and fluid flow often require treatment in three dimensions. In both geophysics and hydrogeology the data is a field or a flux, sampled at various locations, times, or frequencies. Additionally, point samples

of physical properties or (hydro)stratigraphy can be inferred or tested from borehole cores and geologically interpolated between wells and surface observations. These can be included into the inverse problem formulation either implicitly through weightings and reference models (cf. Williams (2008)) or more explicitly by forcings of geologic priors (cf. Linde et al. (2015)). The geologic observations can also be modelled, for example, using radial basis functions (RBFs), to implicitly reproduce the geologic contacts and drillhole data; this results in geologically interpreted interpolations dividing the subsurface into lithological units (Hillier et al., 2014).

Each geophysical technique is sensitive to different physical properties and/or different spatial scales. The differing sensitivities of these techniques motivates combination of methodologies to better understand and image the subsurface and time-lapse processes. This is an active field of study, for example, (a) investigating cooperative electromagnetics inversions in realistic settings by externally combining existing tools through custom workflows (McMillan and Oldenburg, 2014), (b) joint inversion and model fusion algorithms for direct current resistivity and borehole tomography (Haber and Gazit, 2013), (c) integrating multiple types of airborne geophysical data into a consistent geologic model for mineral exploration (Kang et al., 2017a; Fournier et al., 2017; Devriese et al., 2017), and (d) combining one dimensional vadose zone flow and direct current resistivity to invert for hydrological parameters (Hinnell et al., 2010). Many of these studies rely on *externally integrating* existing software tools through purpose-built scripts and workflows; limiting the transferability to other disciplines. However, recent work has seen an increased focus by the geophysical community on a framework approach that targets multiple geophysical and hydrogeologic methodologies (e.g. JInv (Ruthotto et al., 2016) and PyGIMLi (Rücker et al., 2017)). In many electromagnetic geophysical applications, for example, a common model for

electrical conductivity can be produced through cycling a common model through the relevant problems until a sufficient misfit is achieved. In hydrogeophysics, however, the model from a geophysical simulation is the data for a hydrogeologic simulation. As such, for a deterministic large-scale inversion the sensitivity from one problem is empirically coupled to another problem and must be efficiently calculated. In hydrogeophysics, coupled hydrologic and geophysical interpretations are moving into three dimensions, and standard probabilistic and finite difference techniques are becoming "computationally infeasible" (Linde and Doetsch, 2016). The coupling of these methods into a computationally efficient inversion requires attention to the scalability of all individual approaches as well as exposing the geophysical inversions effectively for hydrogeologic parameter estimation. In order to support the custom parameterizations, couplings, and integrations that are necessary for a new application, a general framework must provide combinatorial building blocks that are independently accessible and extensible, while maintaining computational efficiencies. The PEST framework for model independent parameter estimation and uncertainty analysis is a concrete example of where parts of this have been done with success (Doherty, 2004). The software is widely cited in academia ($>$ 2K citations) especially in hydrology and hydrogeophysics, and is heavily used in industry. The advantage of being model independent has given this technique wide application due to the flexibility to adapt to *new scientific questions*. However, this also comes at quite a cost because the structure of the simulation and modelling cannot be used to the advantage of the algorithm. As with vadose zone flow or electromagnetics, when moving to three dimensions there may be hundreds of thousands to millions of parameters to estimate. Not taking the structure of the problem into account severely limits types and size of problems that can be considered. In the context of geophysical simulations and inversions there are

two significant challenges/opportunities for such a framework:

1. to organize the *components* of geophysical simulations and inversions to and expose explicit *interfaces* to components to interdisciplinary manipulation in a *combinatorial* manner; and

2. maintaining computational scalability, especially with respect to efficient calculation of sensitivities.

Adapting interdisciplinary methodologies to formalize geophysical simulations and inversions inherently requires that a diverse suite of methods and applications be considered across geophysics, hydrogeology, and geology. This process will take the form of deriving, from the existing body of literature, a consistent conceptual and computational framework, which supports reproducible inversion workflows. By formalize, I do not mean mathematically, rather taking practices of ontology and computational framework development in biology and other more mature interdisciplinary fields and applying them to geophysical inversions. The ontology literature provides context, albeit abstract, for the approach that I have used to formalize the research around this interdisciplinary problem and is briefly detailed in Appendix A.

1.2.1 Take home points

Sustained, reproducible integration of geophysical simulations and inversions requires that methodologies be accessible, consistent, numerically documented, interoperable, and extensible. This can be enabled by a comprehensive framework that is validated and rigorously tested against reality and leading edge research. To do this, research is required to:

- identify the composable components of geophysical inversions and simulations, as well as the interfaces between the components;
- abstract commonalities between methodologies to a consistent, supporting subset; and
- capture geoscience inversion heuristics in a reproducible manner.

The output of this research will be a computational framework that is numerically tested and demonstrates the capability to support existing and future research directions. Ideally this framework will catalyze and accelerate interdisciplinary collaborations.

1.3 Application to vadose zone parameter estimation

The development of a geophysical framework requires considering a number of disciplines and geophysical problems to ensure generality as well as extensibility. I am working with collaborators in many of these geophysical methods (electromagnetics, direct current resistivity, magnetotellurics, magnetics, gravity) and am ensuring that the framework that I am researching supports these applications. However, the goal is also to have the framework work outside of geophysics and most notably in hydrogeology, as such, I have chosen vadose zone fluid flow as a model problem.

Fluid flow in the vadose zone is described by the Richards equation (eq 3.1) and parameterized by hydraulic conductivity, which is a nonlinear function of pressure head (Richards, 1931; Celia et al., 1990). Investigations in the vadose zone typically require identification of distributed hydraulic properties. This is increasingly being done through an inversion approach, which is also known as data assimilation, model calibration, or history matching (Liu and Gupta, 2007). These methods use changes

in water content or pressure head to infer hydraulic parameters (Binley et al., 2002; Deiana et al., 2007; Hinnell et al., 2010). Hydrogeophysics allows many more proxy measurements, such as direct current resistivity data, to be taken to help characterize these sites spatially, as well as through time. As such, the number of distributed hydraulic parameters to be estimated in a Richards equation inversion will continue to grow. Conceptually this integration is framed as taking the output of a (time-lapse) direct current resistivity inversion (cf. Pidlisecky et al. (2013)), and using this estimate of electrical conductivity as a proxy for water content data in hydraulic parameter estimation. The proxy data can be directly incorporated through an empirical relation (e.g. Archie (1942)) or time-lapse estimations can be structurally incorporated through some sort of regularization technique (Haber and Gazit, 2013; Haber and Oldenburg, 1997; Hinnell et al., 2010). Previous studies have either estimated homogeneous soil profiles estimating less than five parameters (e.g. Binley et al. (2002); Deiana et al. (2007)) or heterogeneous soil profiles, estimating less than thousands of parameters (cf. Irving and Singha (2010); Jardani et al. (2013); Orgogozo et al. (2014)). Parameter estimation is currently completed by trial and error or using stochastic techniques, neither of which scale to the scenario that requires estimation of hundreds of thousands to millions of parameters (Linde and Doetsch, 2016). This limit in scalability, especially in the context of hydrogeophysics has been explicitly noted in the literature (cf. Binley et al. (2002); Deiana et al. (2007); Towara et al. (2015)). To our knowledge, a large scale inversion in three dimensions for distributed hydraulic parameters using the Richards equation has not yet been completed in the literature due to these issues with computational scalability.

There has been much research into the scalability of the inverse problem in geophysical applications (e.g. electromagnetics) that allow the calculation of an optimiza-

tion step in the inversion *without* explicitly calculating or storing the sensitivity matrix (cf. Haber et al. (2000)). This is extremely important in large problems as the computer memory available to store this large, dense matrix can often be a limitation. For example, although there have been significant advances for massively parallel forward simulations of the Richards equation (cf. Orgogozo et al. (2014)), the computational "memory may simply not be large enough" to run the inverse problem using standard automatic differentiation (Towara et al., 2015). Previous work uses either automatic differentiation or finite difference in order to explicitly compute the sensitivity matrix (e.g. PEST) (Finsterle and Zhang, 2011; Bitterlich and Knabner, 2002; Towara et al., 2015). Finite difference is computationally slower and can generate inaccuracies in the sensitivity computation and tarry convergence of the optimization algorithm. With regard to implicit sensitivity calculations, there is an opportunity to apply some of the learnings from the geophysical inversion literature to this hydrogeologic problem. Note that the implicit sensitivity calculation is necessary in *any* gradient based technique as well as modern stochastic methods (Bui-Thanh and Ghattas, 2015).

The application of the implicit sensitivity calculation to the Richards equation, however, is not straightforward. Hydraulic conductivity is the *function* we are inverting for - it is empirically determined and depends on pressure head; pressure head is the field that is calculated using the Richards equation. This nonlinear coupling requires iterative optimization methods in the forward simulation between each time step (e.g. Picard or Newton). This makes the implicit calculation of the effect of the sensitivity on a vector rather involved and intricate. Furthermore, the nonlinear relationship of hydraulic conductivity is empirically determined and depending on the relation used could involve the estimation of up to ten spatially-distributed parameters from a finite dataset. The implicit use of the sensitivity matrix should have the ability

to calculate the sensitivity to any of these model parameters; however, any inversion algorithm must be tested as to the limits of estimating all distributed parameters at once. One goal of the work in this thesis is to tackle the sensitivity calculation *implicitly*. This would further allow for exploration into different inversion methodologies and parameterizations of the empirical relationships. By not storing the sensitivity, and instead computing its effect on a vector, the size of the problem that we can invert will become much larger. This will allow large 3D hydraulic parameter inversions using the Richards equation to be run on modest computational resources. However, directly jumping into a 3D inversion for heterogeneous hydraulic properties, with many parameters per cell in the isotropic case, is highly non-unique. I will explore some inversion schemes, model conceptualizations, and ways to explore interfacing to *a priori* information. Unsaturated hydraulic conductivity as well as the water retention curve are both empirically described functions. The parameterization and estimation of these functions in the context of collecting proxy saturation data will be explored in a number of numerical experiments.

1.3.1 Take home points

With advances in the spatial and temporal density of geophysical data collection, time-lapse water content estimates can be made with increasing accuracy. These proxy time-lapse measurements can be used to estimate hydraulic properties from non-invasive geophysical methods. This is increasingly being completed in field studies, however, conceptual and computational simplifications are consistently made. Part of the bottleneck is the scalability of current inverse methods applied to the Richards equation. Research is required to:

- reframe the Richards equation inversion for computational scalability when mov-

ing to large 3D distributed parameter estimation,

- ensure that any parameter in the empirically determined hydraulic conductivity function and water retention curve can be estimated, and

- investigate and explore the effectiveness of distributed joint inversions for hydraulic parameters from a water content dataset.

This research will inform the conceptual framework and contribute an implicit sensitivity calculation for the Richards equation inverse problem that can be coupled to other geophysical methods.

1.4 Thesis outline

The content of this thesis is divided into three chapters and three appendices; these are shown visually in Figure 1.1. Each chapter and appendix provides a stand alone introduction and conclusion to the specific topic under consideration. Chapter 2 presents a comprehensive framework for geophysical simulations and inversions. This includes an overview of current research and outlines a modular, object-oriented approach for structuring deterministic, large-scale inversion methodology in geophysics that has application to hydrogeology and can incorporate and interface to geologic information. A direct current resistivity forward simulation and inversion are used throughout this chapter as an example. A brief comment on the approach used to research this computational framework is included in Appendix A. An overview of finite volume discretization techniques, which are heavily used throughout this thesis, has been included as Appendix B. In this appendix, I examine and comment on the organization, structure, and formulation of three different classes of mesh: (a) tensor product orthogonal mesh, including formulation in cylindrical coordinates; (b) quadtree and

octree meshes; and (c) logically rectangular, non-orthogonal meshes. These meshes in 1D, 2D, 3D, and 4D are used throughout the thesis as well as extensions to my work. The software used to inform my work and refine my interdisciplinary approach to simulation and parameter estimation in geophysics is open source and available under the permissive MIT license (https://github.com/simpeg/simpeg). This repository includes forward and inverse software by me and my colleagues of the framework for vadose zone flow, time and frequency domain electromagnetics, direct current resistivity, induced polarization, magnetotellurics, magnetics, gravity, and a number of example linear inverse problems; these are described in the online documentation (http://docs.simpeg.xyz).

Chapter 3 focuses on the Richards equation, which is the partial differential equation that describes vadose zone flow. Using the previously developed framework and finite volume tools tailored specifically for inverse problems (Appendix B), I have reframed the Richards equation to be scalable with respect to large-scale inverse problems. The majority of this work is focused on enabling access to the sensitivity implicitly, through multiplication with a vector. The validity of this technique as well as comments on numerical performance are provided. The scalability of the algorithm developed is shown with comparison to other techniques. This work has built upon as well as informed the research into the organization of the framework. The Richards equation is more complex than many other geophysical methods analyzed because of the nonlinear, time dependent forward problem and multiple empirical relationships that may or may not require estimation.

Chapter 4 explores the estimation of hydraulic parameters from water content data. This is motivated by the hydrogeophysical problem where there is an availability of proxy water content data. This chapter explores a set of empirical relations that in-

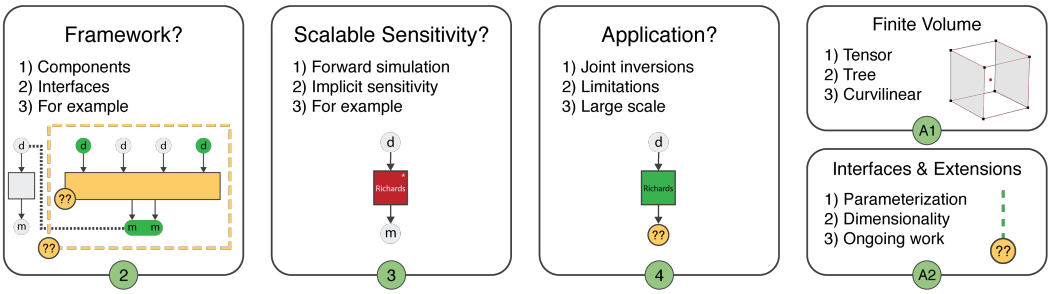

Figure 1.1: Outline of the thesis chapters: (2) the simulation and inversion framework; (3) the sensitivity calculation in Richards equation; (4) applications and exploration into vadose zone inversions; (A1) finite volume techniques on a variety of meshes; and (A2) interfaces to geologic knowledge through parameterizations and a forward simulation framework and extensions to the work presented.

form both the hydraulic conductivity function and the water retention curve. A joint inversion is formulated to estimate for five spatially distributed hydraulic parameters. The number of spatially distributed unknowns are experimented with, and the response of the inversion algorithm is tested under various levels of *a priori* knowledge. The efficacy of these approaches is commented upon, which may help inform laboratory or field based experiments of this kind. Finally, a three dimensional inversion is completed using the Richards equation. Due to computational scalability issues detailed in Chapter 3, an inversion for distributed hydraulic parameters at this scale is computationally infeasible with standard techniques (Linde and Doetsch, 2016). These limitations are overcome by both the framework introduced in Chapter 2 and the implicit sensitivity calculation that allows large-scale parameter estimation problems to be tackled (Chapter 3). Other examples, extensions, and applications of the framework including other geophysical methodologies, case studies, and geoscience integrations are included in Appendix C; much of this work is collaborative in nature and focuses

on the parameterization of the forward problem. As the focus of this thesis is on the geophysical inversion framework, I have distilled my observations from these case studies and provided these learnings in a general form.

Finally, Chapter 5 provides a brief discussion on the thesis contributions and summarizes some opportunities for future research and collaborations. These research areas may seem disparate, but collectively they are united by a common theme of addressing the complexities of bringing together the disciplines of geophysics, hydrology, geology, and inverse theory in a computationally tractable manner that is accessible and extensible by other researchers.

Chapter 2

Simulation and inversion framework

2.1 Introduction and motivation

Geophysical surveys can be used to obtain information about the subsurface, as the measured responses depend on the physical properties and contrasts in the earth. Inversions provide a mathematical framework for constructing physical property models consistent with the data collected by these surveys. The data collected are finite in number, while the physical property distribution of the earth is continuous. Thus, inverting for a physical property model from geophysical data is an ill-posed problem because no unique solution explains the data. Furthermore, the data may be contaminated with noise. Therefore, the goal of a deterministic inversion is not only to find a model consistent with the data, but also to find the 'best' model that is consistent with the data[1]. The definition of 'best' requires the incorporation of assumptions and *a priori* information, often in the form of an understanding of the particular geologic

[1]Alternatively, the inverse problem can be formulated in a probabilistic framework, for example: (Tarantola, 2005; Tarantola and Valette, 1982). In this thesis, we will focus our attention on the deterministic approach.

setting or structures (Constable et al., 1987; Oldenburg and Li, 2005; Lelièvre et al., 2009). Solving the inverse problem involves many moving pieces that must work together, including physical simulations, optimization, linear algebra, and incorporation of geology. Deterministic geophysical inversions have been extensively studied and many components and methodologies have become standard practice. With increases in computational power and instrumentation quality, there is a greater drive to extract more information from geophysical data. Additionally, geophysical surveys are being applied in progressively more challenging environments. As a result, the geosciences are moving towards the integration of geological, geophysical, and hydrological information to better characterize the subsurface (e.g. Haber and Oldenburg (1997); Doetsch et al. (2010); Gao et al. (2012)). This integration is a scientifically and practically challenging task (Li and Oldenburg, 2000a; Lelièvre et al., 2009). These challenges, compounded with inconsistencies between different data sets, often make the integration and implementation complicated and/or non-reproducible. The development of new methodologies to address these challenges will build upon, as well as augment, standard practices; this presupposes that researchers have access to consistent, well-tested tools that can be extended, adapted, and combined.

There are many proprietary codes available that focus on efficient algorithms and are optimized for a specific geophysical application (e.g. Kelbert et al. (2014); Li and Key (2007); Li and Oldenburg (1996, 1998)). These packages are effective for their intended application; for example, in a domain-specific, large-scale geophysical inversion or a tailored industry workflow. However, many of these packages are 'black-box' algorithms; that is, they cannot easily be interrogated or extended. As researchers, we require the ability to interrogate and extend ideas; this must be afforded by the tools that we use. Accessibility and extensibility are the primary motivators for this

work. Other disciplines have approached the development of these tools through open source initiatives, using interpreted languages such as Python (for example, Astropy in astronomy (Astropy Collaboration et al., 2013) and SciPy in numerical computing (Jones et al., 2001)). Interpreted languages facilitate interactive development using scripting, visualization, testing, and interoperability with code in compiled languages and existing libraries. Furthermore, many open source initiatives have led to communities with hundreds of researchers contributing and collaborating by using social coding platforms, such as GitHub (https://github.com). Initiatives also exist in the geophysical forward and inverse modeling community, which target specific geophysical applications (cf. Hansen et al. (2013); Hewett et al. (2013); Uieda et al. (2014); Kelbert et al. (2014); Harbaugh (2005)). Recent work has seen an increased focus by the geophysical community on a framework approach that targets multiple geophysical/hydrogeologic methods (e.g. JInv (Ruthotto et al., 2016) and PyGIMLi (Rücker et al., 2017)). We are interested in creating a community around geophysical simulations and gradient-based inversions. To create a foundation on which to build a community, we require a comprehensive framework that is applicable across domains and upon which researchers can readily develop their own tools and methodologies. To support these goals, this framework must be modular and its implementation must be easily extensible by researchers.

2.1.1 Attribution and dissemination

The goal of this chapter is to present a comprehensive framework for simulation and gradient-based parameter estimation in geophysics. Core ideas from a variety of geophysical inverse problems have been distilled to create this framework. We also provide an open source library, written in Python, called SIMPEG (Simulation and Pa-

rameter Estimation in Geophysics, http://github.com/simpeg/simpeg). Our implementation has core dependencies on SciPy, NumPy, and Matplotlib, which are standard scientific computing packages in Python (Jones et al., 2001; Van Rossum and Drake Jr, 1995; Oliphant, 2007; Hunter, 2007). SIMPEG includes staggered grid, mimetic finite volume discretizations on structured and semi-structured meshes. It interfaces to standard numerical solver packages, convex optimization algorithms, model parameterizations, and visualization routines. We use Python's object-oriented paradigm to create modular code that is extensible through inheritance and subtype polymorphism. SIMPEG follows a fully open source development paradigm (Feller and Fitzgerald, 2000) and uses the permissive MIT license. Throughout its development, we have focused on modularity, usability, documentation, and extensive unit-testing (Wilson et al., 2014). Please see the website (http://simpeg.xyz) for up-to-date code, examples, and documentation of this package. In addition, there are many published use cases across a variety of geophysical applications (Kang et al., 2014, 2015b,a; Kang and Oldenburg, 2015; Heagy et al., 2014, 2015d). We hope that the organization, modularity, minimal dependencies, documentation, and testing in SIMPEG will encourage reproducible research, cooperation, and communication to help tackle some of the inherently multidisciplinary geophysical problems.

To guide the discussion, we start this chapter by outlining gradient-based inversion methodology in Section 2.2. The inversion methodology directly motivates the construction of the SIMPEG framework, terminology, and software implementation, which we discuss in Section 2.3. We weave an example of Direct Current (DC) resistivity throughout the discussion of the SIMPEG framework to provide context for the choices made and highlight many of the features of SIMPEG.

2.2 Inversion methodology

Geophysical inverse problems are motivated by the desire to extract information about the earth from measured data. A typical geophysical datum can be written as:

$$F_i[\mathbf{m}] + \varepsilon_i = d_i, \tag{2.1}$$

where F is a forward simulation operator that incorporates details of the relevant physical equations, sources, and survey design, $\mathbf{m}$ is a generic symbol for the inversion model, ε_i is the noise that is often assumed to have known statistics, and d_i is the observed datum. In a typical geophysical survey, we are provided with the data, $d_i, i = 1...N$, and some estimate of their uncertainties. The goal is to recover the model, $\mathbf{m}$, which is often a physical property. The data provide only a finite number of inaccurate constraints upon the sought model. Finding a model from the data alone is an ill-posed problem since no unique model exists that explains the data. Additional information must be included using prior information and assumptions (for example, downhole property logs, structural orientation information, or known interfaces (Fullagar et al., 2008; Li and Oldenburg, 2000a; Lelièvre et al., 2009)). This prior knowledge is crucial if we are to obtain an appropriate representation of the earth and will be discussed in more detail in Section 2.2.1.

Defining and solving a well-posed inverse problem is a complex task that requires many interacting components. It helps to view this task as a workflow in which various elements are explicitly identified and integrated. Figure 2.1 outlines the inversion methodology that consists of inputs, implementation, and evaluation. The inputs are composed of: the geophysical data; the equations, which are a mathematical description of the governing physics; and, prior knowledge or assumptions about the setting.

The implementation consists of two broad categories: the forward simulation and the inversion. The forward simulation is the means by which we solve the governing equations, given a model, and the inversion components evaluate and update this model. We are considering a gradient-based approach, which updates the model through an optimization routine. The output of this implementation is a model, which, prior to interpretation, must be evaluated. This requires considering, and often re-assessing, the choices and assumptions made in both the input and the implementation stages. In this chapter, our primary concern is the implementation component; that is, how the forward simulation and inversion are carried out numerically. As a prelude to discussing how the SIMPEG software is implemented, we step through the elements in Figure 2.1, considering a Tikhonov-style inversion.

2.2.1 Inputs

Three sources of input are required prior to performing an inversion: (1) the geophysical data and uncertainty estimates; (2) the governing equations that connect the sought model with the observations; and (3) prior knowledge about the model and the context of the problem.

Data and uncertainty estimates

At the heart of the inversion are the geophysical data that consist of measurements over the earth. These data depend on the type of survey, the physical property distribution, and the type and location of the measurements. The details about the survey must also be known, such as the location, orientation, and waveform of a source and which component of a particular wavefield is measured at a receiver. The data are contaminated with additive noise, which can sometimes be estimated by taking mul-

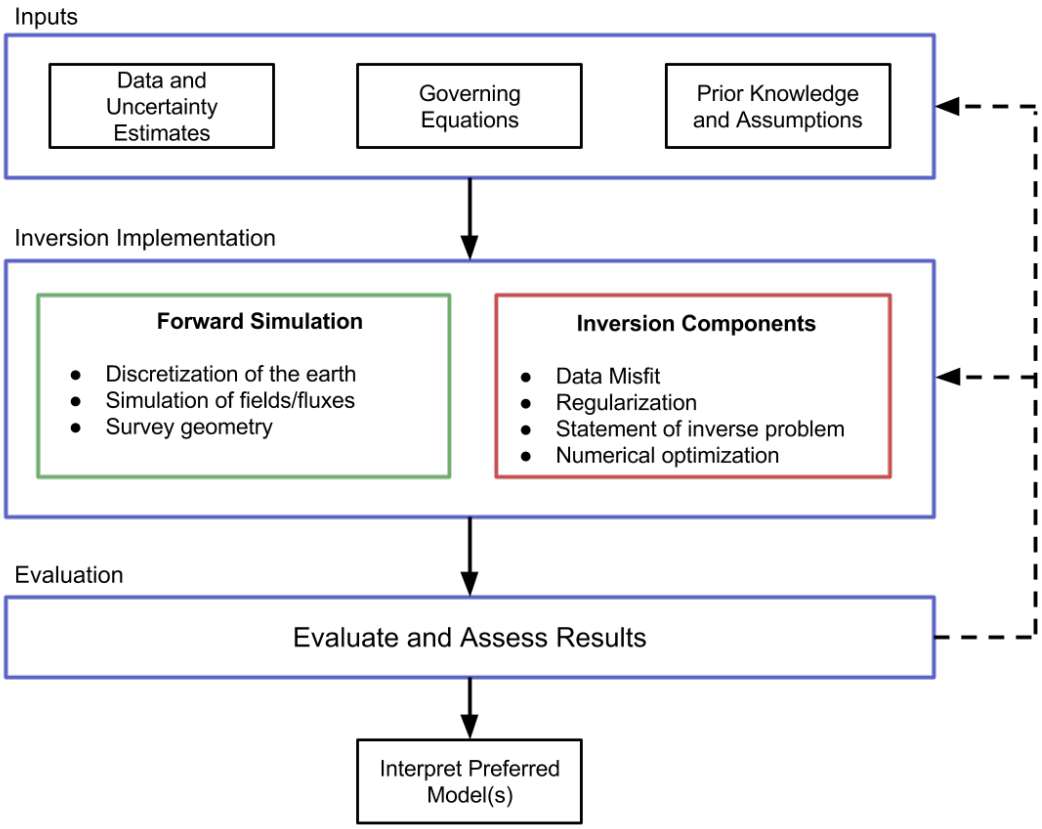

Figure 2.1: Inversion methodology. Including inputs, implementation, evaluation and interpretation.

tiple realizations of the data. However, standard deviations of those realizations only provide a lower bound for the noise. For the inverse problem, the uncertainty in the data must include not only this additive noise, but also any discrepancy between the true earth experiment and our mathematical representation of the data. Including these aspects requires accounting for mis-location of receivers and sources, poor control of the transmitter waveform, electronic gains or filtering applied to signals entering the receivers, incorrect dimensionality in our mathematical model (e.g. working in 2D instead of 3D), neglect of physics in our mathematical equations by introducing assumptions (e.g. using a straight ray tomography vs. a full waveform simulation in

seismic), and discretization errors of our mathematical equations.

Governing equations

The governing equations provide the connection between the physical properties of the subsurface and the data we observe. Most frequently, these are sets of partial differential equations with specific boundary conditions. The governing equations, with specified source terms, can be solved through numerical discretization using finite volume, finite element, or integral equation techniques. Alternatively, they may also be solved through evaluations of analytic functions. Whichever approach is taken, it is crucial that there exists some way to simulate the data response given a model.

Prior knowledge

If one model acceptably fits the data, then infinitely many such models exist. Additional information is therefore required to reduce non-uniqueness. This additional information can be geologic information, petrophysical knowledge about the various rock types, borehole logs, additional geophysical data sets, or inversion results. This prior information can be used to construct reference models for the inversion and also characterize features of the model, such as whether it is best described by a smooth function or if it is discontinuous across interfaces. Physical property measurements can be used to assign upper and lower bounds for a physical property model at points in a volume or in various regions within our 3D volume. The various types of information relevant to the geologic and geophysical questions that we must address must be combined and translated into useful information for the inversion (Lelièvre et al., 2009; Li et al., 2010).

2.2.2 Implementation

In this section, we outline the components necessary to formulate a well-posed inverse problem and solve it numerically. Two major abilities are critical to running the inversion: (1) the ability to simulate data, and (2) the ability to assess and update the model (Figure 2.1).

Forward simulation

The ability to carry out an inversion presupposes the ability to run a forward simulation and create predicted data, given a physical property model. In forward simulation, we wish to compute $F[\mathbf{m}] = \mathbf{d}_{\text{pred}}$. The operator, F, simulates the specific measurements taken in a geophysical survey, using the governing equations. The survey refers to all details regarding the field experiment that we need to simulate the data. The forward simulation of DC resistivity data requires knowledge of the topography, the resistivity of the earth, and the survey details, including locations of the current and potential electrodes, the source waveform, the units of the observations, and the polarity of data (since interchanging negative and positive electrodes may sometimes occur in the field). To complete the simulation, we need to solve our governing equations using the physical property model, $\mathbf{m}$, that is provided. In the DC resistivity experiment, our partial differential equation, with supplied boundary conditions, is solved with an appropriate numerical method; for example, finite volumes, finite elements, integral equations, or semi-analytic methods for 1D problems. In any case, we must discretize the earth onto an appropriate numerical forward simulation mesh, ($mesh_F$). The size of the cells will depend upon the structure of the physical property model, topography, and the distance between sources and receivers. Cells in $mesh_F$ must be small enough, and the domain large enough, to achieve sufficient numerical accuracy. Proper mesh

design is crucial so that numerical modeling errors are below a prescribed threshold value (cf. Haber (2015)).

In general, we can write our governing equations in the form of:

$$C(\mathbf{m}, \mathbf{u}) = 0, \tag{2.2}$$

where $\mathbf{m}$ is the modeled physical property and $\mathbf{u}$ are the fields and/or fluxes. C is often given by a partial differential equation or a set of partial differential equations. Information about the sources and appropriate boundary conditions are included in C. This system is solved for $\mathbf{u}$ and the predicted data are extracted from $\mathbf{u}$ via a projection (or functional), $\mathbf{d}_{\text{pred}} = P[\mathbf{u}]$. The ability to simulate the geophysical problem and generate predicted data is a crucial building block. Accuracy and efficiency are essential, since many forward problems must be evaluated when carrying out any inversion.

Inversion elements

In the inverse problem, we must first specify how we parameterize the earth model. Finding a distributed physical property can be done by discretizing the 3D earth into voxels, each of which has a constant, but unknown, value. It is convenient to refer to the domain on which this model is discretized as the inversion mesh, $mesh_I$. The choice of discretization involves an assessment of the expected dimensionality of the earth model. If the physical property varies only with depth, then the cells in $mesh_I$ can be layers and a 1D inverse problem can be formulated. A more complex earth may require 2D or 3D discretizations. The choice of discretization depends on both the spatial distribution and resolution of the data and the expected complexity of the geologic setting. We note that the inversion mesh has different design criteria and con-

straints than the forward simulation mesh. For convenience, many inverse problems have historically been solved with $mesh_I = mesh_F$ so that only one discretization is needed for the inversion. There is a growing body of work that investigates combinations of inversion discretizations and forward modeling meshes that are geared towards problem-specific formulations as well as efficiency in large-scale problems (Haber and Schwarzbach, 2014; Yang et al., 2014; Haber and Heldmann, 2007). In any formulation, there exists a mapping between $mesh_I$ and $mesh_F$ such that the parameterization chosen can be used to simulate data in a forward simulation.

To formulate a mathematical statement of the inverse problem, there are two essential elements:

1. *data misfit*: a metric that measures the misfit between the observed and predicted data; and

2. *regularization*: a metric that is constructed to evaluate the model's agreement with assumptions and prior knowledge.

The data misfit requires an assessment of the error in each datum. These errors result from anything that yields a discrepancy between the mathematical modeling and the true value. It includes additive noise, errors in the description of survey parameters (e.g. receiver locations, transmitter waveforms, etc.), incorrect choice of governing equations, and numerical errors arising from the simulation technique. Clearly, quantifying the noise for each datum poses a challenge.

The data misfit is a measure of how well the data predicted by a given model reproduce the observed data. To assess *goodness of fit*, we select a norm that evaluates the 'size' of the misfit. This metric must include an uncertainty estimate for each

datum. Often, we assume that the data errors are Gaussian and uncorrelated and then estimate the standard deviation for each datum. The most common norm, and one that is compatible with Gaussian statistics, has the form:

$$\phi_d(\mathbf{m}) = \frac{1}{2}\|\mathbf{W}_\mathrm{d}(F[\mathbf{m}] - \mathbf{d}_\mathrm{obs})\|_2^2. \tag{2.3}$$

Here, $F[\mathbf{m}]$ is a forward modeling that produces predicted data, $\mathbf{d}_\mathrm{pred}$, as in equation: 2.1. $\mathbf{W}_\mathrm{d}$ is a diagonal matrix whose elements are equal to $\mathbf{W}_{d_{ii}} = 1/\varepsilon_i$, where ε_i is an estimated standard deviation of the i^th datum. It is important to think carefully when assigning these estimates. A good option is to assign a $\varepsilon_i = floor + \%|d_i|$. Percentages are generally required when there is a large dynamic range of the data. A percentage alone can cause great difficulty for the inversion if a particular datum acquires a value close to zero; therefore, we include a floor.

In addition to a metric that evaluates the size of the misfit, we also require a tolerance, ϕ_d^*. We consider that models satisfying $\phi_d(\mathbf{m}) \leq \phi_d^*$ adequately fit the data (Parker, 1994). If the data errors are Gaussian and we have assigned the correct standard deviations, then the expected value of $\phi_d^* \approx N$, where N is the number of data. Finding a model that has a misfit substantially lower than this will result in a solution that has excessive and erroneous structure; that is, we are fitting the noise. Finding a model that has a misfit substantially larger than this will yield a model that is missing structure that could have been extracted from the data (see Oldenburg and Li (2005) for a tutorial).

The choice of misfit in equation 2.3 is not the only possibility for a misfit measure. If data errors are correlated, then $\mathbf{W}_\mathrm{d}$ is the square root of the data covariance matrix and it will have off-diagonal terms. Often useful in practice is recognizing if the noise

statistics are non-Gaussian. Incorporating robust statistical measures, like l_p norms with $p \approx 1$, are useful for handling outliers (Ekblom, 1973; Farquharson, 1998).

The second essential inversion element is defining the regularization functional. If there is one model that has a misfit equal to the desired tolerance, then there are infinitely many other models which can fit to the same degree. The challenge is to find the model that has both the desired characteristics and is compatible with *a priori* information. A single model can be selected from an infinite ensemble by measuring the length, or norm, of each model. Then the smallest, or sometimes largest, member can be isolated. Our goal is to design a norm that embodies our prior knowledge and, when minimized, yields a realistic candidate for the solution of our problem. The norm can penalize variation from a reference model, spatial derivatives of the model, or some combination of these. We denote this norm by ϕ_m and write it in a matrix form, for example,

$$\phi_m(\mathbf{m}) = \frac{1}{2}\|\mathbf{W}_\mathrm{m}(\mathbf{m} - \mathbf{m}_\mathrm{ref})\|_2^2. \tag{2.4}$$

$\mathbf{W_m}$ is a matrix and $\mathbf{m}_\mathrm{ref}$ is a reference model (which could be zero). The matrix $\mathbf{W_m}$ can be a stacked combination of matrices weighted by α_*:

$$\mathbf{W}_\mathrm{m} = [\alpha_\mathbf{s}\mathbf{I}, \quad \alpha_\mathbf{x}\mathbf{W}_\mathbf{x}^\top, \quad \alpha_\mathbf{y}\mathbf{W}_\mathbf{y}^\top, \quad \alpha_\mathbf{z}\mathbf{W}_\mathbf{z}^\top]^\top. \tag{2.5}$$

Here, $\mathbf{W}_\mathrm{m}$ is a combination of smallness and first-order smoothness in the x, y, and z directions. Each of the $\mathbf{W}$ matrices is, in fact, a discrete representation of an integral

(cf. Oldenburg and Li (2005)).

$$\begin{aligned}
&\int_\Omega \left(\mathbf{w_s}(\mathbf{m}-\mathbf{m}_{ref})\right)^2 dV && \text{(smallness)},\\
&\int_\Omega \left(\mathbf{w_x}\frac{d\mathbf{m}}{dx}\right)^2 dV && \text{(x-smoothness)},\\
&\int_\Omega \left(\mathbf{w_y}\frac{d\mathbf{m}}{dy}\right)^2 dV && \text{(y-smoothness)},\\
&\int_\Omega \left(\mathbf{w_z}\frac{d\mathbf{m}}{dz}\right)^2 dV && \text{(z-smoothness)},
\end{aligned} \tag{2.6}$$

The final regularization, $\mathbf{W}_\mathrm{m}$, can be a weighted sum of these, with α_* being applied as scalars or diagonal matrices, with varying weights for each cell or cell face (cf. Oldenburg and Li (2005); Haber (2015)). Additional weightings can also be incorporated through $\mathbf{W}_\mathrm{m}$, such as depth weighting, which is important in potential field inversions (such as magnetics and gravity), or sensitivity weightings to prevent model structure from concentrating close to sources or receivers (Li and Oldenburg, 1996, 2000b). The regularization functionals addressed provide constraints on the model in a weak form; that is, a single number is used to characterize the entire model. Strong constraints that work within each cell can often be provided in the form of upper and lower bounds; these bounds will be incorporated in the following section. The l_2 norms referred to above are appropriate for many problems, however models norms, such as l_p-norms, total variation, minimum support stabilizing functionals, or rotated smoothness operators that favor different character and / or include additional information can also be designed (cf. Oldenburg (1984); Oldenburg and Li (2005); Claerbout and Muir (1973); Strong and Chan (2003); Zhdanov (2002); Li and Oldenburg (2000a)). For example:

$$\int |\mathbf{w}_p(\mathbf{m}-\mathbf{m}_{ref})|^p dV \quad \text{(weighted-}l_p\text{ norm)} \tag{2.7}$$

The potential to have different norms tailored to a specific problem, with the additional functionality of localized weightings and reference models, provides the user with the capability of designing a regularization that favors a solution that is compatible with existing knowledge about the model. This task is not trivial, requires careful thought, and must often be re-evaluated and adjusted as the geophysicist iterates over the inversion procedure (Figure 2.1).

Statement of the inverse problem

The purpose of this section is to pose our inverse problem in a mathematically precise way and to provide a methodology by which a solution can be achieved. In the work that follows, we outline a specific methodology that we will later demonstrate. We formulate the inverse problem as a problem in optimization, where we define an objective function, based on the data misfit and model regularization, and aim to find a model which sufficiently minimizes it. Many variants of this formulation are possible.

At this stage of the workflow, we have on hand all of the necessary components for formulating the inverse problem as an optimization problem. We have the capability to forward model and generate predicted data, assess the data misfit using ϕ_d, and a tolerance on the data misfit has already been specified. A regularization functional, ϕ_m, and additional strong constraints on the model have been identified, such as upper and lower bounds: $\mathbf{m}_i^L \leq \mathbf{m}_i \leq \mathbf{m}_i^H$. The sought model is one that minimizes ϕ_m and also reduces the data misfit to some tolerance, ϕ_d^*. However, a reduction in data misfit requires that the model have increased structure, which typically is at odds with the assumptions we impose in the construction of ϕ_m, meaning that the ϕ_d and ϕ_m are antagonistic. To address this and still pose the inversion as an optimization problem,

we design a composite objective function:

$$\phi(\mathbf{m}) = \phi_d(\mathbf{m}) + \beta\phi_m(\mathbf{m}), \tag{2.8}$$

where β is a positive constant. It is often referred to as the trade-off parameter, regression parameter, regularization parameter, or Tikhonov parameter (Tikhonov and Arsenin, 1977). When β is very large, the minimization of $\phi(\mathbf{m})$ produces a model that minimizes the regularization term and yields a large $\phi_d(\mathbf{m})$. Alternatively, when β is very small, minimization of $\phi(\mathbf{m})$ produces a model that fits the data very well but is contaminated with excessive structure so that $\phi_m(\mathbf{m})$ is large. The inverse problem is posed as:

$$\begin{aligned} &\underset{\mathbf{m}}{\text{minimize}} \quad \phi(\mathbf{m}) = \phi_d(\mathbf{m}) + \beta\phi_m(\mathbf{m}) \\ &\text{s.t.} \quad \phi_\text{d} \leq \phi_\text{d}^*, \quad \mathbf{m}_\text{i}^\text{L} \leq \mathbf{m}_\text{i} \leq \mathbf{m}_\text{i}^\text{H}. \end{aligned} \tag{2.9}$$

Since the value of β is not known *a priori*, the above optimization problem can be solved at many values of β to produce a trade-off, or Tikhonov, curve (cf. Parker (1994); Hansen (1998)). An optimum value, β^*, can be found so that minimizing equation 2.8 with β^* produces a model with misfit ϕ_d^*. The value of β^* can be found via cooling techniques where the β is progressively reduced from some high value and the process stopped when the tolerance is reached or by using two-stage methods (cf. Parker (1977)). There are other strategies for selecting the trade-off parameter including the L-curve technique (Hansen, 1992), which attempts to find the point of greatest curvature in the Tikhonov curve and Generalized Cross Validation (Wahba, 1990; Golub et al., 1979; Haber and Oldenburg, 2000; Oldenburg and Li, 2005; Farquharson and Oldenburg, 2004).

The optimization posed in equation 2.9 is almost always non-linear. It is linear only in a special case, where the forward mapping is a linear functional of the model, ϕ_m and ϕ_d are written as l_2 norms, β is known, and there are no imposed bound constraints. This rarely happens in practice, requiring that iterative optimization methods be employed to find a solution. Gradient-based methods are commonly used and we refer the reader to Nocedal and Wright (1999) for background and introductions to the relevant literature. For geophysical problems, Gauss-Newton techniques have proven to be valuable and practical. For l_2 norms, we write the objective function as:

$$\phi(\mathbf{m}) = \frac{1}{2}||\mathbf{W}_\mathrm{d}(F[\mathbf{m}] - \mathbf{d}_\mathrm{obs})||_2^2 + \frac{1}{2}\beta||\mathbf{W}_\mathrm{m}(\mathbf{m} - \mathbf{m}_\mathrm{ref})||_2^2. \tag{2.10}$$

The gradient is given by:

$$g(\mathbf{m}) = J[\mathbf{m}]^\top \mathbf{W}_\mathrm{d}^\top \mathbf{W}_\mathrm{d}(F[\mathbf{m}] - \mathbf{d}_\mathrm{obs}) + \beta \mathbf{W}_\mathrm{m}^\top \mathbf{W}_\mathrm{m}(\mathbf{m} - \mathbf{m}_\mathrm{ref}), \tag{2.11}$$

where $J[\mathbf{m}]$ is the sensitivity or Jacobian. The components, $J[\mathbf{m}]_{ij}$, specify how the i^th datum changes with respect to the j^th model parameter; these changes will be discussed in more detail in the next section. At the k^th iteration, beginning with a model, $\mathbf{m}^k$, we search for a perturbation, $\delta\mathbf{m}$, which reduces the objective function. Linearizing the forward simulation by:

$$F[\mathbf{m}^k + \delta\mathbf{m}] \approx F[\mathbf{m}^k] + J[\mathbf{m}^k]\delta\mathbf{m} \tag{2.12}$$

and setting the gradient equal to zero yields the standard Gauss-Newton equations to

be solved for the perturbation $\delta\mathbf{m}$:

$$(J[\mathbf{m}]^\top \mathbf{W}_\mathrm{d}^\top \mathbf{W}_\mathrm{d} J[\mathbf{m}] + \beta \mathbf{W}_\mathrm{m}^\top \mathbf{W}_\mathrm{m})\delta\mathbf{m} = -g(\mathbf{m}). \tag{2.13}$$

The updated model is given by:

$$\mathbf{m}^{k+1} = \mathbf{m}^k + \gamma\delta\mathbf{m}, \tag{2.14}$$

where $\gamma \in (0,1]$ is a coefficient that can be found by a line search. Setting $\gamma = 1$ is the default and a line search is necessary if $\phi(\mathbf{m}^{k+1}) \geq \phi(\mathbf{m}^k)$.

The iterative optimization process is continued until a suitable stopping criterion is reached. Completion of this iterative process yields a minimization for particular value of the trade-off parameter, β. If we are invoking a cooling schedule, and if the desired misfit tolerance is not yet achieved, β is reduced and the iterative numerical optimization procedure is repeated.

Sensitivities

A central element in the above approach is the computation of the sensitivities. The sensitivity functional is defined by:

$$J[\mathbf{m}] = \frac{\partial F[\mathbf{m}]}{\partial \mathbf{m}} = \mathbf{P}\left(\frac{\partial \mathbf{u}}{\partial \mathbf{m}}\right) \tag{2.15}$$

where $\mathbf{P}$ is a linear projection and $d\cdot$ indicates total difference. There are numerous approaches to computing the sensitivity, but the chosen methodologies are dictated by the size of the problem. The discrete sensitivity matrix, $\mathbf{J}$, is a dense $N \times M$ matrix, where N is the number of data and M is the number of model parameters. For some

problems, $\mathbf{J}$ can be computed directly and stored. Ultimately, this computation and storage demands the solution of numerous forward problems (cf. Haber (2015)). In another approach, we can factor $J[\mathbf{m}]$ in symbolic form. In the general case, we solve for the sensitivity implicitly by taking the derivative of $C(\mathbf{m},\mathbf{u}) = 0$ (equation 2.2) to yield:

$$\nabla_{\mathbf{m}} C(\mathbf{m},\mathbf{u}) d\mathbf{m} + \nabla_{\mathbf{u}} C(\mathbf{m},\mathbf{u}) d\mathbf{u} = 0, \tag{2.16}$$

where $\nabla_{\cdot}$ indicates partial difference and both $\nabla_{\mathbf{m}} C(\mathbf{m},\mathbf{u})$ and $\nabla_{\mathbf{u}} C(\mathbf{m},\mathbf{u})$ are matrices. For a given model, $\nabla_{\mathbf{u}} C(\mathbf{m},\mathbf{u})$ corresponds to the forward simulation operator. If the forward problem is well-posed, then the matrix is invertible (Haber, 2015). Equation 2.16 can be rearranged to:

$$d\mathbf{u} = -\left(\nabla_{\mathbf{u}} C(\mathbf{m},\mathbf{u})\right)^{-1} \nabla_{\mathbf{m}} C(\mathbf{m},\mathbf{u}) d\mathbf{m}, \tag{2.17}$$

and combined with equation 2.15 to obtain a formula for the sensitivity matrix. We note that this matrix is dense, often large, and need not actually be formed and stored and the inverse of $\nabla_{\mathbf{u}} C(\mathbf{m},\mathbf{u})$ need not be formed explicitly.

Inversion as optimization

Once the inverse problem has been stated in an optimization framework (equation 2.9), an appropriate optimization routine can be selected. For example, if bound constraints are incorporated, we can use a projected Gauss-Newton algorithm. In large-scale inversions, special attention may have to be given to ensuring a memory efficient optimization algorithm. However, the underlying mechanics of the algorithms often remain unchanged. In a geophysical inversion, we require a model that is consistent

with *a priori* information and known, or assumed, statistical distributions (e.g. the discrepancy principle). As such, the stopping criteria of the inversion are often implemented differently than traditional optimization algorithms or a series of incomplete optimization algorithms are invoked while changing the objective function (Oldenburg and Li, 2005; Haber, 2015; Haber et al., 2000).

The optimization of the stated inverse problem provides the machinery to obtain a mathematical solution. However, before the model is accepted as a viable candidate, there are numerous questions that should be investigated. For example, some questions to address might include: (a) How well does the recovered model fit the observed data? (b) Is there bias in the misfits between the observed and predicted data? (c) What was the path for the convergence? (d) Is there too much or too little structure? (e) Does the model fit with prior knowledge and other data sets? The final results and details about how the inversion algorithm has performed all provide clues as to whether the constructed model can be accepted or if elements in our procedure or its numerical implementation need to be altered and the inversion rerun. These might include: adjusting the assigned uncertainties in the misfit function; altering the model regularization; or, changing aspects of the numerical computations.

2.2.3 Evaluation/interpretation

In this section, we return to the initial question posed, which the inversion was designed to help answer. Questions of interest might include: (a) Are the interesting features supported by the data or are they artifacts?; (b) Does the result make sense geologically and geophysically?; and (c) Are there interesting features that should be investigated further? Addressing these questions usually involves repeating the inversion process with appropriate modifications (cf. Oldenburg and Li (2005); Pidlisecky

et al. (2011); Lines et al. (1988)). As such, we require an implementation that is inherently and unequivocally modular, with all pieces available for manipulation. Black-box software, where the implementations are hidden, obfuscated, or difficult to manipulate, does not promote experimentation and investigation. Exposing the details of the implementation to the geophysicist in a manner that promotes productivity and question-based interrogation is the goal of SIMPEG and is the topic of the next section.

2.3 Modular implementation

An overwhelming amount of choices must be made while working through the forward modeling and inversion process (Figure 2.1). As a result, software implementations of this workflow often become complex and highly interdependent, making it difficult to interact with other scientists or to ask them to pick up and change the work. Our approach to handling this complexity is to propose a framework, Figure 2.2, which compartmentalizes the implementation of inversions into various units. We present the framework in this specific modular style, as each unit contains a targeted subset of choices crucial to the inversion process. The aim of the SIMPEG framework, and implementation, is to allow users to move between terminology, math, documentation, and code with ease, such that there is potential for development in a scalable way. The SIMPEG implementation provides a library that mimics the framework shown in Figure 2.2, with each unit representing a base class. These base classes can be inherited in specific geophysical problems to accelerate development as well as to create code that is consistent between geophysical applications.

2.3.1 Implementation choices

We chose Python (Van Rossum and Drake Jr, 1995) for the implementation of SIMPEG. Python supports object-oriented practices and interactive coding, has extensive support for documentation, and has a large and growing open source scientific community (Lin, 2012). As an interpreted language, however, there are occasionally bottlenecks on speed or memory. These inefficiencies may mean that the code will not be able to scale to a production quality code. However, these computational bottlenecks can often be identified through profiling and can be written in a lower-level language and wrapped in Python. Additionally, these problems are admissible when the goal of the software is clear: we require an interactive research tool where geophysical problems from many disciplines live in one place to enhance experimentation and dissemination of new ideas. To enhance the dissemination of our work, we have released our work under the permissive MIT license for open source software. The MIT license neither forces packages that use SIMPEG to be open source, nor does it restrict commercial use. We have also ensured that we have followed best practices, with regard to version control, code-testing, and documentation (Wilson et al., 2014).

2.3.2 Overview

As discussed in the previous section, the process of obtaining an acceptable model from an inversion generally requires the geophysicist to perform several iterations of the inversion workflow while rethinking and redesigning each piece of the framework to ensure it is appropriate in the current context. Inversions are experimental and empirical by nature and our software package is designed to facilitate this iterative process. To accomplish this iterative process, we have divided the inversion methodology into eight major components (Figure 2.2). The `Mesh` class handles the discretization

of the earth and also provides numerical operators. The forward simulation is split into two classes: the `Survey` and the `Problem`. The `Survey` class handles the geometry of a geophysical problem as well as sources. The `Problem` class handles the simulation of the physics for the geophysical problem of interest. Although created independently, these two classes must be *paired* to form all of the components necessary for a geophysical forward simulation and calculation of the sensitivity. The `Problem` creates geophysical fields, given a source from the `Survey`. The `Survey` interpolates these fields to the receiver locations and converts them to the appropriate data type (for example, by selecting only the measured components of the field). Each of these operations may have associated derivatives, with respect to the model and the computed field; these associated derivatives are included in the calculation of the sensitivity. For the inversion, a `DataMisfit` is chosen to capture the *goodness of fit* of the predicted data and a `Regularization` is chosen to handle non-uniqueness. These inversion elements and an `Optimization` routine are combined into an inverse problem class (`InvProblem`). `InvProblem` is the mathematical statement (i.e. similar to equation 2.9) that will be numerically solved by running an `Inversion`. The `Inversion` class handles organization and dispatch of *directives* between all of the various pieces of the framework.

The arrows in Figure 2.2 indicate what each class takes as a primary argument. For example, both the `Problem` and `Regularization` classes take a `Mesh` class as an argument. The diagram does not show class inheritance, as each of the base classes outlined have many subtypes that can be interchanged. The `Mesh` class, for example, could be a regular Cartesian mesh or a cylindrical coordinate mesh, which have many common properties. We can exploit these common features, such as both meshes being created from tensor products, through inheritance of base classes; differences can be

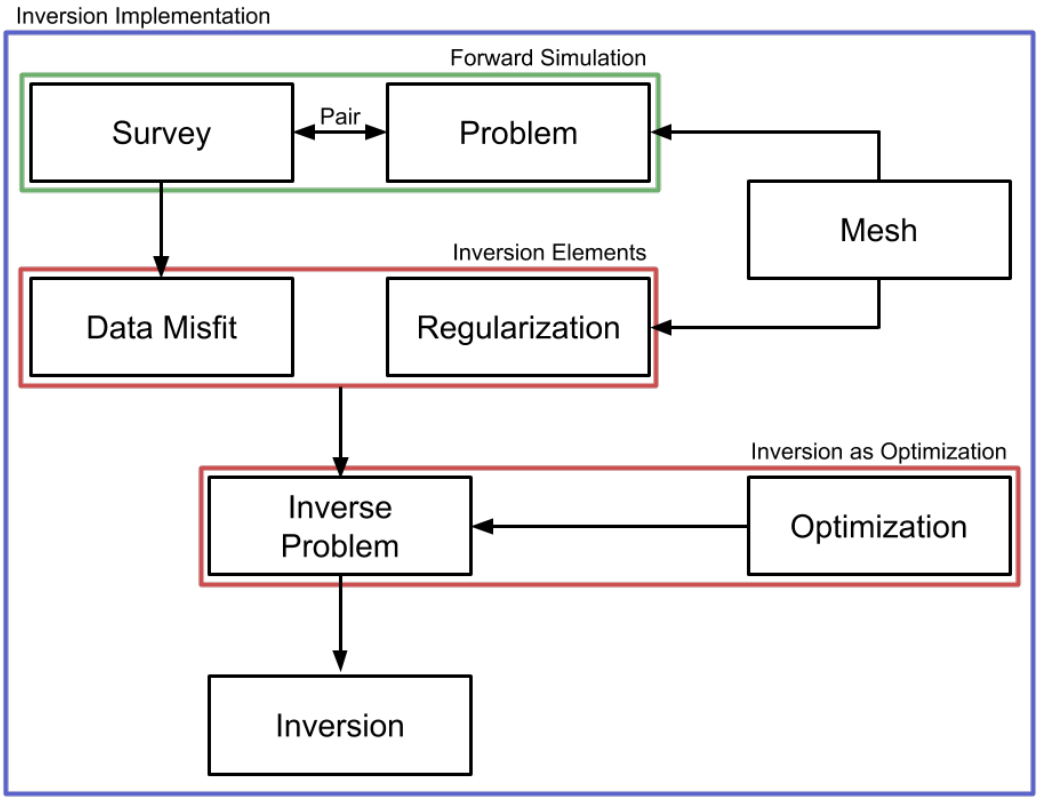

Figure 2.2: SIMPEG framework indicating the flow of information. In the implementation, each of these modules is a base class.

expressed through subtype polymorphism. We refer the reader to the online, up-to-date documentation (http://docs.simpeg.xyz) to observe the class inheritance structure in depth.

2.3.3 Motivating example

We will use the DC resistivity problem from geophysics to motivate and explain the various components of the SIMPEG framework. This example will be referred to throughout this section. We will introduce the example briefly here and refer the reader to Appendix B.2.4 for a more in-depth discussion. The governing equations for DC

resistivity are:

$$\begin{aligned} \nabla \cdot \vec{j} &= I(\delta(\vec{r} - \vec{r}_{s^+}) - \delta(\vec{r} - \vec{r}_{s^-})) = q \\ \frac{1}{\sigma}\, \vec{j} &= -\nabla \phi \end{aligned} \tag{2.18}$$

where σ is the electrical conductivity, ϕ is the electric potential, and I is the input current at the positive and negative dipole locations $\vec{r}_{s^\pm}$, captured as Dirac delta functions. In DC resistivity surveys, differences in the potential field, ϕ, are sampled using dipole receivers to collect observed data. To simulate this partial differential equation (PDE) (or set of PDEs, if there are multiple current injection locations), we must discretize the equation onto a computational mesh.

2.3.4 Mesh

Any numerical implementation requires the discretization of continuous functions into discrete approximations. These approximations are typically organized in a mesh, which defines boundaries, locations, and connectivity. In geophysical simulations, we require the definitions of averaging, interpolation, and differential operators for any mesh. Throughout our work, we have implemented discretization techniques, using a staggered mimetic finite volume approach (Hyman and Shashkov, 1999; Hyman et al., 2002). For an in-depth discussion of the finite volume techniques employed in this thesis, we refer the reader to Appendix B. This work has resulted in an open source package called `discretize`, which provides finite volume techniques abstracted across four mesh types: (1) tensor product mesh; (2) cylindrically symmetric mesh; (3) logically rectangular, non-orthogonal mesh; and (4) octree and quadtree meshes. The techniques and interface to the methodologies are specifically tailored for efficiency and accessibility for geophysical inverse problems. To create a new `Mesh` instance, a `TensorMesh` class can be selected from the `discretize` module and instantiated

with a list of vectors: Here, we import the `discretize` library as well as NumPy

```
import discretize  # See Appendix A and Cockett et al. 2016
import numpy as np
import scipy.sparse as sp
hx = np.ones(30)
hy = np.ones(30)
mesh = discretize.TensorMesh([hx, hy])
```

Program 2.1: Creation of a 2D tensor product mesh using the `discretize` package discussed in Appendix B.

(`np`) and SciPy's sparse matrix package (`sp`) (Oliphant, 2007; Jones et al., 2001). The vectors `hx` and `hy` describe the cell size in each mesh dimension. The dimension of the mesh is defined by the length of the list, requiring very little change to switch mesh dimensions or type. Once an instance of a mesh is created, access to the properties and methods, shown in Table 2.1, is possible. Additional methods and visualization routines are also included in the `Mesh` classes. Of note in Table 2.1 are organizational properties (such as counting and geometric properties), locations of mesh variables as Cartesian grids, differential and averaging operators, and interpolation matrices. We can readily extend the mesh implementation to other types of finite volume meshes (for example, octree (Haber and Heldmann, 2007), logically rectangular non-orthogonal meshes (Hyman et al., 2002), and unstructured meshes (Ollivier-Gooch and Van Altena, 2002)). Additionally, this piece of the framework may be replaced by other methodologies such as finite elements.

With the differential operators readily accessible across multiple mesh types, simulation of a cell-centered discretization for conductivity, σ, in the DC resistivity problem is straightforward. The discretized system of equations, B.4, can be written as:

$$\mathbf{A}(\sigma)\mathbf{u} = \mathbf{D}(\mathbf{M}^{\mathbf{f}}_{\mathbf{1}/\sigma})^{-\mathbf{1}}\mathbf{D}^{\top}\mathbf{u} = -\mathbf{q}, \tag{2.19}$$

Table 2.1: Selected `Mesh` class properties with explanations.

Property or Function	Explanation
dim	Dimension of the mesh
x0	Location of the origin
nC, nN, nF, nE	The number of cells, nodes, faces, or edges. (e.g. nC is the total number of cells)
vol, area, edge	Geometric measurements for the mesh
gridN, gridCC, etc.	Array of grid locations
nodalGrad	Gradient of a nodal variable $\rightarrow$ edge variable
faceDiv	Divergence of a face variable $\rightarrow$ cell-centered variable
edgeCurl	Curl of a edge variable $\rightarrow$ face variable
cellGrad	Gradient of a cell-centered variable $\rightarrow$ face variable
aveF2CC, aveN2CC, etc.	Averaging operators (e.g. F$\rightarrow$CC, takes values on faces and averages them to cell-centers)
getInterpolationMat(loc)	Interpolation matrix for xyz locations

where $\mathbf{D}$ and $\mathbf{D}^\top$ are the divergence and 'gradient' operators, respectively. This equation is assuming Dirichlet boundary conditions and a weak formulation of the DC resistivity equations, as in Section B.2.5. The conductivity, σ, is harmonically averaged from cell-centers to cell-faces to create the matrix $(\mathbf{M}^{\mathbf{f}}_{\mathbf{1}/\sigma})^{-\mathbf{1}}$ (Pidlisecky et al., 2007). Note that the matrix $(\mathbf{M}^{\mathbf{f}}_{\mathbf{1}/\sigma})^{-\mathbf{1}}$ is diagonal when the physical property is isotropic or has coordinate anisotropy on a tensor product mesh, so the inverse is trivial. Using our `discretize` package, this equation is written as:

```
D = mesh.faceDiv
Msig = mesh.getFaceInnerProduct(sigma, invProp=True, invMat=True)
A = D*Msig*D.T
```

Program 2.2: Creation of the matrix $\mathbf{A}(\sigma)$ for the direct current resistivity problem. See Appendix B for details on finite volume.

The code is easy to read, looks similar to the math, can be built interactively using tools such as IPython (Pérez and Granger, 2007), and is not dependent on the dimension of mesh used. Additionally, it is decoupled from the mesh type. For ex-

ample, Figure 2.3 is generated by solving a `DCProblem` for three different mesh types: `TensorMesh`; `TreeMesh`; and, `CurvilinearMesh`. Other than the specific mesh generation code, no other modifications to the DC problem were necessary (see the online examples provided in SIMPEG). Given the electrode locations, a $\mathbf{q}$ can be constructed on each mesh and the system, $\mathbf{A}(\sigma)\mathbf{u} = -\mathbf{q}$, can be solved. There are many excellent packages available to solve matrix equations and we have created a library to interface many of these direct and iterative solvers. The package, `pymatsolver`, comes with a few different types of `Solver` objects that provide a simple and common interface to Super-LU, Paradiso, and Mumps as well as including a few simple preconditioners for iterative solvers (Li, 2005; Schenk and Gartner, 2004; Duff et al., 1986; Balay et al., 2012). The potential field can be projected onto the re-

```
from pymatsolver import PardisoSolver  # Solver wrapping utilities
Ainv = PardisoSolver(A)  # Create a solver object
u = Ainv * (- q)
mesh.plotImage(u)
```

Program 2.3: Solving and plotting the fields (ϕ) for direct current resistivity using `pymatsolver` and visualization utilities in SIMPEG.

ceiver electrode locations through interpolation matrices, which are constructed by the `Mesh` class. Additionally, there are multiple visualization routines that have been included in the `Mesh` class for rapid visualization and interrogation of geophysical fields and physical properties (Figure 2.3). We note that these code snippets can be easily be combined in a script, highlighting the versatility and accessibility of the `Mesh` classes in `discretize`.

This script will be expanded upon and segmented into the various pieces of the framework in the following sections. We find that the development of geophysical codes is often iterative and requires 'scripting' of equations. Only after these equations

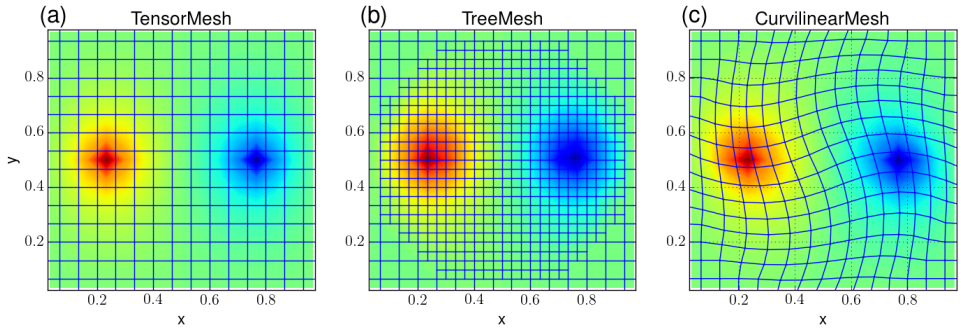

Figure 2.3: Solving the DC resistivity problem for a dipole and using the meshes visualization routine for the potential, ϕ, for three different mesh types: (a) TensorMesh, (b) TreeMesh, and (c) CurvilinearMesh. The potential has been interpolated onto the tensor mesh for visualization.

are correct, as demonstrated by an appropriate test (e.g. `Tests.checkDerivative`), do we formalize and segment our script to enable a geophysical inversion to be run. The toolbox that SIMPEG provides promotes this interactive and iterative style of development.

2.3.5 Forward simulation

The forward simulation in SIMPEG is broken up into a `Survey` class and a `Problem` class. The `Problem` class contains the information and code that capture both the physics used to describe the connection between a physical property distribution and the fields/fluxes that are measured in a geophysical survey. The `Survey` class contains information about the observed data and the geometry of how to collect the data (e.g. locations and types of receivers and sources) given a `Problem` that simulates fields. The `Problem` and the `Survey` must be *paired* together to simulate predicted data. We decided on this separation of the code because it is possible to have multiple mathematical descriptions, of varying complexities, which explain the same observed data. For example, a seismic simulation could have multiple approximations to the

physics, which increase in complexity and accuracy, from straight-ray tomography or Eikonal tomography to full waveform simulation. Additionally, there are often multiple types of geophysical surveys that could be simulated from the same `Problem` class.

Table 2.2: Base `Problem` class properties with explanations.

Property or Function	Explanation
fields(m)	Calculation of the fields given a model
Jvec(m, v)	Sensitivity times a vector
Jtvec(m, v)	Adjoint sensitivity times a vector
Jfull(m)	Full sensitivity matrix
mapping	Maps the model to a physical property

The crucial aspects of the `Problem` class are shown in Table 2.2 and the properties and methods of the `Survey` class are shown in Table 2.3. We note that each of the sub-classes of `Problem` will implement fields and sensitivities in a different way, likely with additional methods and properties. Furthermore, the choice of terminology becomes clearer when these classes are inherited and used in a specific geophysical method (e.g. a `DCProblem` or `EMProblem`). For the `DCProblem`, the `fields` can be created by constructing $\mathbf{A}(\mathbf{m})$ and solving with the source terms, $\mathbf{Q}$, which will be provided by the `DCSurvey`'s source list (`srcList`). Each source has at least one receiver associated with it and the receivers can create a matrix, $\mathbf{P}$, which project the fields, $\mathbf{u}$, onto the data-space. For example, in the DC problem, a dipole receiver samples the potential at each electrode location and computes the difference to give a datum. We note that the process of computing a datum may be more involved and have derivatives with respect the computed fields and, possibly, the model. We solve much of the organizational bottlenecks through general receiver and source classes, which can be inherited and tailored to the specific application. The `mapping`

in the `Problem` provides a transformation from an arbitrary model to a discretized grid function of physical properties. For example, log-conductivity is often used in the inverse problem for DC resistivity, rather than parameterizing directly in terms of conductivity. If this choice is made for the model, an appropriate map (i.e. the exponential) must be provided to transform from the model space to the physical property space (cf. Heagy et al. (2014)).

Table 2.3: Selected `Survey` class properties with explanations.

Property or Function	Explanation
dobs, nD	$\mathbf{d}_{\text{obs}}$, number of data
std	Estimated standard deviations
srcList	List of sources with associated receivers
dpred(m)	Predicted data given a model, $\mathbf{d}_{\text{pred}}(m)$
projectFields(m, u)	Projects the fields, $P(m,u)$
projectFieldsDeriv(m, u)	Derivative of the projection, $\frac{dP(m,u)}{dm}$
residual(m)	$\mathbf{d}_{\text{pred}}(m) - \mathbf{d}_{\text{obs}}$

2.3.6 DC resistivity forward simulation

We present a simple DC-resistivity survey to demonstrate some of the components of SIMPEG in action. We use a set of Schlumberger arrays to complete a vertical sounding. In this example, we have taken our scripts from the previous section describing the forward simulation and combined them in a package called `SimPEG.DC` (http://simpeg.xyz). We use the 3D tensor `mesh` to run the forward simulation for the data of this problem. Here the `srcList` is a list of dipole sources (`DC.SrcDipole`), each of which contains a single receiver, (`DC.RxDipole`). Similar to the illustration in Figure 2.2, the `Problem` and the `Survey` must be paired for either to be used to simulate fields and/or data. These elements represent the major pieces of any forward

```
from SimPEG.EM.Static import DC
survey = DC.SurveyDC(srcList)
problem = DC.ProblemDC(mesh)
problem.pair(survey)
data = survey.dpred(sigma)
```

Program 2.4: Pairing the `Problem` and `Survey` objects to create predicted data, $\mathbf{d}_{\text{pred}}$.

simulation in geophysics; they are crucial and must be well-tested for accuracy and efficiency before any attempt is made at setting up the inverse problem.

2.3.7 Sensitivities

The sensitivity and adjoint will be used in the optimization routine of the inversion. Inefficient or inaccurate calculation of the sensitivities can lead to an extremely slow inversion. This is critical in large-scale inversions, where the dense sensitivity matrix may be too large to hold in memory directly. As discussed in the methodology section, the sensitivity matrix need not be explicitly created when using an iterative optimization algorithm, such as Gauss-Newton (2.13), solved with a conjugate gradient approach. The calculation of vector products with the sensitivity matrices is an important aspect of SIMPEG, which has many tools to make construction and testing of these matrices modular and simple. For the DC resistivity example, the discretized governing equations are written as: $C(\mathbf{m},\mathbf{u}) = \mathbf{A}(\mathbf{m})\mathbf{u} - \mathbf{q} = \mathbf{0}$. We can implement the sensitivity equations 2.15 and 2.17 to yield:

$$\mathbf{J} = -\mathbf{P}(\mathbf{A}(\mathbf{m})^{-1}\nabla_{\mathbf{m}}C(\mathbf{m},\mathbf{u})), \tag{2.20}$$

where $\nabla_{\mathbf{m}}C(\mathbf{m},\mathbf{u})$ is a known sparse matrix, $\mathbf{A}(\mathbf{m})$ is the forward operator and is equivalent to $\nabla_{\mathbf{u}}C(\mathbf{m},\mathbf{u})$, and $\mathbf{P}$ is a projection matrix (cf. Pidlisecky et al. (2007)). The sensitivity matrix is dense and holding it in memory may not be possible. If an

iterative solver is used in the optimization, only matrix vector products are necessary and the sensitivity need not be explicitly calculated or stored. Program 2.5 outlines the calculation of `Jvec`, given a model, `m`, the fields, `u`, and a vector to multiply, `v`. In Program 2.5, we draw the distinction between the model, `m`, and the conductivity, `sig`, which are connected through a mapping, $\sigma = \mathscr{M}(\mathbf{m})$, and associated derivatives. The matrix, $\nabla_{\mathbf{m}} C(\mathbf{m}, \mathbf{u})$, is denoted `dCdm` and formed by looping over each source in the DC resistivity survey.

```
def Jvec(self, m, v, u=None):
    # Set current model; clear dependent property A(m)
    self.curModel = m
    sigma = self.curModel.transform # σ = M(m)
    if u is None:
        # Run forward simulation if u not provided
        u = self.fields(self.curModel)
    else:
        shp = (self.mesh.nC, self.survey.nTx)
        u = u.reshape(shp, order='F')

    D = self.mesh.faceDiv
    G = self.mesh.cellGrad
    # Derivative of model transform, ∂σ/∂m
    dsigdm_x_v = self.curModel.transformDeriv * v

    # Take derivative of C(m,u) w.r.t. m
    dCdm_x_v = np.empty_like(u)
    # loop over fields for each transmitter
    for i in range(self.survey.nTx):
        # Derivative of inner product, (M^f_{1/σ})^{-1}
        dAdsig         = D * self.dMdsig( G * u[:,i] )
        dCdm_x_v[:, i] = dAdsig *  dsigdm_x_v

    # Take derivative of C(m,u) w.r.t. u
    dCdu = self.A
    # Solve for ∂u/∂m
    dCdu_inv = self.Solver(dCdu, **self.solverOpts)
    P        = self.survey.getP(self.mesh)
    J_x_v    = - P * mkvc( dCdu_inv * dCdm_x_v )
    return J_x_v
```

Program 2.5: Sensitivity times a vector method for the `DCProblem`.

2.3.8 Inversion elements

As indicated in the methodology section, there are two key elements needed for a geophysical inversion: `DataMisfit` and `Regularization`. The `DataMisfit` must have a way to calculate predicted data and, as such, it takes a paired survey as an initial argument, which allows forward simulations to be completed. `DataMisfit` and `Regularization` have similar interfaces, which are shown in Table 2.4. The `DataMisfit` class also has a property, `targetMisfit`, for the target misfit, which can be checked by an `InversionDirective` and used as a stopping criteria. As discussed in the methodology section, the `Regularization` is defined independently from the forward simulation. The regularization is with respect to the model, which may or may not be on the same mesh as the forward simulation (i.e. $mesh_I \neq mesh_F$). In this case, a mapping of a model to a physical property on the forward simulation mesh is necessary for the `Problem`. The `Regularization` class also has a `mapping` property, which allows a wide variety of regularizations to be implemented (e.g. an active cell map used to ignore air cells). As such, the `Regularization` `mapping` is often independent from the `mapping` in the `Problem` class, which outputs a physical property. Included in the SIMPEG package are basic Tikhonov regularization routines and simple l_2 norms for both `Regularization` and `DataMisfit` classes. Each of these classes has properties for the appropriate model and data weightings, as discussed in the previous section (e.g. $\mathbf{W}_\mathrm{m}$ and $\mathbf{W}_\mathrm{d}$). These classes are readily extensible, such that they can be customized to specific problems and applications (for example, considering l_1 or l_p norms or customized regularizations).

Table 2.4: Common functions for the `Regularization`, and `DataMisfit` classes.

Function	Explanation
obj(m)	Evaluate the functional given a model when the class is called directly.
obj.deriv(m)	First derivative returns a vector.
obj.deriv2(m, v)	Second derivative as an implicit operator.

2.3.9 Inverse problem and optimization

The `InvProblem` combines the `DataMisfit` and `Regularization` classes by introducing a trade-off parameter, β. In addition to the trade-off parameter, there are methods that evaluate the objective function and its derivatives (Table 2.4). Additional methods can save fields so that information is not lost between evaluation of the objective function and the derivatives. The `InvProblem` may also include bounds on the model properties so that they can be used in the optimization routine. If we consider a joint or integrated inversion, multiple data misfit functions, employing different physics, and that multiple types of regularization functionals may be summed together, possibly with relative weightings, we can define the `InvProblem` (cf. Lines et al. (1988); Holtham and Oldenburg (2010); Heagy et al. (2014)). Once the `InvProblem` can be evaluated to a scalar with associated derivatives, an `Optimization` can either be chosen among the ones included in SIMPEG or provided by an external package. Optimization routines in SIMPEG include steepest descent, L-BFGS, and Inexact Gauss-Newton (cf. Nocedal and Wright (1999)). The components are relatively simple to hook up to external optimization packages (for example, with the optimization package in SciPy (Jones et al., 2001)).

2.3.10 Inversion

The `Inversion` conducts all communication between the various components of the framework and is instantiated with an `InvProblem` object. The `Inversion` has very few external methods but contains the list of directives that are executed throughout the inversion. Each `InversionDirective` has access to the components of the inversion framework and can thus access and change any of these components while the inversion is running. A simple directive may print optimization progress or save models to a database. More complicated directives may change or compute parameters such as, β, reference models, data weights, or model weights. These directives are often guided by heuristics, but versions can often be formalized (see, for example, the iterative Tikhonov style inversion (Tikhonov and Arsenin, 1977; Parker, 1994; Oldenburg and Li, 2005)). There are many computational shortcuts that may be investigated, such as how many inner and outer CG iterations to complete in the inexact Gauss-Newton optimization and whether the number of iterations should change as the algorithm converges to the optimal model. The `directiveList` in the `Inversion` encourages heuristics, which geophysicists often complete 'by hand', to be codified, combined, and shared via a plug-in style framework.

2.3.11 DC resistivity inversion

We will build on the example presented in Section 2.3.6, which has a survey setup that only provides enough information for a vertical sounding. As such, we will decouple our 3D forward mesh and 1D inversion mesh and connect them through a mapping (cf. Kang et al. (2015b)). Additionally, since electrical conductivity is a log-varying parameter, we will also construct a model space that is optimized in log space. Both of these model transformations will be handled with a single map, $\mathscr{M}$,

where $\sigma = \mathscr{M}(\mathbf{m})$. We have provided a number of common mapping transformations

```
from SimPEG import Maps
mapping = Maps.ExpMap(mesh) * Maps.SurjectVertical1D(mesh)
sigma = mapping * model
```

Program 2.6: Creation and chaining together of multiple mapping properties for a model of σ.

in the `SimPEG.Maps` package and these can be easily combined with a multiplication symbol. Additionally, when using these maps, we calculate the derivatives using the chain rule, allowing them to be easily included in the sensitivity calculation (cf. Program 2.5, line 15). Figure 2.5 demonstrates this mapping visually. The 1D model is in $log(\sigma)$, shown in Figure 2.4(a) as a black solid line, and the transformation produces a 3D `sigma` vector, which we plotted in Figure 2.4(b). We can now use the same simulation machinery as discussed in Section 2.3.6, with a single change: Synthetic data,

```
from SimPEG.EM.Static import DC
problem = DC.ProblemDC(mesh, sigmaMap=mapping)
```

Program 2.7: Instantiation of the direct current resistivity problem with a mapping for the σ property.

$\mathbf{d}_{\text{obs}}$, are created using the 1D log-conductivity model and adding 1% Gaussian noise. When creating the regularization inversion element, we note again that the `mapping` parameter can be used to regularize in the space that makes the most sense. In this case, we will regularize on a 1D mesh in log-conductivity space; as such, we will supply only a 1D tensor `mesh` to the regularization. An inversion is run by combining the tools described above. Figure 2.2 illustrates how the components are put together. We note that there are many options and inputs that can enhance the inversion; refer to the online up-to-date documentation (http://docs.simpeg.xyz). The result of this inversion can be seen in Figure 2.5(a) and (b) for the predicted data and model, respectively.

```
mesh1D  = discretize.TensorMesh([mesh.hz])
dmis    = DataMisfit.l2_DataMisfit(survey)
reg     = Regularization.Tikhonov(mesh1D)
opt     = Optimization.InexactGaussNewton()
invProb = InvProblem.BaseInvProblem(dmis, reg, opt)
inv     = Inversion.BaseInversion(invProb)
mopt    = inv.run(m0)
```

Program 2.8: Creating a boiler plate inversion at a low level.

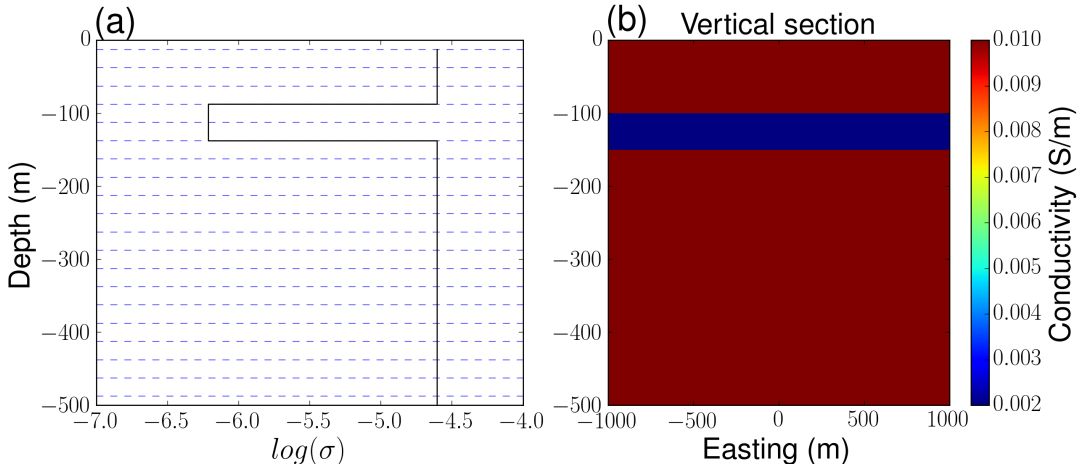

Figure 2.4: Illustration of mapping in DC inversion. (a) 1D log conductivity model. (b) 3D conductivity model.

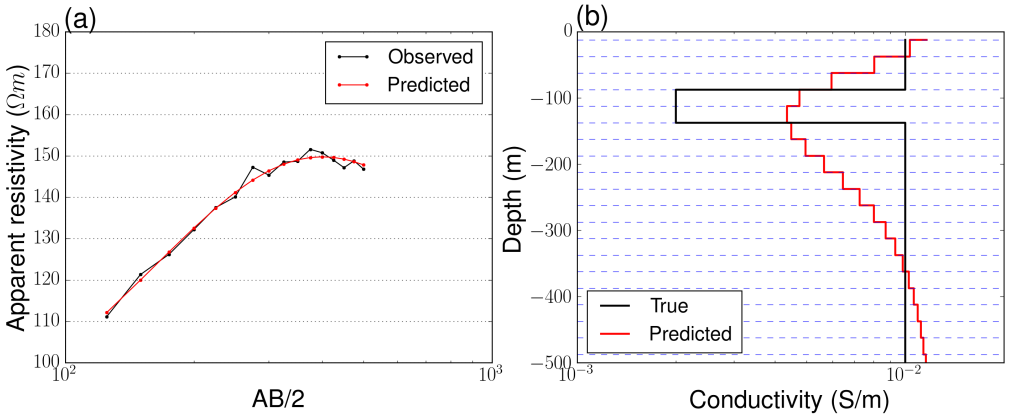

Figure 2.5: (a) Observed (black line) and predicted (red line) apparent resistivity values. (b) True and recovered 1D conductivity model.

2.4 Conclusions

Producing an interpretation from geophysical data through an inversion is an iterative process with many moving pieces. A number of inversion components, techniques, and methodologies have become standard practice. The development of new methodologies to address the evolving challenges in the geosciences will build upon and extend these standard practices, requiring experimentation with, and recombination of, existing techniques. To facilitate this combinatorial experimentation, we have organized the components of geophysical inverse problems in a comprehensive, modular framework. Our implementation of this framework, SIMPEG (http://www.simpeg.xyz), provides an extensible, well-tested toolbox and infrastructure that supports problems, including electromagnetics, fluid flow, seismic, and potential fields. As SIMPEG is formulated with the inverse problem as its core focus, many design choices have been made to ensure that sensitivities are efficient to compute and are readily available; these choices have shown to be advantageous for integrated geophysical inversions. The modular framework that we suggest splits the code into components, which are motivated directly by geophysical methodology and terminology. Splitting the code allows each piece to be improved by specialists, while promoting quantitative communication between researchers.

To accelerate the dissemination and adoption of SIMPEG in the wider community, we have made the entire project open source under the permissive MIT License. The usability of this framework has been a focus of SIMPEG and we strive to use best practices of continuous integration, documentation (http://docs.simpeg.xyz), unit-testing, and version-control. These practices are key to have in place as more modules and packages are created by the community.

Chapter 3

Richards equation

3.1 Introduction

Studying the processes that occur in the vadose zone, the region between the earth's surface and the fully saturated zone, is of critical importance for understanding our groundwater resources. Fluid flow in the vadose zone is described by the Richards equation and parameterized by hydraulic conductivity, which is a nonlinear function of pressure head (Richards, 1931; Celia et al., 1990). Typically, hydraulic conductivity is heterogeneous and can have a large dynamic range. In any site characterization, the spatial estimation of the hydraulic conductivity function is an important step. Achieving this, however, requires the ability to efficiently solve and optimize the nonlinear, time-domain Richards equation. Rather than working with a full, implicit, 3D time-domain system of equations, simplifications are consistently used to avert the conceptual, practical, and computational difficulties inherent in the parameterization and inversion of the Richards equation. These simplifications typically parameterize the conductivity and assume that it is a simple function in space, often adopting a ho-

mogeneous or one dimensional layered soil profile (cf. (Binley et al., 2002; Deiana et al., 2007; Hinnell et al., 2010; Liang and Uchida, 2014)). Due to the lack of constraining hydrologic data, such assumptions are often satisfactory for fitting observed measurements, especially in two and three dimensions as well as in time. However, as more data become available, through spatially extensive surveys and time-lapse proxy measurements (e.g. direct current resistivity surveys and distributed temperature sensing), extracting more information about subsurface hydrogeologic parameters becomes a possibility. The proxy data can be directly incorporated through an empirical relation (e.g. (Archie, 1942)) or time-lapse estimations can be structurally incorporated through some sort of regularization technique (Haber and Gazit, 2013; Haber and Oldenburg, 1997; Hinnell et al., 2010). Recent advances have been made for the forward simulation of the Richards equation in a computationally-scalable manner (Orgogozo et al., 2014). However, the inverse problem is non-trivial, especially in three-dimensions (Towara et al., 2015), and must be considered using modern numerical techniques that allow for spatial estimation of hydraulic parameters. However, this is especially intricate to both derive and implement due to the nonlinear, time-dependent forward simulation and potential model dependence in many aspects of the Richards equation (e.g. multiple empirical relations, boundary/initial conditions). To our knowledge, there has been no large-scale inversion for distributed hydraulic parameters in three dimensions using the Richards equation as the forward simulation.

Inverse problems in space and time are often referred to as history matching problems (see Dean and Chen (2011); Dean et al. (2008); Sarma et al. (2007); Oliver and Reynolds (2001); Šimunek et al. (2012) and reference within). Inversions use a flow simulation model, combined with some a-priori information, in order to estimate a spatially variable hydraulic conductivity function that approximately yields the observed

data. The literature shows a variety of approaches for this inverse problem, including trial-and-error, stochastic methods, and various gradient based methods (Bitterlich et al., 2004; Binley et al., 2002; Carrick et al., 2010; Durner, 1994; Finsterle and Zhang, 2011; Mualem, 1976; Šimunek and van Genuchten, 1996). The way in which the computational complexity of the inverse method scales becomes important as problem size increases (Towara et al., 2015). Computational memory and time often become a bottleneck for solving the inverse problem, both when the problem is solved in 2D and, particularly, when it is solved in 3D (Haber et al., 2000). To solve the inverse problem, stochastic methods are often employed, which have an advantage in that they can examine the full parameter space and give insights into non-uniqueness (Finsterle and Kowalsky, 2011). However, as the number of parameters we seek to recover in an inversion increases, these stochastic methods require that the forward problem be solved many times, which often makes these methods impractical. This scalability, especially in the context of hydrogeophysics has been explicitly noted in the literature (cf. Binley et al. (2002); Deiana et al. (2007); Towara et al. (2015); Linde and Doetsch (2016)).

Derivative-based optimization techniques become a practical alternative when the forward problem is computationally expensive or when there are many parameters to estimate (i.e. thousands to millions). Inverse problems are ill-posed and thus to pose a solvable optimization problem, an appropriate regularization is combined with a measure of the data misfit to state a deterministic optimization problem (Tikhonov and Arsenin, 1977). Alternatively, if prior information can be formulated using a statistical framework, we can use Bayesian techniques to obtain an estimator through the Maximum A Posteriori model (MAP) (Kaipio and Somersalo, 2004). In the context of Bayesian estimation, gradient based methods are also important, as they can be used to efficiently sample the posterior (Bui-Thanh and Ghattas, 2015).

A number of authors have sought solutions for the inverse problem, where the forward problem is the Richards equation (cf. (Bitterlich and Knabner, 2002; Iden and Durner, 2007; Šimunek et al., 2012) and references within). The discretization of the Richards equation is commonly completed by an implicit method in time and a finite volume or finite element method in space. Most work uses a Newton-like method for the resulting nonlinear system, which arises from the discretization of the forward problem. For the deterministic inverse problem using the Richards equation, previous work uses some version of a Gauss-Newton method (e.g. Levenberg-Marquardt), with a **direct** calculation of the sensitivity matrix (Finsterle and Kowalsky, 2011; Šimunek and van Genuchten, 1996; Bitterlich and Knabner, 2002). However, while these approaches allow for inversions of moderate scale, they have one major drawback: the sensitivity matrix is large and dense; its computation requires dense linear algebra and a non-trivial amount of memory (cf. (Towara et al., 2015)). Previous work used either external numerical differentiation (e.g. PEST) or automatic differentiation in order to directly compute the sensitivity matrix (Finsterle and Zhang, 2011; Bitterlich and Knabner, 2002; Doherty, 2004; Towara et al., 2015). Finite difference can generate inaccuracies in the sensitivity matrix and, consequently, tarry the convergence of the optimization algorithm. Furthermore, external numerical differentiation is computationally intensive and limits the number of model parameters that can be estimated.

The goal of this chapter is to suggest a modern numerical formulation that allows the inverse problem to be solved **without explicit** computation of the sensitivity matrix by using **exact** derivatives of the discrete formulation (Haber et al., 2000). Our technique is based on the discretize-then-optimize approach, which discretizes the forward problem first and then uses a deterministic optimization algorithm to solve the inverse problem (Gunzburger, 2003). To this end, we require the discretization of the forward

problem. Similar to the work of (Celia et al., 1990), we use an implicit Euler method in time and finite volume in space. Given the discrete form, we show that we can analytically compute the derivatives of the forward problem with respect to distributed hydraulic parameters and, as a result, obtain an implicit formula for the sensitivity. The formula involves the solution of a linear time-dependent problem; we avoid computing and storing the sensitivity matrix directly and, rather, suggest a method to efficiently compute the product of the sensitivity matrix and its adjoint times a vector. Equipped with this formulation, we can use a standard inexact Gauss-Newton method to solve the inverse problem for distributed hydraulic parameters in 3D. This large-scale distributed parameter estimation becomes computationally tractable with the technique presented in this chapter and can be employed with any iterative Gauss-Newton-like optimization technique.

This chapter is structured as follows: in Section 3.2, we discuss the discretization of the forward problem on a staggered mesh in space and backward Euler in time; in Section 3.3, we formulate the inverse problem and construct the implicit functions used for computations of the Jacobian-vector product. In Section 3.4.1, we demonstrate the validity of the implementation of the forward problem and sensitivity calculation. In Section 3.4, we validate the numerical implementation and compare to the literature. Chapter 4 will expand upon the techniques introduced in this chapter to show the effectiveness of the implicit sensitivity algorithm in comparison to existing numerical techniques.

3.1.1 Attribution and dissemination

To accelerate both the development and dissemination of this approach, we have built these tools on top of an open source framework for organizing simulation and inverse

problems in geophysics (Cockett et al., 2015c). We have released our numerical implementation under the permissive MIT license. Our implementation of the implicit sensitivity calculation for the Richards equation and associated inversion implementation is provided and tested to support 1D, 2D, and 3D forward and inverse simulations with respect to custom empirical relations and sensitivity to parameters within these functions. The source code can be found at https://github.com/simpeg/simpeg and may be a helpful resource for researchers looking to use or extend our implementation. I have presented early versions of this work at two international conferences (Cockett and Haber, 2013a,b) and have submitted a version of this manuscript for peer review (Cockett et al., 2017).

3.2 Forward problem

In this section, we describe the Richards equations and its discretization (Richards, 1931). The Richards equation is a nonlinear parabolic partial differential equation (PDE) and we follow the so-called mixed formulation presented in (Celia et al., 1990) with some modifications. In the derivation of the discretization, we give special attention to the details used to efficiently calculate the effect of the sensitivity on a vector, which is needed in any derivative based optimization algorithm.

3.2.1 Richards equation

The parameters that control groundwater flow depend on the effective saturation of the media, which leads to a nonlinear problem. The groundwater flow equation has a diffusion term and an advection term which is related to gravity and only acts in the z-direction. There are two different forms of the Richards equation; they differ in how they deal with the nonlinearity in the time-stepping term. Here, we use the most

fundamental form, referred to as the 'mixed'-form of the Richards equation (Celia et al., 1990):

$$\frac{\partial \theta(\psi)}{\partial t} - \nabla \cdot k(\psi)\nabla\psi - \frac{\partial k(\psi)}{\partial z} = 0 \quad \psi \in \Omega \tag{3.1}$$

where ψ is pressure head, $\theta(\psi)$ is volumetric water content, and $k(\psi)$ is hydraulic conductivity. This formulation of the Richards equation is called the 'mixed'-form because the equation is parameterized in ψ but the time-stepping is in terms of θ. The hydraulic conductivity, $k(\psi)$, is a heterogeneous and potentially anisotropic function that is assumed to be known when solving the forward problem. In this chapter, we assume that k is isotropic, but the extension to anisotropy is straightforward (Cockett et al., 2015c, 2016a). The equation is solved in a domain, Ω, equipped with boundary conditions on $\partial\Omega$ and initial conditions, which are problem-dependent.

An important aspect of unsaturated flow is noticing that both water content, θ, and hydraulic conductivity, k, are functions of pressure head, ψ. There are many empirical relations used to relate these parameters, including the Brooks-Corey model (Brooks and Corey, 1964) and the van Genuchten-Mualem model (Mualem, 1976; van Genuchten, 1980). The van Genuchten model is written as:

$$\theta(\psi) = \begin{cases} \theta_r + \dfrac{\theta_s - \theta_r}{(1 + |\alpha\psi|^n)^m} & \psi < 0 \\ \theta_s & \psi \geq 0 \end{cases} \tag{3.2a}$$

$$k(\psi) = \begin{cases} K_s \theta_e(\psi)^l (1 - (1 - \theta_e(\psi)^{-m})^m)^2 & \psi < 0 \\ K_s & \psi \geq 0 \end{cases} \tag{3.2b}$$

where

$$\theta_e(\psi) = \frac{\theta(\psi) - \theta_r}{\theta_s - \theta_r}, \qquad m = 1 - \frac{1}{n}, \qquad n > 1 \tag{3.3}$$

Here, θ_r and θ_s are the residual and saturated water contents, K_s is the saturated hydraulic conductivity, α and n are fitting parameters, and, $\theta_e(\psi) \in [0,1]$ is the effective saturation. The pore connectivity parameter, l, is often taken to be $\frac{1}{2}$, as determined by Mualem (1976). Figure 4.1 shows the functions over a range of negative pressure head values for four soil types (sand, loam, sandy clay, and clay). The pressure head varies over the domain $\psi \in (-\infty, 0)$. When the value is close to zero (the left hand side), the soil behaves most like a saturated soil where $\theta = \theta_s$ and $k = K_s$. As the pressure head becomes more negative, the soil begins to dry, which the water retention curve shows as the function moving towards the residual water content (θ_r). Small changes in pressure head can change the hydraulic conductivity by several orders of magnitude; as such, $k(\psi)$ is a highly nonlinear function, making the Richards equation a nonlinear PDE.

3.2.2 Discretization

The Richards equation is parameterized in terms of pressure head, ψ. Here, we describe simulating the Richards equation in one, two, and three dimensions. We start by discretizing in space and then we discretize in time. This process yields a discrete, nonlinear system of equations; for its solution, we discuss a variation of Newton's method.

Spatial Discretization

In order to conservatively discretize the Richards equation, we introduce the flux $\vec{f}$ and rewrite the equation as a first order system of the form:

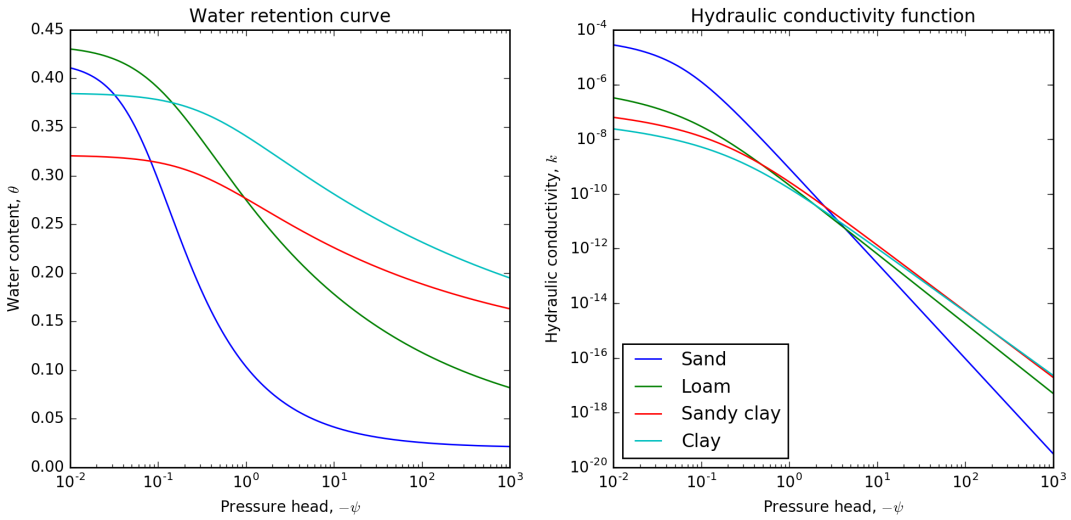

Figure 3.1: The water retention curve and the hydraulic conductivity function for four canonical soil types of sand, loam, sandy clay, and clay.

$$\frac{\partial \theta(\psi)}{\partial t} - \nabla \cdot \vec{f} - \frac{\partial k(\psi)}{\partial z} = 0 \tag{3.4a}$$

$$k(\psi)^{-1}\vec{f} = \nabla \psi \tag{3.4b}$$

We then discretize the system using a standard staggered finite volume discretization (cf. Ascher (2008); Haber (2015); Cockett et al. (2016a), and Appendix B). This discretization is a natural extension of mass-conservation in a volume where the balance of fluxes into and out of a volume are conserved (Lipnikov and Misitas, 2013). Here, it is natural to assign the entire cell one hydraulic conductivity value, k, which is located at the cell center. Such assigning leads to a piecewise constant approximation for the hydraulic conductivity and allows for discontinuities between adjacent cells. From a geologic perspective, discontinuities are prevalent, as it is possible to have

large differences in hydraulic properties between geologic layers in the ground. The pressure head, ψ, is also located at the cell centers and the fluxes are located on cell faces, which lead to the usual staggered mesh or Marker and Cell (MAC) discretization in space (Fletcher, 1988). We demonstrate the discretization in 1D, 2D and 3D on the tensor mesh in Figure 3.2. We discretize the function, ψ, on a cell-centered grid, which results in a grid function, $\boldsymbol{\psi}$. We use bold letters to indicate other grid functions.

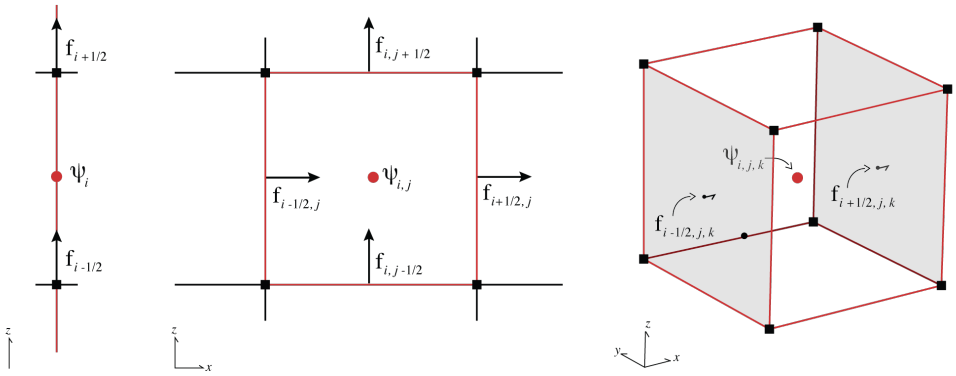

Figure 3.2: Discretization of unknowns in 1D, 2D and 3D space. Red circles are the locations of the discrete hydraulic conductivity K and the pressure head ψ. The arrows are the locations of the discretized flux $\vec{f}$ on each cell face.

The discretization of a diffusion-like equation on an orthogonal mesh is well-known (see (Haber and Ascher, 2001; Fletcher, 1988; Haber et al., 2007; Ascher and Greif, 2011) and reference within). We discretize the differential operators by using the usual mass balance consideration and the elimination of the flux, $\mathbf{f}$ [1]. This spatial discretization leads to the following discrete nonlinear system of ordinary differential

[1]Here we assume an isotropic conductivity that leads to a diagonal mass matrix and this yields easy elimination of the fluxes.

equations (assuming homogeneous Dirichlet boundary conditions):

$$\frac{d\theta(\psi)}{dt} - \mathbf{D}\,\mathrm{diag}\left(\mathbf{k}_{Av}(\psi^{n+1})\right)\mathbf{G}\psi - \mathbf{G}_z\left(\mathbf{k}_{Av}(\psi^{n+1})\right) = 0 \tag{3.5}$$

Here, $\mathbf{D}$ is the discrete divergence operator and $\mathbf{G}$ is the discrete gradient operator. The discrete derivative in the z-direction is written as $\mathbf{G}_z$. The values of ψ and $k(\psi)$ are known on the cell-centers and must be averaged to the cell-faces, which we complete through harmonic averaging (Haber and Ascher, 2001).

$$\mathbf{k}_{Av}(\psi) = \frac{1}{\mathbf{A}_v \frac{1}{\mathbf{k}(\psi)}} \tag{3.6}$$

where $\mathbf{A}_v$ is a matrix that averages from cell-centers to faces and the division of the vector is done pointwise; that is, we use the vector notation, $(1/\mathbf{v})_i = 1/\mathbf{v}_i$. We incorporate boundary conditions using a ghost-point outside of the mesh (Trottenberg et al., 2001).

Time discretization and stepping

The Richards equation is often used to simulate water infiltrating an initially dry soil. At early times in an infiltration experiment, the pressure head, ψ, can be close to discontinuous. These large changes in ψ are also reflected in the nonlinear terms $k(\psi)$ and $\theta(\psi)$; as such, the initial conditions imposed require that an appropriate time discretization be chosen. Hydrogeologists are often interested in the complete evolutionary process, until steady-state is achieved, which may take many time-steps. Here, we describe the implementation of a fully-implicit backward Euler numerical scheme. Higher-order implicit methods are not considered here because the uncertainty associated with boundary conditions and the fitting parameters in the Van Genuchten models

(eq. 4.2) have much more effect than the order of the numerical method used.

The discretized approximation to the mixed-form of the Richards equation, using fully-implicit backward Euler, reads:

$$F(\psi^{n+1}, \psi^n) = \frac{\theta(\psi^{n+1}) - \theta(\psi^n)}{\Delta t} - \mathbf{D}\ \mathrm{diag}\left(\mathbf{k}_{Av}(\psi^{n+1})\right)\mathbf{G}\psi^{n+1} - \mathbf{G}_z\left(\mathbf{k}_{Av}(\psi^{n+1})\right) = 0 \tag{3.7}$$

This is a nonlinear system of equations for ψ^{n+1} that needs to be solved numerically by some iterative process. Either a Picard iteration (as in Celia et al. (1990)) or a Newton root-finding iteration with a step length control can be used to solve the system. Note that to deal with dependence of θ with respect to ψ in Newton's method, we require the computation of $\frac{d\theta}{d\psi}$. We can complete this computation by using the analytic form of the hydraulic conductivity and water content functions (e.g. derivatives of eq. 4.2). We note that a similar approach can be used for any smooth curve, even when the connection between θ and ψ are determined empirically (for example, when $\theta(\psi)$ is given by a spline interpolation of field data).

3.2.3 Solving the nonlinear equations

Regardless of the empirical relation chosen, we must solve 3.7 using an iterative root-finding technique. Newton's method iterates over $m = 1, 2, \ldots$ until a satisfactory estimation of ψ^{n+1} is obtained. Given $\psi^{n+1,m}$, we approximate $F(\psi^{n+1}, \psi^n)$ as:

$$F(\psi^{n+1}, \psi^n) \approx F(\psi^{n,m}, \psi^n) + \mathbf{J}_{\psi^{n+1,m}}\delta\psi \tag{3.8}$$

where the Jacobian for iteration, m, is:

$$\mathbf{J}_{\psi^{n+1,m}} = \left.\frac{\partial F(\psi, \psi^n)}{\partial \psi}\right|_{\psi^{n+1,m}} \tag{3.9}$$

The Jacobian is a large dense matrix, and its computation necessitates the computation of the derivatives of $F(\psi^{n+1,m}, \psi^n)$. We can use numerical differentiation in order to evaluate the Jacobian (or its product with a vector). However, in the context of the inverse problem, an exact expression is preferred. Given the discrete forward problem, we obtain that:

$$\mathbf{J}_{\psi^{n+1,m}} = \frac{1}{\Delta t}\frac{d\theta(\psi^{n+1,m})}{d\psi^{n+1,m}} - \frac{d}{d\psi^{n+1,m}}\left(\mathbf{D}\,\mathrm{diag}\left(\mathbf{k}_{Av}(\psi^{n+1,m})\right)\mathbf{G}\psi^{n+1,m}\right) - \mathbf{G}_z\frac{d\mathbf{k}_{Av}(\psi^{n+1,m})}{d\psi^{n+1,m}} \tag{3.10}$$

Here, recall that $\mathbf{k}_{Av}$ is harmonically averaged and its derivative can be obtained by the chain rule:

$$\frac{d\mathbf{k}_{Av}(\psi)}{d\psi} = \mathrm{diag}\left((\mathbf{A}_v\mathbf{k}^{-1}(\psi))^{-2}\right)\mathbf{A}_v\,\mathrm{diag}\left(\mathbf{k}^{-2}(\psi)\right)\frac{d\mathbf{k}(\psi)}{d\psi} \tag{3.11}$$

Similarly, for the second term in (3.10) we obtain:

$$\frac{\partial}{\partial\psi}\left(\mathbf{D}\ \mathrm{diag}\left(\mathbf{k}_{Av}(\psi)\right)\mathbf{G}\psi\right) = \mathbf{D}\ \mathrm{diag}\left(\mathbf{k}_{Av}(\psi)\right)\mathbf{G} + \mathbf{D}\ \mathrm{diag}\left(\mathbf{G}\psi\right)\frac{\partial\mathbf{k}_{Av}(\psi)}{\partial\psi} \tag{3.12}$$

Here the notation $^{n+1,m}$ has been dropped for brevity. For the computations above, we need the derivatives of functions $\mathbf{k}(\psi)$ and $\theta(\psi)$; note that, since the relations are assumed local (point wise in space) given the vector, ψ, these derivatives are diagonal

matrices. For Newton's method, we solve the linear system:

$$\mathbf{J}_{\psi^{n+1,m}}\,\delta\psi \;=\; -F(\psi^{n+1,m},\psi^{n}) \tag{3.13}$$

For small-scale problems, we can solve the linear system using direct methods; however, for large-scale problems, iterative methods are more commonly used. The existence of an advection term in the PDE results in a non-symmetric linear system in the Newton solve. Thus, when using iterative techniques to solve this system, an appropriate iterative method, such as BICGSTAB or GMRES (Saad, 1996; Barrett et al., 1994), must be used. For a discussion on solver choices in the context of the Richards equation please see Orgogozo et al. (2014).

At this point, it is interesting to note the difference between the Newton iteration and the Picard iteration suggested in (Celia et al., 1990). We can verify that the Picard iteration uses an approximation to the Jacobian $\mathbf{J}_{\psi^{n+1,m}}\,\delta\psi$ given by dropping the second term from (3.12). This term can have negative eigenvalues and dropping it is typically done when considering the lagged diffusivity method (Vogel, 2001). However, as discussed in (Vogel, 2001), ignoring this term can slow convergence.

Finally, a new iterate is computed by adding the Newton update to the last iterate:

$$\psi^{n+1,m+1} = \psi^{n+1,m} + \alpha\,\delta\psi$$

where α is a parameter that guarantees that

$$\|F(\psi^{n,m+1},\psi^{n})\| < \|F(\psi^{n,m},\psi^{n})\|$$

To obtain α, we perform an Armijo line search (Nocedal and Wright, 1999). In our

numerical experiments, we have found that this method can fail when the hydraulic conductivity is strongly discontinuous and changes rapidly. In such cases, Newton's method yields a poor descent direction. Therefore, if the Newton iteration fails to converge to a solution, the update is performed with the mixed-form Picard iteration. Note that Picard iteration can be used, even when Newtons method fails, because Picard iteration always yields a descent direction (Vogel, 2001).

At this point, we have discretized the Richards equation in both time and space while devoting special attention to the derivatives necessary in Newton's method and the Picard iteration as described in (Celia et al., 1990). The exact derivatives of the discrete problem will be used in the following two sections, which outline the implicit formula for the sensitivity and its incorporation into a general inversion algorithm. The implementation is provided as a part of the open source SimPEG project (Cockett et al. (2015c), http://simpeg.xyz).

3.3 Inverse Problem

The location and spatial variability of, for example, an infiltration front over time is inherently dependent on the hydraulic properties of the soil column. As such, direct or proxy measurements of the water content or pressure head at various depths along a soil profile contain information about the soil properties. We pose the inverse problem, which is the estimation of distributed hydraulic parameters, given either water content or pressure data. We frame this problem under the assumption that we wish to estimate hundreds of thousands to millions of distributed model parameters. Due to the large number of model parameters that we aim to estimate in this inverse problem, Bayesian techniques or external numerical differentiation, such as the popular PEST toolbox (Doherty, 2004), are not computationally feasible. Instead, we will employ a

direct method by calculating the exact derivatives of the discrete the Richards equation and solving the sensitivity implicitly. For brevity, we show the derivation of the sensitivity for an inversion model of only saturated hydraulic conductivity, K_s, from pressure head data, $\mathbf{d}_{\text{obs}}$. This derivation can be readily extended to include the use of water content data and inverted for other distributed parameters in the heterogeneous hydraulic conductivity function. We will demonstrate the sensitivity calculation for multiple distributed parameters in the numerical examples (Section 4.5).

The Richards equation simulation produces a pressure head field at all points in space as well as through time. Data can be predicted, $\mathbf{d}_{\text{pred}}$, from these fields and compared to observed data, $\mathbf{d}_{\text{obs}}$. To be more specific, we let $\Psi = [(\psi^1)^\top, \ldots, (\psi^{n_t})^\top]^\top$ be the (discrete) pressure field for all space and n_t time steps. When measuring pressure head recorded only in specific locations and times, we define the predicted data, $\mathbf{d}_{\text{pred}}$, as $\mathbf{d}_{\text{pred}} = \mathbf{P}\Psi(\mathbf{m})$. Here, the vector $\mathbf{m}$ is the vector containing all of the parameters which we are inverting for (e.g. K_s, α, n, θ_r, or θ_s when using the van Genuchten empirical relation). The matrix, $\mathbf{P}$, interpolates the pressure head field, Ψ, to the locations and times of the measurements. Since we are using a simple finite volume approach and backward Euler in time, we use linear interpolation in both space and time to compute $\mathbf{d}_{\text{pred}}$ from Ψ. Thus, the entries of the matrix $\mathbf{P}$ contain the interpolation weights. For linear interpolation in 3D, $\mathbf{P}$ is a sparse matrix that contains up to eight non-zero entries in each row. Note that the time and location of the data measurement is independent and decoupled from the numerical discretization used in the forward problem. A water retention curve, such as the van Genuchten model, can be used for computation of predicted water content data, which requires another nonlinear transformation, $\mathbf{d}^{\theta}_{\text{pred}} = \mathbf{P}\theta(\Psi(\mathbf{m}), \mathbf{m})$. Note here that the transformation to water content data, in general, depends on the model to be estimated in the inversion, which will be addressed

in the numerical examples. For brevity in the derivation that follows, we will make two simplifying assumptions: (1) that the data are pressure head measurements, which requires a linear interpolation that is not dependent on the model; and, (2) that the model vector, $\mathbf{m}$, describes only distributed saturated hydraulic conductivity. Our software implementation *does not* make these assumptions; our numerical examples will use water content data, a variety of empirical relations, and calculate the sensitivity to multiple heterogeneous empirical parameters.

We can now formulate the discrete inverse problem to estimate saturated hydraulic conductivity, $\mathbf{m}$, from the observed pressure head data, $\mathbf{d}_{\text{obs}}$. We frame the inversion as an optimization problem, which minimizes a data misfit and a regularization term. Chapter 2 showed an approach for geophysical inversions where hundreds of thousands to millions of distributed parameters are commonly estimated in a deterministic inversion (Tikhonov and Arsenin, 1977; Oldenburg and Li, 2005; Constable et al., 1987; Haber, 2015). Please refer to the previous chapter for the details of this inversion methodology. The hydrogeologic literature also shows the use of these techniques; however, there is also a large community advancing stochastic inversion techniques and geologic realism (cf. Linde et al. (2015)). Regardless of the inversion algorithm used, an efficient method to calculate the sensitivity is crucial; this method is the focus of our work.

3.3.1 Implicit sensitivity calculation

The optimization problem requires the derivative of the pressure head with respect to the model parameters, $\frac{\partial \Psi}{\partial \mathbf{m}}$. We can obtain an approximation of the sensitivity matrix through a finite difference method on the forward problem (Šimunek and van Genuchten, 1996; Finsterle and Kowalsky, 2011; Finsterle and Zhang, 2011). One

forward problem, or two, when using central differences, must be completed for each column in the Jacobian at every iteration of the optimization algorithm. This style of differentiation proves advantageous in that it can be applied to any forward problem; however, it is highly inefficient and introduces errors into the inversion that may slow the convergence of the scheme (Doherty, 2004). Automatic differentiation (AD) can also be used (Nocedal and Wright, 1999). However, AD does not take the structure of the problem into consideration and often requires that the dense Jacobian be explicitly formed. Bitterlich and Knabner (2002) presents three algorithms (finite difference, adjoint, and direct) to directly compute the elements of the dense sensitivity matrix for the Richards equation. As problem size increases, the memory required to store this dense matrix often becomes a practical computational limitation (Haber et al., 2004; Towara et al., 2015). As we show next, it is possible to explicitly write the derivatives of the Jacobian and evaluate their products with vectors using only sparse matrix operations. This algorithm is much more efficient than finite differencing, especially for large-scale simulations, since it does not require explicitly forming and storing a large dense matrix. Rather, the algorithm efficiently computes matrix-vector and adjoint matrix-vector products with sensitivity. We can use these products for the solution of the Gauss-Newton system when using the conjugate gradient method, which bypasses the need for the direct calculation of the sensitivity matrix and makes solving large-scale inverse problems possible. Other geophysical inverse problems have used this idea extensively, especially in large-scale electromagnetics (cf. Haber et al. (2000)). The challenge in both the derivation and implementation for the Richards equation lies in differentiating the nonlinear time-dependent forward simulation with respect to multiple distributed hydraulic parameters.

The approach to implicitly constructing the derivative of the Richards equation in

time involves writing the whole time-stepping process as a block bi-diagonal matrix system. The discrete Richards equation can be written as a function of the model. For a single time-step, the equation is written:

$$F(\psi^{n+1}(\mathbf{m}), \psi^n(\mathbf{m}), \mathbf{m}) = \frac{\theta^{n+1}(\psi^{n+1}) - \theta^n(\psi^n)}{\Delta t}$$
$$- \mathbf{D}\ \mathrm{diag}\left(\mathbf{k}_{Av}(\psi^{n+1}, \mathbf{m})\right) \mathbf{G}\psi^{n+1} - \mathbf{G}_z \mathbf{k}_{Av}(\psi^{n+1}, \mathbf{m}) = 0 \tag{3.14}$$

In this case, $\mathbf{m}$ is a vector that contains all the parameters of interest. Note that ψ^{n+1} and ψ^n are also functions of $\mathbf{m}$. In general, θ^{n+1} and θ^n are also dependent on the model; however, for brevity, we will omit these derivatives. The derivatives of F to the change in the parameters $\mathbf{m}$ can be written as:

$$\nabla_{\mathbf{m}} F(\psi^n, \psi^{n+1}, \mathbf{m}) = \frac{\partial F}{\partial \mathbf{k}_{Av}} \frac{\partial \mathbf{k}_{Av}}{\partial \mathbf{m}} + \frac{\partial F}{\partial \psi^n} \frac{\partial \psi^n}{\partial \mathbf{m}} + \frac{\partial F}{\partial \psi^{n+1}} \frac{\partial \psi^{n+1}}{\partial \mathbf{m}} = 0 \tag{3.15}$$

or, in more detail:

$$\frac{1}{\Delta t}\left(\frac{\partial \theta^{n+1}}{\partial \psi^{n+1}} \frac{\partial \psi^{n+1}}{\partial \mathbf{m}} - \frac{\partial \theta^n}{\partial \psi^n} \frac{\partial \psi^n}{\partial \mathbf{m}}\right) - \mathbf{D}\ \mathrm{diag}\left(\mathbf{G}\psi^{n+1}\right)\left(\frac{\partial \mathbf{k}_{Av}}{\partial \mathbf{m}} + \frac{\partial \mathbf{k}_{Av}}{\partial \psi^{n+1}} \frac{\partial \psi^{n+1}}{\partial \mathbf{m}}\right)$$
$$- \mathbf{D}\ \mathrm{diag}\left(\mathbf{k}_{Av}(\psi^{n+1})\right) \mathbf{G} \frac{\partial \psi^{n+1}}{\partial \mathbf{m}} - \mathbf{G}_z \left(\frac{\partial \mathbf{k}_{Av}}{\partial \mathbf{m}} + \frac{\partial \mathbf{k}_{Av}}{\partial \psi^{n+1}} \frac{\partial \psi^{n+1}}{\partial \mathbf{m}}\right) = 0 \tag{3.16}$$

The above equation is a linear system of equations and, to solve for $\frac{d\Psi}{d\mathbf{m}}$, we rearrange the block-matrix equation:

$$\overbrace{\left[\frac{1}{\Delta t}\frac{\partial \theta^{n+1}}{\partial \psi^{n+1}} - \mathbf{D}\,\mathrm{diag}\left(\mathbf{G}\psi^{n+1}\right)\frac{\partial \mathbf{k}_{Av}}{\partial \psi^{n+1}} - \mathbf{D}\,\mathrm{diag}\left(\mathbf{k}_{Av}(\psi^{n+1},\mathbf{m})\right)\mathbf{G} - \mathbf{G}_z\frac{\partial \mathbf{k}_{Av}}{\partial \psi^{n+1}}\right]}^{\mathbf{A}_0(\psi^{n+1})}\frac{\partial \psi^{n+1}}{\partial \mathbf{m}} + \underbrace{\left[-\frac{1}{\Delta t}\frac{\partial \theta^{n}}{\partial \psi^{n}}\right]}_{\mathbf{A}_{-1}(\psi^n)}\frac{\partial \psi^{n}}{\partial \mathbf{m}} = \underbrace{\left[-\mathbf{D}\,\mathrm{diag}\left(\mathbf{G}\psi^{n+1}\right)\frac{\partial \mathbf{k}_{Av}}{\partial \mathbf{m}} - \mathbf{G}_z\frac{\partial \mathbf{k}_{Av}}{\partial \mathbf{m}}\right]}_{\mathbf{B}(\psi^{n+1})} \tag{3.17}$$

Here, we use the subscript notation of $\mathbf{A}_0(\psi^{n+1})$ and $\mathbf{A}_{-1}(\psi^n)$ to represent two block-diagonals of the large sparse matrix $\mathbf{A}(\Psi,\mathbf{m})$. Note that all of the terms in these matrices are already evaluated when computing the Jacobian of the Richards equations in Section 3.2 and that they contain only basic sparse linear algebra manipulations without the inversion of any matrix. If ψ_0 does not depend on the model, meaning the initial conditions are independent, then we can formulate the block system as:

$$\overbrace{\begin{bmatrix} \mathbf{A}_0(\psi_1) & & & & \\ \mathbf{A}_{-1}(\psi_1) & \mathbf{A}_0(\psi_2) & & & \\ & \mathbf{A}_{-1}(\psi_2) & \mathbf{A}_0(\psi_3) & & \\ & & \ddots & \ddots & \\ & & & \mathbf{A}_{-1}(\psi_{n_t-1}) & \mathbf{A}_0(\psi_{n_t}) \end{bmatrix}}^{\mathbf{A}(\Psi,\mathbf{m})} \overbrace{\begin{bmatrix} \frac{\partial \psi_1}{\partial \mathbf{m}} \\ \frac{\partial \psi_2}{\partial \mathbf{m}} \\ \vdots \\ \frac{\partial \psi_{n_t-1}}{\partial \mathbf{m}} \\ \frac{\partial \psi_{n_t}}{\partial \mathbf{m}} \end{bmatrix}}^{\frac{\partial \Psi}{\partial \mathbf{m}}} = \overbrace{\begin{bmatrix} \mathbf{B_1}(\psi_1) \\ \mathbf{B_2}(\psi_2) \\ \vdots \\ \mathbf{B_{n-1}}(\psi_{n_t-1}) \\ \mathbf{B_n}(\psi_{n_t}) \end{bmatrix}}^{\mathbf{B}(\Psi,\mathbf{m})} \tag{3.18}$$

This is a block matrix equation; both the storage and solve will be expensive if it is explicitly computed. Indeed, its direct computation is equivalent to the adjoint method (Bitterlich and Knabner, 2002; Dean and Chen, 2011).

Since only matrix vector products are needed for the inexact Gauss-Newton optimization method, the matrix $\mathbf{J}$ is never needed explicitly and only the products of the form $\mathbf{Jv}$ and $\mathbf{J}^\top\mathbf{z}$ are needed for arbitrary vectors $\mathbf{v}$ and $\mathbf{z}$. Projecting the full sensitivity matrix onto the data-space using $\mathbf{P}$ results in the following equations for the Jacobian:

$$\mathbf{J} = \mathbf{PA}(\Psi, \mathbf{m})^{-1}\mathbf{B}(\Psi, \mathbf{m}) \tag{3.19a}$$

$$\mathbf{J}^\top = \mathbf{B}(\Psi, \mathbf{m})^\top\mathbf{A}(\Psi, \mathbf{m})^{-\top}\mathbf{P}^\top \tag{3.19b}$$

In these equations, we are careful to not write $\frac{d\Psi}{d\mathbf{m}}$, as it is a large dense matrix which we do not want to explicitly compute or store. Additionally, the matrices $\mathbf{A}(\Psi, \mathbf{m})$ and $\mathbf{B}(\Psi, \mathbf{m})$ do not even need to be explicitly formed because the matrix $\mathbf{A}(\Psi, \mathbf{m})$ is a triangular block-system, which we can solve using forward or backward substitution with only one block-row being solved at a time (this is equivalent to a single time step). To compute the matrix vector product, $\mathbf{Jv}$, we use a simple algorithm:

1. Given the vector $\mathbf{v}$ calculate $\mathbf{y} = \mathbf{Bv}$
2. Solve the linear system $\mathbf{Aw} = \mathbf{y}$ for the vector $\mathbf{w}$
3. Set $\mathbf{Jv} = \mathbf{Pw}$

Here, we note that we complete steps (1) and (2) using a for-loop with only one block-row being computed and stored at a time. As such, only the full solution, Ψ, is stored and all other block-entries may be computed as needed. There is a complication here if data is in terms of water content or effective saturation, as the data projection is no longer linear and may have model dependence. These complications can be dealt

with using the chain rule during step (3). Similarly, to compute the adjoint $\mathbf{J}^\top \mathbf{z}$ involves the intermediate solve for $\mathbf{y}$ in $\mathbf{A}^\top \mathbf{y} = \mathbf{P}^\top \mathbf{z}$ and then computation of $\mathbf{B}^\top \mathbf{y}$. Again, we solve the block-matrix via backward substitution with all block matrix entries being computed as needed. Note that the backward substitution algorithm can be viewed as time stepping, which means that it moves from the final time back to the initial time. This time stepping is equivalent to the adjoint method that is discussed in Dean and Chen (2011) and references within. The main difference between our approach and the classical adjoint-based method is that our approach yields the *exact* gradient of the discrete system; no such guarantee is given for adjoint-based methods.

The above algorithm and the computations of all of the different derivatives summarizes the technical details of the computations of the sensitivities. Equipped with this "machinery", we now demonstrate that validity of our implementation.

3.4 Numerical results

The focus of this section is to validate and compare our algorithm and implementation to the literature. The following chapter will focus on applications of this work as well as demonstrate computational scalability of the algorithm for realistic field examples (Chapter 4).

3.4.1 Validation

Forward problem

The Richards equation has no analytic solution, which makes testing the code more involved. Here we have chosen to use a fictitious source experiment to rigorously test the code. In this experiment, we approximate an infiltration front by an arctangent

function in one dimension, which is centered over the highly nonlinear part of the van Genuchten curves, with $\psi \in [-60, -20]$ centimeters. The arctangent curve advects into the soil column with time. The advantage of using an analytic function is that the derivative is known explicitly and can be calculated at all times. However, it should be noted that the Richards equation does not satisfy the analytic solution exactly, but differs by a function, $S(x,t)$. We refer to this function as the fictitious source term. The analytic function that we used has similar boundary conditions and shape to an example in Celia et al. (1990) and is considered over the domain $x \in [0,1]$.

$$\Psi_{\text{true}}(x,t) = -20 \arctan(20((x-0.25)-t)) - 40 \tag{3.20}$$

This analytic function is shown at times 0 and 0.5 in Figure 3.3 and has a pressure head range of $\psi \in [-60, -20]$. We can compare these values to the van Genuchten curves in Figure 4.1. We can then put the known pressure head into the Richards equation (3.1) and calculate the analytic derivatives and equate them to a source term, $S(x,t)$. Knowing this source term and the analytic boundary conditions, we can solve discretized form of the Richards equation, which should reproduce the analytic function in Equation 3.20. Table 3.1 shows the results of the fictitious source test when the number of mesh-cells is doubled and the time-discretization is both fixed and equivalent to the mesh size (i.e. $k = h$). In this case, we expect that the backward-Euler time discretization, which is $\mathscr{O}(\delta)$, will limit the order of accuracy. The final column of Table 3.1 indeed shows that the order of accuracy is $\mathscr{O}(\delta)$. The higher errors in the coarse discretization are due to high discontinuities and changes in the source term, which the coarse discretization does not resolve. We can complete a similar procedure in two and three dimensions and these tests show similar results of convergence. The

rigorous testing of the code presented provides confidence in the forward simulation that is used throughout the following sections of this chapter.

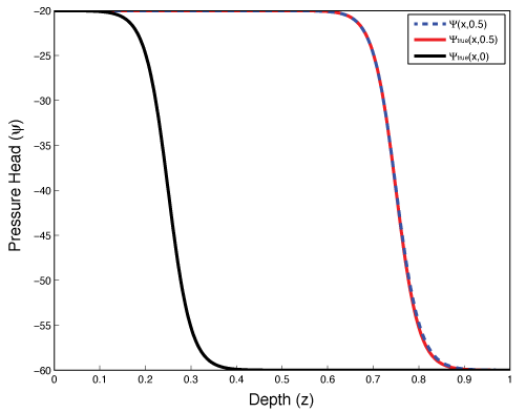

Figure 3.3: Fictitious source test in 1D showing the analytic function Ψ_{true} at times 0.0 and 0.5 and the numerical solution $\Psi(x, 0.5)$ using the mixed-form Newton iteration.

Table 3.1: Fictitious source test for Richards equation in 1D using the mixed-form Newton iteration.

Mesh Size (n)	$\|\|\Psi(x,0.5) - \Psi_{\text{true}}(x,0.5)\|\|_\infty$	Order Decrease, $\mathcal{O}(\delta)$
64	5.485569e+00	
128	2.952912e+00	0.894
256	1.556827e+00	0.924
512	8.035072e-01	0.954
1024	4.086729e-01	0.975
2048	2.060448e-01	0.988
4096	1.034566e-01	0.994
8192	5.184507e-02	0.997

Inverse problem

In order to test the implicit sensitivity calculation, we employ derivative and adjoint tests as described in Haber (2015). Given that the Taylor expansion of a function $f(\mathbf{m}+h\Delta\mathbf{m})$ is

$$f(\mathbf{m}+h\Delta\mathbf{m}) = f(\mathbf{m}) + h\mathbf{J}\Delta\mathbf{m} + \mathscr{O}(h^2), \tag{3.21}$$

for any of the model parameters considered, we see that our approximation of $f(\mathbf{m}+h\Delta\mathbf{m})$ by $f(\mathbf{m})+h\mathbf{J}\Delta\mathbf{m}$ should converge as $\mathscr{O}(h^2)$ as h is reduced. This allows us to verify our calculation of $\mathbf{Jv}$. To verify the adjoint, $\mathbf{J}^\top\mathbf{v}$, we check that

$$\mathbf{w}^\top\mathbf{Jv} = \mathbf{v}^\top\mathbf{J}^\top\mathbf{w} \tag{3.22}$$

for any two random vectors, $\mathbf{w}$ and $\mathbf{v}$. These tests are run for all of the parameters considered in an inversion of the Richards equation. Within our implementation, both the derivative and adjoint tests are included as unit tests which are run on any updates to the implementation (https://travis-ci.org/simpeg/simpeg).

3.4.2 Comparison to literature

Code-to-code comparisons have been completed for comparison to Celia et al. (1990), which can be found in Cockett (2017). The following results are a direct comparison to the results produced by Celia et al. (1990) for the Picard iteration only; Celia et al. (1990) did not implement the Newton iteration. The direct comparison is completed: (a) to give confidence to the numerical results; (b) to compare the Newton iteration to the Picard iteration of the mixed formulation for a well-known example; and (c) demonstrate the use of multiple empirical models. Here, we use the Haverkamp model (Haverkamp et al., 1977) (rather than the classically used van Genuchten model) for

the water retention and hydraulic conductivity functions.

$$\begin{aligned} \theta(\psi) &= \frac{\alpha(\theta_s - \theta_r)}{\alpha + |\psi|^\beta} + \theta_r \\ K(\psi) &= K_s \frac{A}{A + |\psi|^\gamma} \end{aligned} \tag{3.23}$$

We used parameters of $\alpha = 1.611 \times 10^6$, $\theta_s = 0.287$, $\theta_r = 0.075$, $\beta = 3.96$, $A = 1.175 \times 10^6$, $\gamma = 4.74$, and $K_s = 9.44 \times 10^{-5}$ m/s, which are the same as in Celia et al. (1990). The 40 cm high 1D soil column has initially dry conditions with a pressure head $\psi_0(x,0) = -61.5$cm. The boundary conditions applied are inhomogeneous Dirichlet with the top of the soil column, $\psi(40\text{cm},t) = -20.7$cm, and the bottom of the soil column, $\psi(0\text{cm},t) = -61.5$cm. The initial conditions are not consistent with the boundary condition at the top of the soil profile. This inconsistency leads to a boundary layer and a steep gradient in the pressure head at early times; as such, we anticipate that the Newton iteration will converge slowly at these times. The spatial grid is regular and has a spacing of 1.0cm, while the time-stepping, Δt, is manipulated. The three iterative methods described in Section 3.2 are implemented and compared at 360s: (1) head-based form Picard; (2) mixed-form Picard; and, (3) mixed-form Newton. Figure 3.4 shows the solution obtained with the three iterative methods. Comparing the head-based formulation to the mixed-formulation for a large time-step (e.g. $\Delta t = 120$s) shows the degradation of the head-based method. Not only is the infiltration front smoothed, there is underestimate of the front location (Figure 3.4a). The underestimate of the infiltration front location is due to a loss of mass, which can be traced back the initial formulation of the head-based method (Celia et al., 1990). The mixed-formulation, solved with either a Picard iteration or a Newton method, conserves mass and correctly identifies the spatial location of the infiltration front (Figure 3.4b). The

results obtained here show excellent agreement with Celia et al. (1990).

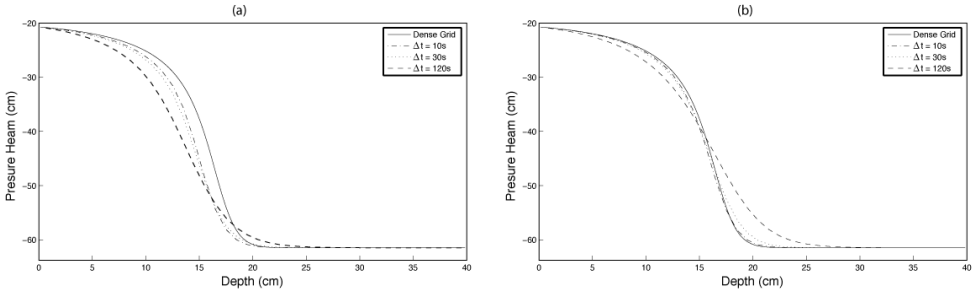

Figure 3.4: Comparison of results to Celia et al. (1990) showing the differences in the (a) head-based and (b) mixed formulations for t=360s.

3.5 Conclusions

The number of parameters that are estimated in the Richards equation inversions has grown and will continue to grow as time-lapse data and geophysical data integration become standard in site characterizations. The increase in data quantity and quality provides the opportunity to estimate spatially distributed hydraulic parameters from the Richards equation; doing so requires efficient simulation and inversion strategies. In this chapter, we have shown a derivative-based optimization algorithm that does not store the Jacobian, but rather computes its effect on a vector (i.e. $\mathbf{Jv}$ or $\mathbf{J}^\top\mathbf{z}$). By not storing the Jacobian, the size of the problem that we can invert becomes much larger. We have presented efficient methods to compute the Jacobian that can be used for all empirical hydraulic parameters, even if the functional relationship between parameters is obtained from laboratory experiments.

Our technique allows a deterministic inversion, which includes regularization, to be formulated and solved for any of the empirical parameters within the Richards equation. For a full 3D simulation, as many as ten spatially distributed parameters may be

needed, resulting in a highly non-unique problem; as such, we may not be able to reasonably estimate all hydraulic parameters. Depending on the setting, amount of a-priori knowledge, quality and quantity of data, the selection of which parameters to invert for may vary. Our methodology enables practitioners to experiment in 1D, 2D and 3D with full simulations and inversions, in order to explore the parameters that are important to a particular dataset. Our numerical implementation is provided in an open source repository (http://simpeg.xyz) and is integrated into the framework presented in Chapter 2. The goal of Chapter 4 is to work with these techniques in an experiment that represents a field infiltration experiment. The following chapter will also further document the scalability of this approach when moving to 3D inversions.

Chapter 4

Vadose zone inversions

4.1 Introduction

Characterizing hydraulic parameters can be completed using direct methods, which require laboratory samples to be taken. However, because much of the vadose zone consists of unconsolidated materials, these invasive measurement techniques may disturb the soil properties, especially porosity and water content (Deiana et al., 2007). Furthermore, these point measurements may not represent the entire hydrologic setting and cannot observe the vadose zone processes *in situ*. Geophysical methods, however, can be used to generate spatially extensive estimates of physical properties, which can be related, through empirical relations, to hydrogeologic properties, such as hydraulic conductivity. We can use geophysical inversions, such as direct current resistivity, to image the electrical conductivity. The electrical conductivity is related to the soil moisture content and the fluid electrical conductivity. This relation is empirically described

through Archie's equation (Archie, 1942):

$$\sigma = a^{-1}\sigma_f \phi^m \theta_e^n \tag{4.1}$$

where σ is the bulk electrical conductivity of the fluid filled soil or rock, σ_f is the electrical conductivity of the fluid, ϕ is the porosity, and θ_e is the effective saturation. The exponents, m and n, and scaling factor, a, are empirical fitting parameters, which are occasionally referred to as the cementation exponent, the saturation exponent, and the tortuosity factor, respectively. Archie's equation, which must be further modified in the presence of clays, has three empirical parameters that are either unknown or poorly constrained. If the imaging for electrical conductivity is completed in a time-lapse experiment, we can capture changes of moisture content over time due to fluid movement. In the vadose zone, the moisture content is the time-stepping term in the Richards equation, which is related to the pressure head through another empirical relation (e.g. the van Genuchten-Mualem empirical relations). Given estimates of saturation, an inversion can be formulated for hydraulic parameters using the Richards equation (Chapter 3). The van Genuchten empirical model has five parameters ($K_s, \theta_r, \theta_s, \alpha,$, and n) that are commonly estimated in laboratory experiments. We can use these hydraulic parameters in groundwater models to make predictions and decisions about groundwater processes.

In summary, we can use geophysical measurements to make time-lapse images of water content, which, in turn, can be used in conjunction with a groundwater simulation to invert for hydraulic properties (cf. Hinnell et al. (2010)). Throughout this process, we require multiple empirical parameterizations as well as extensive hydrogeologic knowledge, including knowledge of boundary and initial conditions. In Binley

et al. (2002), a cross-well tomography experiment was conducted using radar and DC resistivity over the course of a vadose zone tracer test. The center of mass of the tracer was found and tracked over time using the geophysical imaging and subsequent empirical interpretation as water content changes, $\Delta\theta$. A separate groundwater simulation was completed using the Richards equation to estimate the saturated hydraulic conductivity, which was assumed to be homogeneous and isotropic and was determined through "trial and error" because "no automatic data matching was possible" due to the size of the numerical simulation. Similarly, in Deiana et al. (2007), "only the saturated hydraulic conductivity K_s was modified by trial and error to match field data." In this case, anisotropy was introduced to further fit the results obtained by matching the infiltration experiment. Similarly, when investigating coupled hydrogeologic and geophysical simulation, Hinnell et al. (2010) notes that the parameterization was "simpler than reality to make numerical inversion tractable" and that the computational power necessary for a stochastic "approach limits widespread application and use of the hydrogeophysic[s]." The lack of algorithms that formulate and solve the inverse problem for the Richards equation is a hurdle in the analysis and joint quantitative analysis of hydrologic and geophysical data. In Chapter 3, we presented a formulation that reduced a number of these barriers to efficient large-scale parameter estimation. However, the numerical ability to invert for five distributed parameters at once is not immediately practical.

In this chapter, we will use the algorithm and formulation of the inverse problem developed in Chapter 3 to investigate a distributed multi-parameter inversion in both one and three dimensions using water content data. The goal of this work is: (a) to explore the nonlinearities and couplings in the van Genuchten functions; and, (b) to demonstrate that it is now possible to complete an inversion for distributed hydraulic

parameters in a large-scale 3D simulation. We also refer the reader to Appendix C, where we expose the assumptions in a forward simulation framework to manipulation and estimation. This appendix also demonstrates multiple geologic- and physics-based parameterizations that can be used to embed knowledge between diverse disciplines.

4.1.1 Attribution and dissemination

This work extends the previous chapter in terms of applications, but it also extends the forward simulation framework and the inversion framework that is necessary to flexibly invert for multiple distributed parameters at once. We abstracted the necessary advancements in the framework from the organization and implementation of electromagnetics inversions in both time domain and frequency domain. The forward simulation framework used for all numerical examples is derived from this cross-disciplinary, collaborative work in the Richards equation and electromagnetics. This work is presented in Heagy et al. (2016) for eight formulations of Maxwell's equations for geophysical simulations and inverse problems. Appendix C shows an adaption of the forward simulation framework for the Richards equation, along with a number of other collaborative case-studies that build upon or extend this work. The library for implementing the numerical methods and inverse formulation is contained in `SimPEG.FLOW.Richards` (https://github.com/simpeg/simpeg). The code to reproduce the the majority of the results and figures in this chapter is available on FigShare (Cockett and Haber, 2015); the examples are available within the online documentation (http://docs.simpeg.xyz). The numerical examples are inspired by work from my undergraduate thesis, of which two papers were published during the course of my PhD (Pidlisecky et al., 2013; Cockett and Pidlisecky, 2014). These examples involved a field site in California designed for managed aquifer recharge, which col-

lected dense geophysical data to help inform management practices. The two papers' results showed a qualitative comparison to water content in one dimension (Pidlisecky et al., 2013) and a numerical rock physics approach that analyzed soils as they clogged (Cockett and Pidlisecky, 2014). These two studies informed the numerical setup of the following synthetic experiments.

4.2 Empirical relationships

The Richards equation relies upon the correct parameterization of both the water retention curve and the hydraulic conductivity function. A number of empirical models have been proposed to describe these functions, including Brooks and Corey (1964), Haverkamp et al. (1977), van Genuchten (1980), and Mualem (1976). All of these empirical models are loosely based on the physical interpretation of fitting parameters; however, this basis can be misleading and it has been shown that many parameters have no physical meaning and should be considered empirical shape factors (Schaap and Leij, 2000). These functions have also been interpreted as splines, which can be helpful in the inverse formulation (Bitterlich and Knabner, 2002). The van Genuchten model, introduced in Chapter 3, is written as:

$$\theta(\psi) = \begin{cases} \theta_r + \dfrac{\theta_s - \theta_r}{(1 + |\alpha\psi|^n)^m} & \psi < 0 \\ \theta_s & \psi \geq 0 \end{cases} \tag{4.2a}$$

$$k(\psi) = \begin{cases} K_s \theta_e(\psi)^l (1 - (1 - \theta_e(\psi)^{-m})^m)^2 & \psi < 0 \\ K_s & \psi \geq 0 \end{cases} \tag{4.2b}$$

where

$$\theta_e(\psi) = \frac{\theta(\psi) - \theta_r}{\theta_s - \theta_r}, \qquad m = 1 - \frac{1}{n}, \qquad n > 1 \tag{4.3}$$

Here, θ_r and θ_s are the residual and saturated moisture contents, K_s is the saturated hydraulic conductivity, α and n are fitting parameters, and $\theta_e(\psi) \in [0,1]$ is the effective saturation. The pore connectivity parameter, l, is often taken to be $\frac{1}{2}$, as determined by Mualem (1976).

These curves are unknown at every point in space in the inverse problem. We will use a number of canonical parameters for the van Genuchten empirical relation to look at the water retention curve and the hydraulic conductivity function; Table 4.1 shows the values for these parameters. The soil-naming scheme refers to the proportions of sand, silt, and clay. Figure 4.1 shows the functions over a range of negative soil water potentials for four soil types (sand, loam, sandy clay, and clay). The soil water potential varies over the domain $\psi \in (-\infty, 0)$. When the value is close to zero (the left hand side), the soil behaves most like a saturated soil where $\theta = \theta_s$ and $K = K_s$. As the water potential becomes more negative, the soil begins to dry, which the water retention curve shows as the function moving towards the residual water content (θ_r). The parameters α and n determine the slope and shape of this transition. In Figure 4.1, we see that the water retention curve for sand has a high saturated water content but rapidly changes to a low residual water content. For clay, this transition, with respect to the soil water potential, is much more gradual. This difference has to do with the small size of the pores in clay and their ability to retain water, even at high suctions. This difference is reflected in the hydraulic conductivity function: when close to saturated conditions, sand has the highest hydraulic conductivity, however, as the soil dries, the hydraulic conductivity of sand decreases rapidly and becomes relatively lower than a

loam or clay with the same water potential.

Table 4.1: Canonical soil parameters for the water retention and hydraulic conductivity curves (Van Genuchten et al., 1991)

Soil Type	θ_r	θ_s	α (1/m)	n	K_s (m/s)
Sand	0.020	0.417	13.8	1.592	5.8e-05
Loamy sand	0.035	0.401	11.5	1.474	1.7e-05
Sandy loam	0.041	0.412	6.8	1.322	7.2e-06
Loam	0.027	0.434	9.0	1.220	1.9e-06
Silt loam	0.015	0.486	4.8	1.211	3.7e-06
Sandy clay loam	0.068	0.330	3.6	1.250	1.2e-06
Clay loam	0.075	0.390	3.9	1.194	6.4e-07
Silty clay loam	0.040	0.432	3.1	1.151	4.2e-07
Sandy clay	0.109	0.321	3.4	1.168	3.3e-07
Silty clay	0.056	0.423	2.9	1.127	2.5e-07
Clay	0.090	0.385	2.7	1.131	1.7e-07

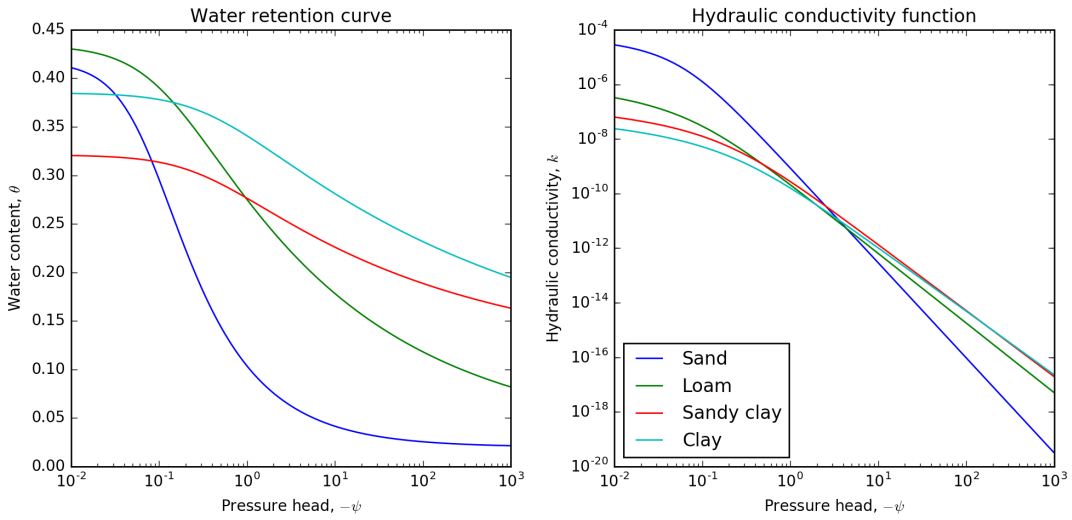

Figure 4.1: The water retention curve and the hydraulic conductivity function for four canonical soil types of sand, loam, sandy clay, and clay.

In this chapter, we are interested in the inverse problem for the soil parameters, which controls the shape and intercepts of these two functions. To solve this problem

systematically, we will first observe how the shape of these curves change as the empirical parameters are modified over the full empirical range. In Figure 4.2, we complete this process for the hydraulic conductivity function for the four soil types (sand, loam, sandy clay, and clay). The bounds show the change from that curve by varying a single parameter and holding the rest constant. The saturated hydraulic conductivity, K_s, can move the entire curve up and down. The parameters α and n control the shape of the curve as the negative soil water potential increases. Figure 4.3 also shows the four soil types for the water retention curve. As expected in this case, θ_s controls the y-intercept and θ_r controls the minimum value of the function. Again, the parameters α and n control the shape of the curve between the bounds of θ_s and θ_r. In this case, the parameter n has much the same effect as θ_r; Šimunek et al. (1998) notes this high correlation. Note that the parameters α and n are involved in both the calculation of k and θ, which links these two curves in a physically reasonable way (Bitterlich et al., 2004). Additionally, the low parameterization of these curves, when using the van Genuchten empirical relationship, means that changing one parameter has a global effect over the domain of the soil water potential, $\psi \in (-\infty, 0)$. When framing the inverse problem, a spline parameterization for each curve can alternatively be used. Although framing in this way decouples the two curves, we need to take care when extrapolating the results to moisture contents not covered by the bounds of the experiment and to ensure that the obtained curves are physically reasonable (Bitterlich et al., 2004). To avoid these potential pitfalls of an inversion, we will use the van Genuchten empirical parameterization for the remainder of the experimentation.

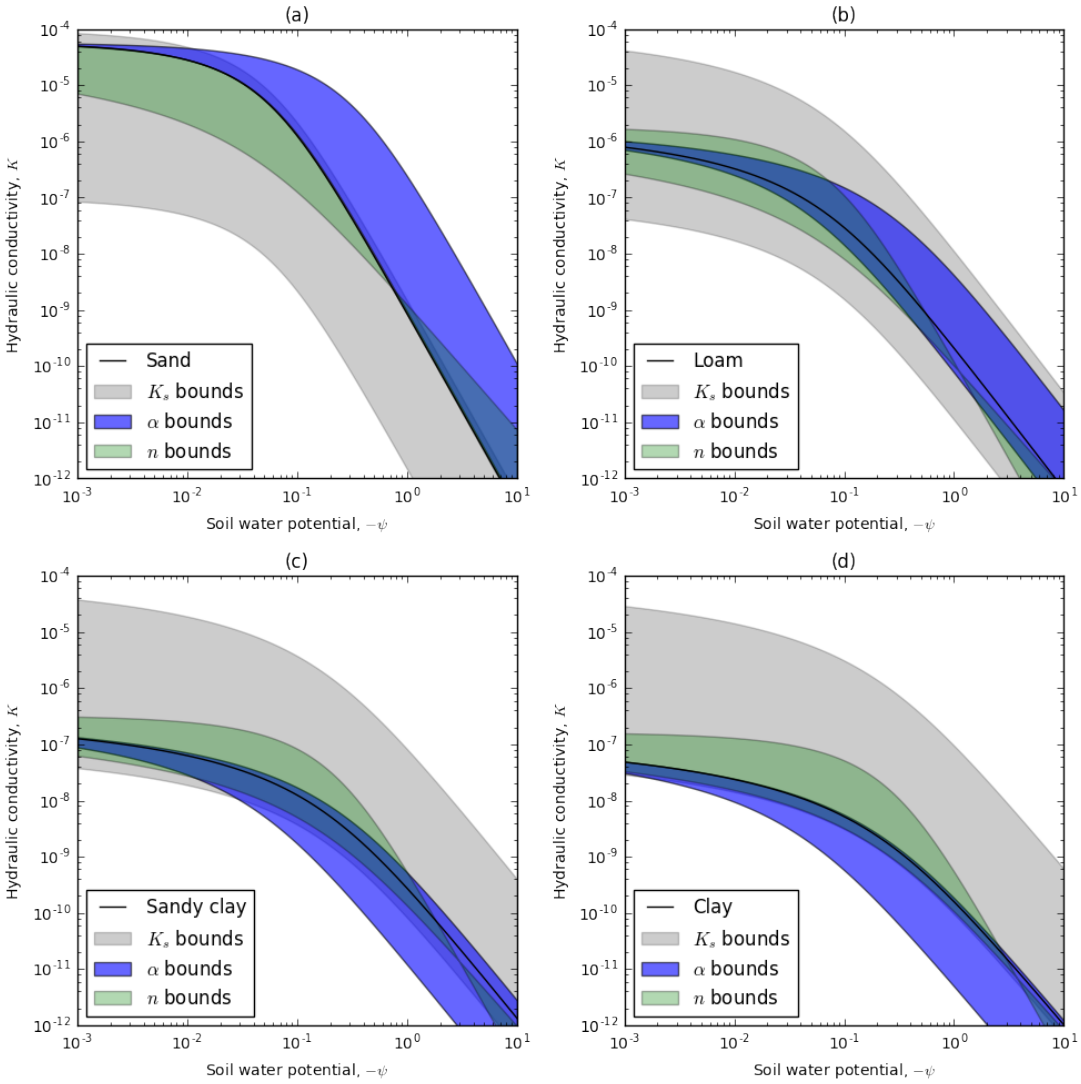

Figure 4.2: The hydraulic conductivity function showing bounds of the various parameters $K_s \in [1 \times 10^{-7}, 1 \times 10^{-4}]$, $\alpha \in [2.5, 13.5]$, and $n \in [1.1, 1.6]$ for four canonical soil types of (a) sand, (b) loam, (c) sandy clay, and (d) clay.

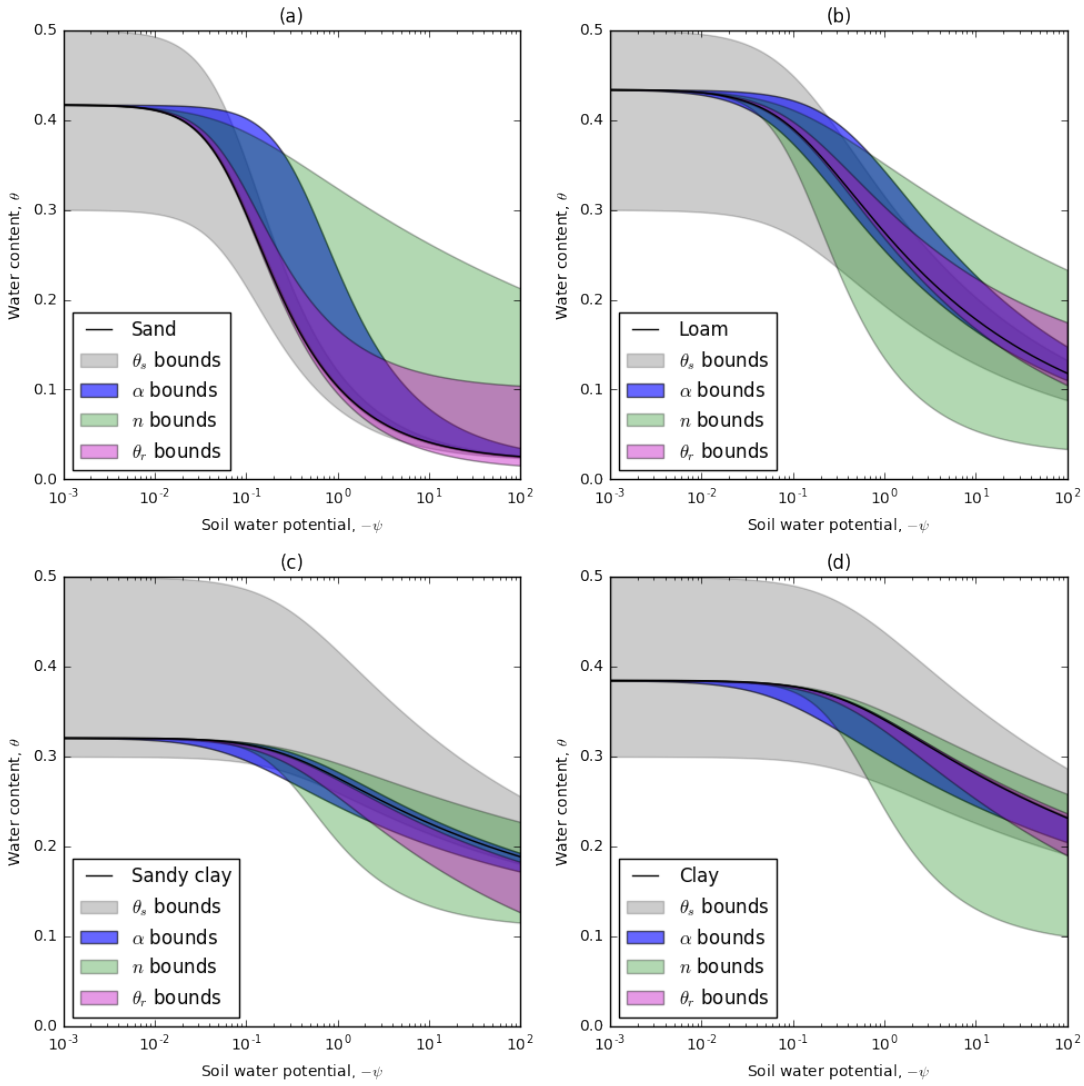

Figure 4.3: The water retention function showing bounds of the various parameters $\theta_s \in [0.3, 0.5]$, $\theta_r \in [0.01, 0.1]$, $\alpha \in [2.5, 13.5]$, and $n \in [1.1, 1.6]$ for four canonical soil types of (a) sand, (b) loam, (c) sandy clay, and (d) clay.

4.2.1 Objective functions

To further explore the van Genuchten parameterizations, we will use a homogeneous sandy clay loam (K_s: 1.2-06 m/s, α: 3.6 1/m, ^{n}: 1.25, θ_r: 0.068, θ_s: 0.33) in a small infiltration experiment. The boundary conditions on the top of a 40 cm deep soil profile

are $\psi = -5$cm at $z = 0$cm and $\psi = -41.5$cm at $z = -40$cm. The Richards equation simulation is run for a time period of 22 hours with an exponentially increasing time step from 5 s to 60 s. Saturation data is collected at nine equally spaced locations from -2 cm to -34 cm. The water content data at each of the nine locations is sampled 23 times over the length of the experiment. This sampling differs from traditionally collected laboratory experiment data, which records water outflow, distributed pressure measurements (using tensiometers), or bulk soil moisture. The analysis here is completed with the assumption that distributed estimates of soil moisture are available from a geophysical method.

In the following sections, we will be estimating the van Genuchten parameters ($K_s, \theta_r, \theta_s, \alpha$, and n) from this water content data. In this section, we will visualize the objective function, l_2 norm of the data difference, around the true model to get a sense of the optimization problem for a homogeneous soil with this data. Even with a homogeneous soil, the objective function is in five-dimensional space, so both visualization and identification of local minima is difficult. Figure 4.4 shows ten profiles through the objective function. These profiles are created by varying two of the parameters while keeping all other parameters constant at the true values. A 40×40 grid, over reasonable bounds of each parameter, requires 16,000 simulations and produces ten cross-sections of the five-dimensional space. The cross-sections of $\theta_s - K_s$, $n - K_s$, $n - \theta_r$, and $n - \alpha$ show well-defined objective functions that are convex along these planes. The structure of the objective functions for α and n, with respect to θ_s, shows a more elongated objective function with contours nearly perpendicular to the θ_s axis, which means that, while keeping θ_s constant in Figures 4.4h and 4.4i, both α and n can vary over their respective ranges at similar objective function values. In Figure 4.4c, α can also change while keeping K_s constant at similar objective function values. Mawer

et al. (2013) also notes low sensitivity to the α parameter when inverting saturation estimates for homogeneous layers. In this case, we also see the contours of the $\theta_r - \theta_s$ objective function as perpendicular to the θ_s axis, which indicates low sensitivity to θ_r in this plane of the objective function (Figure 4.4e). In all cases, when θ_s is involved, except for the cross-section of $\theta_s - K_s$, the objective function is elongated perpendicular to the θ_s axis. This perpendicular elongation means that there is high sensitivity to θ_s, which is not surprising given that our data is water content. Additionally, the boundary conditions of the experiment yield pressure head values in the entire domain between 10^{-2}m and 10^{0}m, which is in the domain where θ_s has the greatest influence over the shape of the water content response to pressure head (Figure 4.3).

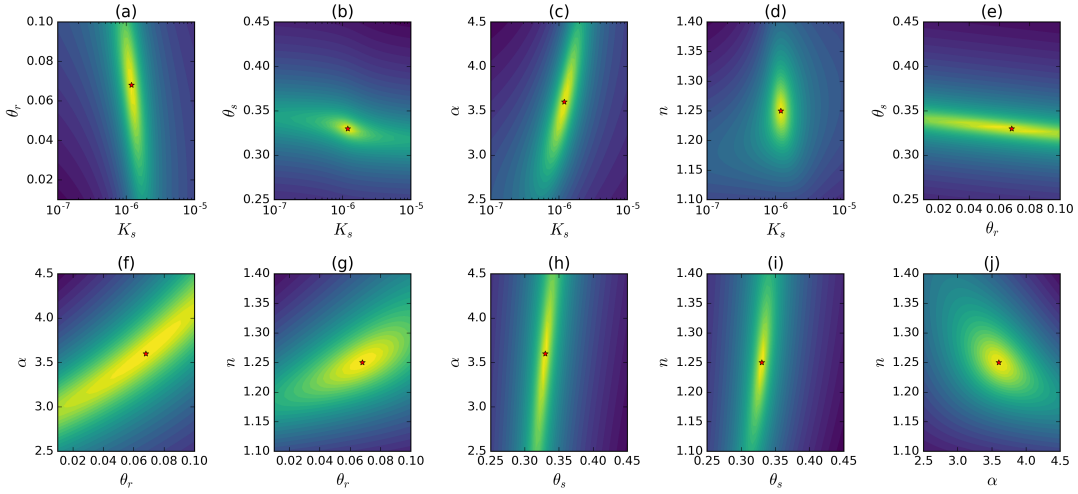

Figure 4.4: Objective function cross sections plotted for all ten cross sections through the five dimensional space of $K_s, \theta_r, \theta_s, \alpha$ and n. Each cross section was simulated with 40×40 simulations and compared using an l_2 data objective function. The results are shown in log_{10}-scale.

This analysis of the objective function shows that all of the cross-sections around the global minima are convex (albeit highly elongated in some axes). If local minima

exist, these are not located on cross sections through the true soil parameters. However, as seen in Figure 4.2 and 4.3, at a single soil water potential, a change in any parameter can raise or lower the functions for hydraulic conductivity and water content. If only a small portion of each curve is examined by the experimental setup (i.e. boundary and initial conditions and measurement locations), then the recovery of these parameters will be non-unique. In the following sections, we will release the assumptions of a homogeneous soil and investigate the recovery of distributed soil parameters in a one-dimensional soil profile.

4.3 Layered soil profile

In this section, we will investigate our ability to recover the van Genuchten parameters of a layered soil. The general setup of this experiment continues to expand on the 40 cm deep soil profile examined in the previous section. In this case, however, we break up the soil profile into three layers: (1) silt loam from 0 cm to -15 cm; (2) loam from -15 cm to -25 cm; and, (3) sandy clay loam from -25 cm to -40 cm. Table 4.1 shows the values for the van Genuchten curves. Figure 4.5 shows the pressure head and water content fields as an image of the 1D soil profile over the length of the experiment. The boundary conditions are set to inhomogeneous Dirichlet with values of -5 cm at the top of the model and -41.5 cm at the bottom of the model. The infiltration front is observed as a pressure increase moving down through the soil profile. The pressure head field is continuous across layer boundaries, as is expected. We can translate the pressure head everywhere in time and space to a saturation profile over time using the known spatially heterogeneous van Genuchten parameters (Figure 4.5b). The most notable difference is the discontinuity between layers, which is mostly due to the differences in θ_s, although the parameters α and n and, to a lesser extent, θ_r, also influence these

discontinuities. This figure shows the infiltration front as an increase in soil water content, which corresponds to the increase in pressure head. We end the experiment when the infiltration front reaches the bottom of the soil profile at 22 hours. The water content figure shows the measurement locations as equally spaced samplings from -2 cm to -34 cm. We took the measurements over the entire time-lapse experiment, as seen in Figure 4.6. There were 225 measurements for the entire experiment. We added 1% noise to the synthetic data, which the figure shows as d_{obs}. This noise is far below what can be expected in any geophysical recovery of the water content; however, by adding more noise, we must conversely reduce our expectations of recovering the true distributions. In this experiment, we are interested in what is possible to recover under the best of circumstances.

4.4 Unconstrained joint inversion

To recover the van Genuchten parameters of $K_s, \theta_r, \theta_s, \alpha$, and n, we will frame the problem as an unconstrained joint inversion for all parameters at once. This framing requires that the model, $\mathbf{m}$, contains all five parameters for every cell in the model and, in this case, has a length of 200. To get the model for hydraulic conductivity, for example, we use a 40×200 projection matrix to select the appropriate rows of the model vector.

$$\mathbf{m}_{K_s} = \mathbf{P}_{K_s}\mathbf{m}$$

We can also complete other model parameterizations at this stage to embed other knowledge (Appendix C). For example, we expect the hydraulic conductivity to be logarithmically varying, so a model of $log(K_s)$ can be created. This mapping, as well

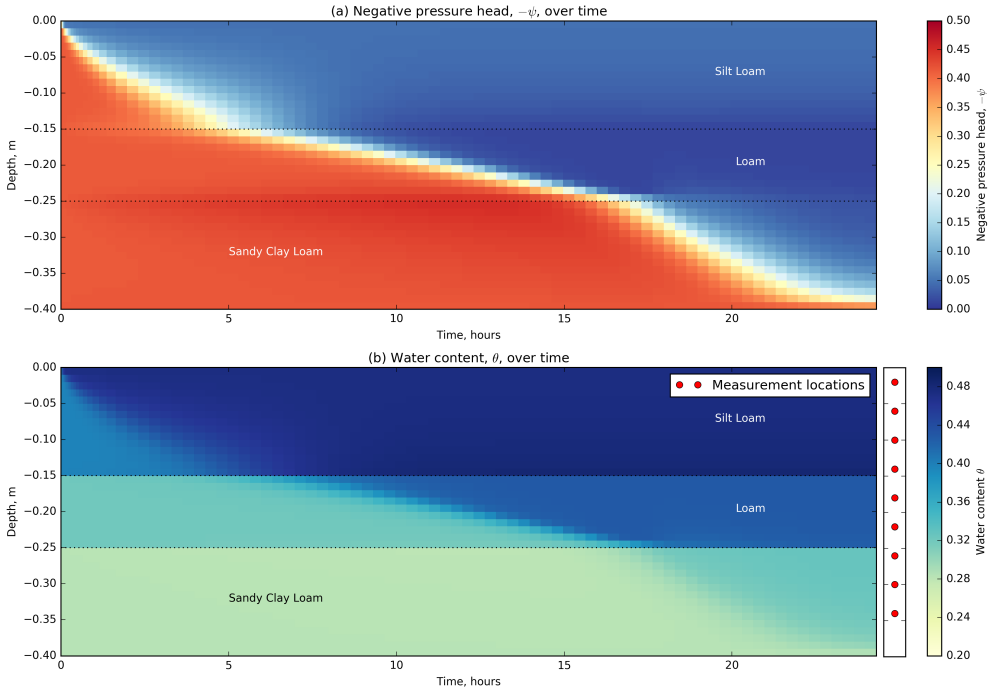

Figure 4.5: Fields from the numerical simulation of a layered one dimensional soil profile, showing (a) pressure head and (b) water content over the full time period. The soil types are shown as annotations on each figure, the spatial location of water content measurements are shown adjacent to the water content fields.

as the projection, must be taken care of in both the translation to the physical property values and the derivative. The entire mapping can be represented as $\mathbf{p} = \mathscr{M}(\mathbf{m})$, where $\mathbf{p}$ is the parameter that is mapped from the model, $\mathbf{m}$. In the previous chapter, we only addressed the model derivative, with respect to saturated hydraulic conductivity. In this chapter, we also require the derivative of the water content curve as well as the derivatives with respect to each parameter in the van Genuchten relationships. The derivative for the water content curve requires attention in both the time stepping terms and in the conversion of the pressure head field to water content for inclusion in the measurement locations.

The inversion for all van Genuchten parameters at once can be written as a sum of weighted objective functions:

$$\begin{aligned}\arg\min_{\mathbf{m}} \ \Phi(\mathbf{m}) = &\frac{1}{2}\left\|\mathbf{W}_d(\mathbf{d}_{\text{pred}}(\mathbf{m}) - \mathbf{d}_{\text{obs}})\right\|_2^2 + \\ &\frac{\beta}{2}\sum_{\{\cdot\}=K_s,\ \alpha,\ n,\ \theta_s,\ \theta_r}\left(\alpha_{\{\cdot\}}\left\|\mathbf{W}_{m_{\{\cdot\}}}(\mathscr{M}_{\{\cdot\}}(\mathbf{m}) - \mathbf{m}_{\text{ref}_{\{\cdot\}}})\right\|_2^2\right)\end{aligned} \tag{4.4}$$

Here, $\mathbf{W}_m$ refers to both model smoothness and smallness for each mapped parameter. The $\mathbf{m}_{\text{ref}}$ can be chosen independently for each parameter. The $\alpha_{\{\cdot\}}$ weightings have significant influence on the inversion; a poor choice of weighting can cause the optimization of the objective function to not converge. We chose the weightings for the inversion results presented by looking at the relative magnitudes of the model resolution matrix ($\mathbf{J}^\top\mathbf{J}$) around the starting model, ($\mathbf{m}_0$). The objective function weights that we used in this inversion were approximately the average sensitivity over the entire soil profile: K_s=1e-3; θ_r=1; θ_s=1; α=1e2; and, n=5e3. The β parameter was chosen by relatively weighting the data misfit and model regularization terms at 1:100 based on a coarse estimate of the major eigenvalue of each inversion component. We used a cooling schedule for β that reduced β by a factor of five every three iterations. The starting model used was a soil with the parameter values of the middle layer of the model: a loam, with the exception of the α parameter, that was set to an initial value of six, which was the midpoint of the values presented in Table 4.1. We optimized the objective function with an inexact Gauss-Newton algorithm that used five conjugate gradient iterations for each step in the inversion. The data misfit function started with a value of 2.8e4, which was reduced two orders of magnitude to the target misfit of 113. The inversion took 117 iterations; however, the majority of the decrease in the objective function occurred in the first 25 iterations of the inversion, which decreased

the misfit to below 200.

4.4.1 Results

Figure 4.6 shows the predicted data for this unconstrained joint inversion for all spatially heterogeneous parameters, which has a good visual fit to the the observed data. This figure shows all of the locations and times in a single figure. There are three saturation measurements in the top and bottom layers and two in the middle layer. Over the course of the infiltration experiment, we can differentiate these saturation profiles by the time at which the infiltration front causes an increase in the water content of the soil.

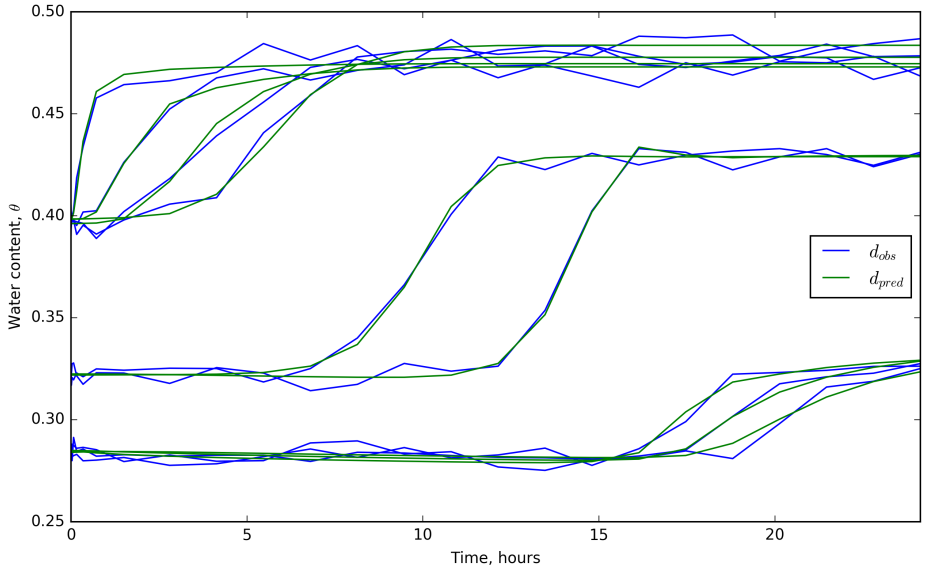

Figure 4.6: Observed and predicted water content data from the one dimensional infiltration experiment showing water content over time.

Figure 4.7 shows the hydraulic conductivity and water retention curves. We allowed each parameter in the van Genuchten curves to vary spatially as shown in Fig-

ure 4.8. However, for visualization purposes, Figure 4.7 uses the average value for each parameter type in each layer to calculate the hydraulic conductivity and water retention curves. The amount of the curve that was interrogated by the infiltration experiments is important to consider. As such, the figure also shows the true pressure head values in the entire layer as a normalized histogram. Information about the curves, outside the bounds of the experiment, will likely be poorly estimated. The curves for hydraulic conductivity in the first two layers were well-recovered for the entire domain of pressure head values. Similarly, the water retention curves were well-recovered over the domain of the experiment, as seen in Figures 4.7d and 4.7e. However, outside the pressure head bounds imposed by the experimental setup, which are shown as histograms, the water retention curves were not fit well for pressure head values less than -1.0 m. The recovered curve over-estimated the water content in the top layer and under-estimated it in the middle layer. The third layer was the last to see the effect of the infiltration event and most of the experiment exposed this layer to approximately -0.5 m of pressure head. The recovered curves for the hydraulic conductivity curve are shown in Figure 4.7c, which overestimates the hydraulic conductivity by nearly an order of magnitude at $-\psi = 10^{-2}$m. Figure 4.7f shows the predicted curve for water retention. Here, the water content is over-estimated at $-\psi = 10^{-2}$m and under-estimated at $-\psi = 10^{2}$m. The location where the layer was sampled was well-recovered and this is seen as the intercept between the true and the predicted models at $-\psi = 0.5$m, which is the peak of the histogram. In summary, the curves were relatively well-recovered in the first two layers, where the pressure head changed by an order of magnitude over the course of the infiltration experiment. Outside these bounds, the water retention curves were less well-recovered, but the hydraulic conductivity showed a good match. The third layer, however, was not matched well and, outside the bounds of the experiment,

the curves for both hydraulic conductivity and water retention were poorly-recovered.

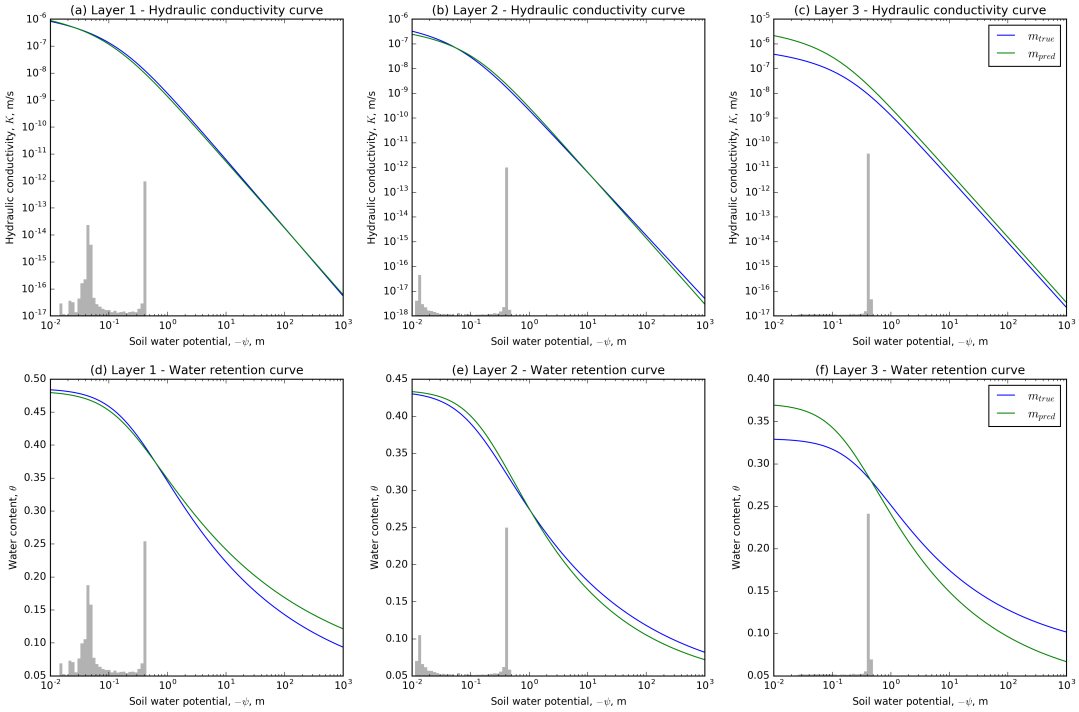

Figure 4.7: Showing the water retention and hydraulic conductivity curves for the true and predicted models for the three soil layers. The histogram in each plow shows the distribution of true pressure head values in each layer.

Figure 4.8 shows the spatially varying van Genuchten parameters through depth. Blue marks the true model parameters and green marks the recovered model parameters at the target misfit. The figure also shows all of the model iterations as thin grey lines of increasing opacity. In Figure 4.8b, the recovered values for θ_s appear to fit in the first few iterations of the inversion. The saturated water content in the top layer is recovered to within 0.012 of the true value; the second layer was also well-recovered, with a maximum difference from the true value of 0.016 for the values inside the layer. The saturated water content in the third layer over-estimated the true value by 0.046. The residual water content for all three layers did not move far from the initial esti-

mate and did not recover the difference in the final layer, but underestimated the third layer by 0.040. The majority of the experiment exposed the soil profile to pressures that were between $-\psi = 10^{-2}$m and $-\psi = 10^{0}$m, which is outside the domain where θ_r influences the water retention curve, as seen in Figure 4.3. In Figure 4.8d, the α parameter did not change from the initial chosen value of 6.0, except in the loam layer where there is a slight increase towards the known value of 9.0. This can be expected from the analysis of the objective functions in Figure 4.4, where α could change over a large range without changing the objective function. Figure 4.8a shows the saturated hydraulic conductivity estimate for the soil profile. The K_s estimate shows the layering of the soil profile, which can lead to other parameterization techniques, as we will see in the next section. The top layer is overestimated by up to a 0.34 in log_{10}-space, the second layer is underestimated by 0.47 in log_{10}-space, and the third layer overestimates K_s by up to an order of magnitude, 0.98 in log_{10}-space. The n parameter is underestimated in the top layer by a maximum of 0.030, in the second layer n is overestimated by a maximum of 0.072, and is slightly overestimated by 0.019 in the third layer.

The parameters estimated by this method demonstrate that the data can be fit by a local minima; that is, multiple values of $K_s - n - \alpha$ can sufficiently minimize the water content data provided. Figure 4.9 further highlights this through the plotting of the fields from the recovered model. As expected, the water content field matches the true fields in Figure 4.5, which contains the data that was recorded and fit. There are some oscillations in the top layer due to the estimate of θ_s; these were added in the final iterations of the optimization and could be considered over-fitting of the inversion. We constructed the data from linear interpolation of the closest two cells; the sensitivity to model parameters outside of this interpolation is up to three orders of magnitude

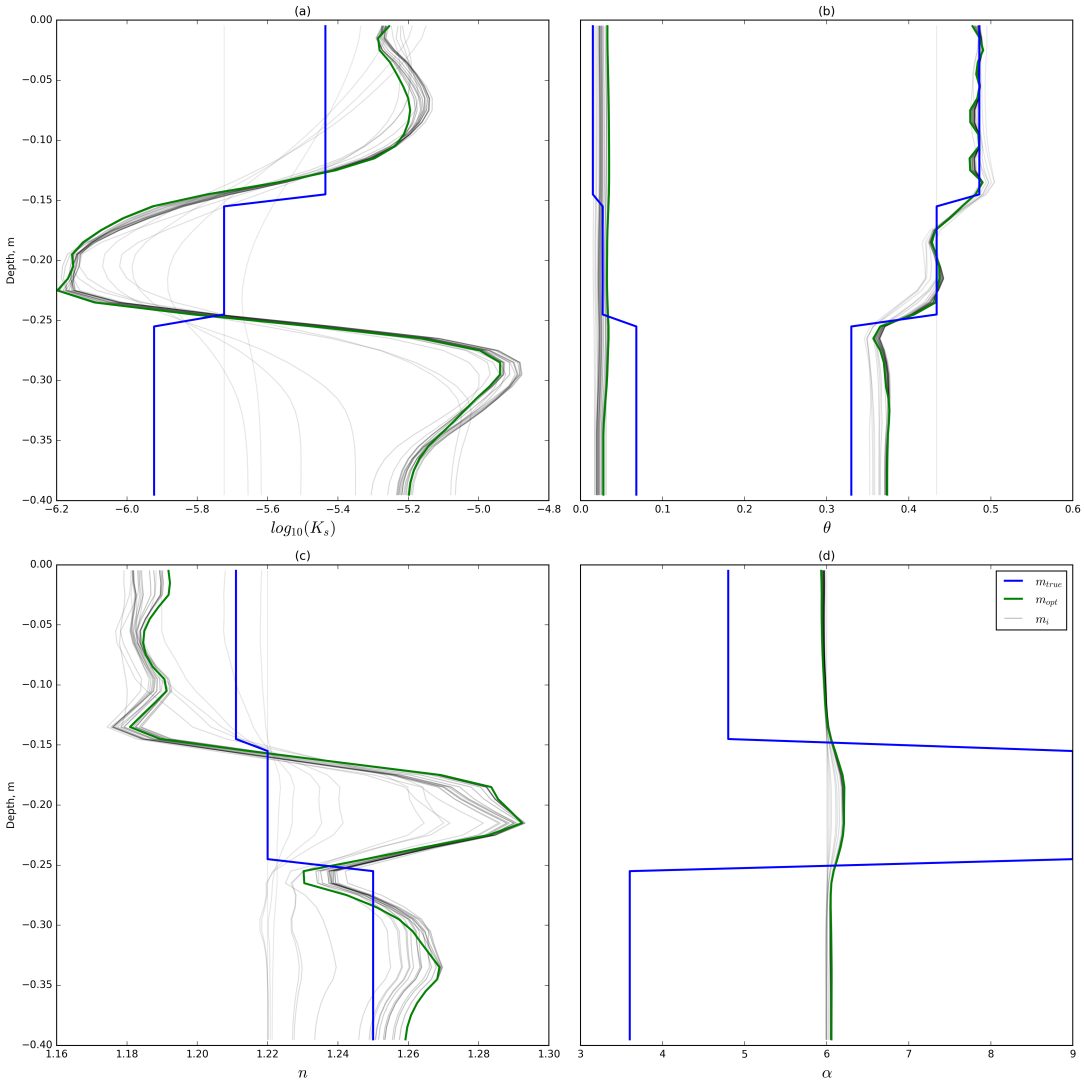

Figure 4.8: The true and recovered soil parameters as a function of depth, showing (a) hydraulic conductivity; (b) residual and saturated water content; (c) the empirical parameter n; and (d) the empirical parameter α. Each plot also shows the full inversion history of the predicted model as a transparent black line.

lower. When considering a geophysical estimation of water content, the 'footprint' of this estimate should be experimented with because the geophysical estimate is an integration over a certain volume of the soil rather than a point estimate, as is assumed here. Comparing to the true fields for the third layer, the water content profile is well-estimated; however, the pressure head recovery has a completely different character. The middle of the bottom layer reaches a maximum pressure head of -16 cm rather than -27 cm in the true model. Considering that the pressure head range of the experiment was 36.5 cm, this estimate was off by 30.1%; this further shows that only collecting saturation data, especially over a small range of pressure heads, can lead to inaccurate results in both the van Genuchten parameters recovered and the pressure head field.

4.4.2 Discussion

The recovery of θ_s is the most robust in the infiltration experiment considered because the majority of the data was collected when the soil was close to saturation. At these pressure head values, θ_s has the greatest control over the van Genuchten water retention curve. We can recover the layering in the system from the saturation data, which can lead to other parameterizations of the model space and exploration of other *a priori* data to be included. The hydraulic conductivity curves for the first two layers were well-recovered within half an order of magnitude. However, there is a trade-off between K_s and n, which could not be isolated over the small pressure ranges that we used for this simulation. We found low sensitivity to α over the range of pressure heads investigated, as Mawer et al. (2013) has previously observed. We found a local minima across the K_s, n, and α parameters. The coverage of a large range (several orders of magnitude) of pressure head values is important for extrapolating the hydraulic conductivity curve and, especially, the water retention curve. Most field studies that

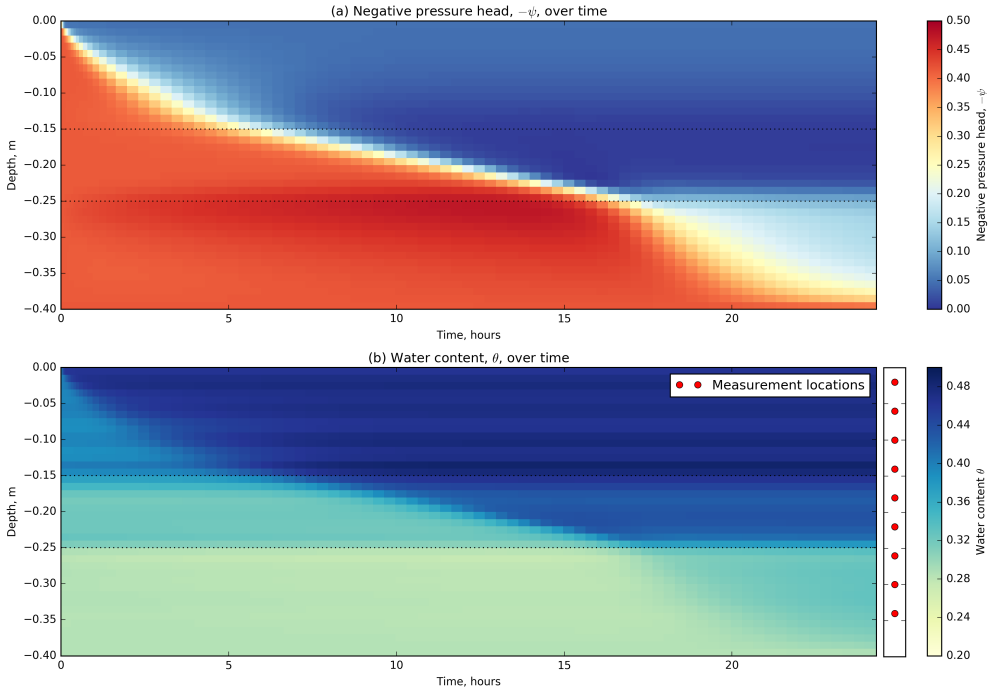

Figure 4.9: Recovered simulation fields from the unconstrained joint inversion for van Genuchten parameters. Showing (a) the pressure head and (b) the water content over the full time period of the one dimensional recovered soil profile. The soil types are shown as annotations on each figure, the spatial location of water content measurements are shown adjacent to the water content fields.

use geophysical data to estimate water content will likely see changes of only a few orders of magnitude. However, if only a small portion of the curve is interrogated in the experiment, data can be fit with incorrect van Genuchten parameters. The domain of the curves that is interrogated should be reported with any estimate of van Genuchten parameters. Furthermore, the addition of pressure head data will be advantageous to the joint inversion presented, as it will reduce non-uniqueness in the water retention model for estimating the pressure head field.

4.5 Three dimensional inversion

In this section, we turn our attention to recovering a three-dimensional soil structure given water content data. The example, motivated by a field experiment introduced in (Pidlisecky et al., 2013), shows a time-lapse electrical resistivity tomography survey completed in the base of a managed aquifer recharge pond. The goal of this management practice is to infiltrate water into the subsurface for storage and subsequent recovery. Such projects require input from geology, hydrology, and geophysics to map the hydrostratigraphy, to collect and interpret time-lapse geophysical measurements, and to integrate all results to make predictions and decisions about fluid movement at the site. As such, the hydraulic properties of the aquifer are important to characterize, and information from hydrogeophysical investigations has been demonstrated to inform management practices (Pidlisecky et al., 2013). We use this context to motivate both the model domain setup of the following synthetic experiment and the subsequent inversion.

For the inverse problem solved here, we assume that time-lapse water-content information is available at many locations in the subsurface. In reality, water content information may only be available through proxy techniques, such as electrical resistivity methods. These proxy data can be related to hydrogeologic parameters using inversion techniques based solely on the geophysical inputs (cf. Mawer et al. (2013)). For the following numerical experiments, we do not address complications in empirical transformations, such as Archie's equation (Archie, 1942). The synthetic numerical model has a domain with dimensions 2.0 m $\times$ 2.0 m $\times$ 1.7 m for the x, y, and z dimensions, respectively. The finest discretization used is 4 cm in each direction. We use padding cells to extend the domain of the model (to reduce the effect of boundary

conditions in the modelling results). These padding cells extend at a factor of 1.1 in the negative z direction. We use an exponentially expanding time discretization with 40 time steps and a total time of 12.3 hours. This choice in discretization leads to a mesh with 1.125×10^5 cells in space ($50\times50\times45$). To create a three-dimensionally varying soil structure, we construct a model for this domain using a three-dimensional, uniformly random field, $\in[0,2]$, that is convolved with an anisotropic smoothing kernel for a number of iterations. We create a binary distribution from this random field by splitting the values above and below unity. Figure 4.10 shows the resulting model, which reveals potential flow paths. We then map van Genuchten parameters to this synthetic model as either a sand or a loamy-sand. The van Genuchten parameters for sand are: K_s: 5.83e-05m/s, α: 13.8, θ_r: 0.02, θ_s: 0.417, and n: 1.592; and for loamy-sand are: K_s: 1.69e-05m/s, α: 11.5, θ_r: 0.035, θ_s: 0.401, and n: 1.474. For this inversion, we are interested in characterizing the soil in three dimensions.

4.5.1 Results

For calculation of synthetic data, the initial conditions are a dry soil with a homogeneous pressure head ($\psi=-30$cm). The boundary conditions applied simulate an infiltration front applied at the top of the model, $\psi=-10\text{cm}\in\delta\Omega^{\text{top}}$. Neumann (no-flux) boundary conditions are used on the sides of the model domain. Figure 4.13 shows the pressure head and water content fields from the forward simulation. Figure 4.11a and 4.11b show two cross-sections at time 5.2 hours and 10.3 hours of the pressure head field. These figures show true soil type model as an outline, where the inclusions are the less hydraulically-conductive loamy sand. The pressure head field is continuous across soil type boundaries and shows the infiltration moving vertically down in the soil column. We can compute the water content field from the pressure

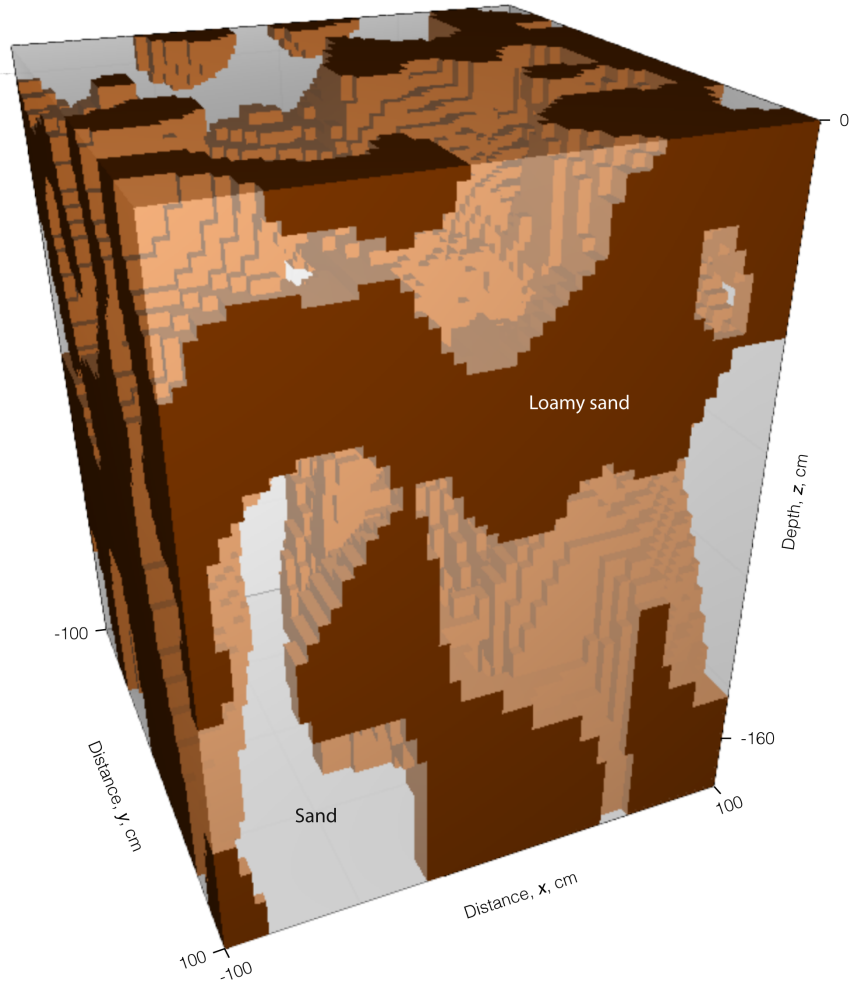

Figure 4.10: Soil structure in three dimensions showing the boundary between two soil types of sand and loamy sand.

head field using the nonlinear van Genuchten model chosen; Figure 4.11c and 4.11d show this computation at the same times. The loamy sand has a higher relative water content for the same pressure head and the water content field is discontinuous across soil type boundaries, as expected.

The observed data, which will be used for the inversion, is collected from the water content field at the points indicated in both Figure 4.11c and 4.11d. The sampling

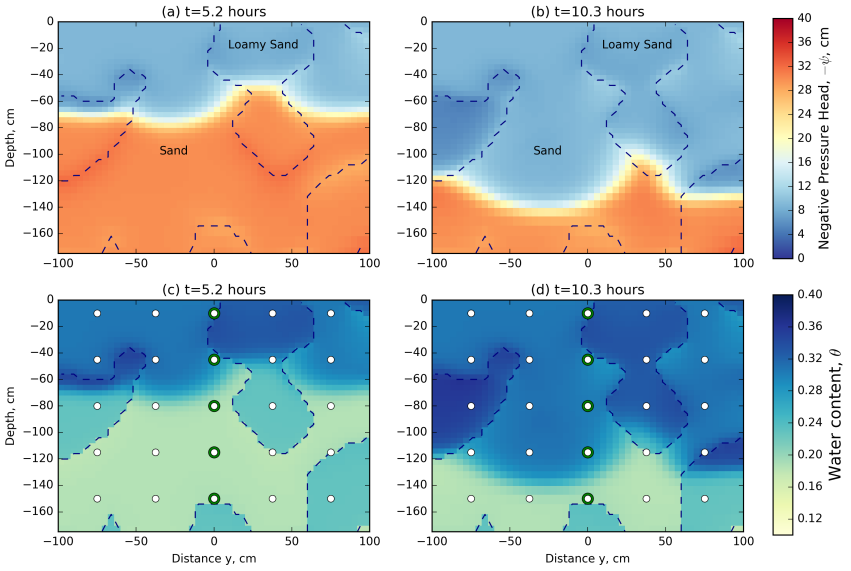

Figure 4.11: Vertical cross sections through the pressure head and saturation fields from the numerical simulation at two times: (a) pressure head field at $t = 5.2$ hours and (b) $t = 10.3$ hours; and (c) saturation field at $t = 5.2$ hours and (d) $t = 10.3$ hours. The saturation field plots also show measurement locations and green highlighted regions that are shown in Figure 4.12. The true location of the two soils used are shown with a dashed outline.

location and density of this three-dimensional grid within the model domain is similar to the resolution of a 3D electrical resistivity survey. Our implementation supports data as either water content or pressure head; however, proxy water content data is more realistic in this context. Similar to the field example in (Pidlisecky et al., 2013), we collect water content data every 18 minutes over the entire simulation, leading to a total of 5000 spatially and temporally extensive measurements. The observed water content data for a single infiltration curve is plotted through depth in Figure 4.12. The green circles in Figure 4.11 show the locations of these water content measurements.

The depth of the observation is colour-coded by depth, with the shallow measurements being first to increase in water content over the course of the infiltration experiment. To create the observed dataset, $\mathbf{d}_{\text{obs}}$, from the synthetic water content field, 1% Gaussian noise is added to the true water content field. This noise is below what can currently be expected from a proxy geophysical measurement of the water content. However, with the addition of more noise, we must reduce our expectations of our ability to recover the true parameter distributions from the data. In this experiment, we are interested in examining what is possible to recover under the best of circumstances, and therefore have selected a low noise level.

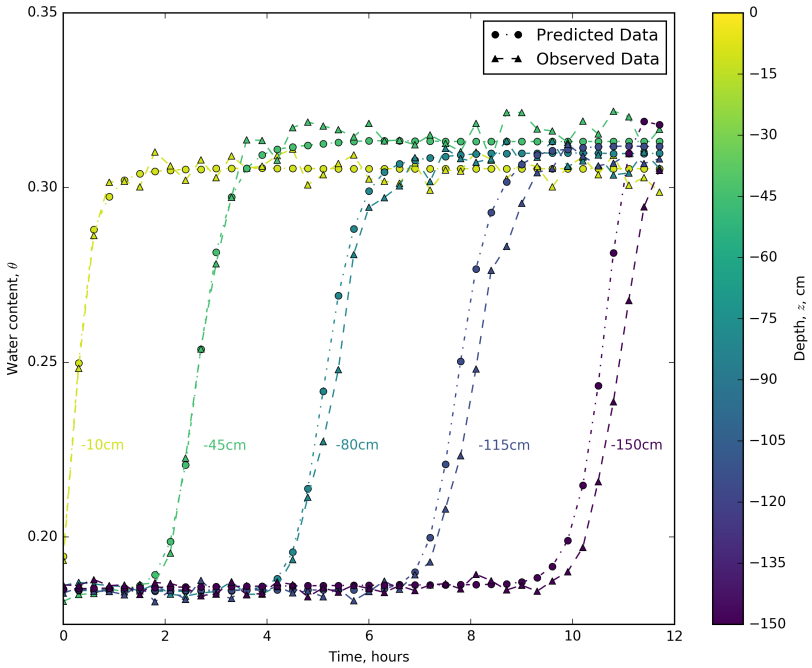

Figure 4.12: Observed and predicted data for five measurements locations at depths from 10cm to 150cm from the center of the model domain.

For the inverse problem, we are interested in the distribution of soil types that fits the measured data. We parameterize these soil types using the van Genuchten empirical model (4.2) with as least five spatially distributed properties. Inverting for all 5.625×10^5 parameters in this simulation with only 5000 data points is a highly underdetermined problem, and thus there are many possible models that may fit those data. In this 3D example, we will invert solely for saturated hydraulic conductivity and assume that all other van Genuchten parameters are equivalent to the sand; that is, they are parameterized to the *incorrect values* in the loamy sand. Note that this assumption, while reasonable in practice, will handicap the results, as the van Genuchten curves between these two soils differ. Better results can, of course, be obtained if we assume that the van Genuchten parameters are known; this assumption is unrealistic in practice, which means that we will not be able to recreate the data exactly. However, the distribution of saturated hydraulic conductivity may lead to insights about soil distributions in the subsurface. Figure 4.13 shows the results of the inversion for saturated hydraulic conductivity as a map view slice at 66 cm depth and two vertical sections through the center of the model domain. The recovered model shows good correlation to the true distribution of hydraulic properties, which is superimposed as a dashed outline. Figure 4.12 shows the predicted data overlaid on the true data for five water content measurement points through time; these data are from the center of the model domain and are colour-coded by depth. As seen in Figure 4.12, we do a good job of fitting the majority of the data. However, there is a tendency for the predicted infiltration front to arrive before the observed data, which is especially noticeable at deeper sampling locations. The assumptions put on all other van Genuchten parameters to act as sand, rather than loamy sand, lead to this result.

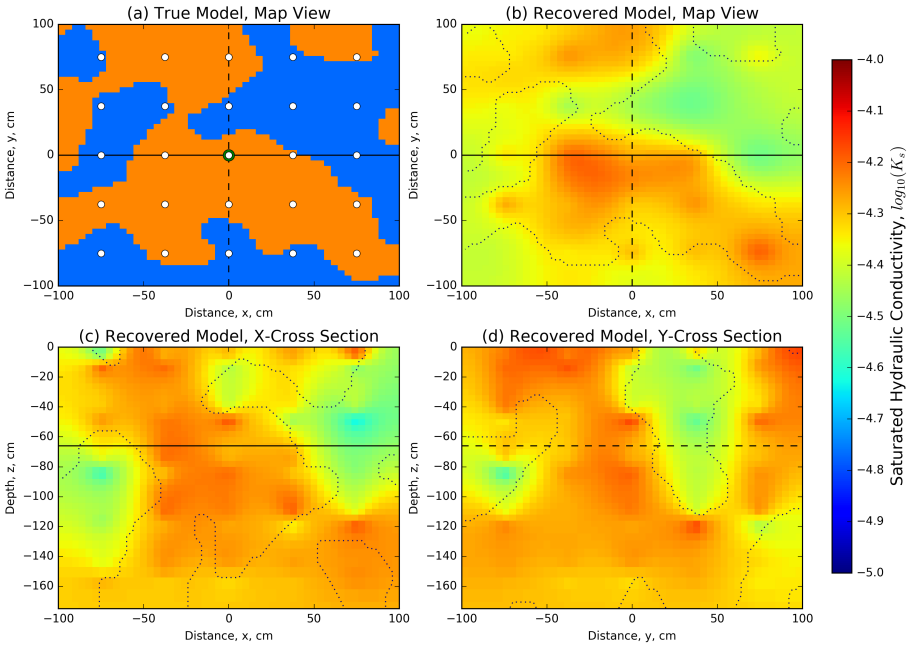

Figure 4.13: The 3D distributed saturated hydraulic conductivity model recovered from the inversion compared to the (a) synthetic model map view section, using (b) the same map view section, (c) an X-Z cross section and (d) a Y-Z cross-section. The synthetic model is show as an outline on all sections, and tie lines are show on all sections as solid and dashed lines, all location measurements are in centimeters.

4.5.2 Scalability of the implicit sensitivity

For the forward simulation presented, the Newton root-finding algorithm took 4-12 iterations to converge to a tolerance of 1×10^{-4}m on the pressure head. The inverse problem took 20 iterations of inexact Gauss-Newton with five internal CG iterations used at each iteration. The inversion fell below the target misfit of 5000 at iteration 20 with $\phi_d = 4.893 \times 10^3$; this led to a total of 222 calls to functions to solve the products $\mathbf{Jv}$ and $\mathbf{J}^\top\mathbf{z}$. Here, we again note that the Jacobian is neither computed nor stored directly, which makes it possible to run this code on modest computational resources; this is not possible if numerical differentiation or direct computation of the Jacobian is

used. For these experiments, we used a single Linux Debian Node on Google Compute Engine (Intel Sandy Bridge, 16 vCPU, 14.4 GB memory) to run the simulations and inversion. The forward problem takes approximately 40 minutes to solve. In this simulation, the dense Jacobian matrix would have 562.5 million elements. If we used a finite difference algorithm to explicitly calculate each of the 1.125e5 columns of the Jacobian with a simple forward difference, we would require a calculation for each model parameter – or approximately 8.5 years of computational time. Furthermore, we would need to recompute the Jacobian at each iteration of the optimization algorithm. In contrast, if we use the implicit sensitivity algorithm presented in Chapter 3, we can solve the entire inverse problem in 34.5 hours.

Table 4.2: Comparison of the memory necessary for storing the dense explicit sensitivity matrix compared to the peak memory used for the implicit sensitivity calculation excluding the matrix solve. The calculations are completed on a variety of mesh sizes for a single distributed parameter (K_s) as well as for five distributed van Genuchten model parameters (K_s, α, n, θ_r, and θ_s). Values are reported in gigabytes (GB).

	Explicit Sensitivity		**Implicit Sensitivity**	
Mesh Size	1 parameter	5 parameters	1 parameter	5 parameters
$32 \times 32 \times 32$	1.31	6.55	0.136	0.171
$64 \times 64 \times 64$	10.5	52.4	0.522	0.772
$128 \times 128 \times 128$	83.9	419	3.54	4.09

Table 4.2 shows the memory required to store the explicit sensitivity matrix for a number of mesh sizes and contrasts them to the memory required to multiply the implicit sensitivity by a vector. These calculations are modifications on the example presented above and use 5000 data points. The memory requirements are calculated for a single distributed parameter (K_s) as well as five spatially distributed parameters (K_s, α, n, θ_r, or θ_s). Neither calculation includes the memory required to solve the matrix system, as such, the reported numbers underestimate the actual memory re-

quirements for solving the inverse problem. The aim of this comparison is to demonstrate how the memory requirements scale, an appropriate solver must also be chosen for either method to solve the forward problem. When using an explicitly calculated sensitivity matrix to invert for additional physical properties, the memory footprint increases proportionally to the number of distributed physical properties; this is not the case for the implicit sensitivity calculation. For example, on a $128 \times 128 \times 128$ mesh, the explicit formation of the sensitivity requires 419 GB for five spatially distributed model parameters, which is five times the requirement for a single distributed model parameter (83.9 GB). For the implicit sensitivity on the same mesh, only 4.09 GB of memory is required, which is 1.2 times the requirement for a single distributed model parameter (3.54 GB). For this mesh, inverting for five spatially distributed parameters requires over 100 times less memory when using the implicit sensitivity algorithm, allowing these calculations to be run on modest computational resources.

4.5.3 Discussion

In the context of managed aquifer recharge, small variations in soil types can cause differences in infiltration rates. Geophysical data that is both spatially and temporally dense can act as a proxy for water content (cf. Pidlisecky et al. (2013)). If hydraulic properties are identified or even spatially delineated, this knowledge can inform and influence management decisions (e.g. where and when to till or dredge a pond to increase infiltration rates). The hydraulic properties determining these infiltration rates are distributed in three dimensions. The inversion presented above demonstrated a large-scale inversion for a 3D distribution of saturated hydraulic conductivity. We fixed the other van Genuchten parameters at values for sand and, as such, the inversion had difficulty fitting late time data in the deeper profile. The inversion qualitatively dis-

criminated between the two geologic units. In this experimentation, we assumed that the initial conditions and boundary conditions were known. This knowledge may be possible in a heavily monitored situation, such as managed aquifer recharge where the pond water height and underlying water table are measured. Further investigation into the conceptualization of the groundwater simulation is important to both the hydrogeologic modelling and any coupling to geophysical methods (Hinnell et al., 2010). The inversion at this scale is computationally possible due to the formulation of the Richards equation presented in Chapter 3 and the implementation both extended and tested the geophysical inversion framework presented in Chapter 2.

4.6 Integrations

The combination of the Richards equation with geophysical responses has been laid out in Hinnell et al. (2010); this includes both directly coupled and uncoupled forms of integrating information. As presented in Hinnell et al. (2010), uncoupled integrations are completed by: (a) using the geophysical data to estimate a physical property, such as electrical conductivity; (b) using an empirical relation, such as Archie's equation, to transform the geophysical estimate into a hydrological parameter, such as water content; and, (c) using hydrological estimates to help inform or test a hydrogeologic simulation. In contrast, a coupled inversion uses geophysical data to directly inform the hydrogeologic simulation through stochastic or deterministic parameter estimation (Finsterle and Kowalsky, 2008; Ferré et al., 2009). We can find increasing instances of these sorts of collaborations and studies in near surface hydrogeophysics (cf. Linde and Doetsch (2016) and references within). As data sizes and numerical domains continue to grow, the relatively few parameters that can be estimated by stochastic inversions may not be sufficient. Alternatively, deterministic inversions can be used, but

will need to draw on improvements across the field of geophysical inverse problems. For example, regularization techniques using fuzzy c-means clustering, developed in other areas of geophysics, have potential to be helpful in introducing known parameter distributions into the Richards equation inversion (Paasche and Tronicke, 2007; Sun et al., 2012). In Hinnell et al. (2010), the authors conclude that, "the coupled approach [for hydrogeophysics] requires that the hydrologic and geophysical models be merged, [which] forces the hydrologist and the geophysicist to formulate a consistent framework," which would require "an uncommon level of collaboration during scientific analysis".

A consistent framework was proposed in Chapter 2; however, to both advance leading research and disseminate leading practice, the integration and arbitrary combination of physical simulations must also be possible while simultaneously calculating derivatives efficiently. Appendix C presents a brief introduction to this collaborative work, as well as several case studies that are striving towards a general formulation. The framework presented in Chapter 2 has been extended to: (a) explicitly expose assumptions in the forward simulation framework to interrogation and inversion; (b) compose custom objective functions, including regularization and multi-physics data objective functions; and, (c) provide extensible parameterizations that are flexible for custom inclusion of *a priori* knowledge. This is ongoing, collaborative work.

4.7 Conclusions

The joint inversion of various hydraulic parameters was explored on a layered 1D soil profile. The water content data was well fit and the soil layers were delineated. Additionally, the van Genuchten curves that were used as the empirical relations were also well recovered over the range of pressure head values that each layer of the soil

profile was exposed to over the experiment. Outside of these ranges, the curves were not reliably recovered. As such, the numeric values of the van Genuchten parameters were occasionally unreliable, especially if pressure head was relatively constant, as was the case in the third layer. In this experiment, the water content data was considered as a point measurement, and the sensitivity of the recovered model varied by orders of magnitude between voxels that were included in that measurement and immediate voxel neighbours. This lead to some artificial layering in the recovered model which was not compensated for by the regularization. The footprint of the water content measurement is likely an integration over a number of voxels, especially if coming from a geophysical estimate, and should be considered and experimented with.

As geophysics is more regularly included in hydrogeology parameter estimation the number of distributed parameters that will be necessary to estimate will increase by several orders of magnitude. The final example in this chapter showed a 3D recovery of a distributed hydraulic parameter (K_s). This inversion would not have been possible through standard finite difference techniques and was significantly more memory efficient than an automatic differentiation algorithm that explicitly forms the Jacobian (up to two orders of magnitude in some cases). These improvements allowed the Richards equation inversion to be run on modest computational resources. The coupling, nesting or otherwise integrating of various geophysical methods with the Richards equation is an obvious extension to this work. This has been extensively explored by several authors, albeit at a smaller scale (cf. Linde and Doetsch (2016) and references within). Appendix C briefly explores various types of integrations between various geophysical methodologies as well as custom model conceptualizations that will be necessary for these integrations.

Chapter 5

Conclusions

Forward and inverse modelling in geophysics requires solving and optimizing large-scale partial differential equations. Thus, many components including linear algebra, optimization routines, discretizations, and model conceptualizations, are required to interact. Advances in instrumentation, the monitoring of time-lapse processes, and acquisition of multiple geophysical and hydrologic data types are driving a need for a more integrated geoscience approach. This includes integrating information through multiphysics simulations and coupled geophysical inversions. In addition to recovering 3D models, geophysics is increasingly being used in time-lapse imaging of fluid flow processes, requiring both computational scalability to 4D inverse problems and interdisciplinary collaborations. There have been significant advances in the past three decades in computational geophysics; researchers are now able to both simulate and invert almost any geophysical method in three dimensions (cf. Oldenburg (2016)). One of the major challenges ahead of geophysics as a discipline is how to systematically improve the quantitative interfaces and integrations between hydrological, geophysical, and geological information and processes. The research necessary to address these

challenges will require the interdisciplinary community to build upon, as well as augment, standard practices; this presupposes that researchers have access to consistent methodologies that can be extended, adapted, and combined.

Adapting interdisciplinary methodologies to geophysical simulations and inversions inherently requires that a diverse suite of methods and applications be considered across hydrogeology, geophysics, and geology. Throughout this work, my colleagues and I have tried to summarize and/or reproduce many methodologies in the geophysical inversion literature. Other communities, such as astrophysics and machine learning (Astropy Collaboration et al., 2013; Pedregosa et al., 2011) have organized these efforts and research communities around open, accessible, and actionable ideas. Adapting these learnings to geoscience, I strive to complete all of my research such that it is immediately reproducible and openly available.

Much of my dissertation required the development of software, which is implementation and engineering by its nature. However, the software, although extremely useful, is not the aim of my research. If the goal is a framework for quantitative geoscience integration, a simplistic, high level conceptualization is easy to present. However, a picture or a paragraph describing a framework cannot be tested, interrogated, nor used beyond its static form. Software is the means by which I test, extend, organize, and abstract inherently computational ideas in a rigorous, scientific way. The aim of my research is to identify, explore, and formalize a framework for simulation and parameter estimation in geophysics. I applied this framework to vadose zone flow and, with the help of collaborators, to several other geophysical methodologies.

5.1 Contributions and dissemination

The conclusions from each component of my thesis are contained within each chapter and each appendix. However, the central aim of my dissertation was to develop a framework for geophysical inversions: this has been disseminated through three publications (Cockett et al., 2017; Heagy et al., 2016; Cockett et al., 2015c), four extended abstracts (Heagy et al., 2014; Kang et al., 2015a; Heagy et al., 2015c, 2017), over twenty conference presentations, and a dedicated international workshop[1]. Furthermore, this framework has demonstrated value in several new research areas, methodologies, and case studies (cf. Kang et al. (2017a); Miller et al. (2017); Kang and Oldenburg (2016); Rosenkjaer et al. (2016)). Overall, the contributions of my thesis are twofold:

1. a conceptual organization and synthesis of geophysical simulations and inversions into a framework that has been rigorously, numerically tested; and

2. an algorithm for large-scale vadose zone parameter estimation for any distributed hydraulic parameter, regardless of the empirical relationship used.

One outcome of a framework approach is the accelerated transfer of ideas from one discipline to another. For example, the implicit sensitivity calculation for the Richards equation (Cockett et al., 2017) was heavily inspired by research completed in time-domain electromagnetics simulations and inversions (Heagy et al., 2016). The refinement and application of this algorithm to hydrology significantly improved numerical scalability for the 3D inverse problem. Chapter 4 showed significant improvements in memory over explicitly forming the sensitivity matrix by over two orders of magnitude

[1] At the Banff International Research Station, see http://www.birs.ca/events/2016/2-day-workshops/16w2695

for the example shown, bringing this inversion into the range of possibility on modest computational resources. Additionally, the complexities of the Richards equation were generalized and synthesized to improve other geophysical methods in the framework. These improvements were especially demonstrated with regard to dealing with multiple physical properties that may or may not be estimated and occur in distributed empirical relations throughout the forward simulation framework.

The framework presented is designed to decouple concerns and expose well-defined interfaces between the many components necessary in geophysical simulations and inversions. Chapter 4 and Appendix C showed many demonstrations of these ideas, for example: (a) exposing model conceptualization that are decoupled from the sensitivity calculation allows custom, parameterized inversions to be completed by combining various, predefined mappings; (b) the declarative interface of differential operators and derivatives is decoupled from the structure and type of mesh, which allows the physics to be written once and used across many types of meshes; or (c) the decoupling of the physics from the definition of the geophysical survey, which allows many types of geophysical surveys to be combined with a single physical problem (e.g. in electromagnetics) or conversely different approximations of a physical problem (e.g. dimensionality) to be combined with a single survey. Given the number of choices in geophysical simulations and inversions this type of combinatorial, decoupled approach could provide significant acceleration to the unique geoscience integration problems of the future.

5.1.1 Software

The conceptual organization developed could not have been created without the aid of software. Furthermore, even if it were, it would be of limited utility, difficult, or

impossible to validate, and would not make significant progress towards sustained, reproducible, quantitative geoscience integrations. It is not until a framework is implemented and tested from a number of non-overlapping geoscience perspectives that the assumptions, inconsistencies, or redundancies come to light and are available to interrogation. The main software package that was, and continues to be, developed is SIMPEG (https://github.com/simpeg/simpeg), which defines the framework and hosts a collection of other geophysical methods written by many collaborators across six universities. Currently there are methods for: vadose zone flow (Cockett et al., 2017), direct current resistivity and induced polarization (Kang and Oldenburg, 2016), time-domain and frequency-domain electromagnetics (Heagy et al., 2016), magnetotellurics (Rosenkjaer et al., 2016), magnetics and gravity (Fournier et al., 2016; Miller et al., 2017), and a number of example linear inverse problems. Many of these geophysical methods also have different formulations (e.g. integral equation, differential equation, etc.), dimensionalities (e.g. 1D, 2.5D, 3D), and survey components (e.g. sources and receivers). All software is disseminated with the MIT license to encourage permissionless innovation.

5.2 Outlook and continuing work

This thesis constructed a preliminary organization and synthesis of simulations and deterministic inversions in a few subdisciplines of geophysics and hydrogeology. This conceptual framework and computational implementation has demonstrated utility in advancing current and future research, however, caveats and qualifiers abound.

Some of the more intricate parallel algorithms (e.g. stochastic optimization) would require updates to the implementation and framework; a lack of these scalable parallel algorithms is limiting when tackling problems with many sources (e.g. airborne elec-

tromagnetics). There is still significant work to do in coupling and nesting various geophysical problems in a robust way. Due to the current focus on enabling researchers, the framework offers little in high-level interfaces to inversions that are typical of *black box* industry use (i.e. data in, model out). Regardless of these shortcomings, many of my colleagues rely upon and extend this framework and implementation in their day to day research. My goal was to accelerate their work and connect their research to a community of collaborators who are explicitly working towards common goals.

This thesis is positioned from a perspective of looking out to a future of multidisciplinary, multi-data-type, quantitatively integrated geoscience simulations and inversions. A future where joint, cooperative, coupled, parameterized, multiphysics inversions and simulations are the norm rather than the exception. Where multiple existing, robust, and computationally-efficient methodologies are combined to extract all possible information from disparate geoscience datasets. To realize this sort of ubiquitous, quantitative communication between disciplines and methodologies requires an organized and integrated community that can effectively work together. My research is *aimed* here. This future will not be realized by one person nor by one research group. In the field of machine learning, Olah and Carter (2017) note that "[t]he maintainable size of the field is controlled by how its members trade off the energy between communicating and understanding." The curation of ideas is just as important as their creation. There is a *research debt* created by an exclusive focus on research novelty; future advances also require distillation, synthesis, and explicit communication. There is a significant amount of effort ahead of us to achieve effective communication and collaboration with our geoscience peers in geology and hydrogeology. This communication and quantitative integration is the webbing on which the future of our *geoscience* field depends. My approach, therefore, has been to research and disseminate

a numerical framework that attempts to support and enable a number of these diverse interdisciplinary collaborations. I hope that a lasting contribution of my work is the open, modular approach that I have curated and the community that I have helped to seed around these ideas.

Bibliography

Alexe, M., Sandu, A., 2014. Space-time adaptive solution of inverse problems with the discrete adjoint method. Journal of Computational Physics 270, 21–39. → pages 171

Archie, G., 1942. The Electrical Resistivity Log as an Aid in Determining Some Reservoir Characteristics. Transactions of the AIME 146 (1), 54–62. → pages 5, 14, 61, 89, 111

Armstrong, R., Gannon, D., Geist, A., Keahey, K., Kohn, S., McInnes, L., Parker, S., 1999. Toward a Common Component Architecture for High Performance Scientific Computing. Proc. 8th IEEE Symp. on High Performance Distributed Computing, 115–124. → pages 141

Ascher, U. M., 2008. Numerical methods for evolutionary differential equations. Society for Industrial and Applied Mathematics. → pages 68

Ascher, U. M., Greif, C., jun 2011. A First Course in Numerical Methods. Society for Industrial and Applied Mathematics, Philadelphia, PA. → pages vi, 69, 144

Astropy Collaboration, Robitaille, T., Tollerud, E., Greenfield, P., Droettboom, M., Bray, E., Aldcroft, T., Davis, M., Ginsburg, A., Price-Whelan, A., Kerzendorf, W., Conley, A., Crighton, N., Barbary, K., Muna, D., Ferguson, H., Grollier, F., Parikh, M., Nair, P., Unther, H., Deil, C., Woillez, J., Conseil, S., Kramer, R., Turner, J., Singer, L., Fox, R., Weaver, B., Zabalza, V., Edwards, Z., Azalee Bostroem, K., Burke, D., Casey, A., Crawford, S., Dencheva, N., Ely, J., Jenness, T., Labrie, K., Lim, P., Pierfederici, F., Pontzen, A., Ptak, A., Refsdal, B., Servillat, M., Streicher, O., oct 2013. Astropy: A community Python package for astronomy. AAP 558, A33. → pages 23, 124, 142

Balay, S., Brown, J., Buschelman, K., Eijkhout, V., Gropp, W., Kaushik, D., Knepley, M., McInnes, L. C., Smith, B., Zhang, H., 2012. PETSc users manual revision 3.3. Computer Science Division, Argonne National Laboratory, Argonne, IL. → pages 48

Bard, J. B. L., Rhee, S. Y., 2004. Ontologies in biology: design, applications and future challenges. Nature reviews. Genetics 5 (3), 213–222. → pages 141

Barrett, R., Berry, M., Chan, T. F., Demmel, J., and J. Donato, Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., der Vorst, H. V., 1994. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. SIAM, Philadelphia. → pages 73

Binley, A., Cassiani, G., Middleton, R., Winship, P., 2002. Vadose zone flow model parameterisation using cross-borehole radar and resistivity imaging. Journal of Hydrology 267 (3), 147–159. → pages 5, 14, 61, 62, 89

Bitterlich, S., Durner, W., Iden, S. C., Knabner, P., aug 2004. Inverse Estimation of the Unsaturated Soil Hydraulic Properties from Column Outflow Experiments Using Free-Form Parameterizations. Vadose Zone Journal 3 (3), 971–981. → pages 62, 95

Bitterlich, S., Knabner, P., 2002. An efficient method for solving an inverse problem for the Richards equation. Journal of Computational and Applied Mathematics 147, 153–173. → pages 15, 63, 77, 79, 92

Brooks, R. H., Corey, A. T., 1964. Hydraulic properties of porous media and their relation to drainage design. Trans. ASAE 7.1 26 (28). → pages 66, 92

Bui-Thanh, T., Ghattas, O., jan 2015. A scalable algorithm for MAP estimators in Bayesian inverse problems with Besov priors. Inverse Problems and Imaging 9 (1), 27–53. → pages 15, 62

Burstedde, C., Wilcox, L. C., Ghattas, O., 2011. p4est: Scalable Algorithms For Parallel Adaptive Mesh Refinement On Forests Of Octrees. SIAM J. on Sci. Comp. 33 (3), 1103–1133. → pages 144, 146

Calhoun, D. A., Helzel, C., Leveque, R. J., 2008. Logically Rectangular Grids and Finite Volume Methods for PDEs in Circular and Spherical Domains. SIAM Review 50 (4), 723–752. → pages 147

Carrick, S., Almond, P., Buchan, G., Smith, N., dec 2010. In situ characterization of hydraulic conductivities of individual soil profile layers during infiltration over long time periods. European Journal of Soil Science 61 (6), 1056–1069. → pages 62

Celia, M. A., Bouloutas, E. T., Zarba, R. L., 1990. A general mass-conservative numerical solution for the unsaturated flow equation. Water Resources Research 26 (7), 1483–1496. → pages xvi, 13, 60, 64, 65, 66, 71, 73, 74, 82, 84, 85, 86

Claerbout, J., Muir, F., 1973. Robust modeling with erratic data. Geophysics 38, 826–844. → pages 34

Cockett, R., 2012. Visible Geology - Interactive online geologic block modelling. In: AGU Fall Meeting. → pages 198

Cockett, R., 2013. A Interactive Online Geologic Block Modeling. In: AAPG Hedberg Research Conference - 3D Structural Geologic Interpretation: Earth, Mind, and Machine. AAPG, Reno. → pages 198

Cockett, R., 2015. Developing, deploying and reflecting on a web-based geologic simulation tool. In: AGU Fall Meeting. → pages 198

Cockett, R., 2017. Richards equation comparison to Celia 1990. FigShare http://dx.doi.org/10.6084/m9.figshare.4833605.v1. → pages 84

Cockett, R., Funning, G. J., Pratt-Sitaula, B., 2014a. Visible Earthquakes : An open visualization and modeling platform for InSAR earthquake data. In: Southern California Earthquake Center - Annual Meeting. → pages 198

Cockett, R., Haber, E., 2013a. A Numerical Method for Large Scale Estimation of Distributed Hydraulic Conductivity from Richards Equation. In: AGU Fall Meeting. → pages v, 65, 174

Cockett, R., Haber, E., 2013b. Inversion for hydraulic conductivity using the unsaturated flow equations. In: SIAM Conference on Mathematical & Computational Issues in the Geosciences. → pages v, 65

Cockett, R., Haber, E., 2015. Richards Equation Inversion. FigShare http://dx.doi.org/10.6084/m9.figshare.1450842. → pages 91

Cockett, R., Heagy, L. J., Haber, E., 2017. A numerical method for efficient 3D inversions using Richards equation. arXiv. → pages v, vi, vii, 65, 125, 127

Cockett, R., Heagy, L. J., Kang, S., Oldenburg, D. W., 2015a. Coupling geophysical terminology and package development. In: SciPy. → pages v

Cockett, R., Heagy, L. J., Oldenburg, D. W., 2016a. Pixels and their neighbors: Finite volume. The Leading Edge 35 (8), 703–706. → pages vi, 66, 68, 144, 174

Cockett, R., Heagy, L. J., Rosenkjaer, G., Kang, S., 2015b. Development practices and lessons learned in developing SimPEG. In: AGU Fall Meeting. → pages v

Cockett, R., Kang, S., Heagy, L. J., Pidlisecky, A., Haber, E., Oldenburg, D. W., 2014b. SimPEG: A framework for simulation and parameter estimation in geophysics. In: SciPy. Austin. → pages v, 174

Cockett, R., Kang, S., Heagy, L. J., Pidlisecky, A., Oldenburg, D. W., 2015c. SimPEG: An open source framework for simulation and gradient based parameter estimation in geophysical applications. Computers & Geosciences 85, 142–154. → pages v, 65, 66, 74, 125, 144, 145, 147, 185, 190

Cockett, R., Moran, T., Pidlisecky, A., 2016b. Visible Geology: Creative online tools for teaching, learning, and communicating geologic concepts. In: Krantz, R., Ormand, C. J., Freeman, B. (Eds.), Earth, Mind, and Machine: 3D Structural Interpretation, American Association of Petroleum Geologists Hedberg Series. No. 6. → pages vi, 198

Cockett, R., Pidlisecky, A., mar 2014. Simulated electrical conductivity response of clogging mechanisms for managed aquifer recharge. Geophysics 79 (2), D81–D89. → pages vi, 91, 92

Constable, S. C., Parker, R. L., Constable, C. G., 1987. Occam's inversion: A practical algorithm for generating smooth models from electromagnetic sounding data. Geophysics 52 (3), 289–300. → pages 22, 76

Daily, W., Ramirez, A., LaBrecque, D., Nitao, J., 1992. Electrical resistivity tomography of vadose water movement. Water Resources Research 28 (5), 1429–1442. → pages 1

Dean, O. S., Chen, Y., 2011. Recent progress on reservoir history matching: a review. Computational Geosciences 15, 185–221. → pages 61, 79, 81

Dean, O. S., Reynolds, A. C., Liu, N., 2008. Inverse Theory for Petroleum Reservoir Characterization and History Matching. Cambridge University Press, Cambridge. → pages 61

Deiana, R., Cassiani, G., Kemna, A., Villa, A., Bruno, V., Bagliani, A., 2007. An experiment of non-invasive characterization of the vadose zone via water injection and cross-hole time-lapse geophysical monitoring. Near Surface Geophysics, 183–194. → pages 14, 61, 62, 88, 90

Devi, G. K., Ganasri, B. P., Dwarakish, G. S., 2015. A Review on Hydrological Models. International Conference On Water Resources, Coastal and Ocean Engineering (ICWRCOE'15) 4 (Icwrcoe), 1001–1007. → pages 7

Devriese, S. G. R., Davis, K., Oldenburg, D. W., aug 2017. Inversion of airborne geophysics over the DO-27/DO-18 kimberlites - Part 1: Potential fields. Interpretation 5 (3), T299–T311. → pages 3, 10

Doetsch, J., Linde, N., Coscia, I., Greenhalgh, S. A., Green, A. G., 2010. Zonation for 3D aquifer characterization based on joint inversions of multimethod crosshole geophysical data 75 (6). → pages 22

Doherty, J., 2004. PEST: Model independent parameter estimation. Fifth edition of user manual. Watermark Numerical Computing. → pages 11, 63, 74, 77, 143, 184

Duff, I. S., Erisman, A. M., Reid, J. K., 1986. Direct methods for sparse matrices. Clarendon press, Oxford. → pages 48

Durner, W., 1994. Hydraulic conductivity estimation for soils with heterogeneous pore structure. Water Resources Research 30 (2), 211. → pages 62

Egbert, G. D., Kelbert, A., apr 2012. Computational recipes for electromagnetic inverse problems. Geophysical Journal International 189 (1), 251–267. → pages 147

Ekblom, H., 1973. Calculation of linear best Lp-approximations. BIT Numerical Mathematics 13 (3), 292–300. → pages 33

Farquharson, C., 1998. Nonlinear inversion using general measures of data misfit and model structure. Geophysical Journal 134, 213–227. → pages 33

Farquharson, C. G., Oldenburg, D. W., mar 2004. A comparison of automatic techniques for estimating the regularization parameter in non-linear inverse problems. Geophysical Journal International 156 (3), 411–425. → pages 36

Feller, J., Fitzgerald, B., 2000. A Framework Analysis of the Open Source Software Development Paradigm. In: Proceedings of the Twenty First International Conference on Information Systems. ICIS '00. Association for Information Systems, Atlanta, GA, USA, pp. 58–69. → pages 24

Fensel, D., Motta, E., Decker, S., Zdrahal, Z., 1997. Using ontologies for defining tasks, problem-solving methods and their mappings. Lecture Notes in Computer Science 1319, 113–128. → pages 141

Ferré, T., Bentley, L., Binley, A., Linde, N., Kemna, A., Singha, K., Holuger, K., Huisman, J. A., Minsley, B., 2009. Critical steps for the continuing advancement of hydrogeophysics. Eos 90 (23), 200. → pages 6, 7, 120

Finsterle, S., Kowalsky, M. B., 2008. Joint HydrologicalGeophysical Inversion for Soil Structure Identification. Vadose Zone Journal 7 (1), 287. → pages 6, 120

Finsterle, S., Kowalsky, M. B., jun 2011. A truncated LevenbergMarquardt algorithm for the calibration of highly parameterized nonlinear models. Computers & Geosciences 37 (6), 731–738. → pages 62, 63, 76

Finsterle, S., Zhang, Y., jul 2011. Solving iTOUGH2 simulation and optimization problems using the PEST protocol. Environmental Modelling & Software 26 (7), 959–968. → pages 15, 62, 63, 76

Fletcher, C. A. J., 1988. Computational Techniques for Fluid Dynamics. Vol. II. Springer-Verlag. → pages 69

Fournier, D., Kang, S., McMillan, M. S., Oldenburg, D. W., aug 2017. Inversion of airborne geophysics over the DO-27/DO-18 kimberlites - Part 2: Electromagnetics. Interpretation 5 (3), T397–T409. → pages 3, 10

Fournier, D., Oldenburg, D., Davis, K., sep 2016. Robust and flexible mixed-norm inversion. In: SEG Technical Program Expanded Abstracts 2016. No. 5. Society of Exploration Geophysicists, pp. 1542–1547. → pages 127

Fullagar, P. K., Pears, G. A., McMonnies, B., 2008. Constrained inversion of geologic surfaces pushing the boundaries. The Leading Edge 27 (1), 98–105. → pages 25

Funning, G. J., Cockett, R., 2012. Visible Earthquakes: a web-based tool for visualizing and modeling InSAR earthquake data. In: AGU Fall Meeting. American Geophysical Union, Fall Meeting 2012, abstract #ED43E-06. → pages 198

Gao, G., Abubakar, A., Habashy, T., 2012. Joint petrophysical inversion of electromagnetic and full-waveform seismic data. GEOPHYSICS 77 (3), WA3–WA18. → pages 22

Golub, G., Heath, M., Wahba, G., 1979. Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter. Technometrics 21, 215–223. → pages 36

Gruber, T. R., 1993. A Translation Approach to Portable Ontology Specifications. Knowledge acquisition 5 (2), 199–220. → pages 140

Gunzburger, M. D., 2003. Prespectives in flow control and optimization. SIAM. → pages 63

Guyer, J. E., Wheeler, D., Warren, J. A., 2009. FiPy : Partial Differential (September 2014). → pages 171

Haber, E., 2015. Computational Methods in Geophysical Electromagnetics. Mathematics in industry. → pages vi, 30, 34, 39, 40, 68, 76, 84, 144, 154, 172, 188

Haber, E., Ascher, U., Oldenburg, D., 2000. On Optimization Techniques for Solving Nonlinear Inverse Problems. Inverse problems 16, 1263–1280. → pages 15, 40, 62, 63, 77

Haber, E., Ascher, U., Oldenburg, D., 2004. Inversion of 3D electromagnetic data in frequency and time domain using an inexact all-at-once approach. Geophysics 69, 1216–1228. → pages 77

Haber, E., Ascher, U. M., jan 2001. Fast Finite Volume Simulation of 3D Electromagnetic Problems with Highly Discontinuous Coefficients. SIAM Journal on Scientific Computing 22 (6), 1943–1961. → pages 69, 70, 147

Haber, E., Gazit, M. H., 2013. Model Fusion and Joint Inversion. Reviews in Geophysics To Appear, 35–54. → pages 5, 10, 14, 61

Haber, E., Heldmann, S., 2007. An OcTree Multigrid Method for Quasi-Static Maxwell's Equations with Highly Discontinuous Coefficients. Journal of Computational Physics 65, 324–337. → pages 3, 31, 46, 146

Haber, E., Heldmann, S., Ascher, U., 2007. Adaptive finite volume method for the solution of discontinuous parameter estimation problems. Inverse Problems. → pages 69

Haber, E., Oldenburg, D., 1997. Joint Inversion A Structural Approach. Inverse Problems 13, 63–67. → pages 14, 22, 61

Haber, E., Oldenburg, D. W., 2000. A GCV based method for nonlinear ill-posed problems. Computational Geosciences 4 (1), 41–63. → pages 36

Haber, E., Schwarzbach, C., 2014. Parallel inversion of large-scale airborne time-domain electromagnetic data with multiple OcTree meshes. Inverse Problems 30 (5), 55011. → pages 3, 31

Hansen, P. C., 1992. Analysis of discrete ill-posed problems by means of the L-curve. SIAM review 34 (4), 561–580. → pages 36

Hansen, P. C., 1998. Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion. Vol. 4. Siam. → pages 36

Hansen, T. M., Cordua, K. S., Looms, M. C., Mosegaard, K., mar 2013. SIPPI: A Matlab toolbox for sampling the solution to inverse problems with complex prior information. Computers & Geosciences 52, 481–492. → pages 23

Harbaugh, A. W., 2005. The U.S. Geological Survey modular ground-water model the Ground-Water Flow Process. U.S. Geological Survey Techniques and Methods (6), A16. → pages 23

Harder, M., Scott Smith, B. H., Hetman, C. M., Pell, J., 2009. The evolution of geological models for the DO-27 kimberlite, NWT, Canada: Implications for evaluation. Lithos 112, 61–72. → pages 7

Haverkamp, R., M. Vauclin, J. Touma, P.J. Wierenga, Vachaud, G., 1977. A Comparison of Numerical Simulation Models For One-Dimensional Infiltration. Soil Sci. Soc. Am. J. 41, 285–294. → pages 84, 92

Heagy, L. J., Cockett, A. R., Oldenburg, D. W., aug 2014. Parametrized Inversion Framework for Proppant Volume in a Hydraulically Fractured Reservoir. In: SEG Technical Program Expanded Abstracts 2014. Society of Exploration Geophysicists, pp. 865–869. → pages vi, 24, 51, 55, 125, 144, 174, 195

Heagy, L. J., Cockett, R., Kang, S., Rosenkjaer, G., Oldenburg, D. W., 2015a. simpegEM: An open-source resource for simulation and parameter estimation problems in electromagnetic geophysics. In: AGU Fall Meeting. → pages 174

Heagy, L. J., Cockett, R., Kang, S., Rosenkjaer, G. K., Oldenburg, D. W., 2016. A framework for simulation and inversion in electromagnetics. Computers and Geosciences. → pages vi, vii, 91, 125, 127, 147, 174, 185, 187, 188, 190, 196, 199, 200, 201

Heagy, L. J., Cockett, R., Oldenburg, D. W., 2015b. Using Python to span the gap between education, research, and industry applications in geophysics. In: SciPy. → pages 145, 174

Heagy, L. J., Cockett, R., Oldenburg, D. W., 2017. Modular electromagnetic simulations with applications to steel cased wells. In: Proceedings of the Sixth International Symposium in Three-Dimensional Electromagnetics. pp. 125–129. → pages vii, 125

Heagy, L. J., Cockett, R., Oldenburg, D. W., Wilt, M., aug 2015c. Modelling electromagnetic problems in the presence of cased wells. In: SEG Technical Program Expanded Abstracts 2015. Society of Exploration Geophysicists, pp. 699–703. → pages vi, 125, 144, 174, 188

Heagy, L. J., Cockett, R., Oldenburg, D. W., Wilt, M., 2015d. Modelling electromagnetic problems in the presence of cased wells, 2–6. → pages 24

Heagy, L. J., Oldenburg, D. W., Geophysical, U. B. C., Facility, I., Columbia, B., 2013. Investigating the potential of using conductive or permeable proppant particles for hydraulic fracture characterization. SEG Houston 2013 Annual Meeting i, 576–580. → pages 195

Hewett, R. J., Demanet, L., the PySIT Team, 2013. PySIT: Python Seismic Imaging Toolbox. → pages 23

Hillier, M. J., Schetselaar, E. M., de Kemp, E. A., Perron, G., 2014. Three-Dimensional Modelling of Geological Surfaces Using Generalized Interpolation with Radial Basis Functions. Mathematical Geosciences 46 (8), 931–953. → pages 10, 198

Hindmarsh, A. C., Brown, P. N., Grant, K. E., Lee, S. L., Serban, R., Shumaker, D. E., Woodward, C. S., sep 2005. SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers. ACM Transactions on Mathematical Software 31 (3), 363–396. → pages 172

Hinnell, A. C., Ferré, T. P. a., Vrugt, J. a., Huisman, J. a., Moysey, S., Rings, J., Kowalsky, M. B., apr 2010. Improved extraction of hydrologic information from geophysical data through coupled hydrogeophysical inversion. Water Resources Research 46 (4), W00D40. → pages 5, 6, 10, 14, 61, 89, 90, 120, 121

Holtham, E., Oldenburg, D. W., 2010. Three-dimensional inversion of MT and ZTEM data SEG Denver 2010 Annual Meeting SEG Denver 2010 Annual Meeting (2), 655–659. → pages 55

Hunter, J. D., 2007. Matplotlib: A 2D graphics environment. Computing In Science & Engineering 9 (3), 90–95. → pages 24

Hyman, J., Morel, J., Shashkov, M., Steinberg, S., 2002. Mimetic finite difference methods for diffusion equations . Computational Geosciences, 333–352. → pages 45, 46, 145

Hyman, J. M., Shashkov, M., dec 1997. Adjoint operators for the natural discretizations of the divergence, gradient and curl on logically rectangular grids. Applied Numerical Mathematics 25 (4), 413–442. → pages 144

Hyman, J. M., Shashkov, M., 1999. Mimetic Discretizations for Maxwell's Equations 909, 881–909. → pages 45, 144, 145

Iden, S. C., Durner, W., jul 2007. Free-form estimation of the unsaturated soil hydraulic properties by inverse modeling using global optimization. Water Resources Research 43 (7), W07451. → pages 63

Irving, J., Singha, K., nov 2010. Stochastic inversion of tracer test and electrical geophysical data to estimate hydraulic conductivities. Water Resources Research 46 (11), W11514. → pages 14

Jahandari, H., Ansari, S., Farquharson, C. G., jan 2017. Comparison between staggered grid finitevolume and edgebased finiteelement modelling of geophysical electromagnetic data on unstructured grids. Journal of Applied Geophysics. → pages 147, 178

Jardani, A., Revil, A., Dupont, J., feb 2013. Stochastic joint inversion of hydrogeophysical data for salt tracer test monitoring and hydraulic conductivity imaging. Advances in Water Resources 52, 62–77. → pages 14

Jasak, H., Jemcov, A., Tukovic, Z., 2007. OpenFOAM : A C ++ Library for Complex Physics Simulations. International Workshop on Coupled Methods in Numerical Dynamics m, 1–20. → pages 172

Jones, E., Oliphant, T., Peterson, P., Others, 2001. SciPy: Open source scientific tools for Python. → pages 23, 24, 46, 55, 142

Kaipio, J., Somersalo, E., 2004. Statistical and Computational Inverse Problems. Springer Verlag. → pages 62

Kang, S., Cockett, R., Heagy, L. J., 2014. Moving between dimensions in electromagnetic inversions with a consistent framework. In: AGU Fall Meeting. → pages 24, 145, 174, 199

Kang, S., Cockett, R., Heagy, L. J., Oldenburg, D. W., aug 2015a. Moving between dimensions in electromagnetic inversions. In: SEG Technical Program Expanded Abstracts 2015. No. 2. Society of Exploration Geophysicists, pp. 5000–5004. → pages vi, 24, 125, 144, 174, 182, 190, 198

Kang, S., Cockett, R., Heagy, L. J., Oldenburg, D. W., 2015b. Moving between dimensions in electromagnetic inversions (2). → pages 24, 56

Kang, S., Fournier, D., Oldenburg, D. W., aug 2017a. Inversion of airborne geophysics over the DO-27/DO-18 kimberlites - Part 3: Induced polarization. Interpretation 5 (3), T345–T358. → pages 3, 10, 125

Kang, S., Heagy, L. J., Cockett, R., Oldenburg, D. W., 2017b. Exploring nonlinear inversions : A 1D magnetotelluric example. The Leading Edge (August), 696–699. → pages vii

Kang, S., Oldenburg, D. W., jan 2015. Recovering IP information in airborne-time domain electromagnetic data. ASEG Extended Abstracts 2015 (1), 1–4. → pages 24

Kang, S., Oldenburg, D. W., 2016. On recovering distributed IP information from inductive source time domain electromagnetic data (In Review). Geophysical Journal International. → pages vii, 125, 127, 145

Kelbert, A., Meqbel, N., Egbert, G. D., Tandon, K., feb 2014. ModEM: A Modular System for Inversion of Electromagnetic Geophysical Data. Computers & Geosciences. → pages 22, 23, 147

Ketcheson, D. I., Mandli, K., Ahmadia, A. J., Alghamdi, A., de Luna, M. Q., Parsani, M., Knepley, M. G., Emmett, M., jan 2012. PyClaw: Accessible, Extensible, Scalable Tools for Wave Propagation Problems. SIAM Journal on Scientific Computing 34 (4), C210–C231. → pages 171

Knight, R., Irving, J., van der Kruk, J., jan 2013. Studying Hydrological Properties and Processes at Scales From Centimeters to Watersheds. Eos, Transactions American Geophysical Union 94 (2), 21–21. → pages 7

Lelièvre, P. G., Oldenburg, D. W., Williams, N. C., 2009. Integrating geological and geophysical data through advanced constrained inversions. Exploration Geophysics 40 (4), 334–341. → pages 22, 25, 28

LeVeque, R. J., 1997. Wave Propagation Algorithms for Multidimensional Hyperbolic Systems. Journal of Computational Physics 131 (2), 327–353. → pages 171

Li, M., Abubakar, A., Habashy, T. M., Zhang, Y., 2010. Inversion of controlled-source electromagnetic data using a model-based approach. Geophysical Prospecting 58 (3), 455–467. → pages 28, 182

Li, X. S., sep 2005. An overview of SuperLU. ACM Transactions on Mathematical Software 31 (3), 302–325. → pages 48

Li, Y., Key, K., 2007. 2D marine controlled-source electromagnetic modeling: Part 1 An adaptive finite-element algorithm. GEOPHYSICS 72 (2), WA51–WA62. → pages 22

Li, Y., Oldenburg, D., 2000a. Incorporating geological dip information into geophysical inversions. GEOPHYSICS 65 (1), 148–157. → pages 22, 25, 34

Li, Y., Oldenburg, D. W., 1996. 3D Inversion of magnetic data. Geophysics 61, 394–408. → pages 22, 34, 147, 182

Li, Y., Oldenburg, D. W., jan 1998. 3-D inversion of gravity data. Geophysics 63 (1), 109–119. → pages 22, 182

Li, Y., Oldenburg, D. W., 2000b. Joint inversion of surface and three-component borehole magnetic data. Geophysics 65 (2), 540–552. → pages 34

Liang, L., Abubakar, A., Habashy, T., 2012. Joint inversion of controlled-source electromagnetic and production data for reservoir monitoring. GEOPHYSICS 77 (5), ID9–ID22. → pages 5

Liang, W.-L., Uchida, T., mar 2014. Effects of topography and soil depth on saturated-zone dynamics in steep hillslopes explored using the three-dimensional Richards' equation. Journal of Hydrology 510, 124–136. → pages 61

Lin, J. W.-B., dec 2012. Why Python Is the Next Wave in Earth Sciences Computing. Bulletin of the American Meteorological Society 93 (12), 1823–1824. → pages 42

Linde, N., Doetsch, J., apr 2016. Joint Inversion in Hydrogeophysics and Near-Surface Geophysics. pp. 117–135. → pages 4, 5, 6, 7, 11, 14, 19, 62, 120, 122

Linde, N., Renard, P., Mukerji, T., Caers, J., 2015. Geological realism in hydrogeological and geophysical inverse modeling: A review. Advances in Water Resources 86, 86–101. → pages 10, 76

Lines, L. R., Schultz, A. K., Treitel, S., 1988. Cooperative inversion of geophysical data. Geophysics 53 (1), 8–20. → pages 41, 55

Lipnikov, C., Misitas, O., 2013. Second-order accurate monotone finite volume scheme for Richards' equation. J. Comp. Phys 239, 123–137. → pages 68

Liu, Y., Gupta, H. V., 2007. Uncertainty in hydrologic modeling: Toward an integrated data assimilation framework. Water Resources Research 43 (7), 1–18. → pages 7, 13

Ma, X., 2011. Ontology Spectrum for Geological Data Interoperability. Ph.D. thesis. → pages 140

Mawer, C., Kitanidis, P., Pidlisecky, A., Knight, R., 2013. Electrical Resisivity for Characterization and Infiltration Monitoring beneath a Managed Aquifer Recharge Pond. Vadose Zone Journal 12 (1), 1–20. → pages 98, 109, 111

McDonald, M. G., Harbaugh, A. W., mar 2003. The History of MODFLOW. Ground Water 41 (2), 280–283. → pages 147

McMillan, M. S., Oldenburg, D. W., 2014. Cooperative constrained inversion of multiple electromagnetic data sets. Geophysics 79 (4), B173–B185. → pages 10

McMillan, M. S., Schwarzbach, C., Haber, E., Oldenburg, D. W., 2015. 3D parametric hybrid inversion of time-domain airborne electromagnetic data. Geophysics 80 (6), K25–K36. → pages 182, 183

Mendelson, K. S., Cohen, M. H., feb 1982. The effect of grain anisotropy on the electrical properties of sedimentary rocks. GEOPHYSICS 47 (2), 257–263. → pages 5

Miller, C. A., Williams-Jones, G., Fournier, D., Witter, J., 2017. 3D gravity inversion and thermodynamic modelling reveal properties of shallow silicic magma reservoir beneath Laguna del Maule, Chile. Earth and Planetary Science Letters 459, 14–27. → pages vii, 125, 127

Mualem, Y., 1976. A new model for predicting the hydraulic conductivity of unsaturated porous media. Water Resources Research 12 (3), 513. → pages 62, 66, 67, 92, 93

Nocedal, J., Wright, S. J., 1999. Numerical Optimization. Springer, New York, NY. → pages 37, 55, 73, 77

Noy, N. F., McGuinness, D. L., 2002. Ontology Development 101: A Guide to Creating Your First Ontology. In: Stanford Medical Informatics Report. Stanford University, Stanford, CA. → pages 140

Olah, C., Carter, S., 2017. Research Debt. Distill 1 (1), 1–5. → pages 128

Oldenburg, D. W., 1984. An introduction to linear inverse theory. Geoscience and Remote Sensing, IEEE Transactions on (6), 665–674. → pages 34

Oldenburg, D. W., 2016. Looking back and moving forward. Banff International Research Station 16 (2695). → pages 3, 9, 123

Oldenburg, D. W., Li, Y., 2005. 5. Inversion for Applied Geophysics: A Tutorial. Ch. 5, pp. 89–150. → pages 7, 22, 32, 34, 36, 40, 56, 76, 182

Oliphant, T. E., 2007. Python for Scientific Computing. Computing in Science & Engineering 9 (3). → pages 24, 46

Oliver, Y. A. D., Reynolds, A., 2001. Efficient History-Matching Using Subspace Vectors. Computational Geosciences 5, 151–172. → pages 61

Ollivier-Gooch, C., Van Altena, M., sep 2002. A High-Order-Accurate Unstructured Mesh Finite-Volume Scheme for the AdvectionDiffusion Equation. Journal of Computational Physics 181 (2), 729–752. → pages 46, 147

Orgogozo, L., Renon, N., Soulaine, C., Hénon, F., Tomer, S., Labat, D., Pokrovsky, O., Sekhar, M., Ababou, R., Quintard, M., 2014. An open source massively parallel solver for Richards equation: Mechanistic modelling of water fluxes at the watershed scale. Computer Physics Communications 185 (12), 3358–3371. → pages 14, 15, 61, 73

Paasche, H., Tronicke, J., 2007. Cooperative inversion of 2D geophysical data sets: A zonal approach based on fuzzy c-means cluster analysis. Geophysics 72 (3), A35. → pages 6, 121

Park, S., 1998. Fluid migration in the vadose zone from 3-D inversion of resistivity monitoring data. Geophysics 63 (1), 41–51. → pages 1

Parker, R., 1994. Geophysical Inverse Theory. Princeton University Press, Princeton NJ. → pages 32, 36, 56

Parker, R. L., may 1977. Understanding Inverse Theory. Annual Review of Earth and Planetary Sciences 5 (1), 35–64. → pages 36

Peckham, S. D., Hutton, E. W., Norris, B., apr 2013. A component-based approach to integrated modeling in the geosciences: The design of CSDMS. Computers & Geosciences 53, 3–12. → pages 141

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12, 2825–2830. → pages 124

Pérez, F., Granger, B. E., may 2007. IPython: a System for Interactive Scientific Computing. Computing in Science and Engineering 9 (3), 21–29. → pages 47

Pidlisecky, A., Cockett, a. R., Knight, R., 2013. Electrical Conductivity Probes for Studying Vadose Zone Processes: Advances in Data Acquisition and Analysis. Vadose Zone Journal 12 (1), 1–12. → pages vi, 5, 14, 91, 92, 111, 114, 119, 147, 199

Pidlisecky, A., Haber, E., Knight, R., 2007. RESINVM3D : A 3D resistivity inversion package. Geophysics 72 (2), H1—-H10. → pages 47, 52, 152, 182

Pidlisecky, A., Singha, K., Day-Lewis, F. D., 2011. A distribution-based parametrization for improved tomographic imaging of solute plumes. Geophysical Journal International 187, 214–224. → pages 40, 182, 183

Pollock, D., Cirpka, O. a., jan 2012. Fully coupled hydrogeophysical inversion of a laboratory salt tracer experiment monitored by electrical resistivity tomography. Water Resources Research 48 (1), W01505. → pages 5, 7

Porwal, A. K., Kreuzer, O. P., 2010. Introduction to the Special Issue: Mineral prospectivity analysis and quantitative resource estimation. Ore Geology Reviews 38 (3), 121–127. → pages 7

Racz, A. J., Fisher, A. T., Schmidt, C. M., Lockwood, B. S., Huertos, M. L., nov 2011. Spatial and Temporal Infiltration Dynamics During Managed Aquifer Recharge. Ground water, 1–9. → pages 1

Richards, L. A., 1931. Capillary conduction of liquids through porous mediums. Journal of Applied Physics 1 (5), 318–333. → pages 13, 60, 65

Rosenkjaer, G., Cumming, W., Christopherson, K., 2016. The CSIMT method combining natural MT signals and cultural noise as sources. SEG Technical Program Expanded Abstracts 2016, 889–894. → pages vii, 125, 127, 145, 174, 200

Rosenkjaer, G., Heagy, L. J., Cockett, R., 2015a. Practical integration of processing, inversion and visualization of magnetotelluric geophysical data. In: SciPy. → pages 174

Rosenkjaer, G., Heagy, L. J., Cockett, R., Kang, S., Oldenburg, D., 2015b. Using simpegMT to demonstrate a modular modelling and inversion framework applied in a workflow for a hydrothermal system. In: AGU Fall Meeting. → pages 145

Rücker, C., Günther, T., Wagner, F. M., 2017. pyGIMLi: An open-source library for modelling and Inversion in geophysics. Computers & Geosciences. → pages 10, 23

Ruthotto, L., Treister, E., Haber, E., 2016. jInv – a flexible Julia package for PDE parameter estimation. arXiv, 1–19. → pages 10, 23, 172

Saad, Y., 1996. Iterartive Methods for Sparse Linear Systems. PWS Publishing Company. → pages 73

Sarma, P., Durlofsky, L. J., Aziz, K., Chen, W., 2007. A New Approach to Automatic History Matching using Kernel PCA. SPE Reservoir Simulation Symposium, Houston, Texas. → pages 61

Schaap, M. G., Leij, F. J., 2000. Improved Prediction of Unsaturated Hydraulic Conductivity with the Mualem-van Genuchten Model. Soil Science Society of America Journal 64 (3), 843–851. → pages 92

Schenk, O., Gartner, K., 2004. Solving unsymmetric sparse systems of linear equations with PARDISO. Future Generation Computer Systems 20 (3), 475–487. → pages 48

Sharman, R., Kishore, R., Rames, R., 2004. Computational Ontologies and Information Systems: II. Formal Specification. Communications of AIS 2004 (14), 184–205. → pages 140, 141, 142

Šimunek, J., van Genuchten, M. T., 1996. Estimating Unsaturated Soil Hydraulic Properties from Tension Disc Infiltrometer Data by Numerical Inversion. Water Resources Research 32 (9), 2683. → pages 62, 63, 76

Šimunek, J., van Genuchten, M. T., Senja, M., 2012. HYDRUS: MODEL USE, CALIBRATION, AND VALIDATION. American Society of Agricultural and Biological Engineers 55 (4), 1261–1274. → pages 5, 61, 63

Šimunek, J., Wendroth, O., van Genuchten, M. T., 1998. Parameter Estimation Analysis of the Evaporation Method for Determining Soil Hydraulic Properties. Soil Science Society of America Journal 62, 894–905. → pages 95

Stein, L. D., 2008. Towards a cyberinfrastructure for the biological sciences: progress, visions and challenges. Nature reviews. Genetics 9 (9), 678–688. → pages 141

Strong, D., Chan, T., 2003. Edge-preserving and scale-dependent properties of total variation regularization. Inverse problems 19 (6), S165. → pages 34

Sun, J., Li, Y., Studies, M., 2012. Joint inversion of multiple geophysical data : A petrophysical approach using guided fuzzy c-means clustering SEG Las Vegas 2012 Annual Meeting SEG Las Vegas 2012 Annual Meeting (4), 1–5. → pages 6, 121

Tarantola, A., 2005. Inverse Problem Theory and Methods for Model Parameter Estimation. Society for Industrial and Applied Mathematics. → pages 21

Tarantola, A., Valette, B., 1982. Generalized nonlinear inverse problems solved using the least squares criterion. Reviews of Geophysics 20 (2), 219–232. → pages 21

Tikhonov, A., Arsenin, V., 1977. Solutions of Ill-Posed Problems. W.H. Winston and Sons. → pages 36, 56, 62, 76

Towara, M., Schanen, M., Naumann, U., 2015. MPI-Parallel Discrete Adjoint OpenFOAM. Procedia Computer Science 51 (1), 19–28. → pages 5, 14, 15, 61, 62, 63, 77

Trottenberg, U., Oosterlee, C., Schuller, A., 2001. Multigrid. Academic Press. → pages 70

Uieda, L., Jr, V. C. O., Ferreira, A., dos Santos, H. B., Jr., J. F. C., 2014. Fatiando a Terra: a Python package for modeling and inversion in geophysics. → pages 23

van Genuchten, M. T., 1980. A Closed-form Equation for Predicting the Hydraulic Conductivity of Unsaturated Soils. Soil Science Society of America Journal 44 (5), 892. → pages 66, 92

Van Genuchten, M. T., Leij, F. J., Yates, S. R., Others, 1991. The RETC code for quantifying the hydraulic functions of unsaturated soils. → pages xiv, 94

Van Rossum, G., Drake Jr, F. L., 1995. Python reference manual. Centrum voor Wiskunde en Informatica Amsterdam. → pages 24, 42

Vogel, C., 2001. Computational methods for inverse problem. SIAM, Philadelphia. → pages 73, 74

Wahba, G., 1990. Spline Models for Observational Data. SIAM, Philadelphia. → pages 36

Williams, N. C., 2008. Geologically-constrained UBCGIF Gravity And Magnetic Inversions With Examples From The Agnew-Wiluna Greenstone Belt, Western Australia. Phd, University of British Columbia. → pages 10, 182

Wilson, G., Aruliah, D. a., Brown, C. T., Chue Hong, N. P., Davis, M., Guy, R. T., Haddock, S. H. D., Huff, K. D., Mitchell, I. M., Plumbley, M. D., Waugh, B., White, E. P., Wilson, P., jan 2014. Best practices for scientific computing. PLoS biology 12 (1), e1001745. → pages 24, 42

Yang, D., Oldenburg, D. W., Haber, E., mar 2014. 3-D inversion of airborne electromagnetic data parallelized and accelerated by local mesh and adaptive soundings. Geophysical Journal International 196 (3), 1492–1507. → pages 3, 31

Yee, K. S., 1966. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. IEEE Trans. on antennas and propagation 14, 302–307. → pages 144, 145

Zhdanov, M. S., 2002. Preface. In: Zhdanov, M. S. (Ed.), Geophysical Inverse Theory and Regularization Problems. Vol. 36 of Methods in Geochemistry and Geophysics. Elsevier, pp. XIX – XXIII. → pages 34

Appendix A

Frameworks and ontologies

A computational science framework provides a set of standards such that individual scientists can contribute software components, in this case components used in simulation or inversion routines, with the confidence that those components will work with other components in that framework. As such, the standards of the framework define the responsibilities of each component (or class of component) and the required interfaces of all components. The term framework is commonly used in the software community, however, the formal term for the component organization and their properties is an ontology. The use of ontologies in the sciences to formally describe domain knowledge has exploded in recent decades, especially in domains of artificial intelligence, chemistry, and biology, but also more recently in the geosciences (Sharman et al., 2004; Ma, 2011). A computational ontology (rather than the underlying discipline of philosophical ontology) is a "formal explicit specification of a shared conceptualization" (Gruber, 1993). Noy and McGuinness (2002) summarizes the purpose of computational ontologies as enabling a shared understanding of the structure of information and systematically enabling knowledge and information reuse. Practically

ontologies can (a) provide access and discoverability to heterogeneous information; (b) act as a common language to lower the barrier to transfer of ideas; and (c) act as a specification for interoperability, for example, as a communication protocol or application programming interface (Sharman et al., 2004). The techniques for building ontologies amount to capturing, synthesizing, organizing, and digitizing the relationships between concepts, conceptual inheritance patterns, and behaviour. Ontologies are most commonly used for storing and organizing data, for example, connecting genetic data with phenotypic data in bioinformatics. However, ontologies are also used in defining tasks, workflows, and problem solving methods (Fensel et al., 1997; Bard and Rhee, 2004). In more mature interdisciplinary fields, this research is becoming core to scientists' day to day research; for example, Stein (2008) notes that "all current biomedical cyberinfrastructure efforts use ontologies." As a result of successes in other fields, geoscience integration is currently the target of major funding initiatives across the world (e.g. EarthCube - 11 year NSF project, $35M in 2015; CIMIC Footprints Project - NSERC Project, 24 Universities, 30 Industry, $13M). Many of the current efforts are focused on computational science frameworks, formally describing geoscientific data (using ontologies), and formally describing methods of integrating disciplines. For example, a Common Component Architecture for high performance scientific computing (Armstrong et al., 1999) has been used as the basis for coupled forward integration of a number of geoscience simulation tools written by different authors (Peckham et al., 2013). The research into these domain specific standards for interoperability is critical for sustainable interdisciplinary research.

The growth in complexity of geophysical data and analysis and the necessity for cross-disciplinary integrations is also coincident with the revolution of open source software communities, largely enabled through web-based interactions. Other re-

search communities, for example `Astropy` in astronomy and `SciPy` in numerical computing, have embraced the open source approach for collaboration and research (Astropy Collaboration et al., 2013; Jones et al., 2001). These pioneering efforts are now complemented by easy-to-use, ubiquitous web-based repositories and version-control systems (e.g. GitHub), that have removed many of the barriers associated with management and collaboration. The growth of such systems, coupled with the maturity of individual geophysical subdisciplines (e.g. potential fields, electromagnetics), presents an opportunity to develop a computational framework and associated ontology for geophysical simulation and inversion. An ontology is an embodiment of concepts, relationships, and behaviours in a specific scientific domain and can be (a) captured in special purpose languages (e.g. Web Ontology Language, Resource Description Framework), or (b) captured in general purpose computer programming languages (e.g. Python, Java, C++) (Sharman et al., 2004). To research a geophysical simulation and inversion framework, I have chosen the latter approach for the purposes of utility, testing, and creating a framework/ontology that can be openly used and evolved by the geoscience community.

Appendix B

Finite volume techniques

B.1 Introduction

Inverse problems are common across the geosciences: for example, in geophysical imaging, history matching, and parameter estimation. Many of these inverse problems require constrained optimization using partial differential equations (PDEs), which requires derivatives with respect to mesh variables in addition to simulation of the PDE. Finite difference, finite element, and finite volume techniques allow subdivision of continuous differential equations into discrete domains. Knowledge and the appropriate application of these methods is fundamental to simulating physical processes. Many inverse problems in the geosciences are solved using stochastic techniques or external finite difference-based tools (e.g. Doherty (2004)), which are robust to local minima and the programmatic implementation, respectively. However, these methods do not scale to situations where millions of parameters are to be estimated. This sort of scale is necessary for solving many of the inverse problems in geophysics and, increasingly, hydrogeology (e.g. electromagnetics, gravity, and fluid flow problems).

In the context of the inverse problem, when the physical properties, the domain, and the boundary conditions are not necessarily known, the simplicity and efficiency in mesh generation are important criteria. Complex mesh geometries, such as body fitted grids, commonly used when the domain is explicitly given, are less appropriate. Additionally, when considering the inverse problem, it is important that operators and their derivatives are accessible for interrogation and extension. The goal of this work is to provide a high-level background to finite volume techniques, which are abstracted across four mesh types: (1) tensor product mesh; (2) cylindrically symmetric mesh; (3) logically rectangular, non-orthogonal mesh; and (4) octree and quadtree meshes. This work contributes an overview of finite volume techniques, in the context of geoscience inverse problems, which are treated in a consistent way across various mesh types in order to highlight similarities and differences.

B.1.1 Attribution and dissemination

The numerical implementations underlying this work have been created throughout my PhD in multiple programming languages (i.e. Matlab, Julia, and Python) and have been influenced by course material and instruction from Dr. Eldad Haber, Dr. Uri Ascher and Dr. Chen Grief (Haber, 2015; Ascher and Greif, 2011). Further references can be found throughout the scientific literature (cf. Yee (1966); Hyman and Shashkov (1999, 1997)). I have published aspects of this work in Cockett et al. (2015c), in many *Society of Exploration Geophysics* abstracts (Heagy et al., 2014; Kang et al., 2015a; Heagy et al., 2015c) and in a tutorial paper in *The Leading Edge* (Cockett et al., 2016a). Dave Marchant and Lindsey Heagy influenced the development of the cylindrical mesh, and the octree storage algorithm was loosely based on the implementation by Burstedde et al. (2011). These techniques and implementations proved successful in the Sim-

PEG project (Cockett et al., 2015c) and are used in applications of frequency and time domain electromagnetics, direct current resistivity, gravity, magnetics, fluid flow, and seismic across industry, academia, and education (Rosenkjaer et al., 2016; Kang and Oldenburg, 2016; Heagy et al., 2015b; Rosenkjaer et al., 2015b; Cockett et al., 2015c; Kang et al., 2014). The techniques discussed below, as well as a number of accompanying utilities for mesh generation, import, export, visualization, documentation, and testing are provided in an open source package for Python, called: `discretize` (https://github.com/simpeg/discretize). The `discretize` package is released using the permissive MIT license to encourage reuse and future improvement of this work. My collaborators and I have also generalized across these types of meshes, in order to have both a high-level and a standard programmatic interface, differing only in mesh instantiation. The generalization of meshes allows both ourselves and others to build upon this work as we continue to improve and expand its capabilities.

B.2 Terminology

To simulate differential equations in any computational domain, we must approximate the continuous equations through discretization onto a mesh. The mesh defines boundaries, locations of variables, and connectivity between cells. In this section we will discuss a staggered mimetic finite volume approach (Yee, 1966; Hyman and Shashkov, 1999; Hyman et al., 2002).

B.2.1 Mesh types

The topographic interface, the location of boundary conditions, and the location of sources/receivers are often the only non-numerical constraints on mesh generation in geophysics. These constraints require meshes that that can pad efficiently to suf-

ficiently distant boundaries (as in electromagnetics), align or refine to topographic features, and refine around the locations of sources. Numerical efficiency generally translates to minimizing the number of cells used in any computational domain. Here, we will consider four mesh types: (1) tensor product mesh; (2) logically rectangular; non-orthogonal mesh (curvilinear mesh); (3) octree and quadtree meshes; and (4) cylindrically symmetric mesh. We use different techniques for dealing with padding, alignment, and refinement for each mesh. Orthogonal vectors of spacings define a tensor product mesh (tensor mesh). In the 2D example in Figure B.1a, the mesh is created from two vectors, $\mathbf{h}_x$ and $\mathbf{h}_y$, which define constant spacing, orthogonal to each direction. As the spacing is fixed, any refinement in one dimension means that that refinement is completed everywhere in the domain. These refinement constraints often lead to meshes with many cells and unnecessary resolution far from the domain of interest. Tree meshes are built through successively dividing mesh cells into four or eight cells in 2D quadtree meshes and 3D octree meshes, respectively (Figure B.1b). Octree meshes are used extensively in electromagnetic geophysical inversions (Haber and Heldmann, 2007). These meshes can also be built on a variable tensor spaced grid, but have the advantage of not refining in locations far from areas of interest, resulting in meshes with fewer cells. Unlike the tensor mesh, tree meshes are not logically rectangular; that is, each cell does not necessarily have two neighbours in each dimension (x_+, x_-, y_+, y_-, etc.). A quadtree cell may have additional neighbours if it is coarser than its direct neighbours. Using a mesh leveling algorithm, the level of refinement can be enforced to be a maximum of one level change between cells (Burstedde et al., 2011). The tensor mesh is a logically rectangular orthogonal mesh, where 'orthogonal' means that the tensors are orthogonal and define a local Cartesian coordinate system. Another mesh type under consideration is a logically rectangular *non*-orthogonal mesh,

where each cell still has two neighbours in each dimension but the cells are neither required to be axes aligned nor to have orthogonal faces. Here, we will refer to logically rectangular non-orthogonal meshes as curvilinear meshes, as seen in Figure B.1c. As these meshes are no longer constrained to have orthogonal cells, topographic layers can be better approximated, without the staircase effect that is present on both tensor and tree meshes. Additionally, curvilinear meshes allow different ways of padding to 'infinity', and can be used to approximate spherical domains (Calhoun et al., 2008). Finally, we will also consider a cylindrically symmetric tensor mesh. The cylindrically symmetric tensor mesh is defined in a cylindrical coordinate system where the radial r, azimuthal θ, and vertical z dimensions are in the following domains:

$$r \in [0, \infty), \qquad \theta \in [0, 2\pi), \qquad z \in (-\infty, \infty) \tag{B.1}$$

Cylindrical symmetry is enforced through a single cell in θ. With the exception of calculations for boundary conditions, volume and area are formulated similarly to tensor meshes. Cylindrical meshes are often used for electromagnetics problems for layered systems or cylindrically symmetric problems, such as geophysics or fluid flow around a borehole (Pidlisecky et al., 2013; Heagy et al., 2016). Fully unstructured (tetrahedral) meshes will not be considered here, but are commonly used in geophysics and hydrogeology (e.g. Ollivier-Gooch and Van Altena (2002); Jahandari et al. (2017)). We chose the meshes used in this appendix for their common use in electromagnetic geophysics and fluid flow (Haber and Ascher, 2001; Li and Oldenburg, 1996; Egbert and Kelbert, 2012; McDonald and Harbaugh, 2003; Kelbert et al., 2014; Cockett et al., 2015c). All meshes are easy to parameterize, which is an advantage when relatively little is known about the simulation domain, as is the case in the context in geophysical

inverse problems.

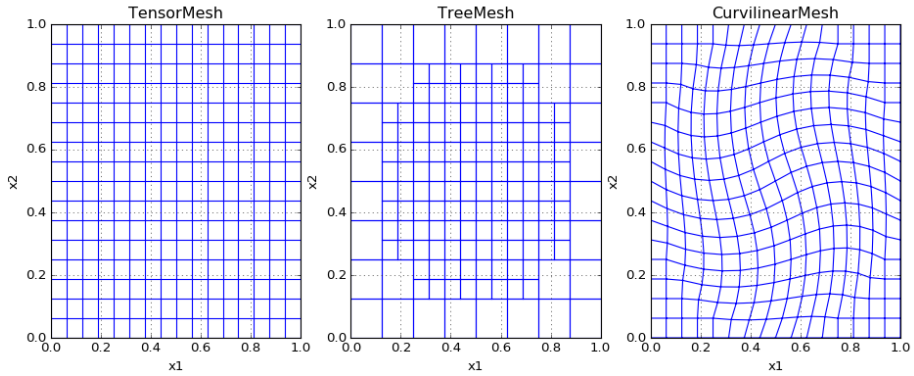

Figure B.1: Three mesh types in two dimensions on the domain of a unit square: (a) a tensor product mesh, (b) a quadtree mesh, and (c) a curviliear mesh.

B.2.2 Cell anatomy

This approach requires defining variables at either cell centers, nodes, faces, or edges, as described in Figure B.2. The finite volume technique is derived geometrically from studying the control volume of a mesh 'cell'. The cell center is often used for scalar variables or anisotropic tensors that represent physical properties. This shows that a single value fills the entire cell, allowing discontinuities between adjacent cells. From a geologic perspective, discontinuities are prevalent, as large differences in physical properties may exist between geologic layers. Cell nodes, alternatively, are often used for variables that are continuously varying in space; that is, internal to a cell values between nodes can be found through bi/tri-linear interpolation. Vector quantities are held on the faces or edges.

A cell face variable represents a vector that is a flux into or out of that face; the vector is pointed in the face normal direction, $\vec{n}$. As seen in the curvilinear cell in Figure B.3, the face normal directions may not be orthogonal, nor parallel to the Cartesian

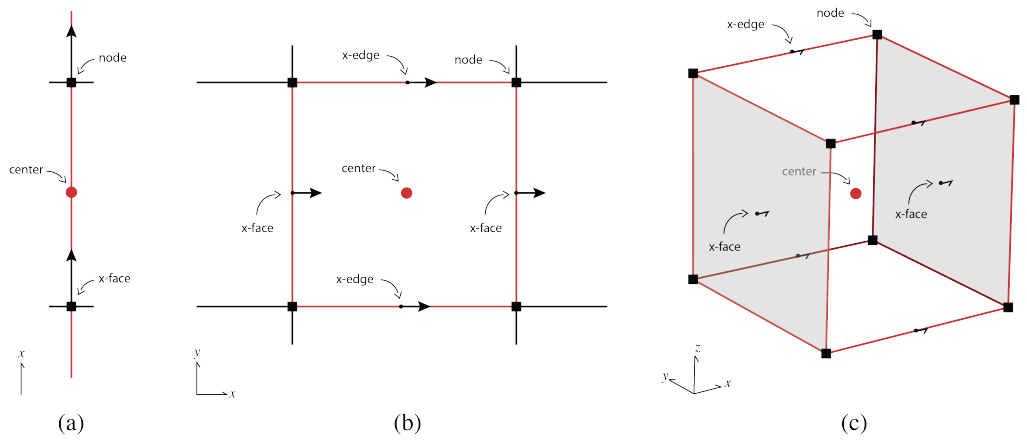

Figure B.2: Names of a finite volume cell on a tensor mesh in (a) one dimension, (b) two dimensions, and (c) three dimensions.

axes. However, as the direction of the face normal is a property of the mesh, the face variables only store the magnitude of the vector. A face variable on a single rectilinear cell is a length four array in 2D and a length six array in 3D. There are twelve edges in 3D, four in 2D, and one in 1D for each cell, all holding vector quantities that point in the tangent directions, $\vec{t}$. While cell faces represent fluxes, edges represent vector fields, as is the case in electromagnetics.

Figure B.4 displays a tree mesh cell, which shows the location of hanging faces and nodes when two cells of different refinement levels share an interface. These hanging nodes, faces, and edges become important when computing the differential operators and inner products. A cylindrically symmetric mesh has the same structure as a tensor product mesh cell, except that all cells must be in the radial domain $r \in [0, \infty)$. It is tempting to conceptually locate the cell center of the first radial cell at $r = 0$, as this would be at the center of the cylinder. However, locating the cell center here violates our staggered grid in *cylindrical* coordinates and operators, such as the divergence, do not converge with second order accuracy. For consistency throughout the following

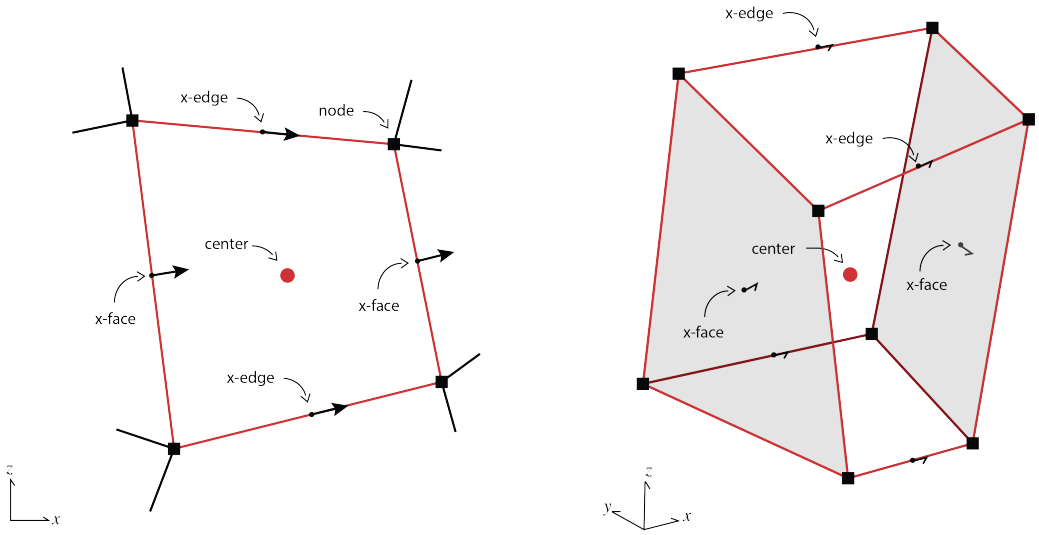

Figure B.3: Names of a finite volume cell on a curvilinear mesh in (a) two dimensions, and (b) three dimensions. Note that the cell faces and edges are no longer orthogonal.

sections, we will use terminology derived from a 3D cell. A cell's volume will refer to: volume in 3D; area in 2D; and length in 1D. Face areas will refer to the area perpendicular to a 3D face, which is a length in 2D and unity in 1D. Edge lengths will refer to lengths in 2D and will be in the same spatial locations as the cell faces (although with a different numbering and vector direction). In 1D, edge lengths will be the cell 'volumes' (lengths) and will be located at the cell centers. As such, the length of the 'volume' array will always be equal to the number of cells in the mesh.

B.2.3 Numbering

The numbering of any mesh must be explicit in order to define arrays of properties, fields, and fluxes. The numbering of the mesh is arbitrary but has a number of consequences for the resulting differential operator matrices in terms of their structure and

construction. In all logically rectangular meshes under consideration, we count first in the x, then y, then z dimensions. Counting in this way results in column vectorization and allows the use of Kronecker products for many of the matrix equations, specifically, the vectorization identity:

$$\text{vec}(AB) = (I_m \otimes A)\text{vec}(B) \tag{B.2}$$

where B is the discretized grid function, which is useful in building differential operators recursively from 1D operators. For non-logically rectangular meshes, such as quadtree and octree meshes, we sort the numbering first by distance along the x-axis, then y- and then z-. In the case of faces and edges where there are x-, y-, and z-components, we order these components separately and then concatenate them. The numbering is shown in Figure B.8a for a quadtree mesh for the cell centers, x-faces, and y-faces. Although Kronecker products can be used for logically rectangular meshes, this is not possible for tree-based meshes and the indexes must be kept track of 'by hand'.

It is important to know the number of variables for the discretization of grid variables. For a 3D logically rectangular mesh, the number of cells, nodes, faces, and

edges are:

$$
\begin{aligned}
n_c &= n_{c_x} \times n_{c_y} \times n_{c_z} && \text{cells} \\
n_n &= (n_{c_x}+1) \times (n_{c_y}+1) \times (n_{c_z}+1) && \text{nodes} \\
n_{f_x} &= (n_{c_x}+1) \times n_{c_y} \times n_{c_z} && \text{x-faces} \\
n_{f_y} &= n_{c_x} \times (n_{c_y}+1) \times n_{c_z} && \text{y-faces} \\
n_{f_z} &= n_{c_x} \times n_{c_y} \times (n_{c_z}+1) && \text{z-faces} \\
n_{e_x} &= n_{c_x} \times (n_{c_y}+1) \times (n_{c_z}+1) && \text{x-edges} \\
n_{e_y} &= (n_{c_x}+1) \times n_{c_y} \times (n_{c_z}+1) && \text{y-edges} \\
n_{e_z} &= (n_{c_x}+1) \times (n_{c_y}+1) \times n_{c_z} && \text{z-edges}
\end{aligned}
\tag{B.3}
$$

When comparing this to a cylindrically symmetric mesh, it is interesting to note that neither nodes nor θ faces exist, and edges only exist in the θ direction. A tree mesh has an added complication, which occurs when two adjacent cells have different refinement levels, leading to hanging nodes, edges, and faces. Figure B.4 schematically shows the locations of the hanging faces and nodes. When not dealt with, these complications cause numerical inaccuracies, which we will discuss further in Section B.3 on differential operators and Section B.4 on inner products.

B.2.4 DC resistivity equations

We will use the direct current (DC) resistivity problem from geophysics to motivate discretization of a parabolic partial differential equation and explain the various operators and operations necessary to consider for the finite volume technique. The equations for DC resistivity are derived in Figure B.5 and are further discussed in (Pidlisecky et al., 2007). Conservation of charge (which can be derived by taking the divergence of Amperes law at steady state) connects the divergence of the current den-

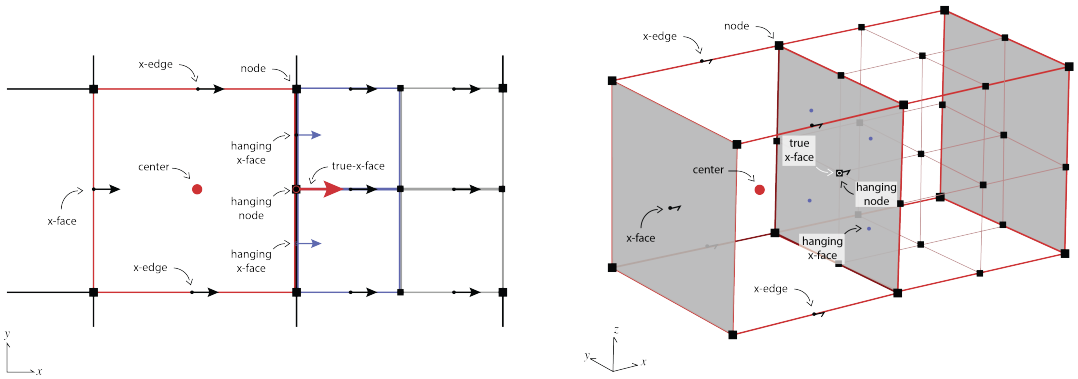

Figure B.4: Names of a finite volume cell on a tree mesh in (a) two dimensions, and (b) three dimensions. Note the location of hanging x-faces from the refined cells; hanging edges are not shown.

sity everywhere in space to the source term, which consists of two point sources: one positive and one negative. The flow of current sets up electric fields according to Ohms law, which relates current density to electric fields through the electrical conductivity, σ. From Faradays law for steady state fields, we can describe the electric field in terms of a scalar potential, ϕ, which in a DC resistivity experiment is sampled at potential electrodes to obtain data in the form of potential differences. The first order form of the governing equations for DC resistivity are:

$$\begin{aligned} \nabla \cdot \vec{j} &= I(\delta(\vec{r} - \vec{r}_{s^+}) - \delta(\vec{r} - \vec{r}_{s^-})) = q \\ \frac{1}{\sigma}\,\vec{j} &= -\nabla\phi \end{aligned} \tag{B.4}$$

where I is the input current at the positive and negative dipole locations, $\vec{r}_{s^\pm}$, captured as Dirac delta functions. To motivate the discretization of the DC resistivity equations, we will write the equations in weak form in the following section.

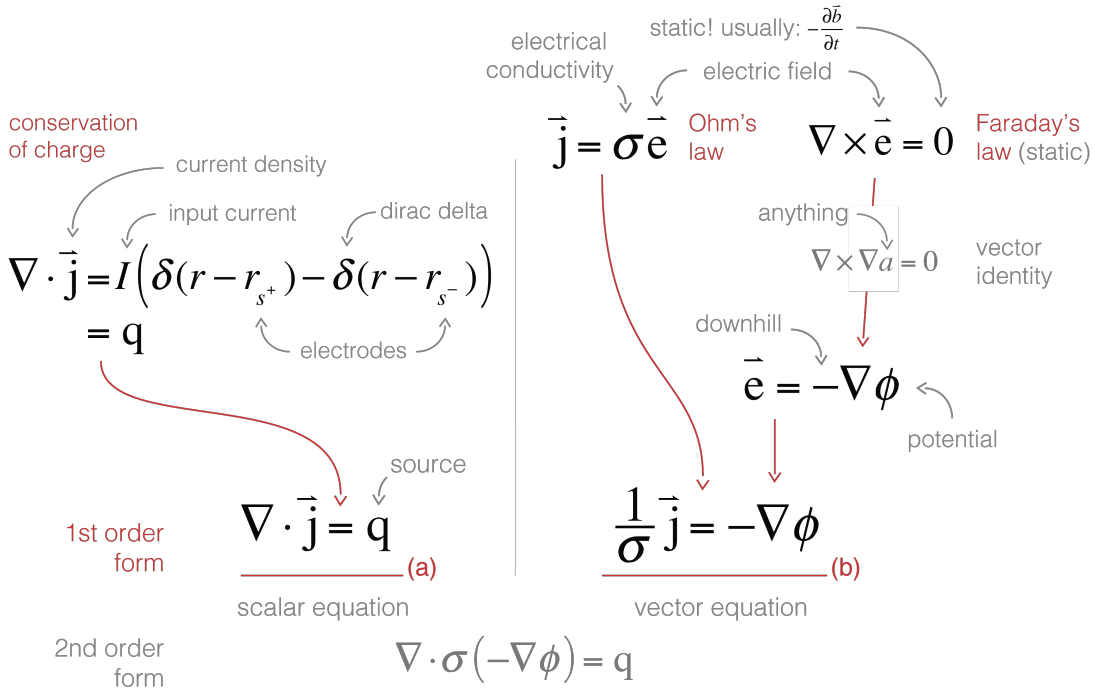

Figure B.5: Derivation of the direct current resistivity equations.

B.2.5 Weak formulation

The weak formulation integrates the DC resistivity equations with a test function, $\vec{f}$, which reduces the requirement of differentiability (more details are available in Haber (2015)). To keep the notation clean, we also introduce notation $(\cdot,\ \cdot)$, which we refer to as an *inner product*.

$$(a, b) = \int_\Omega a(\vec{x}) \cdot b(\vec{x})\ \partial v \tag{B.5}$$

where the vectors $\vec{a}$ and $\vec{b}$ are arbitrary. The vector part of the DC resistivity equations (written in first order form) can be written in weak form as:

$$\left(\frac{1}{\sigma}\ \vec{j}, \vec{f}\ \right) = \left(-\nabla \phi, \vec{f}\ \right) \tag{B.6}$$

where $\vec{f}$ is the test function. We can now employ a vector identity:

$$\nabla \cdot \left(a\, \vec{f} \right) = a \left(\nabla \cdot \vec{f} \right) + \left(\vec{\nabla}\, a \right) \cdot \vec{f} \tag{B.7}$$

to integrate the right-hand side by parts. This integration results in the discretization of the DC resistivity equations entirely in terms of the divergence operator.

$$\left(\frac{1}{\sigma}\, \vec{j},\, \vec{f} \right) = \int_{\Omega} \phi \left(\nabla \cdot \vec{f} \right) - \nabla \cdot \left(\phi \vec{f} \right) dv \tag{B.8}$$

Here, if we assume Dirichlet boundary conditions for $\phi \mid_{\partial\Omega} = 0$, that is, the potentials are zero far away from the domain of interest, we can use the divergence theorem to eliminate the second term on the right-hand side of the equation. This results in the following equation for DC resistivity with Dirichlet boundary conditions on ϕ:

$$\left(\frac{1}{\sigma}\, \vec{j},\, \vec{f} \right) = \int_{\Omega} \phi \left(\nabla \cdot \vec{f} \right) dv \tag{B.9}$$

We use Dirichlet for simplicity in this example. In practice, Neumann conditions are often used because 'infinity' needs to be further away, if applying Dirichlet boundary conditions, since potential falls off as $1/r^2$ and current density as $1/r^3$. Similar techniques for the weak formulation of Maxwell equations can be derived by applying the appropriate vector identities and boundary conditions. In the following two sections, we will discuss the differential operators and the discretization of inner products that are prevalent in the weak formulation.

B.3 Operators

With the terminology and structure of the meshes well-defined, we can now create operators for the meshes. Although operators for averaging and interpolation are critical to any implementation, in this section, we will focus on the differential operators for the divergence, curl, and gradient. These operators take the form of sparse matrices, which are properties of each mesh. Although nuances exist in creating the operators for each mesh type, the basic building blocks come from the geometric concepts of individual cells, specifically the cell volume, face areas, and edge lengths. Given the cell spacings of tensor meshes, the computation of these properties is straightforward. For the cylindrical mesh, these values must be calculated in cylindrical coordinates. The volume and area calculations on the curvilinear mesh are straightforward in two dimensions. However, in three dimensions, the faces of the cell may not lie on a plane and, as a result, both the volume and face areas may not be well-defined. For face area, we use the average of the four parallelograms, which are calculated at each node of the face. As seen in Figure B.6, the cell volume is calculated by dividing the cell into five tetrahedrons and calculating the volume of each.

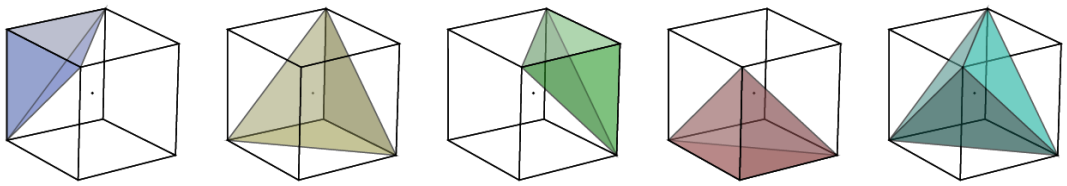

Figure B.6: Volume calculation using five tetrahedra.

B.3.1 Divergence

The divergence is the integral of a flux through a closed surface as that enclosed volume shrinks to a point.

$$\nabla \cdot \vec{f}(p) \stackrel{\text{def}}{=} \lim_{v \to \{p\}} \iint_{S(v)} \frac{\vec{f} \cdot \vec{n}}{|v|} \, dS \tag{B.10}$$

Since we have discretized and no longer have continuous functions, we cannot take the limit fully to a point. Instead, we approximate the limit around a finite volume: the cell. The flux out of the surface ($\vec{f} \cdot \vec{n}$) is exactly how we discretized $\vec{f}$ onto our mesh (i.e. $\mathbf{f}$), except that the face normal points out of the cell (rather than in the coordinate directions). We can readily calculate the surface areas and normals of all cells in the mesh. The flux values on each cell face are discretized and represented by a scalar value pointing in the direction of the face normal. As such, we need only multiply these values by the face area and multiply by ± 1 to ensure an outward facing normal with respect to the cell under consideration. To construct the divergence operator, $\mathbf{D}_1$, in one-dimension, the following discretization is used:

$$\mathbf{D}_1 \, \mathbf{f} = \mathbf{V}^{-1} \mathbf{D}_{\pm} \mathbf{S} \, \mathbf{f} = \mathsf{diag}\,(\mathbf{v})^{-1} \underbrace{\begin{bmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}}_{\mathbf{D}_{\pm}} \mathsf{diag}\,(\mathbf{s}) \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_n \\ \mathbf{f}_{n+1} \end{bmatrix} \tag{B.11}$$

where $\mathbf{V}$ is a sparse matrix with the cell volumes on the diagonal, $\mathbf{D}_{\pm}$ is a 'stencil' matrix that ensures outward pointing normals, and $\mathbf{S}$ is a diagonal matrix including the surface areas of each face. To move to higher dimensions, we exploit the logically rectangular structure and the column-ordered vectorization of the face variable. Kro-

necker products are used to place the difference matrix, $\mathbf{D}_{\pm}$, on the correct cells and faces. Not only is this conceptually efficient, when using an interpreted programming language, this also allows use of lower-level functions in compiled languages. For example, the divergence matrix in three-dimensions may be formed by:

$$\mathbf{D}_x = \mathbf{I}_3 \otimes \mathbf{I}_2 \otimes \mathbf{D}_1 \tag{B.12a}$$

$$\mathbf{D}_y = \mathbf{I}_3 \otimes \mathbf{D}_2 \otimes \mathbf{I}_1 \tag{B.12b}$$

$$\mathbf{D}_z = \mathbf{D}_3 \otimes \mathbf{I}_2 \otimes \mathbf{I}_1 \tag{B.12c}$$

where $\mathbf{D}_i$ is the difference matrix in one-dimension for the i^{th} dimension. The $\mathbf{I}_i$ is the identity matrix that has the length of the cells in the n^{th} dimension. Here the full divergence operator can be formed by:

$$\mathbf{D} = \mathbf{V}^{-1} \begin{bmatrix} \mathbf{D}_x & \mathbf{D}_y & \mathbf{D}_z \end{bmatrix} \mathbf{S} \tag{B.13}$$

The diagonal matrix, $\mathbf{S}$, contains the surface areas for each cell in the x, y, and z directions, concatenated on the matrix diagonal. As the divergence only takes account of fluxes into and out of a cell in the direction of the face normal, this concatenation works for any logically rectangular mesh, regardless of orthogonality. For a cylindrical mesh, we need to give attention to the middle cylindrical cell where the flux at $r = 0$ is known to be zero and, as such, this column can be removed from the $\mathbf{D}_r$ matrix.

For a tree mesh, we need to pay special attention to the hanging faces to achieve second-order convergence for the divergence operator. Although the divergence cannot be constructed through Kronecker product operations, the initial steps are exactly the same for calculating the stencil, volumes, and areas. These steps yield a divergence

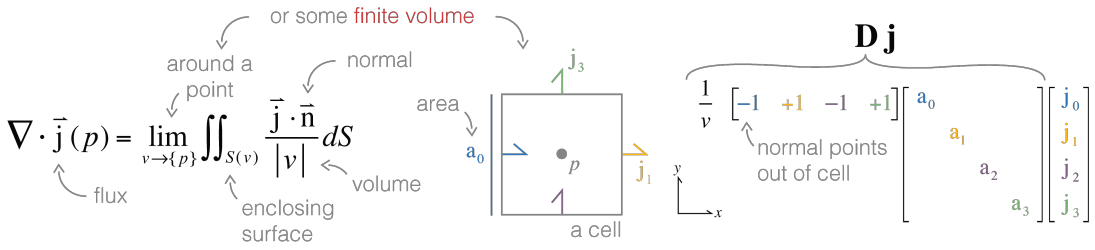

Figure B.7: Visual connection between the continuous and discrete representations of the divergence.

defined for every cell in the mesh using all faces. However, redundant information exists when including hanging faces. As seen in Figure B.8, the x-face between the cells $\{1,3\}$ and cell 4 has three locations for an x-face variable, but there is conceptually only a single flux at that location: x-face 4. As such, we can construct a matrix that identifies these hanging faces and assigns them to the same face variable. This matrix includes only ones and can be multiplied on the right-hand side of the unreduced divergence. This ensures that the flux into the negative x-face of cell 4 in Figure B.8 has a single numeric value.

B.3.2 Curl

Similar to the divergence operator, we rely on the geometric interpretation of the curl:

$$(\nabla \times \vec{e}) \cdot \hat{\mathbf{n}} \overset{\text{def}}{=} \lim_{s \to 0} \left(\frac{1}{|s|} \oint_c \vec{e} \cdot d\mathbf{r} \right) \tag{B.14}$$

where, $\vec{n}$ is the outward facing normal, s is the area of the face, and the line integral direction c is oriented positively with respect to the normal (i.e. right-hand rule). Figure B.9 shows the integrating directions for a unit cube in each unit direction. To discretize the curl of an edge variable, we must integrate along each edge in the appro-

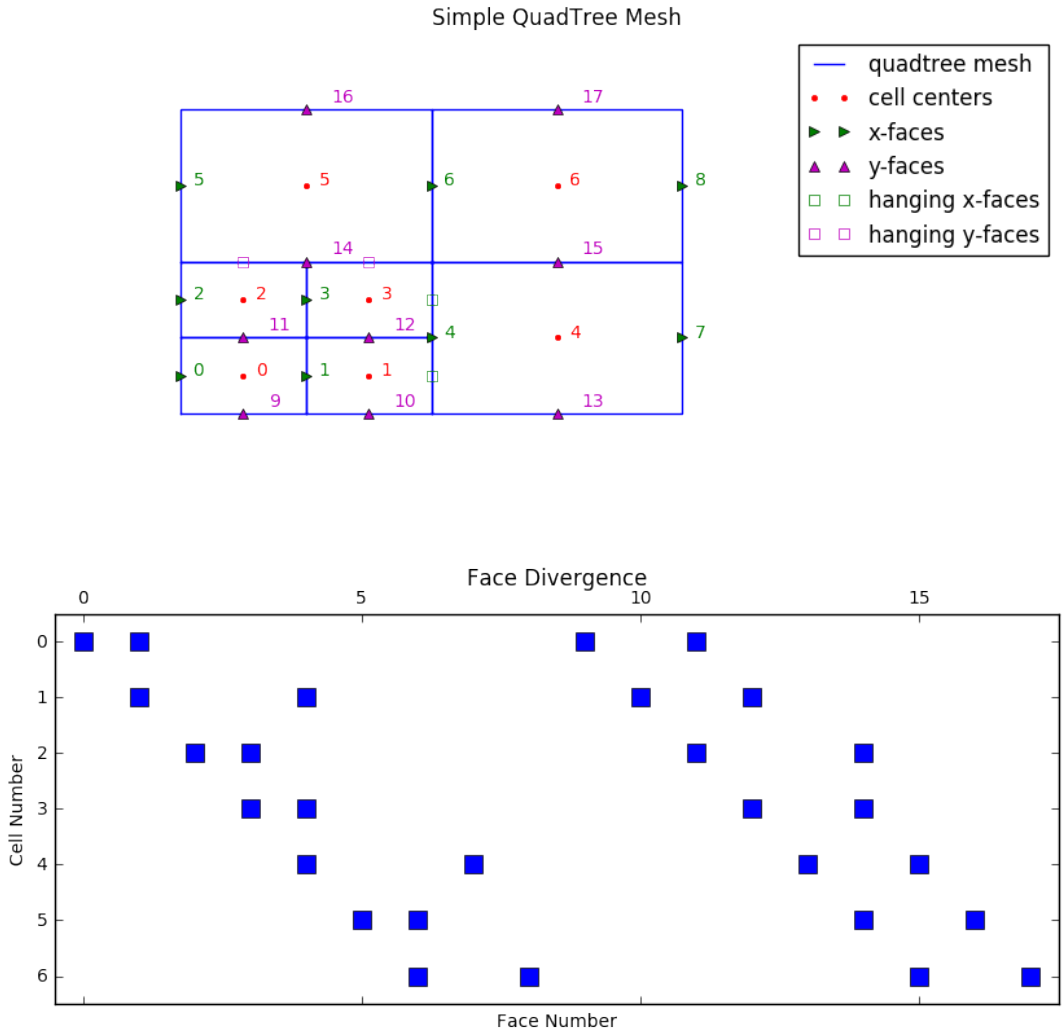

Figure B.8: Simple quadtree mesh showing (a) the mesh structure, cell numbering, and face numbering in both the x and y directions; and (b) the structure of the face divergence matrix that has eliminated the hanging faces.

priate direction (i.e. multiply by ± 1) and divide by the face area:

$$\mathbf{C} = \text{diag}\,(\mathbf{s})^{-1}\,\mathbf{C}_{\pm}\text{diag}\,(\mathbf{l}) \tag{B.15}$$

where $\mathbf{l}$ is an edge length vector, and $\mathbf{s}$ is the face area vector. The numeric curl on edges yields a vector variable on cell faces. Similar to the divergence operator,

this definition can exploit the logically rectangular nature of the mesh and create the difference matrix, $\mathbf{C}_{\pm}$, using Kronecker products. For the octree mesh, we need to

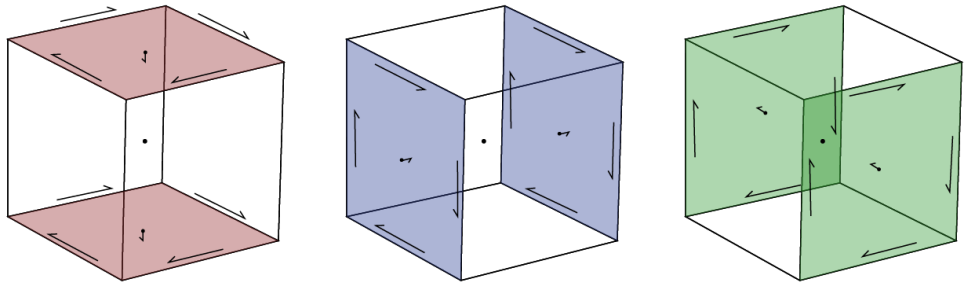

Figure B.9: Edge path integration or definition of the curl operator.

treat both the hanging edges and faces to remove redundant information. Hanging edges are treated differently than faces. We can average the resulting flux from the curl operation through a face, from the five estimates of the flux to a single value over the larger face. We multiply this averaging matrix on the left-hand side of the unreduced curl matrix. The edges, however, need to be eliminated through linear interpolation of the coarse edges to the six refined edges in each direction on each coarse face. We multiply this interpolation on the right-hand side of the unreduced curl matrix. These two operations result in a curl that has eliminated both hanging edges and hanging faces through interpolation and averaging, respectively.

B.3.3 Gradient

The gradient of a scalar function, $f(x)$, is a vector field whose dot product with an arbitrary vector, $v \in \mathbb{R}^d$, yields the directional derivative in that direction.

$$\big(\nabla f(x)\big) \cdot \mathbf{v} = D_{\mathbf{v}} f(x) \tag{B.16}$$

In Cartesian coordinates ($d = 3$), this has the much more familiar form:

$$\nabla f = \frac{\partial f}{\partial x}\mathbf{i} + \frac{\partial f}{\partial y}\mathbf{j} + \frac{\partial f}{\partial z}\mathbf{k} \tag{B.17}$$

To discretize the gradient of a nodal variable, we can do a forward difference along all edges of the mesh and divide by the length of the edge tangents

$$\mathbf{G} = \text{diag}\,(\mathbf{l})^{-1}\,\mathbf{G}_{\pm} \tag{B.18}$$

where $\mathbf{G}_{\pm}$ is the gradient stencil identifying the correct nodes, with ± 1, and $\mathbf{l}$ is the lengths of the edges. Multiplying the gradient by some nodal variable, $\mathbf{Gn}$, results in a vector quantity that is the directional derivative along the edge tangent, which is a central difference and results in a variable located on the edges on the mesh. Alternatively, we can formulate the gradient for cell-centered, rather than nodal, variables. This formulation explicitly considers neighbouring cells, rather than just a single finite volume, and is a finite difference method. Although possible, this formulation is more cumbersome to represent on both tree meshes and curvilinear meshes or when anisotropy is considered. When a gradient is necessary in a differential equation, and the variable is located on cell centers instead of nodes, it is usually possible to rearrange the equations, using vector identities, to use the negative transpose of the divergence, as previously discussed.

For both the quadtree and octree meshes, we must once again deal with the hanging nodes and hanging edges. The treatment here is similar to that in the curl matrix. We interpolate hanging nodes from their neighbours (two in 2D, four in 3D) and the result of the gradient, an edge vector on all edges, is averaged to exclude the hanging edges.

Since we are using a staggered grid with centered differences, the discretization of the differential operators is second-order. That is, as we refine the mesh, our approximation of the divergence should improve by a factor of two. We can verify this numerical convergence using simple test functions with analytic expressions, well-chosen boundary conditions, and known derivatives. The assertion of this expectation of order is a critical piece of any numerical implementation.

B.4 Inner products

Evaluating volume integrals over the cell becomes important when formulating equations in weak form, as in Section B.2.5. We can numerically evaluate these volume integrals in many different ways, however, the method must take into account the location and number of approximations of the integrand variables that are available. For two cell-centered variables, for example, the evaluation of the integral is simple and can be calculated by the midpoint approximation. The result is an *inner product* that includes a volume term, $\mathbf{v}$, for each cell:

$$\begin{aligned}(a,b) &= \int_\Omega a(\vec{x}) \cdot b(\vec{x})\ \partial v \\ &\approx \mathbf{a}^\top \operatorname{diag}(\mathbf{v})\, \mathbf{b}\end{aligned} \tag{B.19}$$

Complications arise when we approximate the inner product but the variables do not live in the same location; that is, on the edges, faces, and cell centers.

B.4.1 Face inner product

We use the face inner product when there is a multiplication between a cell-centered variable and a vector variable located on the faces. In the following example, we will use a cell-centered variable, σ, which is fully anisotropic, meaning that, in 3D, each

cell is represented by a 3×3 tensor. We multiply the tensor by $\vec{j}$ and take the inner product with a face variable $\vec{f}$. When discretizing this integral, recall that the discretization has approximations, $\mathbf{j}$ and $\mathbf{f}$, on each face of the cell. In 2D, that means two approximations of $\mathbf{j}_x$ and two approximations of $\mathbf{j}_y$. In 3D, we also have two approximations of $\mathbf{j}_z$. Additionally, in the general case, these vectors may not be orthogonal nor axis aligned and so we must first transform them into their Cartesian components before multiplying by the discretized tensor, Σ. Regardless of how we choose to approximate the vectors, $\mathbf{j}_{\text{cart}}$ and $\mathbf{f}_{\text{cart}}$, we can represent this in vector form for every cell.

$$\begin{aligned} \left(\sigma^{-1}\ \vec{j}, \vec{f}\ \right) &= \int_{\Omega} \sigma^{-1} \vec{j} \cdot \vec{f}\ \partial v \\ &\approx \mathbf{j}_{\text{cart}}^{\top} \left(\sqrt{v_{\text{cell}}}\ \Sigma^{-1} \sqrt{v_{\text{cell}}}\right) \mathbf{f}_{\text{cart}} \end{aligned} \tag{B.20}$$

We multiply by square-root of volume on each side of the tensor, Σ to keep symmetry in the system. Here, $\mathbf{j}_{\text{cart}}$ is an approximation of the Cartesian $\vec{j}$ that must be calculated from known locations on the mesh, $\mathbf{j}_{\text{mesh}}$. There are many different ways to evaluate the inner product $\left(\sigma^{-1}\ \vec{j}, \vec{f}\ \right)$: we could approximate the integral using trapezoidal, midpoint, or higher order approximations, for example. A simple, second-order method is to break the integral into a sum of 2^d sections and apply the midpoint rule, where d is the dimension of the mesh (i.e. two in 1D, four in 2D, and eight in 3D). For each of these sections, the midpoint rule uses the closest components of $\mathbf{j}$ to compose a Cartesian vector. We use a $\mathbf{Q}_i$ matrix of size 2×2^d consisting of 0s and 1s, to select the appropriate faces to compose the corresponding vector $\mathbf{j}_{\text{cart}}$.

$$\mathbf{j}_{\text{cart}}^{i} = \mathbf{N}_i^{-1} \mathbf{Q}_i\ \mathbf{j}_{\text{mesh}} \tag{B.21}$$

Here, the i index refers to the section where we choose to approximate this integral. In a curvilinear mesh, the fluxes chosen by $\mathbf{Q}_i$ are not necessarily axis-aligned and also may not be mutually orthogonal. As such, we must use a normalization to get back to Cartesian components, where multiplication with the Σ is defined. These projection matrices, $\mathbf{N}_i$, are completed 2^d times for every cell of the mesh and, in 2D, have the form:

$$\begin{aligned}
\overbrace{\begin{bmatrix} j^{(1)} \\ j^{(3)} \end{bmatrix}}^{\mathbf{Q}_{(i)}\mathbf{j}_{\text{mesh}}} &= \overbrace{\begin{bmatrix} n_x^{(1)} & n_y^{(1)} \\ n_x^{(3)} & n_y^{(3)} \end{bmatrix}}^{\mathbf{N}_{(i)}} \overbrace{\begin{bmatrix} j_x \\ j_y \end{bmatrix}_{(1)}}^{\mathbf{j}_{\text{cart}}}, \\
\begin{bmatrix} j^{(1)} \\ j^{(4)} \end{bmatrix} &= \begin{bmatrix} -\mathbf{n}_{(1)}^\top - \\ -\mathbf{n}_{(4)}^\top - \end{bmatrix} \begin{bmatrix} j_x \\ j_y \end{bmatrix}_{(2)}, \\
\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{j}_{\text{mesh}} &= \begin{bmatrix} -\mathbf{n}_{(2)}^\top - \\ -\mathbf{n}_{(3)}^\top - \end{bmatrix} \mathbf{j}_{\text{cart}}^{(3)}, \\
\mathbf{Q}_{(4)}\, \mathbf{j}_{\text{mesh}} &= \mathbf{N}_{(4)}\, \mathbf{j}_{\text{cart}}^{(4)}
\end{aligned} \tag{B.22}$$

where $\mathbf{n}_{(i)}$ is the normal to face i. Solving for the Cartesian flux, $\mathbf{j}_{\text{cart}}^{(i)}$, requires inverting a small matrix (2×2 or 3×3) for each section. We now have eight evaluations in 3D of the midpoint rule, using various approximations for $\mathbf{j}_{\text{cart}}$. We can sum these approximations together to define the face inner product matrix, $\mathbf{M}^f_{\Sigma^{-1}}$, for a logically rectangular mesh of any dimension, d.

$$\begin{aligned}
\mathbf{M}^f_{\Sigma^{-1}} &= \frac{1}{2^d} \left(\sum_{i=1}^{2^d} \mathbf{Q}_i^\top \mathbf{N}_i^{-\top} \sqrt{v_{\text{cell}}}\, \Sigma^{-1} \sqrt{v_{\text{cell}}}\, \mathbf{N}_i^{-1} \mathbf{Q}_i \right) \\
&= \sum_{i=1}^{2^d} \mathbf{P}_i^\top \Sigma^{-1} \mathbf{P}_i, \qquad \text{where} \\
\mathbf{P}_i &= \sqrt{\frac{1}{2^d} \mathbf{I}^d \otimes \text{diag}\,(\mathbf{v})} \mathbf{N}_i^{-1} \mathbf{Q}_i
\end{aligned} \tag{B.23}$$

Here, each $\mathbf{P} \in \mathbb{R}^{(d*n_c,\, n_f)}$ is a combination of the projection, volume, and any normalization to Cartesian coordinates. In a numerical implementation, it is often more

efficient to complete this operation for each cell in the mesh at the same time, and return the sparse matrices for caching as this is a common operation in any inverse problem that estimates a cell-centered variable. For the tree mesh, we complete the face inner product identically as above, except for a final step to exclude the redundant hanging faces in the same way as the divergence matrix.

B.4.2 Edge inner product

The edge inner product with a vector that is defined on the *edges* of a cell (rather than on the faces, as above) has a similar derivation. The difference comes in 3D, when selecting the edges around each node, as there are twelve edges instead of only six faces; these edges are selected by the $\hat{\mathbf{Q}}_i$ matrix. Similarly, in the normalization to Cartesian coordinates, we use the edge tangent directions instead of using the face normals. The Cartesian edge values, $\mathbf{e}^c_{(i)}$, are projected using the tangents:

$$\mathbf{e}^{(i)}_{\text{cart}} = \mathbf{T}^{-1}_{(i)}\ \hat{\mathbf{Q}}_{(i)}\ \mathbf{e}_{\text{mesh}} \tag{B.24}$$

where $\hat{\mathbf{Q}}_{(i)}$ is the projection matrix that selects the edges closest to each midpoint approximation. Once again, we compute the inner product by the mass matrix acting on the edges:

$$\mathbf{M}^e_\rho = \frac{1}{2^d}\left(\sum_{i=1}^{2^d} \hat{\mathbf{Q}}_i^\top \mathbf{T}_i^{-\top} \sqrt{v_{\text{cell}}}\ \rho\ \sqrt{v_{\text{cell}}}\ \mathbf{T}_i^{-1} \hat{\mathbf{Q}}_i\right) \tag{B.25}$$

For the tree mesh, we complete the edge inner product as above, except with an added step to deal with the hanging edges. This step excludes the hanging edges through linear interpolation in the same way as the curl matrix.

B.4.3 Tensor product mesh

The generality of this equation can be reduced when dealing with an axis-aligned tensor mesh with an isotropic physical property, σ. In this case, we do not need to take into account the normalization from $\mathbf{N}_i^{-1}$. For a tensor mesh, the face inner product can be calculated as follows and can be interpreted as averaging the physical property between neighboring cells:

$$\mathbf{M}^f_{\sigma^{-1}} = \text{diag}\left(\mathbf{A}_v^\top (\mathbf{v} \odot \sigma^{-1})\right) \tag{B.26}$$

where $\odot$ is a Hadamard product for point-wise multiplication and $\mathbf{A}_v \in \mathbb{R}^{(n_f,\, n_c)}$ is an averaging matrix from faces to cell centers. In one dimension this matrix has the form

$$\mathbf{A}_v^{(1)} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & & \\ & \ddots & \ddots & \\ & & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{B.27}$$

The matrix has slight differences on a cylindrical tensor mesh to exclude the $r = 0$ face. The 'averaging' matrix can be made for higher dimensions using Kronecker products and horizontal concatenation:

$$\mathbf{A}_v = \left[\mathbf{I}_3 \otimes \mathbf{I}_2 \otimes \mathbf{A}_v^{(1)}, \quad \mathbf{I}_3 \otimes \mathbf{A}_v^{(2)} \otimes \mathbf{I}_1, \quad \mathbf{A}_v^{(3)} \otimes \mathbf{I}_2 \otimes \mathbf{I}_1\right] \tag{B.28}$$

Note that this matrix is often referred to as the face-to-cell-center averaging matrix and, as such, is often divided by the dimension of the mesh to be a true average (i.e. there are three face averages above). If this is the case, care needs to be taken to restore this constant when calculating the inner product. Although general anisotropy is not easily added in this form, coordinate anisotropy can be added by composing this matrix as

block diagonals instead of horizontal concatenation. Similarly, on a cylindrical mesh, if anisotropy is necessary, care should be taken that the anisotropy uses Cartesian, rather than cylindrical, coordinates, unless intended.

B.4.4 Anisotropy

For defining isotropic, coordinate anisotropic, and fully anisotropic parameters, the following conventions are used in 3D:

$$\vec{\sigma} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_1 & 0 \\ 0 & 0 & \sigma_1 \end{bmatrix} \qquad \vec{\sigma} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} \qquad \vec{\sigma} = \begin{bmatrix} \sigma_1 & \sigma_4 & \sigma_5 \\ \sigma_4 & \sigma_2 & \sigma_6 \\ \sigma_5 & \sigma_6 & \sigma_3 \end{bmatrix} \tag{B.29}$$

In 2D, these conventions are similarly defined as:

$$\vec{\sigma} = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_1 \end{bmatrix} \qquad \vec{\sigma} = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \qquad \vec{\sigma} = \begin{bmatrix} \sigma_1 & \sigma_3 \\ \sigma_3 & \sigma_2 \end{bmatrix} \tag{B.30}$$

Both the isotropic and coordinate anisotropic material properties result in a diagonal mass matrix on a tensor mesh. This is easy to invert if necessary. However, in the fully anisotropic case, or for any curvilinear mesh, the inner product matrix is not diagonal, as can be seen for a 3D mesh in the figure below.

B.4.5 Derivatives

In the context of parameter estimation, we are often interested in the parameters inside inner products and require efficient matrix-free derivatives for these elements. The derivative of these inner products is actually a tensor. However, the derivative is a matrix if we only require the computation of this product with a vector (as is usually

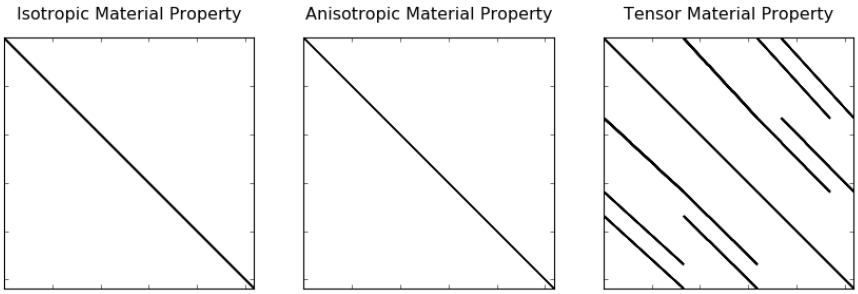

Figure B.10: Matrix structure of a face inner product of a cell centered physical property on a tensor mesh.

the case). To show the computation of the inner product derivatives, we will consider a fully anisotropic tensor for a 3D logically rectangular mesh. Let us start with one part of the sum which makes up $\mathbf{M}_\Sigma^f$ and take the derivative when this matrix is multiplied by some vector, $\mathbf{w}$:

$$\mathbf{P}_i^\top \Sigma \mathbf{P}_i \mathbf{w} \tag{B.31}$$

Here, we will let $\mathbf{P}_i\mathbf{w} = \mathbf{y}$ and $\mathbf{y}$ will have the form:

$$\mathbf{y} = \mathbf{P}_i\mathbf{w} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \end{bmatrix} \tag{B.32}$$

This matrix can subsequently be multiplied by $\mathbf{P}_i^\top \Sigma$. When multiplying these matrices by hand, we can see that they have the form:

$$\mathbf{P}_i^\top \Sigma \mathbf{y} = \mathbf{P}_i^\top \begin{bmatrix} \sigma_1 & \sigma_4 & \sigma_5 \\ \sigma_4 & \sigma_2 & \sigma_6 \\ \sigma_5 & \sigma_6 & \sigma_3 \end{bmatrix} \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \end{bmatrix} = \mathbf{P}_i^\top \begin{bmatrix} \sigma_1 \odot \mathbf{y}_1 + \sigma_4 \odot \mathbf{y}_2 + \sigma_5 \odot \mathbf{y}_3 \\ \sigma_4 \odot \mathbf{y}_1 + \sigma_2 \odot \mathbf{y}_2 + \sigma_6 \odot \mathbf{y}_3 \\ \sigma_5 \odot \mathbf{y}_1 + \sigma_6 \odot \mathbf{y}_2 + \sigma_3 \odot \mathbf{y}_3 \end{bmatrix} \tag{B.33}$$

where $\odot$ is the Hadamard product, and represents element-wise multiplication. We can now take the derivative with respect to any one of the σ parameters, for example, $\frac{\partial}{\partial \sigma_1}$

$$\frac{\partial}{\partial \sigma_1}\left(\mathbf{P}_i^\top \Sigma \mathbf{y}\right) = \mathbf{P}_i^\top \begin{bmatrix} \text{diag}(\mathbf{y}_1) \\ 0 \\ 0 \end{bmatrix} \tag{B.34}$$

Meanwhile, the derivative for $\frac{\partial}{\partial \sigma_4}$ is:

$$\frac{\partial}{\partial \sigma_4}\left(\mathbf{P}_i^\top \Sigma \mathbf{y}\right) = \mathbf{P}_i^\top \begin{bmatrix} \text{diag}(\mathbf{y}_2) \\ \text{diag}(\mathbf{y}_1) \\ 0 \end{bmatrix} \tag{B.35}$$

These derivatives are calculated for each of the eight components of the discretized inner product (four in 2D).

For a tensor product mesh with an isotropic physical property, this is again not the most efficient method of calculation. Instead, we can recall an alternative formula for calculating the inner product and take the derivative of that formula. We still must

multiply by an arbitrary vector, $\mathbf{w}$:

$$\begin{aligned}\frac{d}{d\boldsymbol{\sigma}}\left(\mathbf{M}^f_{\boldsymbol{\sigma}}\mathbf{w}\right) &= \frac{d}{d\boldsymbol{\sigma}}\left(\text{diag}\left(\mathbf{A}_v^\top(\mathbf{v}\odot\boldsymbol{\sigma})\right)\mathbf{w}\right) \\ &= \text{diag}\left(\mathbf{w}\right)\mathbf{A}_v^\top\,\text{diag}\left(\mathbf{v}\right)\end{aligned} \tag{B.36}$$

Here, we calculate the derivative by exploiting the fact that a diagonal matrix and a vector can be multiplied in either order; that is, they are commutative.

$$\begin{aligned}\mathbf{a}\odot\mathbf{b} &= \mathbf{b}\odot\mathbf{a} \\ &= \text{diag}\left(\mathbf{a}\right)\mathbf{b} \\ &= \text{diag}\left(\mathbf{b}\right)\mathbf{a}\end{aligned} \tag{B.37}$$

The derivative, with respect to physical properties, becomes critical for the inverse problem, and is often not provided when implementations are solely focused on the forward simulation. In the implementation presented, we have given thought to accessing the derivatives of isotropic and anisotropic physical properties in an efficient way for all mesh types under consideration.

B.5 Implementation

The `discretize` package (http://discretize.simpeg.xyz) allows for the explicit access to derivatives with respect to the inner product operations. This is due to our focus on the inverse problem and the construction of sensitivities and adjoint sensitivities implicitly, through a matrix vector multiplication. Many other packages for finite volume or finite element simulation either focus entirely on the forward simulation (e.g. Guyer et al. (2009)) or explicitly create the sensitivity matrices with automatic differentiation and/or finite differencing (Ketcheson et al., 2012; LeVeque, 1997; Alexe and

Sandu, 2014; Hindmarsh et al., 2005; Jasak et al., 2007). Haber has developed implementations in Ruthotto et al. (2016) and Haber (2015) which are written in Julia and Matlab, respectively. We chose Python for the implementation because: (a) Python has a large and growing scientific community, including existing and maintained tools for matrix solvers and sparse matrix operations (eg. SciPy, pymatsolver); (b) it is an object-oriented language (unlike Julia), which allows the relationships between the mesh types to be expressed through base class inheritance and subtype polymorphism, where appropriate; and (c) it is a high-level language that has the ability to interface with low-level codes in Fortran and C, which allows for efficient creation of declarative scripts while leveraging existing work in lower-level languages. If our primary interest was in finite volume techniques, it may have been more sensible to choose a language such as Julia (or Matlab, which is proprietary), which has better built-in capabilities to represent sparse matrix operators and defaults to matrix multiplication, rather than array multiplication. However, finite volume techniques are not the endgame of this work. Rather, we wish to use these techniques in conjunction with geophysical inverse problems, which consists of much more than the numerical implementation and requires many additional utilities for critical tasks, such as scripting, interactive programming, reading files, 3D visualization, and communicating results.

B.5.1 Organization

The object-oriented programming model in Python allows organization of the finite volume methods for the various meshes into a class inheritance structure. This organization has highlighted the similarities between techniques through elimination of redundant code between shared concepts and methods. Such elimination leads to a concise description of the differences between the methods, which was outlined in the

previous section. We show our chosen organization in its entirety in Figure B.11. This figure shows the class inheritance and class properties of the `discretize` package and is best viewed in digital form.

There is a `BaseMesh` that has properties such as number of cells, nodes, faces, and edges `nC, nN, nF` and `nE`, respectively. We have decided to separate the differential, averaging and interpolation operators into their own class such that they can be included as a mixin (rather than inherited) into the final mesh classes. Methods for the inner products, IO to common formats, and visualization are also separated out and included as mixins. We have separated the concepts of being logically rectangular and being a tensor product mesh because these are different concepts. The basic counting of cells is on the rectangular mesh and the concept of a 'cell-centered vector in the x dimension' is only for tensor product meshes. This inheritance demonstrates the concepts that: (a) the curvilinear mesh is rectangular, but not a tensor mesh; (b) the tree mesh is a tensor mesh, but not logically rectangular; and (c) the cylindrical mesh is both logically rectangular and a tensor mesh, however, through subtype polymorphism the cylindrical mesh overwrites the concepts for geometric calculations of volume, surface area, etc. as well as the counting for nodes, faces, and edges.

B.5.2 User interface

Our goal for the implementation is to create a common programmatic terminology for working with finite volume techniques. By sharing the numerical implementation, not only is this 'language' precise, but it can also be tested for accuracy. A brief description of the implementation and use in practice, as well as a look at some of the major properties on the abstract mesh types in Table 2.1, was previously given in Section 2.3.4. As the implementation is openly available, we will point the reader

to both the online, up-to-date documentation of each specific property and method (http://discretize.simpeg.xyz) and to the 400+ unittest results of numerical convergence (https://travis-ci.org/simpeg/discretize). As a brief overview, we use the convention of `C, N, F, E` to refer to cells, nodes, faces and edges, respectively. For example, for a tensor mesh the number of cells in the x-dimension would be called `nCx`, the nodal tensor of x-locations as a vector is called `vectorNx`, and the location of all face variables with an x-component is called `gridFx`. We named differential operators by the variable location that they act upon; for example: `faceDiv` and `edgeCurl`. These names allow us to define multiple operators. For example, the gradient can either operate on cell centers or nodes, `cellGrad` and `nodalGrad`, respectively. This language is common across all meshes considered and extends to other mesh types not yet implemented. As such, geophysical simulation codes can be built on top of this work to write PDEs in a declarative way, which is agnostic to the mesh implementation actually used. In collaboration with Heagy, Kang, Rosenkjaer and Mitchell simulations have been completed for time, frequency, and static implementations of electromagnetics using 1D, 2D, and 3D versions of the tensor, tree, curvilinear and cylindrically symmetric meshes (Rosenkjaer et al., 2016; Kang et al., 2015a; Heagy et al., 2015c, 2014; Cockett et al., 2014b; Kang et al., 2014; Heagy et al., 2015a; Cockett and Haber, 2013a; Cockett et al., 2016a; Heagy et al., 2016; Rosenkjaer et al., 2015a; Heagy et al., 2015b). The terminology developed defines a clear interface, which allows for improvements in speed and functionality of the `discretize` package to transparently improve geophysical simulations.

The `discretize` interface allows for lazy loading of properties; that is, all properties of the mesh are created on demand and then stored for later use. This caching of properties is important, as not all operators are useful in all problems and, as such, are

not created. The implementation here is different from some other finite volume implementations, as the operators are held in memory as matrices and are readily available for interrogation. We find this feature beneficial for educational and research purposes, as the discretization remains visually very close to the math, and the matrices can be manipulated, visually inspected, and readily combined.

The major difference between the mesh types is the instantiation of each type. A regular tensor mesh that is defined on the unit cube can be instantiated with integers. For example, `TensorMesh([4, 8])` will create a 2D tensor mesh with equal spacings (a regular mesh) with `nCx = 4` and `nCy = 8` that has a domain over the unit square. Similarly, `TensorMesh([hx, hy, hz])`, where $\mathbf{h}_x$ is a vector of variable cell spacings in the x dimension, will create an irregularly spaced tensor mesh in 3D. To produce a cylindrically symmetric mesh (i.e. with a single cell in the azimuthal, θ, dimension), we can mix these notations. For example, `CylMesh([hr, 1, hz])`. For the definition of a `CurvilinearMesh`, all node locations must be specified in each dimension. These node locations may be provided as a list of either 2D or 3D matrices. For a tree mesh, a base mesh of tensor products are provided (which must be a power of 2), which represents the location of nodes at the lowest refinement level. A refinement function must also be provided. The refinement function is recursively passes cell centers and chooses whether that cell should be refined. This functional construction of the tree meshes encourages composable functions that refine based on, for example, location from a source and/or distance from a topographic interface. Alternatively, the mesh can be loaded from several formats (e.g. the UBC mesh and model files or the Open Mining Format (*.omf)). Once we have constructed the mesh, there is access to the differential operators, averaging and interpolation functions, and utilities for visualization and export.

B.6 Numerical examples

Here we will briefly explore the application of the curvilinear mesh for a DC resistivity problem, which was introduced in Section B.2.4. We will explore the equations for a tensor mesh and a curvilinear mesh over a unit cube with Neumann boundary conditions. We tested the forward operators for analytical potential fields with the appropriate boundary conditions. A series of electrode arrays (surveys) were written to produce and collect data from the forward model. The survey used in this paper considered all receiver permutations in a grid on the top surface of the model. We note that it is not possible to experimentally collect data at the same location as the source electrodes; we discarded these permutations.

For the numerical experiments presented here, we use a true model with a geologic interface with varying elevation. A cross-section through the 3D model at a mid-range discretization is seen in Figure B.12. The layer above the interface has a conductivity of 1 Sm^{-1} and the layer below the interface has a conductivity of 100 Sm^{-1}. We produced data from the forward operator, with the true model discretized, using $45 \times 45 \times 45$ cells. We created a series of models that ranged from $5 \times 5 \times 5$ to $40 \times 40 \times 40$, over the same domain. At each discretization level, the true model was down-sampled onto a regular mesh as well as a curvilinear mesh that was aligned to the interface. The survey setup was a grid of 4×4 equally spaced electrodes centered on the top surface of the model. There were a total of 16 electrodes, 120 source configurations, and 91 active measurements per source dipole. This gave rise to 10,920 total measurements, half of which are symmetric and likely would not have been collected in a field experiment, but were collected in this numerical experiment. We compare the data collected from each of the test models directly to the large model's data and plot

the norm in Figure B.13. The mesh that is aligned to the layer performs significantly better at lower discretizations because it is more accurate at resolving the topographic interface. For example, at a norm data error of 10^0, the mesh aligned to the layer needed 16^3 cells, versus 27^3 when we used a rectangular discretization - or nearly five times the number of cells. The changes in error at coarse discretizations is due to the low accuracy in modeling the location of the sources and receivers.

The logically rectangular mesh allows increased degrees of freedom when placing the nodes of the mesh. In simple situations it is possible to significantly improve accuracy of the numerical model at reduced computational costs. However, the increased freedoms in picking node locations forces additional thought in mesh creation and alignment. Issues of mesh creation can be complex and these problems must be handled appropriately. It is suggested that simple meshes (i.e. regular) be used when possible and to only use the logically rectangular mesh when known layers, such as topography, are well-defined and known to significantly influence the solution of the problem and data collected.

B.7 Conclusions

Discretization techniques are necessary in every aspect of computational geophysics and have been used extensively throughout my thesis and collaborative work. Many of the components in the discretization require special attention when considering the inverse problem; most notably, derivatives to the inner products and choosing when to cache operators. In this chapter, I have provided a description of the finite volume techniques that are necessary to discretize and simulate many of the elliptic and parabolic partial differential equations that are common in electromagnetic geophysics and hydrogeologic fluid flow. I have provided the derivations in a general

form, such that they apply to four different types of meshes in common use in geophysical inverse problems. This generality allows for differences between the mesh types to be highlighted and discussed. I have also provided an open source, permissively licensed implementation of this work. The Python implementation, called `discretize` (http://discretize.simpeg.xyz), is object-oriented and highlights the concepts and inheritance structure of the meshes under consideration. The numerical example highlights these meshes in use for a direct current resistivity problem and briefly discusses some of the numerical advantages and disadvantages of each mesh type.

B.7.1 Continuing work

A number of improvements and extensions that have yet to be tackled at the time of writing. Among these are (in no particular order): (a) improved ease of use around boundary conditions; (b) more utilities for mesh creation especially for more complicated meshes (i.e. curvilinear and tree); (c) an increase in the combinatorial nature of existing meshes, for example, cylindrical or curvilinear octrees; (d) extension to unstructured meshes, voronoi meshes, and other coordinate systems (e.g. spherical); (e) a more rigorous comparison to finite element codes (cf. Jahandari et al. (2017)); and (f) integration to existing mesh generation packages. By providing this package in an open, standalone, tested, documented form we hope that the implementation can be improved by the growing community of contributors.

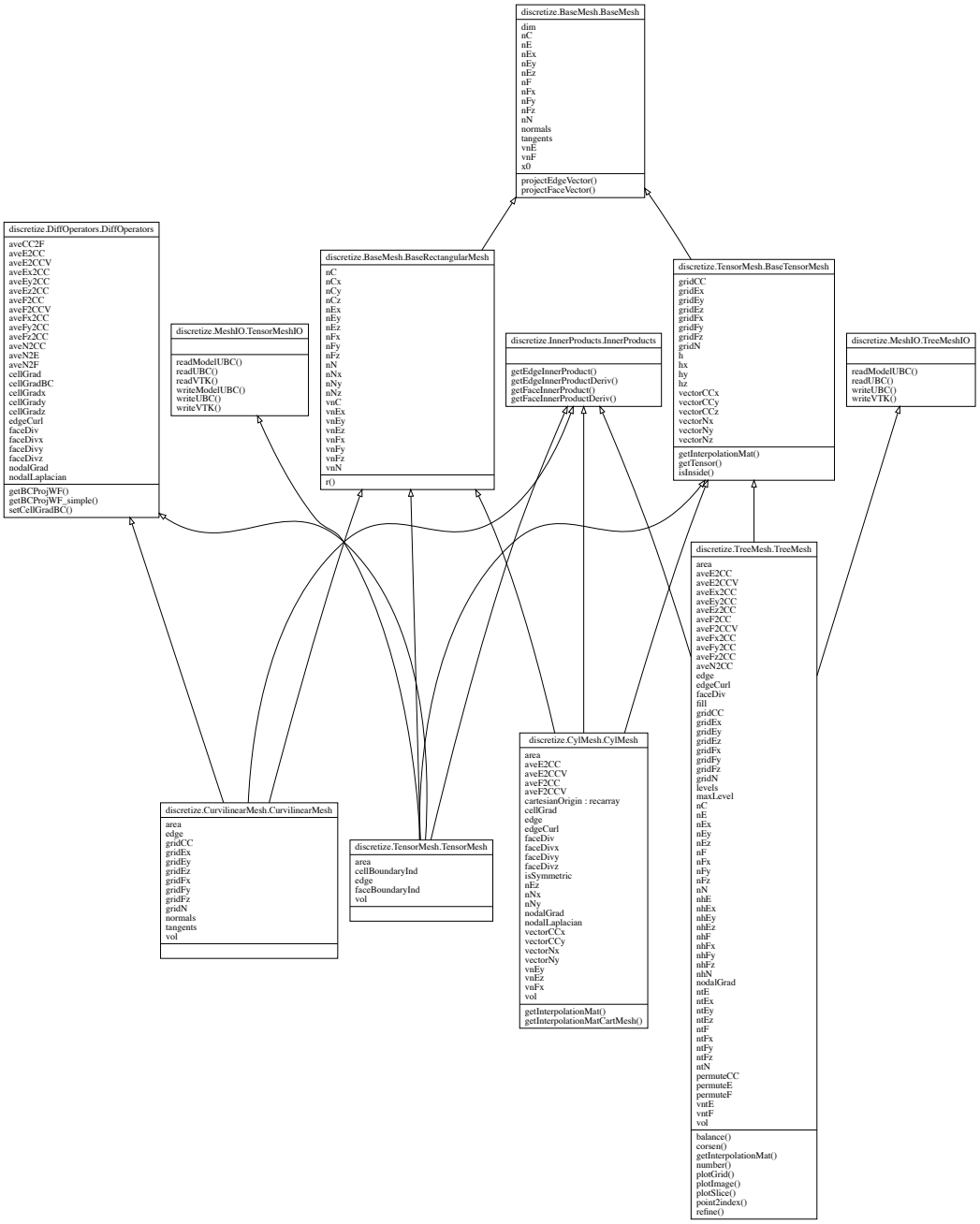

Figure B.11: The computational ontology developed for the `discretize` package showing inheritance and commonalities between the four mesh types.

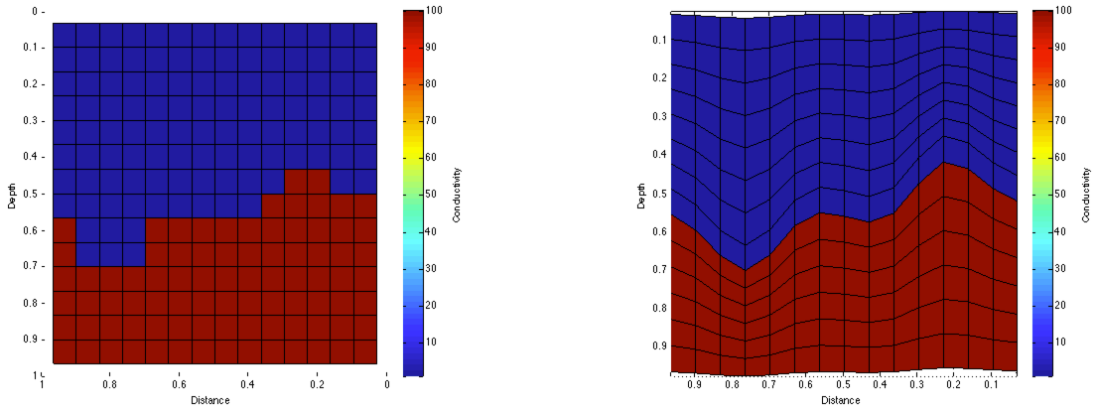

Figure B.12: Regular mesh and mesh aligned to layer for a simple conductivity model at $14 \times 14 \times 14$.

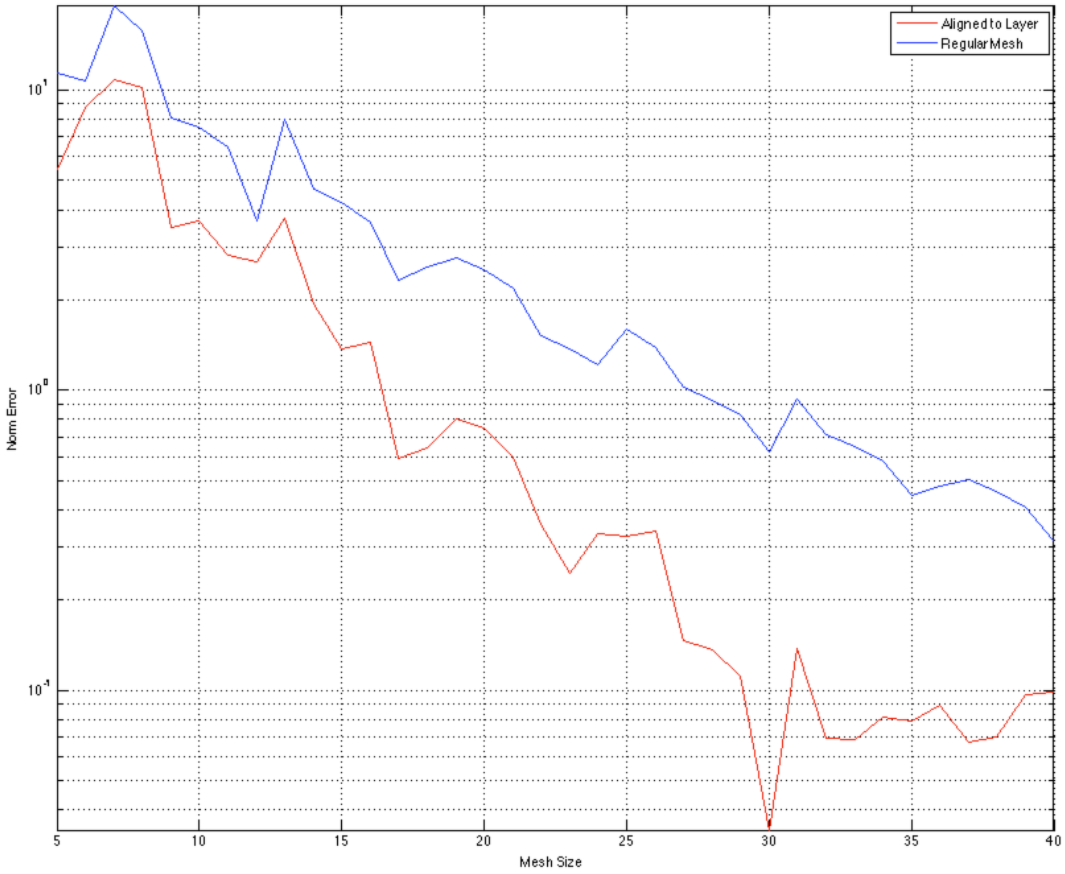

Figure B.13: Comparison of norm data error for the regular mesh and the mesh aligned to the interface.

Appendix C

Interfaces and extensions

C.1 Introduction

Incorporating and quantitatively capturing a-priori and hydrologic and geologic information is a perennial problem in geophysics. Many methods already exist to include geologic information in geophysical simulations and inversions. Broadly, these methods include ways of formulating: (a) the inverse problem: through regularization, objective functions, or constraints (cf. Williams (2008)) and (b) the forward simulation: through parameterization of physical properties, model conceptualization, and the dimensionality or physical equations used in the simulation (e.g. Oldenburg and Li (2005); Li and Oldenburg (1998, 1996); Pidlisecky et al. (2007); Li et al. (2010); Pidlisecky et al. (2011); McMillan et al. (2015); Kang et al. (2015a)). Section 2.2.2 gives the inversion elements, inverse problem formulation, and a brief discussion of choices in the regularization. Furthermore, the inverse formulation and the addition of regularization can largely be done in an external way. The intricacies inherent in the forward simulation, however, were largely neglected in the general discussion of the

framework. In this appendix, we will explore a number of case studies that are driven by the context of the inverse problem. The case studies span vadose zone flow, direct current resistivity, time and frequency domain electromagnetics, and simple structural geologic modelling.

C.1.1 What is your model?

In all of these case studies, we focus on the question **'What is your model?'**; that is, how does the framework approach enable custom model conceptualizations? The standard approach in exploration geophysics of a 3D distribution of voxels is both general and widely applicable. However, this approach falls short when integrating with fields of hydrology or geology. For example, Pidlisecky et al. (2011) uses a hydrologic conceptualization for a geophysical model and inverts directly for the spatial morphology of a solute plume through a moment-based description. McMillan et al. (2015) takes a similar approach to invert time domain electromagnetic data for a thin geologic unit of variable dip. In direct coupling of geophysical and hydrologic inverse problems, the parameterization of the model must be flexible and extensible by researchers who are asking *new scientific questions*.

The focus on the forward simulation is to expand on the capabilities of the framework developed in Chapter 2. The forward simulation framework must have the flexibility to support and interface to arbitrary parameterizations. A general architecture requires that derivatives be calculated with respect to: (a) multiple physical properties in the physical equations (e.g. electrical conductivity, hydraulic conductivity, and magnetic permeability); (b) the sources (e.g. waveform, transmitter location, and boundary conditions); (c) estimation of additional fields/fluxes from the solution of the PDE (e.g. saturation field from pressure head); and, (d) from the measurements

(e.g. location and orientation of data collection). In order to support the custom parameterizations that are necessary for a unique decision or prediction, this architecture must provide building blocks that are independently extensible. The PEST framework for model independent parameter estimation and uncertainty analysis provides a concrete example of where this has previously been done with success (Doherty, 2004). The software is widely cited in academia ($>$ 2K citations) especially in hydrology and hydrogeophysics, and is heavily used in industry. The advantage of being model-independent has given this technique wide application due to the flexibility to adapt to new scientific questions. However, this flexibility also comes at quite a cost because the structure of the simulation and modelling cannot be used to the advantage of the algorithm. As with the Richards equation or electromagnetics, when moving to three dimensions there may be hundreds of thousands to millions of parameters to estimate. Not taking the structure of the problem into account severely limits the types and sizes of problems that can be considered. In the following sections, we will briefly describe some of the necessary components to simultaneously support a breadth of custom parameterizations. These abstractions have been possible through collaborative work across multiple fields, including electromagnetics, fluid flow, and parameterized geologic modeling.

C.2 Forward simulation framework

The aim of the forward simulation is to compute predicted data, $\mathbf{d}_{\text{pred}}$, provided an inversion model, $\mathbf{m}$, and sources, which may come in the form of boundary conditions, are used. Here, we use the term, *inversion model*, to describe a parameterized representation of the earth (voxel-based or some other parametric representation; that is, the model is an array of numbers). Even if the model is solely used for forward modelling,

its form sets the context for the inverse problem and the parameter-space that is to be explored. Additionally, it is often an advantage to explore the sensitivity of predicted data or fields with respect to physical properties, sources, or receivers (for example, in survey design or uncertainty estimation). The forward simulation framework was lead primarily from the conceptual pieces necessary from the Richards equation (Chapter 3) as well as time and frequency domain electromagnetics, which are presented in Heagy et al. (2016). Figure C.1 shows the framework for the forward simulation as distilled from this collaborative work, which extends the general framework presented in Chapter 2 and, similarly, consists of two overarching categories (Cockett et al., 2015c):

- the `Problem`, which is the implementation of the governing equations,
- the `Survey`, which provides the source(s) to excite the system as well as the receivers to sample the fields and produce predicted data at receiver locations.

C.2.1 Concepts

Here, we provide a *brief* overview of each of the components in the forward simulation. An in-depth discussion about each component in this framework has been published in the context of electromagnetic problems (Heagy et al., 2016). Here, we present a succinct adaptation of this work in the context of the Richards equation by walking through Figure C.1. To compute pressure head responses everywhere in space and time, the forward simulation requires the definition of two *physical property* functions, which describe the water retention curve ($\theta(\psi)$) and hydraulic conductivity function ($K(\psi)$) on the simulation mesh. This method differs from many geophysical methods where physical properties are not dependent on the fields/fluxes (for example, electrical conductivity in direct current resistivity). Analogies exist here between

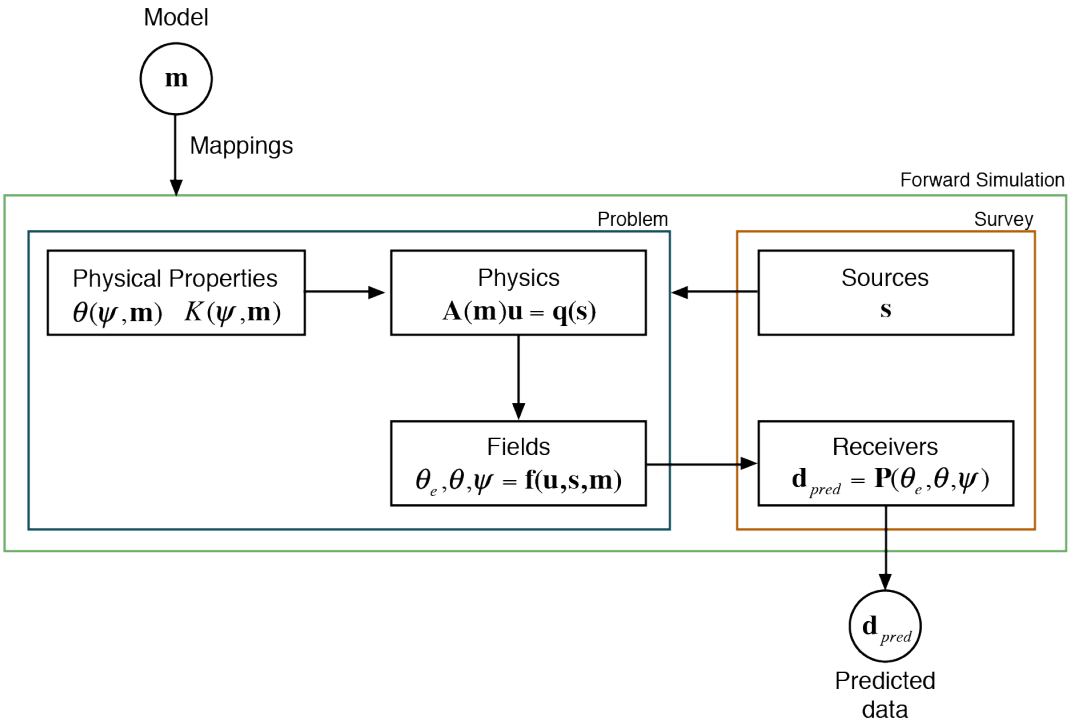

Figure C.1: The forward simulation framework that is used for Richards equation.

induced polarization geophysical methods and the Richards equation, where the measured electrical conductivity depends upon the frequency of the alternating current. Regardless of the physics used, *physical properties* (or functions) must be defined throughout the computational domain. The *physics* defines the equation, in this case the Richards equation, which can be written as a matrix equation:

$$\mathbf{A}(\mathbf{m})\mathbf{u} = \mathbf{q}(\mathbf{s})$$

where $\mathbf{u}$ is the solution of the matrix solve (possibly a field) and $\mathbf{q}(\mathbf{s})$ is both the right-hand side and the function of a source. In the time domain case, when backward Euler

is used, $\mathbf{A}$ amounts to a block bidiagonal matrix. The boundary conditions can be seen as *sources*, $\mathbf{s}$, when the discretization is written in weak-form (Section B.2.5). The sources must be included in both the physics and, in the case of electromagnetics, the *fields*. In the case of the Richards equations, the evaluation of the pressure head field to the water content field is dependent on the water retention curve, which may, in turn, depend on the *model*. These fields, defined everywhere in the computational domain, can be spatially and temporally interpolated onto the measurement locations of the *receivers* to create predicted data. In other geophysical methodologies, the data may be produced through a spatial or temporal derivative (e.g. gravity gradiometry), a potential difference (e.g. direct current resistivity), or a ratio involving multiple geophysical fields (e.g. magnetotellurics). Figure C.1 shows the conceptual steps involved in going from a model vector to predicted data. Heagy et al. (2016) discusses some additional intricacies; however, this conceptual organization into modular, exchangeable, and testable components is helpful when tackling the implicit derivatives necessary in the inverse problem.

C.2.2 Derivatives

The process we follow to compute matrix-vector products with the sensitivity is shown in Figure C.2, where $\mathbf{Jv}$ is built in stages by taking matrix vector products with the relevant derivatives in each component. This process is shown schematically in Figure C.2 for both $\mathbf{Jv}$ and $\mathbf{J}^\top\mathbf{v}$. As an example, let us consider our model to be the van Genuchten parameter, α, which is defined as a heterogeneous parameter inside the computational domain. In this case, $\mathbf{m} = \alpha$ and no other parameters are considered to be active. The

sensitivity of the forward simulation to the model takes the form:

$$\begin{aligned}\mathbf{J}[\mathbf{m}] = \frac{d\mathscr{F}[\mathbf{m}]}{d\mathbf{m}} &= \frac{\mathrm{d}\mathbf{P}(\mathbf{f})}{\mathrm{d}\mathbf{f}} \frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\alpha} \frac{\mathrm{d}\alpha}{\mathrm{d}\mathbf{m}} \\ &= \underbrace{\frac{\mathrm{d}\mathbf{P}(\mathbf{f})}{\mathrm{d}\mathbf{f}}}_{\text{Receivers}} \underbrace{\left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}} \overbrace{\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\alpha}}^{\text{Physics}} + \frac{\partial \mathbf{f}}{\partial \mathbf{s}} \overbrace{\frac{\mathrm{d}\mathbf{s}}{\mathrm{d}\alpha}}^{\text{Sources}} + \frac{\partial \mathbf{f}}{\partial \alpha} \right)}_{\text{Fields}} \underbrace{\frac{\mathrm{d}\alpha}{\mathrm{d}\mathbf{m}}}_{\text{Properties}}\end{aligned} \tag{C.1}$$

The annotations in Figure C.2 denote which of the elements shown are responsible for computing the respective contribution to the sensitivity. If the model provided is instead in terms of $\mathbf{K}_s$, this property replaces the role of α. The flexibility to invoke distinct properties of interest (e.g. α, K_s, source location, etc.) in the inversion requires quite a bit of 'wiring' to keep track of which model parameters are associated with which properties (physical properties, location properties, boundary conditions, etc.); this 'wiring' is achieved through a general `Wires` class that keeps track of the connections between properties and the model (Program C.2).

Although typically the source terms do not have model dependence and thus their derivatives are zero, the derivatives of $\mathbf{s}$ must be considered in a general implementation (Heagy et al., 2016). For example, if one wishes to use a nested approach, where source fields or boundary conditions are constructed by solving a simplified or different physics problem, the source terms may have dependance on the model. The source terms' dependance on the model means that their derivatives have a non-zero contribution to the sensitivity (cf. Haber (2015); Heagy et al. (2015c, 2016)).

The derivative of the solution, vector $\mathbf{u}$ with respect to the model, is found by

implicitly taking the derivative of the physics with respect to $\mathbf{m}$, giving:

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\mathbf{m}} = \mathbf{A}^{-1}(\mathbf{m}) \left(-\underbrace{\frac{\partial \mathbf{A}(\mathbf{m})\mathbf{u}^{\mathrm{fix}}}{\partial \mathbf{m}}}_{\texttt{getADeriv}} + \underbrace{\frac{\partial \mathbf{q}}{\partial \mathbf{s}} \frac{\mathrm{d}\mathbf{s}}{\mathrm{d}\mathbf{m}} + \frac{\partial \mathbf{q}}{\partial \mathbf{m}}}_{\texttt{getRHSDeriv}} \right) \tag{C.2}$$

The annotations below the equation indicate the methods of the `Problem` class that are responsible for calculating the respective derivatives. If multiple physical properties are internal to the $\mathbf{A}$, then this is indicated by an underscore in the method name (`getADeriv_sigma`). Typically the model dependence of the system matrix is through the physical properties. (for example, σ, μ in electromagnetics or K_s, α, and other van Genuchten parameters in the Richards equation). Thus, to compute derivatives with respect to $\mathbf{m}$, we first take the derivatives with respect to α and we treat the dependence of α on $\mathbf{m}$ using chain rule.

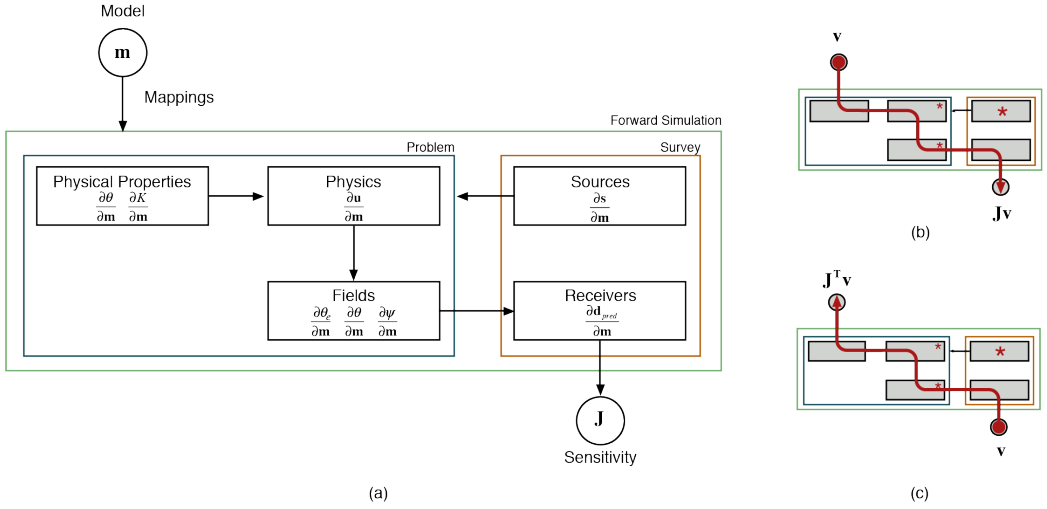

Figure C.2: The components required in calculating the derivatives of the forward simulation, showing (a) the modular nature of each derivative; (b) the process of multiplying each derivative in the forward sense with $\mathbf{Jv}$; and (c) in the adjoint sense with $\mathbf{J}^\top \mathbf{v}$.

C.2.3 Properties and Mappings

Often, in solving an inverse problem, the model which we choose to invert for (the vector $\mathbf{m}$) is some discrete representation of the earth that is decoupled from the physical property model or perhaps represents multiple physical properties. This decoupling requires the definition of a `Mapping` that is capable of translating $\mathbf{m}$ to physical properties on the simulation mesh. For instance, if the inversion model is chosen to be log-conductivity, an exponential mapping is required to obtain electrical conductivity (i.e. $\sigma = \mathscr{M}(\mathbf{m})$). To support this abstraction and integration to model conceptualization, the framework defines a number of extensible `Mapping` classes (Cockett et al., 2015c; Kang et al., 2015a; Heagy et al., 2016). These `Mappings` must also be able to deal with the potential for multiple physical properties, potentially in different locations, in the forward simulation framework. For example, in the Richards equation, the water retention function is required in both the physics and the fields to translate pressure head to water content or saturation. As we are in the process of developing a computational ontology that is extensible and applicable across disciplines, our framework is explicit about where the physical properties (or functions and their properties) are involved in the forward simulation framework. For example, in Program C.1, we can define a base class that represents the hydraulic conductivity function. Many variants (subclasses) of this function exist, which define parameters in different ways, including van Genuchten, Brooks-Corey, splines, interpolation, etc. The properties on these subclasses are the parameters that one might be interested in inverting for (in van Genuchten, these would include K_s, α, n, and I). The `Problem` class, which contains the physics, can declaratively define that it requires a valid instance of a hydraulic conductivity function. This inheritance is similar for the water

retention, which can be defined on both the problem and the fields. The declarative nature of these `properties` is the backbone of extracting an ontology, which can be used in a variety of other ways (specification, search, documentation, collaboration, etc.). In defining the properties, we also define a map, $\mathscr{M}(\mathbf{m})$, to that property as well as a derivative with respect to the model.

```
import properties
from SimPEG import Props, Problem


class HydraulicConductivity(Props.HasModel):
    """The base hydraulic conductivity function"""


class Vangenuchten_k(HydraulicConductivity):

    Ks, KsMap, KsDeriv = Props.Invertible(
        "Saturated hydraulic conductivity",
        default=1e3
    )

    alpha, alphaMap, alphaDeriv = Props.Invertible(
        "inverse of the air entry suction [L-1]",
        default=6.0, min=0
    )


class BrooksCorey_k(HydraulicConductivity):
    """Or a spline, or something custom"""


class RichardsProblem(Problem.BaseTimeProblem):

    hydraulic_conductivity = properties.Instance(
        'hydraulic conductivity function',
        HydraulicConductivity
    )
```

Program C.1: Definition of a hydraulic conductivity model with multiple invertible properties that is declaratively attached to the Richards problem class.

The mappings from the model space to a physical property on the computational domain are a key way to interface to domain knowledge in other disciplines (e.g. structural geology). Additionally, they allow the inversion methodology to turn on or off sensitivity to the model at a property by property basis; this is completed by a `Wires`

utility, which creates projection matrices that sample the model vector at the appropriate location(s). Program C.2 demonstrates this utility in Python code and shows the outcome of assumptions using assert statements. In this case, we define the model vectors to be K_s and α and use the wires to attach these vectors to the physical properties. In the inversion, however, we may wish to invert in log conductivity space for K_s, as this property varies logarithmically rather than linearly. To do this, we use an additional `Mapping` to take the exponential of the model before setting the parameter in the hydraulic conductivity function. The length of these transformations is arbitrary and extremely specific to the case study or geoscience hypothesis (e.g. survey design, parametric inversions, or sensitivity analysis). The mappings between model conceptualization and physical properties can be broken into composable, reusable pieces, which can use the chain rule to evaluate the derivative of the transformation. Figure C.3 shows this evaluation in action, where a logarithmically scaled conductivity in a layered earth is transformed into the physical property on the entire computational domain.

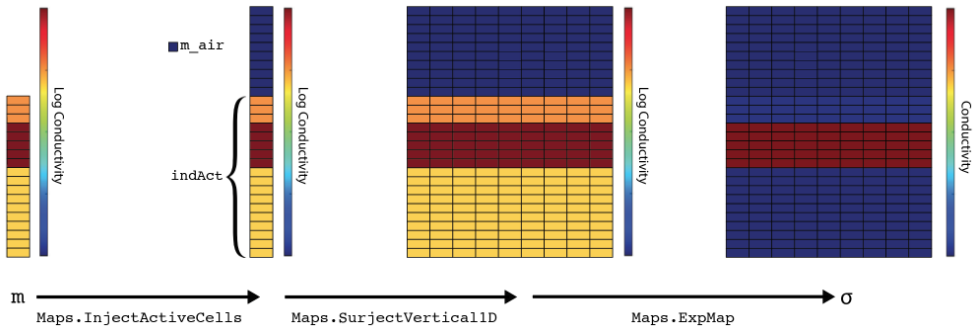

Figure C.3: Mapping an inversion model, a 1D layered, log conductivity model defined below the surface, to electrical conductivity defined in the full simulation domain.

```python
import numpy as np
from SimPEG import Mesh, Maps
from SimPEG.FLOW.Richards import Empirical

mesh = Mesh.TensorMesh([[(1, 40)]], 'N')            # Create mesh with 40 cells
k_fun, theta_fun = Empirical.van_genuchten(mesh)    # Create empirical models

k_sat = 1e-3                                        # Define homogeneous physical properties
alpha = 6.0

k_sat_model = [np.log(k_sat)]*mesh.nC               # Put values into array, log(Ks) is the model
alpha_model = [alpha]*mesh.nC
model = np.array(k_sat_model + alpha_model)

assert len(model) == mesh.nC * 2                    # We have 80 model parameters

theta_r = np.array([0.02]*mesh.nC)                  # Define other properties (not in model)
theta_s = np.array([0.30]*mesh.nC)

wires = Maps.Wires(                                 # Create wires from model to properties
    ('Ks', mesh.nC),                                # Ks gets 40 parameters
    ('alpha', mesh.nC)                              # alpha gets 40 parameters
)

# Use the maps to define Ks and alpha
k_fun.KsMap = Maps.ExpMap(mesh) * wires.Ks          # Add the exponential mapping
k_fun.alphaMap = theta_fun.alphaMap = wires.alpha   # Note that alpha is in both functions

theta_fun.theta_r = theta_r                         # Use the properties to define θr & θs
theta_fun.theta_s = theta_s

k_fun.model = theta_fun.model = model               # Set the model for each function

# Test the setup of the functions
assert np.isclose(k_fun.alpha[0], alpha)            # Check that the mappings are working
assert np.isclose(theta_fun.alpha[0], alpha)
assert np.isclose(k_fun.Ks[0], k_sat)               # Including the exponential
assert k_fun.KsDeriv.shape == (40, 80)              # Check the Ks derivative is the correct shape
assert theta_fun.theta_rMap is None                 # Check that there are no Maps for θr
assert theta_fun.theta_sMap is None                 # Nor for θs
assert theta_fun.theta_rDeriv == 0                  # Which means the derivative, ∂θr/∂m, is zero

print(k_fun.summary())                              # Print a summary of the function

# >> Physical Properties:
# >>  [*] I: set by default value
# >>  [*] Ks: set by the 'KsMap': ComboMap[ExpMap(40,40) * Projection(40,80)] * model(80)
# >>  [*] alpha: set by the 'alphaMap': Projection(40,80) * model(80)
# >>  [*] n: set by default value
```

Program C.2: Demonstration of the ability to choose arbitrary parameters to include in a model, and use the chain rule to compose parameterizations.

C.3 Parameterizations

In the following case studies, I will touch briefly on some key questions and how the framework developed can answer these *classes* of questions. Many of the case studies have been published and represent collaborative work with many colleagues. In this section, rather than repeating the conclusion of each study, I will highlight what I have learned and how the framework developed can be used.

Some geoscience questions:

- What is the distribution of physical properties?
  - What space?
  - What about known distributions?
- What is the sensitivity to a conceptual model?
  - For example, in survey design?
  - How does this extend to structural geologic modelling?
- What is the dimensionality (1D, 2D, 3D, or 4D) of the problem?
  - For example, in electromagnetics?
  - How do we keep control variables when testing assumptions?
- How can we integrate, nest, couple, and join different problems?
  - For example, in a primary secondary formulation?
  - In the general case?

C.3.1 Expected distributions

Electrical or hydraulic conductivity are often parameterized logarithmically. This logarithmic parameterization changes the space in which the inversion 'searches' for an answer to the optimization problem. In the Richards equation, for example, by choosing the parameterization of van Genuchten, we constrain the inversion to pulling only from a set of parameterizations of this function. In Heagy et al. (2014), we investigated two approaches for identifying the extent and location of an electrically conductive proppant. Hydraulic fracturing uses sand or ceramic beads to *prop* newly created fractures open; as such, the location of the proppant represents the volume of a reservoir that can be effectively drained. The standard approach involves using an uncoupled geophysical inversion to create an image of electrical conductivity and then, subsequently, interpret that distribution either qualitatively or quantitatively (Figure C.4). In Heagy et al. (2013), the authors investigated the geophysical responses of conductive and permeable proppant particles. Later research by Heagy lead to a parameterization using effective medium theory to analytically describe the relationship between conductivity and volume fraction of the proppant. In Heagy et al. (2014), we used this relation directly in a coupled geophysical inversion for the volume of the proppant. Furthermore, by parameterizing the inversion in terms of volume, rather than conductivity, we can use the known volume of the proppant pumped into the synthetic reservoir as another datum. This coupled methodology was tested on a synthetic example and the joint inversion of geophysical and volume data showed promising results.

This coupled geophysical method can be completed through a single, custom made `Mapping` that codifies the effective medium theory parameterization and the derivative. We can then attach this mapping to a physical property, such as electrical con-

ductivity. By changing the space in which we choose to conceptualize the model, it is possible to add more *a priori* information and other datum.

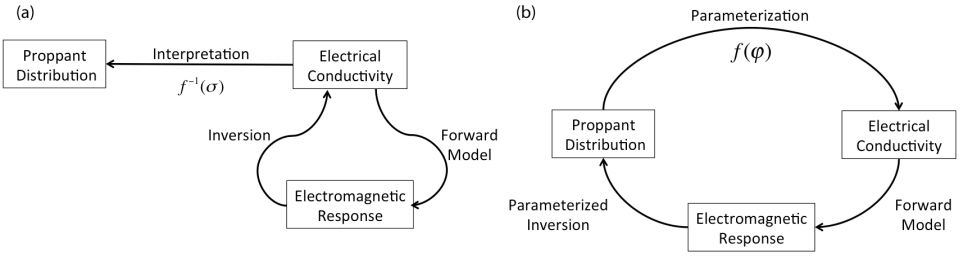

Figure C.4: (a) Traditional approach to inversion, where the model space, electrical conductivity, is mapped to data space, the electromagnetic response, through a forward model. The inversion then provides a method by which we estimate a model that is consistent with the observed data. The recovered conductivity model is then used to infer information about the reservoir properties of interest, in this case, the distribution of proppant. (b) Parametrized inversion, where we parametrize the model space, electrical conductivity, in terms of the property of interest, the distribution of proppant. By defining such a parametrization, the inversion can provide a means of estimating the properties of interest directly from the data.

C.3.2 Survey design

Heagy et al. (2016) presented the forward simulation framework for the context of electromagnetic simulations and inversions. One of the examples in this paper dealt with a steel-cased well, which was used to deliver a galvanic or inductive source to a target reservoir (Figure C.5). Here, we posed the question: How sensitive are the data to the location, depth, and conductivity of a target in a reservoir? To answer this question, we conceptualized a simple model of location, dimensions, and conductivities of an idealized block in a reservoir layer; all model parameters are visualized in Figure C.5. The challenge here is that the steel-cased well has a thickness on the order of millimeters, while being kilometers long. Tackling this challenge in a computa-

tionally efficient manner required a primary-secondary approach, where source fields were constructed by solving a simplified problem without the target on a cylindrically symmetric mesh. However, this approach required that the contribution of the model sensitivity had to be efficiently traced all the way back through the primary fields. Figure C.5 shows the *nesting* of two forward modelling frameworks. By looking at the sensitivity to these model parameters, the paper drew conclusions about potential survey designs and to what extent the conceptual model could be resolved by noisy data.

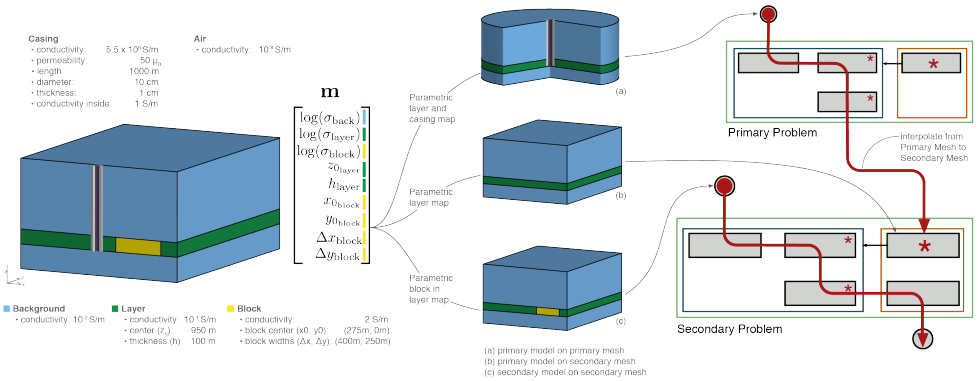

Figure C.5: Setup of a parametric models for a steel cased well and a reservoir target. The calculation of sensitivity for using a primary secondary approach is shown using the forward simulation framework.

This example required multiple formulations of Maxwell's equations on two different mesh types. The model could be conceptualized and potentially mapped to both electrical conductivity and magnetic permeability. The mappings required for this model conceptualization are largely reusable for other contexts. The sensitivity to the model parameters required that two problems be nested, which showed the value of composable pieces in a geophysical inversion framework.

C.3.3 Geologic modeling

While completing my studies, I was involved in creating a number of structural geologic modelling tools (Cockett et al., 2014a; Cockett, 2012, 2015; Funning and Cockett, 2012; Cockett et al., 2016b; Cockett, 2013). Over 300,000 people around the world have used these tools, primarily in introductory geoscience education (Cockett et al., 2016b). These tools allow rapid conceptual modelling of geologic scenarios (Figure C.6). Again, this process can be thought of as a mapping that codifies geologic knowledge and provides physical properties as a function of space. Using these mappings directly in an inversion, however, is difficult as the parameterization is spatially coupled; that is, a single parameter, such as rotation of a tilting event or period of a folding event, can change almost all physical properties at once. This spatial coupling is in contrast to a spatially *decoupled* voxel-based parameterization, where each cell centered parameter has no effect on its neighbours. Due to the spatial coupling, the explicit geologic parameterizations developed in (Cockett et al., 2016b) creates a non-convex objective function, which is difficult to minimize with deterministic optimization. Implicit geologic modelling, in contrast, solves an inverse problem by using radial basis functions to create a spatially extensive geologic parameterization (cf. Hillier et al. (2014)). Including this implicit modelling into the geophysics inversion framework as a mapping could be promising way to include geologic information.

C.3.4 Dimensionality and controlled variables

In Kang et al. (2015a), we explored the advantages to increasing the dimensionality of the physical problem being solved (in this case, ground loop, time domain electromagnetics). We explored these advantages over a three-dimensional synthetic model of seawater intrusion into a confined aquifer (Figure C.7). The experiment allowed

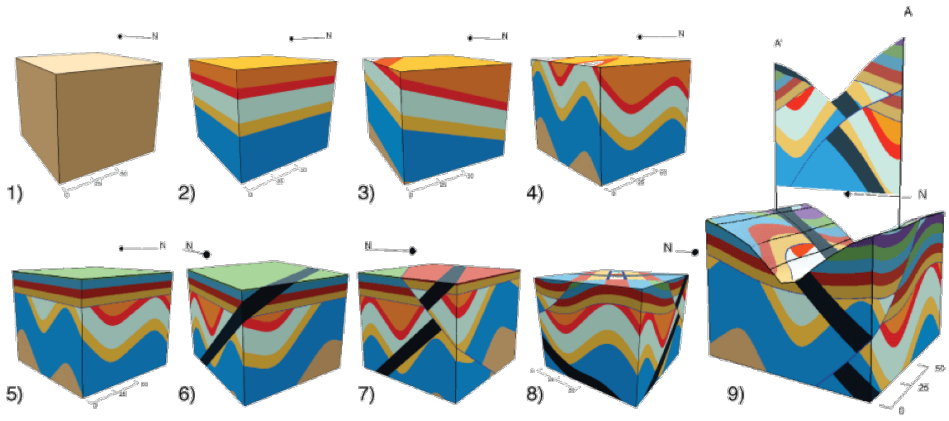

Figure C.6: Parameterized geologic models using a series of analytic functions. Models were created using Visible Geology (http://app.visiblegeology.com).

for comparisons between the various dimensions because the tools were written in a consistent framework where only a single variable was changing at a time. The increased dimensionality, in this case, led to a better recovery of the seawater intrusion interface. (Kang et al., 2014) subsequently fit the saltwater intrusion by a parametric surface. In Pidlisecky et al. (2013), however, we used a decrease in dimensionality (from 2.5D to 1D), which was tested for the ability to similarly fit collected data. In Heagy et al. (2016), data inversions were explored in both time domain and frequency domain electromagnetics. Figure C.8 shows the similarities between the two forward simulation frameworks. Although the internals of the implementation differ slightly, the inversion framework that we used is identical.

The flexibility to explore the dimensionality of the problem under consideration is important; this can be enabled by a consistent set of tools that allow you to control all variables except the dimensionality of the computational domain. When comparing between formulations of the same physics, the forward problems can inherit from and utilize many of the same components and the inversion machinery is identical. Rather

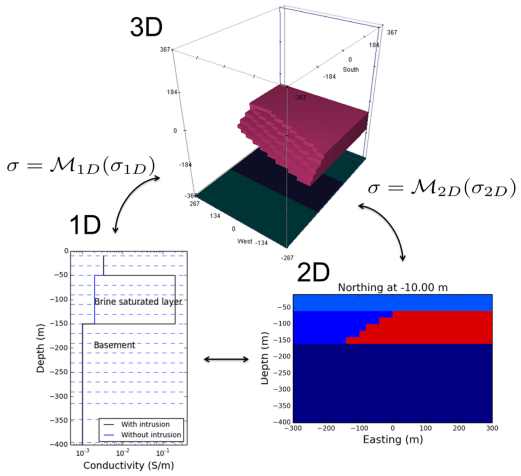

Figure C.7: Conceptual diagram of moving between 1D, 2D, and 3D models.

than comparing inversion results to a completely different black-box implementation, these shared components of the framework promotes controlled variables and more rigorous comparisons of inversion methodologies across physical problems.

C.3.5 Nesting

There are many future research opportunities for combining, coupling, nesting, or otherwise integrating different physical problems. An example of these opportunities was completed in Heagy et al. (2016), where we nested a physics problem inside another one (Figure C.5). In this case, we nested the same physics problem; however, in Rosenkjaer et al. (2016), the authors used the framework to combine electromagnetic dipole sources with the magnetotelluric forward modelling to investigate coherent source interference. The growing field of hydrogeophysics will continue to combine geophysics and fluid flow problems; the use of a consistent and integrated framework should help to further these goals.

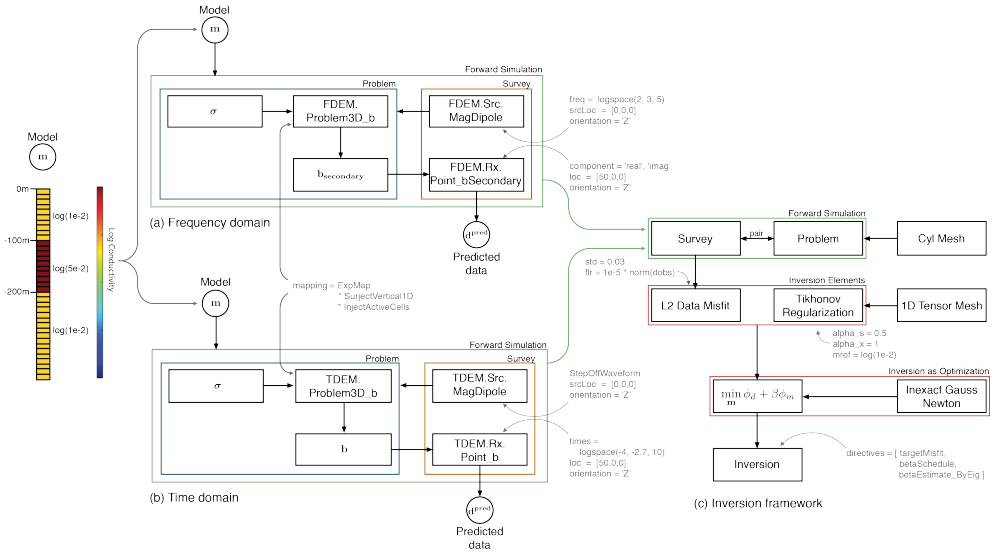

Figure C.8: Diagram showing the entire setup and organization of (a) the frequency domain simulation; (b) the time domain simulation; and (c) the common inversion framework used for each example. The muted text shows the programmatic inputs to each class instance.

C.4 Conclusions

Geologic and *a priori* assumptions are often given through parameterizations. The parameterizations necessary for a specific case study will often need to be unique to answer or predict a specific geoscience hypothesis. Through the exploration of numerous case studies across the geophysical, hydrologic, and geologic literature, we presented a forward modelling framework in Heagy et al. (2016). The important aspects of this framework are: (a) rigorously defining the properties that can be inverted for; (b) allowing these properties to be defined directly or wiring the property to the model through a mapping; and, (c) providing a number of extensible and reusable mapping components, which, when chained together, allow for efficient calculation of the sensitivity. The forward modelling framework presented maintains the computational scalability that is necessary to invert for large-scale 3D distributions of param-

eters (such as in electromagnetics or vadose zone flow). It is important to note that this framework also allows for customization through extensible mappings to custom model conceptualizations. By defining these components in a common framework, we enable hypothesis testing and exploration of assumptions by changing single variables (e.g. formulation or dimensionality) while keeping other variables controlled. The nesting and coupling of forward simulations, combined with these mappings between model conceptualizations, is important to any framework that aims to enhance quantitative geoscience collaboration.