Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

Samy Wu Fung	Date

Large-Scale Parameter Estimation in Geophysics and Machine Learning

By

Samy Wu Fung

Doctor of Philosophy

Mathematics

Lars Ruthotto

Advisor

James Nagy

Committee Member

Joyce Ho

Committee Member

Accepted:

Lisa A. Tedesco, Ph.D.

Dean of the James T. Laney School of Graduate Studies

Date

Large-Scale Parameter Estimation in Geophysics and Machine Learning

By

Samy Wu Fung

A.A., Miami Dade College, 2011

B.Sc., Brown University, 2014

Advisor: Lars Ruthotto, Ph.D.

An abstract of

A dissertation submitted to the Faculty of the

James T. Laney School of Graduate Studies of Emory University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in Mathematics

2019

Abstract

Large-Scale Parameter Estimation in Geophysics and Machine Learning

By Samy Wu Fung

The ability to collect large amounts of data with relative ease has given rise to new opportunities for scientific discovery. It has led to a new class of large-scale parameter estimation problems in geophysics, machine learning, and numerous other applications. Traditionally, parameter estimation aims to infer parameters in a physical model from indirect measurements, where the model is often given by a partial differential equation (PDE). Here, we also phrase machine learning as a parameter estimation problem, where rather than having a PDE as the model, we have a hypothesis function; for instance, the hypothesis function may be a neural network with the parameters of interest corresponding to the weights of the network. A common thread in these problems is their massive computational expense. In both cases, the underlying parameter space is typically very high-dimensional, making the optimization computationally demanding, sometimes intractable, when large amounts of data are available. This thesis addresses two general approaches to reduce the computational burdens of large-scale parameter estimation in geophysics and machine learning: 1) model order reduction (MOR), which aims to reduce the computational complexity of the model, and 2) parallel/distributed optimization which aims to reduce the time-to-solution in parameter estimation.

For the MOR approach, we present an adaptive scheme tailored to problems in geophysics, where the number of PDE simulations required to accurately reconstruct the parameter is correlated to the amount of measurements. To this end, we apply the multiscale finite volumes (MSFV) to solve high-dimensional geophysics parameter estimation problems. Given a finite volume discretization of the PDE on a fine mesh, the MSFV method reduces the problem size by computing a parameter-dependent

projection onto a nested coarse mesh. A novelty in our work is the integration of MSFV into a PDE-constrained optimization framework, which updates the reduced space in each iteration. This adaptivity of the projection basis allows us to project to an aggressively coarsened mesh while achieving highly accurate solutions. We also present a computationally tractable way of explicitly differentiating the MOR solution that acknowledges the change of basis. We illustrate the effectiveness of this approach on the direct current resistivity survey.

For the parallel/distributed approach, we propose two methods. In the first method, we present an asynchronous, uncertainty-weighted alternating direction method of multipliers (ADMM). In particular, we consider a global variable consensus ADMM algorithm to estimate parameters in geophysics and machine learning asynchronously and in parallel. Motivated by problems with many measurements, we partition the data and distribute the resulting subproblems among the available workers. Since each subproblem can be associated with different models and right-hand-sides, this provides ample options for tailoring the method to different applications. Our contribution is a novel weighting scheme that empirically improves the progress made in the early iterations of the consensus ADMM scheme and is attractive when using a large number of subproblems. The weights in our scheme are related to the uncertainties associated with the solutions of each subproblem and can be computed efficiently using iterative schemes. We exemplarily show that the weighting scheme leads to accelerated convergence for a series of linear and nonlinear parameter estimation problems. We also show that the asynchronous implementation further reduces the time-to-solution of 3D problems in geophysics. In the second method, we present an ADMM-based technique (ADMM-Softmax) which aims to efficiently learn the weights in multinomial logistic regression (MLR) problems. In each iteration, our algorithm decomposes the training into three steps; a linear least-squares problem for the weights, a global variable update involving a separable MLR problem, and

a trivial dual variable update. The least-squares problem can be factorized in the off-line phase, and the separability in the global variable update allows for efficient parallelization, leading to faster convergence. We outline the potential of our method for the MNIST and CIFAR-10 datasets, and show that ADMM-Softmax leads to improved generalization and convergence compared to the current state-of-the art methods.

Large-Scale Parameter Estimation in Geophysics and Machine Learning

By

Samy Wu Fung
A.A., Miami Dade College, 2011
B.Sc., Brown University, 2014

Advisor: Lars Ruthotto, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Mathematics
2019

Acknowledgments

I would like to give special thanks to my advisor, Lars Ruthotto. Under his guidance, I have learned to think with clarity, approach work with integrity, and set high standards for myself. His support and mentorship stepped beyond the boundaries of science, and his contagious passion and enthusiasm for research has been a great source of motivation. I will always be grateful for the significant impact he has had on my intellectual and personal development.

I would like to thank my committee members, James Nagy and Joyce Ho, for their insight and words of encouragement, and whose discussions helped shape this thesis. I would also like to express my appreciation to Johnny Guzmán, Michele Benzi, Zichao (Wendy) Di, and Eldad Haber, who also played central roles in my professional development.

I am also very lucky to have shared these years with an incredible group of friends. Thanks to Alessandro, Sofia, Massi, Vicman, Yunyi (Larry), Lea, Jackson, Kelvin, Jessi, Isabel, Pooya, María, Max, Xiaoyun, and Jon to name a few. I wish I could name each and every one by name. Without them, graduate school would have been much harder.

Most importantly, I would like to thank my family. Thanks to my sisters, Meily and Karina, for their unending love and support. Thanks to my parents, Jaime and Celia, for the immense sacrifices they have made so that my sisters and I could have a better life. Finally, I dedicate this thesis to my grandparents, Li Xiqin and Feng Kegang, who raised me to be a person of principle and integrity, and for instilling in me the value of scholarship and hard work.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The ever-increasing ability to collect data at massive scales with relative ease has led to new opportunities for scientific discovery. This trend, often referred to as *big data*, has led to a new class of large-scale parameter estimation problems in geophysics [14, 34, 42, 60, 127, 131, 154], machine learning [102, 107, 144, 151], medical imaging [5, 6, 22, 31, 111], hydrology [32, 89], signal processing [128], and numerous other applications. Traditionally, parameter estimation is an inverse problem that aims to infer parameters in a physical model from indirect measurements, where the model is often given by a partial differential equation (PDE). In this dissertation, we define parameter estimation more broadly by also including machine learning, where rather than having a PDE as the forward model, we have a hypothesis function, e.g., a neural network, and the parameters of interest correspond to the weights.

One way to estimate the parameter of interest is to phrase it as a Bayesian inverse problem and compute the *maximum a posteriori* (MAP) estimate, which after taking into account prior information about the parameter, conveys the most probable parameters that led to the measured observations. To this end, we first construct the posterior probability density function (PDF), which assigns a given set of parameters the probability that a certain candidate is the true parameter that gave rise

to the observed data. This is done using the data likelihood PDF, which describes the relationship between the data and the parameters, and the prior PDF, which describes any *apriori* knowledge we may have about the parameter. The MAP estimate is then obtained by finding the parameter that maximizes the posterior density function. A common thread in MAP estimation for machine learning and geophysics applications is their massive computational expense. The underlying parameter space is usually high-dimensional, making the MAP estimation very computationally demanding, sometimes intractable, when large amounts of data are available.

1.1 Contribution and Related Works

This thesis focuses on two general approaches to reduce the computational burdens of large-scale parameter estimation in geophysics and machine learning.

1.1.1 Model Order Reduction

The first approach is via model order reduction (MOR) techniques aimed at reducing the complexity of the parameter-to-observables map (or forward model). Our proposed MOR method can be used in geophysics and medical imaging where every evaluation of the forward model requires solving a partial differential equation (PDE); this incurs immense computational costs as the PDEs must be solved in every iteration of the optimization scheme. In MOR, the idea is to project the *controls* (i.e., the parameter) and/or the *states* (i.e., the PDE solutions) onto lower-dimensional subspaces [12]. Several techniques have been used recently to compute subspaces with good approximation properties, e.g., reduced basis methods [122], moment matching [44], empirical interpolation [8, 31], Krylov subspace methods [124], and perhaps most commonly proper orthogonal decomposition (POD) [18, 91, 100, 111]. In order for MOR to be effective, the solutions obtained using MOR need to accurately

approximate solutions of the original problem for a sufficiently large set of parameters [18]. For example, the success of POD-based methods relies on the availability of sufficiently many well-distributed *snapshots* - solutions of the original problem for given parameters that are used to build an MOR basis. While sampling the parameter space is feasible in low-dimensional settings, the problem becomes more difficult or intractable for high-dimensional parameter spaces. Similar restrictions apply to interpolatory MOR techniques.

Our contribution in this work is the use of multiscale finite volume methods (MSFV) [24–26, 39, 69, 70, 86, 87, 105, 129] to build a parameter-dependent interpolation between grid functions on a nested coarse mesh and solutions on the original fine mesh. The interpolation is computed by solving fine-mesh versions of the original problem separately for each coarse-mesh block, which can be parallelized, and, in contrast to, e.g., POD, can avoid solving the fine-mesh problem altogether. The MSFV method bears similarity with other adaptive meshing techniques, e.g., [79, 110], however, the homogenization applied to obtain the reduced problem is operator-dependent and can thus capture larger variety in the parameters. MSFV methods have been employed successfully to reduce the cost of PDE solves in porous media flow problems [30, 37, 81, 84, 85, 90, 113, 116, 120] and more recently in electromagnetics [65].

Rather than using a fixed reduced space throughout the iterative scheme, we use a computationally tractable way to adaptively update the basis within the derivative-based optimization process [30]. In our discretize-then-optimize framework, this requires explicit differentiation of the MSFV method. Derivatives of MSFV methods have also been approximately computed using interpolation [99], adjoint methods [48, 49], and more recently using a general algebraic framework [30]. In contrast to these works, we explicitly differentiate the solution of the discretized problem obtained with the MSFV solver with respect to the parameter of interest. Our contribution in this part of the dissertation is closely related to to [30], however, we use local sensitiv-

ity equations to alleviate the need for automatic differentiation. We demonstrate the potential of our proposed adaptive multiscale method on the direct current resistivity (DCR) survey.

1.1.2 Parallel and Distributed Optimization

The second approach consists of parallel and distributed techniques to reduce the time-to-solution of large-scale parameter estimation problems.

Uncertainty-Weighted Consensus ADMM

We consider the global variable consensus reformulation of the MAP estimate and use the alternating direction method of multipliers (ADMM) [16, 55, 83] as well as its asynchronous variant (async-ADMM) [162] for its computation. Consensus ADMM has previously been applied to high-dimensional inverse problems in data sciences [114, 119], machine learning [16, 55, 148, 159], and imaging [54, 75, 88]. The algorithm tackles large-scale problems by partitioning the data into, say, N smaller batches that can be solved in parallel, and in some cases explicitly. This often leads to an improved ratio of local computation and communication. More specifically, each iteration of the algorithm breaks down into:

1. N subproblems using parts of the data that are solved locally and independently,
2. an averaging step that is performed once their corresponding processors have solved all N subproblems, and
3. an explicit update of the dual variable.

In the async-ADMM variant, the averaging step is performed once $N_a < N$ subproblems have been solved, reducing the overall latency. As we demonstrate in our numerical experiments, a straightforward implementation of consensus ADMM converges slowly, particularly when the information contained in the split data sets is

complementary and the number of batches, N, is large. One problem in these cases is that the averaging step in consensus ADMM gives equal weight to all the solutions corresponding to each batch, leading to an uninformed averaged reconstruction. This renders consensus ADMM prohibitive for large-scale problems since often only a few iterations are affordable.

Our contribution is a novel weighting scheme that improves the convergence of consensus ADMM. The weights are obtained in a systematic and efficient way using the framework of uncertainty quantification (UQ) proposed in [46]. We demonstrate the effect of the weights on a collection of linear inverse problems, a multinomial logistic regression problem, and 3D single-physics and multiphysics problems consisting of the direct current resistivity (DCR) and travel-time tomography surveys.

Classification using Multinomial Logistic Regression and ADMM

We also present an efficient learning algorithm for solving large-scale classification via multinomial logistic regression (MLR) and ADMM (ADMM-Softmax). There are a variety of algorithms that can be used to train the classifying weights for the MLR problem, e.g., steepest descent [117], Newton and quasi-Newton methods [157], and perhaps most commonly, stochastic gradient descent (SGD) [101,163]. However, parallelization of these methods is not trivial as methods like SGD are sequential; this can result in very slow convergence when the problem is ill-conditioned.

Our contribution consists of a reformulation of the MLR problem into a constrained one whose objective function is separable along the examples and whose coupling is enforced by the constraints. This allows for efficient parallelization of the optimization via ADMM. Specifically, each iteration of ADMM decomposes the optimization into three subproblems:

1. a weight update that involves solving a least-squares problem,

2. a global variable update that involves a cross-entropy problem that is separable

along the examples, and

3. a trivial dual update variable.

The least-squares problem arising from the weights can be efficiently solved using direct or iterative solvers [56, 137]. The convex and smooth softmax problem arising from the global variable update can also be solved efficiently since the separability along examples renders it highly parallelizable. Finally, the dual variable update is a trivial step requiring only a matrix-vector product. The reformulation of the MLR problem in this case is different from that of the global variable consensus formulation previously described. More details can be found in Section 5.3.

We test our method on two common machine learning datasets: MNIST [104], which consists of 60,000 images of handwritten digits, and CIFAR-10 [97], which consists of 60,000 images containing ten different classes: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. To improve accuracy and generalization, we adjust the inputs propagating the features through either a fixed random layer (extreme learning machines [82]), or propagating the features through a network which has been pre-trained on a simiar data set (transfer learning [130]). In particular, for transfer learning we extract the features from a pre-trained AlexNet [97] on the Imagenet dataset [33] to classify CIFAR-10 images. This is done by removing the classification layer in AlexNet, and re-training the classification layer using the CIFAR-10 data [35].

1.2 Thesis Overview

This thesis is organized as follows. In Chapter 2, we present the mathematical background relevant for this dissertation. The general formulation of MAP estimation and uncertainty quantification is reviewed as well as some common numerical optimization techniques for their computation. We conclude this chapter with a description of

the applications we will use in our numerical experiments. In Chapter 3, we motivate the use of model reduction techniques with an overview of the discretized DCR forward and inverse problem. We then present our proposed adaptive multiscale model reduction technique based on MSFV, and show a tractable way for computing the projection bases and their derivatives. We compare our model reduction scheme using adaptive and fixed bases on the DCR survey. In Chapter 4, we shift focus to parallel and distributed methods. We present an uncertainty-weighted, asynchronous ADMM method whose weights depend on the uncertainties of the model and can be computed efficiently using iterative schemes. We test this scheme on a series of least-squares problems, a multinomial logistic regression problem, and 3D geophysics parameter estimation problems consisting of the travel-time tomography survey as well as a multi-physics problem consisting of travel-time tomography and DCR. We also present an additional ADMM-based scheme (ADMM-Softmax) in Chapter 5, which is tailored to classification problems via multinomial logistic regression problems. We test our method on the MNIST and CIFAR-10 dataset, and compare its performance to that of SGD and Newton-PCG. We improve generalization by propagating our features on a fixed hidden layer for the MNIST dataset and through a pre-trained Alex-Net [98] for the CIFAR-10 dataset. We conclude with a summary and discussion of future work in Chapter 6.

Chapter 2

Preliminaries

In this chapter, we present the mathematical background relevant for this thesis. We first present MAP estimation and uncertainty quantification (UQ) in the context of geophysics and machine learning. For simplicity, we limit the discussion to the finite-dimensional case since in our examples, we follow the discretize-then-optimize approach and have a fixed discretization. However, an overview of the infinite-dimensional case can be found in [143]. We then review two numerical optimization techniques, the Gauss-Newton-PCG and ADMM, which will be pertinent for our experiments, and conclude with an overview of the geophysics and machine learning applications we will use to test our proposed schemes.

2.1 MAP Estimation and UQ

We consider additive noise-corrupted observations

$$D = \mathcal{F}(X) + E, \tag{2.1}$$

where $\mathcal{F}: \mathbb{R}^n \mapsto \mathbb{R}^m$ is the parameter-to-observable map, and D, X, and E are random vectors corresponding to the observations, the model parameter, and the measurement

noise, respectively. We denote their corresponding realizations by $\mathbf{d} \in \mathbb{R}^m$, $\mathbf{x} \in \mathbb{R}^n$, and $\boldsymbol{\epsilon} \in \mathbb{R}^m$.

In MAP estimation, one seeks to infer the parameter $\mathbf{x}$ using prior information and indirect observations $\mathbf{d}$. To compute the MAP estimate, we first to construct the posterior density function (PDF) $\pi_{\text{post}}\colon \mathbb{R}^n \mapsto \mathbb{R}$, which assigns to a given set of parameters the probability that the candidate $\mathbf{x}$ of this set is the true parameter that gave rise to the observed data $\mathbf{d}$. The MAP estimate thus consists of the most probable (or maximum) point in the posterior density function.

There are two ingredients to construct the posterior density. The first one is the prior PDF, $\pi_{\text{prior}}\colon \mathbb{R}^n \mapsto \mathbb{R}$, which allows us to encode prior information we may have about our model. The second one is the likelihood function $\pi(\mathbf{d}|\mathbf{x})\colon \mathbb{R}^n \mapsto \mathbb{R}$, which describes the relationship between the measurements $\mathbf{d}$ and the unknown model parameters $\mathbf{x}$. With these two at hand, we can then use Bayes' theorem to obtain the posterior PDF. Namely, Bayes' theorem states that

$$\begin{aligned} \pi_{\text{post}}(\mathbf{x}) &= \frac{\pi_{\text{prior}}(\mathbf{x})\pi(\mathbf{d}|\mathbf{x})}{\pi(\mathbf{d})} \\ &\propto \pi_{\text{prior}}(\mathbf{x})\pi(\mathbf{d}|\mathbf{x}), \end{aligned} \tag{2.2}$$

where, as commonly done, the marginal density of the data, $\pi(\mathbf{d})$, is dropped since it does not depend on the parameter $\mathbf{x}$. The MAP point can thus by maximizing the posterior PDF, that is,

$$\mathbf{x}_{\text{MAP}} = \underset{\mathbf{x}}{\operatorname{argmax}}\, \pi_{\text{post}}(\mathbf{x}). \tag{2.3}$$

For simplicity, we assume that X and E are statistically independent and limit the discussion to the case where the prior PDF is Gaussian, and E is a random vector whose entries are independently and identically distributed so that $E \sim \mathcal{N}(\mathbf{0}, \Gamma_{\text{noise}})$, where $\Gamma_{\text{noise}} \in \mathbb{R}^{m\times m}$ is the diagonal noise-covariance matrix. In this case, the likeli-

hood and prior PDFs are given by

$$\pi(\mathbf{d}|\mathbf{x}) \propto \exp(-\Phi(\mathbf{x})) \quad \text{and} \quad \pi_{\text{prior}}(\mathbf{x}) \propto \exp(-\mathcal{R}(\mathbf{x})), \tag{2.4}$$

respectively, where due to the assumptions above,

$$\Phi(\mathbf{x}) = \frac{1}{2}\|\mathcal{F}(\mathbf{x}) - \mathbf{d}\|^2_{\mathbf{\Gamma}^{-1}_{\text{noise}}} \quad \text{and} \quad \mathcal{R}(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{x}_{\text{ref}}\|^2_{\mathbf{\Gamma}^{-1}_{\text{prior}}}. \tag{2.5}$$

Here, $\mathbf{x}_{\text{ref}}$ is the mean of the model parameter prior PDF, and $\Gamma_{\text{prior}} \in \mathbb{R}^{n\times n}$ is the covariance matrix of the prior PDF. Using (2.2) and (2.4), we can restate the posterior PDF in closed form as

$$\pi_{\text{post}}(\mathbf{x}) \propto \exp\Big(- \Phi(\mathbf{x}) - \mathcal{R}(\mathbf{x})\Big). \tag{2.6}$$

Computing $\mathbf{x}_{\text{MAP}}$ in this case is equivalent to minimizing the negative log-posterior; this is done for conventional purposes. The MAP estimate can thus be found by solving

$$\mathbf{x}_{\text{MAP}} = \underset{\mathbf{x}}{\text{argmin}} \ \Phi(\mathbf{x}) + \mathcal{R}(\mathbf{x}). \tag{2.7}$$

In this instance, computing the MAP estimate is exactly the deterministic solution of the Tikhonov functional [121]. In particular, the likelihood term corresponds to the data misfit, and adding prior knowledge corresponds to regularizing the problem. We note that encoding prior information into our MAP computation serves to remedy the ill-posedness of the problem so that a unique solution exists, and if possible, the problem is more stable. Examples of prior knowledge of a desired solution include regularity, sparsity, or smallness of the solution. Since solving (2.7) is a deterministic inverse problem, we will use the terms commonly used in the literature [121]. In

particular, we will refer to Φ as the misfit, and $\mathcal{R}$ as the regularizer in the remainder of this dissertation. However, it should be noted that their design is based on statistical knowledge of of the problem (when available).

In this thesis, we are also interested in quantifying the uncertainties of our model, which are given by the diagonal entries of the posterior covariance Γ_{post} (see Chapter 4). When $\mathcal{F}$ is a linear operator, that is, $\mathcal{F} = \mathbf{A} \in \mathbb{R}^{m\times n}$, the posterior PDF is also Gaussian and we can write its covariance matrix $\mathbf{\Gamma}_{\text{post}} \in \mathbb{R}^{n\times n}$ in closed form as

$$\mathbf{\Gamma}_{\text{post}} = (\mathbf{A}^\top \mathbf{\Gamma}_{\text{noise}}^{-1}\mathbf{A} + \mathbf{\Gamma}_{\text{prior}}^{-1})^{-1}, \tag{2.8}$$

which can be used for quantifying uncertainties of the model parameter $\mathbf{x}$. In the nonlinear setting, we approximate the Hessian with the Gauss-Newton Hessian (see (2.17)), evaluated at some reference model. In the context of large-scale parameter estimation, however, the Hessian of Φ, let alone its inverse, is seldom constructed. Thus, naively computing the diagonals of $\mathbf{\Gamma}_{\text{post}}$ is intractable. To quantify the uncertainties of our model efficiently, we follow [46] and use efficient iterative schemes to obtain a low-rank approximation of $\mathbf{\Gamma}_{\text{post}}$. More details can be found in Section 4.2.2.

2.1.1 MAP Estimation in Classification

In machine learning classification problems, where there are a finite number of discrete outcomes, the aforementioned approach to compute the MAP estimate is known as a *generative* classifier, where assumptions on the model are made apriori, e.g., Gaussian noise. A more common alternative to compute the MAP estimate in classification problems is through multinomial logistic regression (MLR). MLR is known as a *discriminant* classifier, which makes no assumptions on the underlying distribution of the model [47]. The MAP estimate in this case is computed by minimizing the cross-entropy loss function with the softmax function as the parameter-to-observables map.

In particular, given training data $(\mathbf{D}, \mathbf{C})$, where $\mathbf{D} = [\mathbf{d}_1\, \mathbf{d}_2 \ldots \mathbf{d}_{n_e}]^\top \in \mathbb{R}^{n_e \times n_f}$ is the feature matrix and $\mathbf{C} = [\mathbf{c}_1\, \mathbf{c}_2 \ldots \mathbf{c}_{n_e}]^\top \in \mathbb{R}^{n_e \times n_c}$ the class label matrix, the MAP estimate is given by

$$\mathbf{x}_{\text{MAP}} = \underset{\mathbf{X}}{\text{argmin}} \ \Big(\Phi_{\text{CE}}(\mathbf{X}) + \mathcal{R}(\mathbf{X})\Big), \tag{2.9}$$

where $\mathbf{X} \in \mathbb{R}^{n_f \times n_c}$ are the weights to be trained, $\mathcal{R}\colon \mathbb{R}^{n_f \times n_c} \mapsto \mathbb{R}$ is a regularization term, $\Phi_{\text{CE}}\colon \mathbb{R}^{n_f \times n_c} \mapsto \mathbb{R}$ is the cross-entropy loss function is given by

$$\Phi_{\text{CE}}(\mathbf{X}) = -\text{trace}(\mathbf{C}^\top \log(h_{\mathbf{X}}(\mathbf{D}))), \tag{2.10}$$

and $h_{\mathbf{X}}$ is the softmax function given by

$$\mathbf{C}_{\text{pred}} = h_{\mathbf{X}}(\mathbf{D}) = \text{diag}\left(\frac{1}{\exp(\mathbf{D}\mathbf{X})\mathbf{e}_{n_c}}\right)\exp(\mathbf{D}\mathbf{X}). \tag{2.11}$$

Here, $\mathbf{C}_{\text{pred}} \in \mathbb{R}^{n_e \times n_c}$ are the predicted classes generated by the softmax function, $\mathbf{e}_{n_c} \in \mathbb{R}^{n_c}$ is a vector of all ones, and the diagonal, exponential, and division are performed element-wise. The softmax function is a natural forward model used in classification problems since it outputs a probability distribution, and each entry in its outputs corresponds to the probability the observed data belongs to a particular class. We note that we treat the variable $\mathbf{X}$ as a matrix since it simplifies the notation for the MLR problem. We will revisit the MLR problem in Chapter 5.

2.2 Numerical Optimization

In this section, we discuss some of the current state-of-the-art optimization methods for computing the MAP estimate in geophysics [60, 136] and classification problems [16, 147]. There are many ways to solve (2.7) and (2.9), including stochastic op-

timization methods, e.g., stochastic approximation [133], stochastic average approximation [94], and the method of simultaneous sources [62]. Deterministic methods (potentially applied to a stochastic average approximation) include quasi-Newton methods, e.g., l-BFGS [157], nonlinear conjugate gradient methods [68], Gauss-Newton methods [60, 92], and distributed methods [16]. These methods are well studied, and summaries of these methods can be found in optimization textbooks [16, 92, 157].

In this section, we limit the discussion to two methods of particular relevance to this dissertation. The first method is Gauss-Newton-PCG, which has proven effective for large-scale problems in geophysics [60, 136], and the second method is the alternating direction method of multipliers (ADMM), whose parallel properties are attractive to problems in geophysics [50] and machine learning [16, 147].

2.2.1 Gauss-Newton-PCG

The Gauss-Newton-PCG algorithm is an inexact Gauss-Newton method that has proven to work well for these large-scale problems in geophysics [60, 136]. The method is a gradient-based local optimization scheme that only requires first-order derivative information of the residual. In particular, consider again the objective function from (2.7) given by

$$\Phi(\mathbf{x}) = \frac{1}{2}\mathbf{r}(\mathbf{x})^\top \boldsymbol{\Gamma}_{\text{noise}}^{-1}\mathbf{r}(\mathbf{x}), \tag{2.12}$$

where $\mathbf{r}(\mathbf{x}) = \mathcal{F}(\mathbf{x}) - \mathbf{d}$ is the residual term. The gradient is then given by

$$\nabla_{\mathbf{x}}\Phi = \mathbf{J}(\mathbf{x})^\top \boldsymbol{\Gamma}_{\text{noise}}^{-1}\mathbf{r}(\mathbf{x}), \tag{2.13}$$

where $\mathbf{J} = \dfrac{\partial \mathbf{r}}{\partial \mathbf{x}}^\top \in \mathbb{R}^{m \times n}$ is the sensitivity matrix (also known as the Jacobian of the resdual). Finally, the Hessian can be computed as

$$\begin{aligned} \nabla^2_{\mathbf{x}} \Phi &= \mathbf{J}(\mathbf{x})^\top \boldsymbol{\Gamma}^{-1}_{\text{noise}} \mathbf{J}(\mathbf{x}) + \frac{\partial \left(\mathbf{J}(\mathbf{x})^\top \boldsymbol{\Gamma}^{-1}_{\text{noise}} \mathbf{r}^{\text{fix}} \right)}{\partial \mathbf{x}}, \\ &= \mathbf{J}(\mathbf{x})^\top \boldsymbol{\Gamma}^{-1}_{\text{noise}} \mathbf{J}(\mathbf{x}) + \mathbf{C}(\mathbf{x}), \end{aligned} \tag{2.14}$$

where $\mathbf{r}^{\text{fix}} = \mathbf{r}(\mathbf{x})$ is the residual assumed to be fixed as part of the product rule. Here, the Hessian consists of two parts. The first is a symmetric positive semidefinite (SPSD) matrix $\mathbf{J}(\mathbf{x})^\top \boldsymbol{\Gamma}^{-1}_{\text{noise}} \mathbf{J}(\mathbf{x})$. The second term, $\mathbf{C}$, is a nonlinear term which depends on the curvature of the objective function, and may be indefinite.

The Gauss-Newton step offers an alternative to the Newton's method step update. The method solves a Newton-like system where the Hessian is approximated by dropping the nonlinear term $\mathbf{C}$, that is,

$$\mathbf{H}(\mathbf{x}) = \mathbf{J}(\mathbf{x})^\top \boldsymbol{\Gamma}^{-1}_{\text{noise}} \mathbf{J}(\mathbf{x}) \approx \nabla^2_{\mathbf{x}} \Phi(\mathbf{x}). \tag{2.15}$$

In particular, the search direction, $\partial \mathbf{x}$, is given by solving the Gauss-Newton system

$$\mathbf{H}(\mathbf{x}) \partial \mathbf{x} = -\nabla_{\mathbf{x}} \Phi(\mathbf{x}). \tag{2.16}$$

We show the entire Gauss-Newton scheme in Algorithm 2.1.

Convergence

It is possible to show that $\mathbf{C}$ is small when the residual is small at the solution, or for problems that are not very nonlinear [92]. In this case, the Gauss-Newton method can be shown to have similar convergence properties to Newtons method [157]. The Gauss-Newton method is guaranteed to converge to a local minimum under the following assumptions [92]:

Algorithm 2.1: Gauss-Newton-PCG

- initialize $\mathbf{x}^{(0)}$
- for $k = 0, 1, 2, \ldots$
  1. compute $\Phi(\mathbf{x}), \mathcal{R}(\mathbf{x}), \nabla_{\mathbf{x}}\Phi(\mathbf{x}), \nabla_{\mathbf{x}}\mathcal{R}(\mathbf{x})$
  2. compute $\mathbf{H}(\mathbf{x})$ from (2.17)
  3. solve system (2.16) to obtain $\delta\mathbf{x}$ using PCG
  4. set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \gamma\partial\mathbf{x}$, where γ is set by Armijo linesearch
  5. check convergence criteria

1. the line search satisfies the Wolfe conditions,
2. the Jacobian $\mathbf{J}(\mathbf{x})$ has full rank, i.e, $\mathbf{J}(\mathbf{x})^\top \mathbf{\Gamma}_{\text{noise}}^{-1}\mathbf{J}(\mathbf{x})$ is symmetric positive definite (SPD), for every $\mathbf{x}$ in the region of interest, and
3. each entry in the residual function $\mathbf{r}(\mathbf{x})$ is Lipschitz continuous within a neighborhood of the level set $L = \{\mathbf{x} \mid \Phi(\mathbf{x}) \leq \Phi(\mathbf{x}_0)\}$.

In practice, $\mathbf{J}$ may be rank-deficient for some $\mathbf{x}$ in the region of interest (this is expected in our case). Thus, adding a regularization term helps the Gauss-Newton converge. In particular, the Gauss-Newton Hessian approximation is given by

$$\mathbf{H}(\mathbf{x}) = \mathbf{J}(\mathbf{x})^\top \mathbf{\Gamma}_{\text{noise}}^{-1}\mathbf{J}(\mathbf{x}) + \nabla_{\mathbf{x}}^2\mathcal{R}(\mathbf{x}). \tag{2.17}$$

The regularization term $\mathcal{R}$ is typically convex, so that even if $\mathbf{J}$ is rank-deficient for some $\mathbf{x}$, the Hessian can remain SPD, and a descent direction can be obtained. In our numerical experiments, we solve (2.16) iteratively using the preconditioned conjugate gradients method (PCG), which makes our method an inexact Gauss-Newton method. This adds further regularization if stopped early enough to avoid small or zero eigenvalues since the PCG method has a built-in mechanism to approximate the eigenvalues of $\mathbf{H}$ [137]. More details can be found in [157].

Stopping Criteria

In practice, there are different ways to stop the Gauss-Newton algorithm. As is commonly done in iterative gradient-based algorithms [157], one way to stop the algorithm is to compute the norm of the gradient at the current iterate and stop the method when this value falls below a chosen tolerance,

$$\|\nabla_{\mathbf{x}}\Phi(\mathbf{x}^{(k)})\|_2^2 < \epsilon. \tag{2.18}$$

This choice follows from the fact that the first-order optimality condition for Gauss-Newton states that the gradient should converge to the zero vector when the method reaches a local minimum. Another way to stop the method is when the algorithm stagnates. That is, the norm of the difference between successive iterates fall below some selected tolerance,

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_2^2 < \epsilon, \tag{2.19}$$

or similarly, the objective function values stagnate

$$\|\Phi(\mathbf{x}^{(k+1)}) - \Phi(\mathbf{x}^{(k)})\|_2^2 < \epsilon. \tag{2.20}$$

As a final note, neither of the latter two criteria guarantee that the method has reached a minimum. For instance, these values may fall below the prescribed tolerance if the step length γ in Algorithm 2.1 is too short.

2.2.2 Alternating Direction Method of Multipliers

The alternating direction method of multipliers (ADMM) is a distributed optimization scheme that has gained popularity in the machine learning community [16], and more recently, in geophysics [50]. It takes the form of a decomposition-coordination

procedure, where small local subproblems are solved in a coordinated manner to find a solution of a large global problem. ADMM can be viewed as an attempt to blend the benefits of dual decomposition [11, 28] and augmented Lagrangian methods for problems [76] of the form

$$\begin{aligned} \underset{\mathbf{x}_1,\mathbf{x}_2}{\text{argmin}} \quad & \Phi_1(\mathbf{x}_1) + \Phi_2(\mathbf{x}_2) \\ \text{s.t.} \quad & \mathbf{B}_1\mathbf{x}_1 + \mathbf{B}_2\mathbf{x}_2 = \mathbf{c}, \end{aligned} \tag{2.21}$$

with variables $\mathbf{x}_1 \in \mathbb{R}^{n_1}, \mathbf{x}_2 \in \mathbb{R}^{n_1}$, where $\mathbf{B}_1 \in \mathbb{R}^{p\times n_1}, \mathbf{B}_2 \in \mathbb{R}^{p\times n_2}$, and $\mathbf{c} \in \mathbb{R}^p$. ADMM aims at solving (2.21) by finding a saddle point of the augmented Lagrangian

$$\begin{aligned} &\mathcal{L}_\rho(\mathbf{x}_1, \mathbf{x}_2, \mathbf{y}) = \\ &\quad \Phi_1(\mathbf{x}_1) + \Phi_2(\mathbf{x}_2) + \mathbf{y}^\top(\mathbf{B}_1\mathbf{x}_1 + \mathbf{B}_2\mathbf{x}_2 - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{B}_1\mathbf{x}_1 + \mathbf{B}_2\mathbf{x}_2 - \mathbf{c}\|_2^2, \end{aligned} \tag{2.22}$$

via the following iterations

$$\mathbf{x}_1^{(k+1)} = \underset{\mathbf{x}_1}{\text{argmin}}\, \mathcal{L}_\rho(\mathbf{x}_1, \mathbf{x}_2^{(k)}, \mathbf{y}^{(k)}), \tag{2.23}$$

$$\mathbf{x}_2^{(k+1)} = \underset{\mathbf{x}_2}{\text{argmin}}\, \mathcal{L}_\rho(\mathbf{x}_1^{(k+1)}, \mathbf{x}_2, \mathbf{y}^{(k)}), \tag{2.24}$$

$$\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + \rho\left(\mathbf{B}_1\mathbf{x}_1^{(k+1)} + \mathbf{B}_2\mathbf{x}_2^{(k+1)} - \mathbf{c}\right), \tag{2.25}$$

where $\rho > 0$ is the penalty parameter. When $\rho = 0$, $\mathcal{L}_\rho$ corresponds to the standard unaugmented Lagrangian. The necessary and sufficient optimality conditions for (2.21) are determined by the primal feasibility,

$$\mathbf{r}_{\text{pri}} = \mathbf{B}_1\mathbf{x}_1^{(k)} + \mathbf{B}_2\mathbf{x}_2^{(k)} - \mathbf{c}, \tag{2.26}$$

and dual feasibility, which can be shown to be

$$\mathbf{r}_{\text{dual}} = \rho \mathbf{B}_1^\top \mathbf{B}_2 (\mathbf{x}_2^{(k+1)} - \mathbf{x}_2^{(k)}). \tag{2.27}$$

Convergence

The method is guaranteed to converge under the following assumptions [16].

1. The (extended-real-valued) functions $\Phi_1 \colon \mathbb{R}^{n_1} \mapsto \mathbb{R} \cup \{+\infty\}$ and $\Phi_2 \colon \mathbb{R}^{n_2} \mapsto \mathbb{R} \cup \{+\infty\}$ are closed, proper, and convex.

2. The unaugmented Lagrangian $\mathcal{L}$ has a saddle point.

The first assumption implies that the local problems (2.23) and (2.24) are solvable, and the second implies existence of a solution for (2.21). Under these assumptions, the ADMM iterates satisfy the following:

- Residual convergence: $\mathbf{r}_{\text{pri}}^{(k)} \to \mathbf{0}$ as $k \to \infty$, that is, the iterates approach feasibility.

- Objective convergence. $\Phi_1(\mathbf{x}_1^{(k)}) + \Phi_2(\mathbf{x}_2^{(k)}) \to \mathbf{p}^*$, where $\mathbf{p}^*$ is the optimal value of (2.21). That is, the objective function of the iterates approaches the optimal value.

- Dual variable convergence. $\mathbf{y}^{(k)} \to \mathbf{y}^*$ as $k \to \infty$, where $\mathbf{y}^*$ is a dual optimal point.

For non-convex problems, it has been shown that ADMM converges to a local minimum under some modest assumptions, most importantly, requiring ρ to be sufficiently large [78, 115, 157] These assumptions ensure that the Hessian of the Lagrangian of (2.22) remains positive definite throughout the ADMM iterations.

Stopping Criteria

For ADMM, a common termination criterion is that the primal and dual residuals be small, i.e.,

$$\|\mathbf{r}_{\text{pri}}\|_2 \leq \epsilon_{\text{pri}} \quad \text{and} \quad \|\mathbf{r}_{\text{dual}}\|_2 \leq \epsilon_{\text{dual}}, \tag{2.28}$$

where $\epsilon_{\text{pri}} > 0$ and $\epsilon_{\text{dual}} > 0$ are feasibility tolerances for the primal and dual feasibility conditions. These tolerances can be chosen using an absolute and relative criterion, for instance,

$$\begin{aligned} \epsilon_{\text{pri}} &= \sqrt{n_1}\epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\{\|\mathbf{B}_1\mathbf{x}_1^{(k)}\|_2, |\mathbf{B}_2\mathbf{x}_2^{(k)}\|_2, \|\mathbf{c}\|_2\} \\ \epsilon_{\text{dual}} &= \sqrt{n_2}\epsilon_{\text{abs}} + \epsilon_{\text{rel}}\|\mathbf{B}_1^\top \mathbf{y}^{(k)}\|_2, \end{aligned} \tag{2.29}$$

where $\epsilon_{\text{abs}} > 0$ is an absolute tolerance and $\epsilon_{\text{rel}} > 0$ is a relative tolerance.

Varying Penalty Parameter

To improve the performance of ADMM, a standard extension is to use adaptive penalty parameters for each iteration. This makes the performance of the algorithm less dependent on the initial choice of the penalty parameter. Though it can be difficult to prove the convergence of ADMM when ρ varies by iteration, the same theory for the case of fixed ρ still applies when one assumes ρ becomes fixed after a finite number of iterations. A simple, yet effective adaptive scheme recommended in [16, 74, 153] is given by

$$\rho^{(k+1)} = \begin{cases} \tau^{\text{incr}}\rho^{(k)} & \text{if} \quad \|\mathbf{r}_{\text{pri}}\|_2 > \mu\|\mathbf{r}_{\text{dual}}\|_2 \\ \rho^{(k)}/\tau^{\text{decr}} & \text{if} \quad \|\mathbf{r}_{\text{dual}}\|_2 > \mu\|\mathbf{r}_{\text{pri}}\|_2 \\ \rho^{(k)} & \text{otherwise,} \end{cases} \tag{2.30}$$

where $\mu > 1, \tau^{\text{incr}} > 1$, and $\tau^{\text{decr}} > 1$ are parameters commonly chosen to be $10, 2$, and 2, respectively [16]. This updating scheme aims at balancing the primal and dual residual norms within a factor of μ of one another as they both converge to zero.

2.3 Applications in Geophysics and Machine Learning

We conclude this chapter with a description of the applications that will be used in our numerical experiments. We begin with a description of two geophysical imaging techniques, the direct current resistivity (DCR) and travel-time tomography survey, as well as their associated PDEs. We also describe image classification in machine learning and two commonly used datasets used in the field: MNIST [104] and CIFAR-10 [97].

Direct Current Resistivity

The DCR survey is an exploration technique used in many geophysical applications [34,118]. DCR is commonly used to estimate a spatial image of the conductivity of the subsurface from indirect measurements obtained on the earth's surface. It uses electrical sources to introduce direct currents into the ground, thereby creating electric potential fields that are sensitive to variations of the conductivity in the subsurface. Measurements of these potential fields are then collected with receivers on the surface and are used to reconstruct a three-dimensional image of the conductivity in the subsurface. Because it is an inverse problem aiming at estimating the conductivity, the DCR survey is also known as the inverse conductivity problem and is closely related to, e.g., electrical impedance tomography [5, 6, 22, 111].

The governing equations for DCR are given by the steady-state heterogeneous

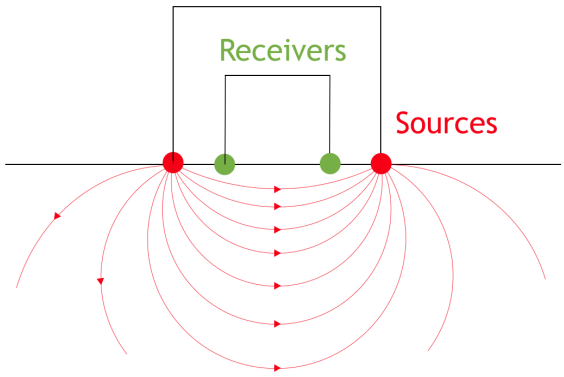


Figure 2.1: Experimental setup of the direct current resistivity survey.

diffusion operator, particularly,

$$\begin{aligned} \nabla \cdot (\sigma(x(\xi))\nabla u_j(\xi)) &= q_j(\xi), & \xi &\in \Omega, \\ \nabla u_j(\xi) \cdot \vec{n}(\xi) &= 0, & \xi &\in \partial\Omega, \\ u_j(\xi_0) &= u_{j,0} & j &= 1, \ldots, N_s. \end{aligned} \tag{2.31}$$

Here, $x \in \Omega \mapsto \mathbb{R}$ is the parameter of interest, $\sigma \colon \mathbb{R} \mapsto \mathbb{R}$ is the conductivity function, which is parameterized by x, e.g., $\sigma(x) = \exp(x)$, and u_j is the potential field induced by the jth source q_j. An illustration of the experiment is shown in Figure 2.1

Travel-Time Tomography

Another exploration technique is travel-time tomography (also known as first-arrival travel-time tomography), which uses seismic waves to map the structure of the subsurface. In this technique, the travel time of seismic waves emitted from sources on the surface are measured, and a velocity model is estimated from the seismic data. The experiment in this case is modeled by the eikonal equation, namely,

$$\begin{aligned} |\nabla \tau_j(\xi)|^2 &= x(\xi) & \xi &\in \Omega, \\ \tau_j(\xi_j) &= 0 & j &= 1, \ldots, N_s, \end{aligned} \tag{2.32}$$

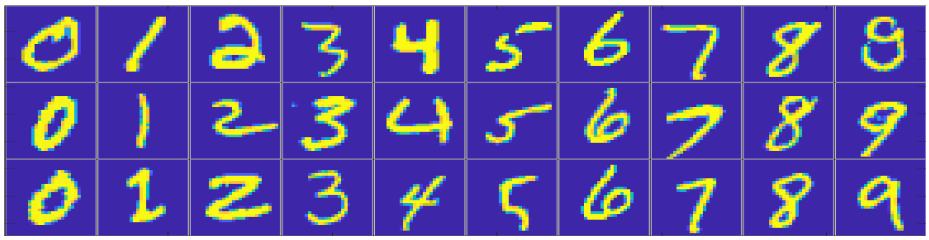


Figure 2.2: Example of 30 hand-written images from MNIST dataset [104].

where $|\cdot|$ is the Euclidean norm in $\mathbb{R}^d$, $\tau_j : \Omega \mapsto \mathbb{R}$ is the arrival time of the first wave that evolves from the j^{th} source that is located at $\xi_j \in \Omega$, and the parameter of interest, x is the compressibility parameter of the subsurface, also known as the squared slowness. The experimental setup for travel-time tomography is similar to that of the DCR survey shown in Figure 2.1.

Image Classification

Parameter estimation in geophysics is done for scientific interpretation purposes, e.g., subsurface imaging. Parameter estimation in machine learning, on the other hand, serves more predictive purposes, e.g., classification of unseen text or images. In this thesis, we will focus on image classification problems.

Classification is the process of identifying the class to which a new observation belongs. There are different ways to perform classification in machine learning, including k-nearest neighbors [1], artificial neural networks [47], support vector machines [27], and multinomial logistic regression (MLR) [47]. Thus, unlike in geophysics, where the forward model is fixed with a PDE, classification in machine learning can be done using different forward models. As seen in Section 2.1.1, in this thesis we focus on MLR, where the forward model is given by the softmax function shown in (2.11). Details about other classifiers can be found in [47]. Indeed, the softmax function in (2.11) is a natural forward model in classification, since it outputs a probability distribution. More specifically, given an example $\mathbf{d} \in \mathbb{R}^{n_f}$, the softmax function returns a vector $h_{\mathbf{X}}(\mathbf{d}) = \mathbf{c}_{\text{pred}} \in \mathbb{R}^{n_c}$ whose i^{th} entry corresponds to the probability the observed data

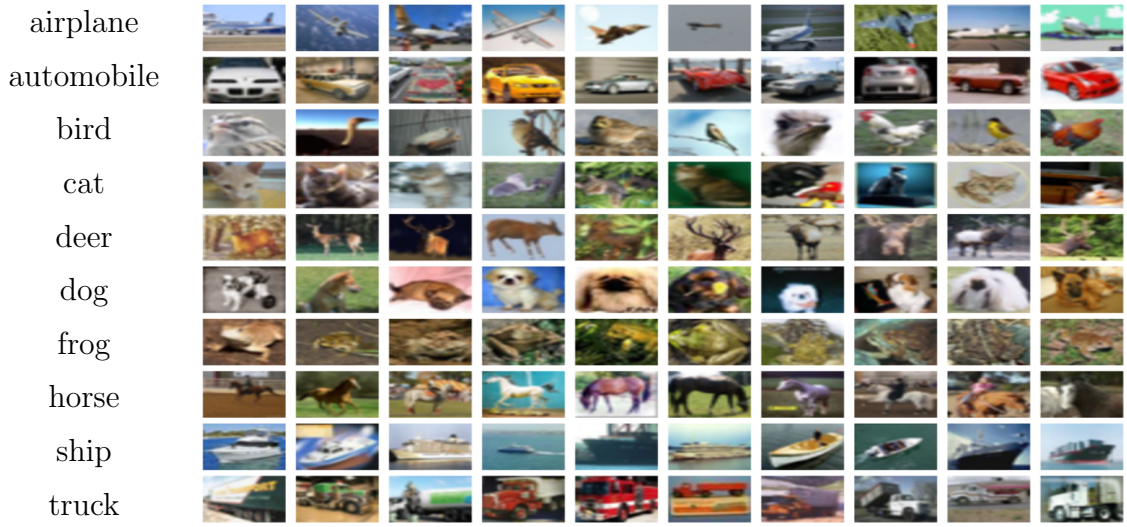


Figure 2.3: Example images for the CIFAR-10 dataset obtained from the official CIFAR-10 dataset

belongs to i^{th} class. The goal is then to train the softmax classifier by solving (2.9), which aims at finding the weights $\mathbf{X}$ that classifies any given data correctly. We note that for training data, we typically have a corresponding class vector $\mathbf{c}$ for each data $\mathbf{d}$. However, we also wish for $h_{\mathbf{X}}$ to correctly classify a given data $\mathbf{d}$ for which we may not have a label, that is, we wish for $h_{\mathbf{X}}$ to generalize well.

For classification problems in this thesis, we experiment on two datasets. The first is the MNIST dataset [104], which database consists of 60,000 gray-scale hand-written images of digits ranging from 0 to 9. Some examples are shown in Figure 2.2. The second dataset is CIFAR-10 [97], which consists of 60,000 32×32 RGB-valued images in the following 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

Chapter 3

Adaptive Multiscale Model Reduction

The results and content in this chapter are based on [51] and are done in collaboration with L. Ruthotto. In this chapter, we present an adaptive model order reduction technique to reduce the computational costs of large-scale geophysics parameter estimation problems. The key novelty of our method is the combination of MSFV and the numerical optimization scheme used for parameter estimation. This chapter is organized as follows. We first motivate our approach with an overview of the discretized DCR forward and inverse problem. We then present our proposed adaptive multiscale model reduction technique for problems in geophysics. We conclude the chapter with numerical experiments using the DCR survey.

3.1 DCR Forward and Inverse Problem

Let $\Omega \subset \mathbb{R}^d$ be the computational domain with $d = 2, 3$ and let $x: \Omega \to \mathbb{R}$ be a model function describing the parameter of interest. We consider measurement data of the form

$$\mathbf{D}_{ij} = \mathcal{F}_{ij}(x) + \epsilon_{ij}, \tag{3.1}$$

where the forward operator (or parameter-to-observables map) is given by

$$\mathcal{F}_{ij}(x) = (p_i, u_j(x)), \tag{3.2}$$

for each $i = 1, \ldots, N_r,\ j = 1, \ldots, N$. Here, $p_i \colon \Omega \to \mathbb{R}$ is the ith receiver function, $u_j \colon \Omega \to \mathbb{R}$ is the field induced by the jth source $q_j \colon \Omega \to \mathbb{R}$, $\mathbf{D} \in \mathbb{R}^{N_r \times N}$ are the discrete measurements, and N_r and N are the number of receivers and sources, respectively. We assume that the measurements are given by the L_2 inner product denoted by $(\cdot, \cdot)$ between p_i and u_j plus some additive noise ϵ_{ij} that for simplicity is assumed to be Gaussian white noise. The field u_j (also called the *state*) satisfies

$$\mathcal{A}(x)u_j = q_j, \tag{3.3}$$

where $\mathcal{A}(x)$ is the underlying PDE operator that includes the boundary conditions.

To solve the DCR problem, we follow the discretize-optimize approach in [60, 65]. We discretize the domain Ω on a uniform mesh $\mathcal{M}_h$ consisting of n cells and m nodes. Here, $h > 0$ is a parameter that denotes the discretization size. Moreover, let $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u}_j$, $\mathbf{q}_j$, $\mathbf{p}_i \in \mathbb{R}^m$, and $\boldsymbol{\epsilon}_j \in \mathbb{R}^{N_r}$ be the discretizations of the model x, the field u_j, the receiver p_i, the source q_j, and the measurement noise ϵ_{ij} given in (3.1) on $\mathcal{M}_h$. The discrete data measurements are given by

$$\mathbf{d}_j = \mathcal{F}_j(\mathbf{x}) + \boldsymbol{\epsilon}_j, \quad j = 1, \ldots, N, \tag{3.4}$$

with the forward problem

$$\mathcal{F}_j(\mathbf{x}) = \mathbf{P}^\top \mathbf{A}(\mathbf{x})^{-1} \mathbf{q}_j = \mathbf{P}^\top \mathbf{u}_j, \tag{3.5}$$

where $\mathbf{A} \in \mathbb{R}^{m \times m}$ is the discretization of (3.3) using finite volume methods pre-

sented in [60], $\mathbf{P} \in \mathbb{R}^{m \times N_r}$ is the receiver matrix that maps the fields to the data, $\{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_N\}$ are the sources, $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N\}$ are the induced fields, and $\{\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2, \ldots, \boldsymbol{\epsilon}_N\}$ are the measurement noise. Upon suitable discretization, $\mathbf{A}$ is non-singular because of the boundary conditions in (2.31), which consists of homogeneous Neumann boundary conditions as well as Dirichlet boundary conditions. This renders the fields $\mathbf{u}_j$ uniquely defined and differentiable with respect to the model parameter $\mathbf{x}$. In the optimal control literature, $\mathbf{x}$ corresponds to the discrete *control* and the fields, $\mathbf{u}_j$, are the discrete *states* [53, 139].

The DCR problem aims at estimating the underlying model $\mathbf{x}$ given the data $\mathbf{d}_j$, sources $\mathbf{q}_j$, and receivers $\mathbf{P}$. In this case, (2.7) can be formulated as the PDE-constrained optimization problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & \sum_{j=1}^{N} \Phi_j \left(\mathbf{P}^\top \mathbf{A}(\mathbf{x})^{-1} \mathbf{q}_j - \mathbf{d}_j \right) + \mathcal{R}(\mathbf{x}), \\ \text{s.t.} \quad & \mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_H, \end{aligned} \tag{3.6}$$

where $\Phi_j : \mathbb{R}^{N_r} \times \mathbb{R}^{N_r \times N} \to \mathbb{R}$ are the misfits and $\mathcal{R} : \mathbb{R}^n \to \mathbb{R}$ is Tikhonov regularization terms corresponding to the, likelihood and priors in (2.7), respectively, and $\mathbf{x}_L$ and $\mathbf{x}_H$ can be used to enforce physical bounds for the model parameters. Since the dimensions of $\mathbf{x}$ are typically very large, we approximately solve (3.6) using the Gauss-Newton-PCG method described in Section 2.2.1

3.1.1 Sensitivity Computation

A key ingredient of derivative-based methods for solving (3.6) is the sensitivity matrix $\mathbf{J}(\mathbf{x})$ (see Section 2.2.1), which characterizes how small changes in the model affect the measurements in (3.5). Since similar techniques will be used to differentiate the multiscale basis in Chapter 3, we review the concept of sensitivity computations discussed in [60]. To simplify notation, we consider the case for a single source $\mathbf{q}$. We

know from Taylor series approximations that for any small perturbation $\delta\mathbf{x} \in \mathbb{R}^n$, the sensitivity matrix $\mathbf{J}(\mathbf{x})$ satisfies

$$\mathbf{P}^\top \mathbf{u}(\mathbf{x} + \delta\mathbf{x}) \approx \mathbf{P}^\top \mathbf{u}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\delta\mathbf{x} + \mathcal{O}(\|\delta\mathbf{x}\|^2). \tag{3.7}$$

As a consequence, we have that the sensitivity matrix is given by

$$\mathbf{J}(\mathbf{x}) = \mathbf{P}^\top (\boldsymbol{\nabla}_{\mathbf{x}}\, \mathbf{u}(\mathbf{x}))^\top \tag{3.8}$$

To obtain the derivatives of the fields with respect to the model parameter, we use implicit differentiation. We begin by writing the discretized PDE-constraint as

$$\mathbf{A}(\mathbf{x})\mathbf{u}(\mathbf{x}) = \mathbf{q}.$$

Applying the product rule to differentiate both sides with respect to $\mathbf{x}$, we obtain

$$\boldsymbol{\nabla}_{\mathbf{x}}(\mathbf{A}(\mathbf{x})\mathbf{u}^{\text{fix}})^\top + \mathbf{A}(\mathbf{x})(\boldsymbol{\nabla}_{\mathbf{x}}\, \mathbf{u}(\mathbf{x}))^\top = \mathbf{0}, \tag{3.9}$$

where we use the notation $\mathbf{u}^{\text{fix}} = \mathbf{u}(\mathbf{x})$ to denote the current values of the fields assumed to be constant as part of the product rule. For nonsingular $\mathbf{A}(\mathbf{x})$ this is equivalent to

$$\boldsymbol{\nabla}_{\mathbf{x}}\, \mathbf{u}(\mathbf{x})^\top = -\mathbf{A}(\mathbf{x})^{-1}(\, \boldsymbol{\nabla}_{\mathbf{x}}(\mathbf{A}(\mathbf{x})\mathbf{u}^{\text{fix}}))^\top. \tag{3.10}$$

Finally, inserting this into (3.8) gives the sensitivity matrix

$$\mathbf{J}(\mathbf{x}) = -\mathbf{P}^\top \mathbf{A}(\mathbf{x})^{-1}(\, \boldsymbol{\nabla}_{\mathbf{x}}(\mathbf{A}(\mathbf{x})\mathbf{u}^{\text{fix}}))^\top. \tag{3.11}$$

As can be seen in (3.6) and (3.11), each evaluation of Φ and product with $\mathbf{J}$ or

its transpose requires one PDE simulation per source. This renders solving (3.6) very expensive, particularly for parameter estimation problems involving hundreds of thousands or even millions of sources; see, e.g, [61, 152]. This observation motivates lowering the cost of the PDE solves through model order reduction techniques as we will see in Section 3.2

3.2 Model Order Reduction

MOR can be applied both to the model, reducing the dimensionality of the nonlinear optimization problem (3.6), and the fields, reducing the dimensionality of the PDE-constraint. In the following, we assume that the model is represented efficiently, e.g., using a tensor or OcTree mesh [64], and focus on the latter part. A common theme in MOR is to project the PDEs onto a small, k-dimensional subspace (where $k \ll m$) that is spanned by the basis

$$\mathbf{S} = [\mathbf{s}_1 \ \mathbf{s}_2 \ \dots \ \mathbf{s}_k] \in \mathbb{R}^{m \times k}. \tag{3.12}$$

We can then replace the forward problem in (3.5) with the following reduced forward approximation

$$\mathcal{F}_{j,\text{red}}(\mathbf{x}) = \mathbf{P}^\top \mathbf{u}_{j,\text{red}}(\mathbf{x}), \tag{3.13}$$

where

$$\mathbf{u}_{j,\text{red}}(\mathbf{x}) = \mathbf{S}\mathbf{A}_{\text{red}}(\mathbf{x})^{-1}\mathbf{S}^\top \mathbf{q}_j \tag{3.14}$$

is the reduced approximation of the fields, $j = 1, \dots, N$, and

$$\mathbf{A}_{\text{red}}(\mathbf{x}) = \mathbf{S}^\top \mathbf{A}(\mathbf{x})\mathbf{S} \in \mathbb{R}^{k \times k} \tag{3.15}$$

is the reduced PDE. Inserting the reduced forward problem into (3.6) yields the surrogate problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & \Phi\left(\mathbf{P}^\top \mathbf{S}\mathbf{A}_{\text{red}}(\mathbf{x})^{-1}\mathbf{S}^\top \mathbf{Q}, \mathbf{D}\right) + R(\mathbf{x}), \\ \text{s.t.} \quad & \mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_H. \end{aligned} \tag{3.16}$$

Since the approximate fields $\mathbf{U}_k(\mathbf{x})$ change for every iteration in the optimization scheme (3.16), a key challenge in solving the reduced optimization problem is finding a good basis $\mathbf{S}$ so that $\mathbf{U}_k(\mathbf{x})$ is a good approximation of $\mathbf{U}(\mathbf{x})$ for all possible $\mathbf{x}$.

Prominent examples for finding these bases are moment matching [44], superposition of locally reduced models [112], matrix interpolation [4,126], Interpolatory Model Order Reduction [9,31], Proper Orthogonal Decomposition [59,156] and a Greedy procedure [18,41,108], all of which have been explored in parameter estimation. Most of these techniques use an offline phase in which the PDEs are solved for a large number of right hand sides and a basis is constructed from these solutions. While these methods have been shown to be effective for many problems, they are more difficult to apply for the problem at hand in which the dimensionality of $\mathbf{x}$ is typically in the order of millions and sampling the parameter space becomes intractable.

To avoid the challenge of sampling the high-dimensional parameter space, we propose using an adaptive basis

$$\mathbf{S}(\mathbf{x}) = [\mathbf{s}_1(\mathbf{x})\ \mathbf{s}_2(\mathbf{x})\ \ldots\ \mathbf{s}_k(\mathbf{x})],$$

where now the projection basis depends on the model parameter $\mathbf{x}$. This leads to the optimization problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & \Phi\left(\mathbf{P}^\top \mathbf{S}(\mathbf{x})\mathbf{A}_{\text{red}}(\mathbf{x})^{-1}\mathbf{S}(\mathbf{x})^\top \mathbf{Q}, \mathbf{D}\right) + R(\mathbf{x}), \\ \text{s.t.} \quad & \mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_H. \end{aligned} \tag{3.17}$$

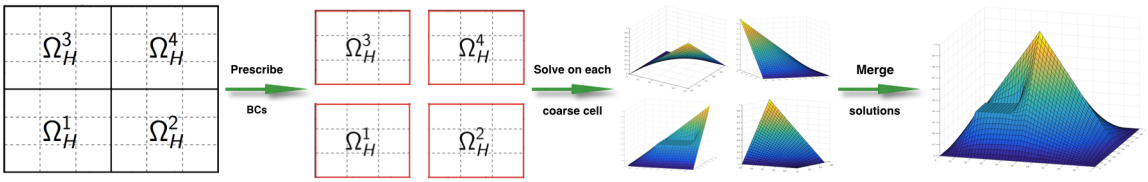


Figure 3.1: Outline of the construction of a multiscale basis function using a bilinear piecewise Lagrangian polynomial as boundary conditions. Here, $m(x) = 10$ in Ω_H^1, and $m(x) = 1$ everywhere else. The adaptivity of the basis is evident by looking at its values on Ω_H^1.

In the following subsection, we use the multiscale finite volume method [39] to construct the basis $\mathbf{S}$. As we show in the next section, where we compute the sensitivities of (3.17) with MSFV, this leads to a computationally tractable smooth optimization scheme. While the adaptive basis approach is not limited to MSFV, computing derivatives might be more involved for other techniques.

3.3 Multiscale Finite Elements/Volumes

We now introduce the basic steps of the MSFV method to construct the basis $\mathbf{S}$ used in (3.13). Our discussion closely follows [39]. Each multiscale basis function is constructed in the following manner.

1. Partition the fine mesh $\mathcal{M}_h$ into a nested coarse mesh $\mathcal{M}_H = \bigcup_{j=1}^{N_c} \Omega_H^j$ where Ω_H^j is the jth coarse cell, and N_c is the number of coarse cells.

2. Choose a forcing term q, and prescribe values to a particular multiscale basis function $s: \mathbb{R}^m \to \mathbb{R}$ on the boundary of the coarse grid cells – denote these block-boundary values by s_{bc}.

3. Obtain the values of the multiscale basis function inside each coarse cell by solving the underlying PDEs on the local fine mesh using the prescribed boundary

conditions for each coarse cell from step 2, i.e.,

$$\mathcal{A}(x(\xi))s(\xi) = q(\xi), \;\; \xi \in \Omega_H^i, \;\; s(\xi) = s_{\text{bc}}, \; \xi \in \partial\Omega_H^i, \tag{3.18}$$

for each $i = 1, ..., N_c$, where in the case of the DCR survey, $\mathcal{A}$ corresponds to the diffusion operator, and s is a particular multiscale basis function. In general, non-zero boundaries for the coarse cells are prescribed with $q = 0$ (or vice-versa). See below for some common options.

The multiscale basis functions are thus obtained by solving the underlying PDEs *locally* and *independently* on each coarse-mesh cell given specialized boundary conditions and/or forcing terms. Each column in the projection basis $\mathbf{S}$ corresponds to one multiscale basis function; the number of columns in $\mathbf{S}$ therefore depends on the number of boundary conditions and/or forcing terms assigned. Furthermore, (3.18) shows that $\mathbf{S}$ is an operator-induced interpolation, thus allowing for the multiscale projection basis to adapt to the current optimal parameter in the optimization scheme. For a detailed convergence analysis, we refer to [23, 40, 80].

The scheme is illustrated in Figure 3.1. The specific choice of boundary conditions and/or forcing terms offers the flexibility to introduce application-specific prior knowledge into the construction. We consider the following techniques:

- **Piecewise Lagrange Polynomials**: We construct multiscale basis functions by assuming s_{bc} to be piecewise Lagrange polynomials on the coarse mesh, and solving (3.18) with $q = 0$. The polynomial basis is a modular component of multiscale methods [39] and other choices have been used, e.g., in [65]. Here, Lagrange polynomials are adequate since piecewise linear finite elements are known to be effective for solving the PDE problem at hand. Due to the dependency of $\mathcal{A}$ on m, the obtained basis function captures the local conductivity structure. For example, unless the PDE parameters are constant in the coarse-

mesh cell the obtained multiscale basis functions generally differ considerably from the generic FEM basis functions [39] .

- **Source Bases**: multiscale basis functions are constructed by solving (3.18) with q as the restricted sources from (3.3) and setting $s_{\text{bc}} = 0$ [39, 65].

- **Global Skeleton**: multiscale basis functions are constructed by first computing the fields u_{ref} using a fixed reference parameter m_{ref}. The values of u_{ref} on the coarse mesh (global skeleton) are then used as s_{bc} in (3.18) with $q = 0$.

- **Local Bases**: multiscale basis functions are constructed by solving for the fields u_{ref} using a fixed reference parameter m_{ref}. Subsequently, boundary conditions for each coarse-mesh block are constructed. For example, on the jth cell, we apply a principal component analysis to the values of the fields u_{ref} on the boundary $\partial\Omega_H^j$ to identify the r most important boundary conditions $u_1^j, u_2^j, \ldots, u_r^j$. The associated multiscale basis functions are then obtained by solving (3.18) in Ω_H^j with $s_l^j = u_l^j$ on $\partial\Omega_H^j$ for each $l = 1, 2, \ldots, r$, $q = 0$, and by keeping the values of the basis as zero in the rest of the domain.

3.4 Optimization with MSFV Methods

We now revisit the sensitivity computation of the reduced misfit in (3.17). In this section, we derive the gradient, or sensitivities, of the reduced misfit and design an efficient mechanism for their implementation. A similar derivation has also been proposed recently in [30], however, we avoid the use of automatic differentiation by using local sensitivity equations (see Section 3.4.5).

3.4.1 Reduced Optimization

We exemplarily compute the derivative of a data vector $\mathbf{d}$ obtained from a single source, $\mathbf{q}$, in (3.13). The general case can be obtained by decomposing the misfit function into a sum over the sources. Furthermore, we assume the sum-of-squared misfit

$$\Phi(\mathbf{x}) = \frac{1}{2}\|\mathbf{r}_{\text{red}}(\mathbf{x})\|^2,$$

where the residual function is

$$\mathbf{r}_{\text{red}}(\mathbf{x}) = \mathbf{P}^\top \mathbf{S}(\mathbf{x})\mathbf{A}_{\text{red}}(\mathbf{x})^{-1}\mathbf{S}(\mathbf{x})^\top \mathbf{q} - \mathbf{d}. \tag{3.19}$$

In this case, the gradient is simply

$$\boldsymbol{\nabla}_{\mathbf{x}}\, \Phi = \mathbf{J}_{\text{red}}(\mathbf{x})^\top\ \mathbf{r}_{\text{red}}(\mathbf{x}),$$

where $\mathbf{J}_{\text{red}}(\mathbf{x})$ is the sensitivity (or Jacobian) of the *reduced* forward model, that is,

$$\mathbf{J}_{\text{red}}(\mathbf{x})^\top = \boldsymbol{\nabla}_{\mathbf{x}} \left(\mathbf{P}^\top \mathbf{S}(\mathbf{x})\mathbf{A}_{\text{red}}(\mathbf{x})^{-1}\mathbf{S}(\mathbf{x})^\top \mathbf{q}\right). \tag{3.20}$$

In the remainder of this section, we derive the sensitivity of the misfit for the case of a fixed basis in the reduced forward model. We then give a detailed derivation of the (more complicated) gradient of the misfit when the basis is adapted to the model. Finally, we compare both results and provide an intuition about the difference of the gradients using the fixed and adaptive basis.

3.4.2 Optimization with Fixed Reduced Space

For problems where $\mathbf{S}$ does not depend on $\mathbf{x}$, $\mathbf{J}_{\text{red}}(\mathbf{x})$ in (3.20) can be computed by the following sensitivity calculation. Writing

$$\mathbf{S}^\top \mathbf{A}(\mathbf{x})\mathbf{S}\mathbf{u}(\mathbf{x}) = \mathbf{S}^\top \mathbf{q},$$

and differentiating both sides with respect to $\mathbf{x}$, we obtain that

$$\mathbf{S}^\top \mathbf{G} + \mathbf{S}^\top \mathbf{A}(\mathbf{x})\mathbf{S}\ \ \boldsymbol{\nabla}_{\mathbf{x}}(\mathbf{u}(\mathbf{x}))^\top = \mathbf{0},$$

where the matrix $\mathbf{G}$ is obtained by differentiating $\mathbf{A}(\mathbf{x})\mathbf{S}\mathbf{u}$ assuming $\mathbf{u}$ is constant, that is,

$$\mathbf{G} = \boldsymbol{\nabla}_{\mathbf{x}}(\mathbf{A}(\mathbf{x})\mathbf{S}\mathbf{u}^{\text{fix}})^\top. \tag{3.21}$$

This implies that

$$\boldsymbol{\nabla}_{\mathbf{x}}\, \mathbf{u}(\mathbf{x})^\top = -(\mathbf{S}^\top \mathbf{A}(\mathbf{x})\mathbf{S})^{-1}\mathbf{S}^\top \mathbf{G}.$$

Multiplying by the receiver matrix we see that the Jacobian of the reduced residual is

$$\mathbf{J}_{\text{red}}(\mathbf{x}) = -\mathbf{P}^\top \mathbf{S}(\mathbf{S}^\top \mathbf{A}(\mathbf{x})\mathbf{S})^{-1}\mathbf{S}^\top \mathbf{G}. \tag{3.22}$$

3.4.3 Optimization with Adaptive Reduced Space

Computing the sensitivity is more involved when using an adaptive reduced basis in which $\mathbf{S}$ depends on the model [30]. In this case, we need to differentiate the basis vectors, $\mathbf{s}_1(\mathbf{x}), \ldots, \mathbf{s}_k(\mathbf{x})$, with respect to the model $\mathbf{x}$. This derivation is not standard and provided in detail below. We also provide a description of our implementation that makes computing matrix-vector products with these derivatives tractable.

Similar to the previous section, the sensitivities for the general case are computed

using implicit differentiation. Consider the following equation:

$$\mathbf{S}(\mathbf{x})^\top \mathbf{A}(\mathbf{x})\mathbf{S}(\mathbf{x})\mathbf{u}(\mathbf{x}) = \mathbf{S}\left(\mathbf{x}\right)^\top \mathbf{q}. \tag{3.23}$$

Differentiating both sides, we get

$$\boldsymbol{\nabla}_{\mathbf{x}} \left(\mathbf{S}(\mathbf{x})^\top \mathbf{A}(\mathbf{x})\mathbf{S}(\mathbf{x})\mathbf{u}(\mathbf{x})\right) = \boldsymbol{\nabla}_{\mathbf{x}} \left(\mathbf{S}\left(\mathbf{x}\right)^\top \mathbf{q}\right). \tag{3.24}$$

Applying the product rule to (3.24), we obtain

$$\boldsymbol{\nabla}_{\mathbf{x}} \left(\mathbf{S}(\mathbf{x})^\top \mathbf{A}^{\text{fix}}\mathbf{S}_k^{\text{fix}}\mathbf{u}^{\text{fix}}\right) + \mathbf{S}(\mathbf{x})^\top \boldsymbol{\nabla}_{\mathbf{x}} \left(\mathbf{A}(\mathbf{x})\mathbf{S}(\mathbf{x})\mathbf{u}(\mathbf{x})\right)^\top = \boldsymbol{\nabla}_{\mathbf{x}} \left(\mathbf{S}\left(\mathbf{x}\right)^\top \mathbf{q}\right), \tag{3.25}$$

where $\mathbf{A}^{\text{fix}} = \mathbf{A}(\mathbf{x}), \mathbf{S}_k^{\text{fix}} = \mathbf{S}(\mathbf{x})$, and $\mathbf{u}^{\text{fix}} = \mathbf{u}(\mathbf{x})$ are treated as constants as part of the product rule.

For ease of presentation, we denote the operators that compute the directional derivatives of $\mathbf{S}(\mathbf{x})$ and its transpose by

$$\mathcal{Y}(\mathbf{v}, \mathbf{x}) = \boldsymbol{\nabla}_{\mathbf{x}} \left(\mathbf{S}(\mathbf{x})\mathbf{v}\right)^\top \tag{3.26}$$

and

$$\mathcal{X}(\mathbf{w}, \mathbf{x}) = \boldsymbol{\nabla}_{\mathbf{x}} \left(\mathbf{S}(\mathbf{x})^\top \mathbf{w}\right)^\top. \tag{3.27}$$

Using this notation, and applying the product rule to (3.25), we obtain

$$\mathcal{X}(\mathbf{q}, \mathbf{x}) = \mathcal{X}((\mathbf{A}(\mathbf{x})\mathbf{S}(\mathbf{x})\mathbf{u}(\mathbf{x}), \mathbf{x}) + \mathbf{S}(\mathbf{x})^\top \mathbf{G} + \mathbf{S}(\mathbf{x})^\top \mathbf{A}(\mathbf{x})\mathcal{Y}(\mathbf{u}, \mathbf{x})\mathbf{A}_{\text{red}}(\mathbf{x})(\boldsymbol{\nabla}_{\mathbf{x}}\, \mathbf{u}(\mathbf{x}))^\top,$$

where $\mathbf{G}$ is as defined in (3.21). Assuming that the reduced discrete PDE is non-

singular, this implies that

$$\boldsymbol{\nabla}_{\mathbf{x}}\,\mathbf{u}(\mathbf{x})^{\top} = \mathbf{A}_{\text{red}}(\mathbf{x})^{-1}\left(\mathcal{X}(\mathbf{q}-\mathbf{A}(\mathbf{x})\mathbf{S}(\mathbf{x})\mathbf{u},\mathbf{x}) - \mathbf{S}(\mathbf{x})^{\top}\left(\mathbf{G}+\mathbf{A}(\mathbf{x})\mathcal{Y}(\mathbf{u},\mathbf{x})\right)\right). \tag{3.28}$$

Multiplying by the receiver matrix, $\mathbf{P}^{\top}$, the current basis, $\mathbf{S}(\mathbf{x})$, and applying the product rule to (3.19) yields the Jacobian of the reduced forward problem

$$\mathbf{J}_{\text{red}}(\mathbf{x}) = \mathbf{P}^{\top}\left(\mathcal{Y}(\mathbf{t}(\mathbf{x})) + \mathbf{S}(\mathbf{x})(\,\boldsymbol{\nabla}_{\mathbf{x}}\,\mathbf{u}(\mathbf{x})^{\top})\right), \tag{3.29}$$

where the coefficients of the fields with respect to the multiscale basis are denoted by

$$\mathbf{t}(\mathbf{x}) = \left(\mathbf{S}(\mathbf{x})^{\top}\mathbf{A}(\mathbf{x})\mathbf{S}(\mathbf{x})\right)^{-1}\mathbf{S}(\mathbf{x})^{\top}\mathbf{q}. \tag{3.30}$$

Clearly, there is a difference between the Jacobian obtained for a fixed basis in (3.22) and the Jacobian obtained for the adaptive basis given in (3.29). Note that even if $\mathbf{S}(\mathbf{x})$ changes slowly with respect to $\mathbf{x}$, the term $\mathbf{A}(\mathbf{x})\mathcal{Y}(\mathbf{u},\mathbf{x})$ may not be small since $\mathbf{A}(\mathbf{x})$ is the discretization of a differential operator. This might cause problems in the optimization when computing the gradients between iterates using the Jacobian in (3.22). Indeed, if one ignores the dependence of $\mathbf{S}(\mathbf{x})$ on $\mathbf{x}$ and uses the Jacobian in (3.22), the error in the gradient can be rather large.

3.4.4 Illustrating the Error

Ignoring the dependency of $\mathbf{S}(\mathbf{x})$ on the model $\mathbf{x}$ might not always lead to a large error in the gradient computation. Consider first the ideal case where the subspace

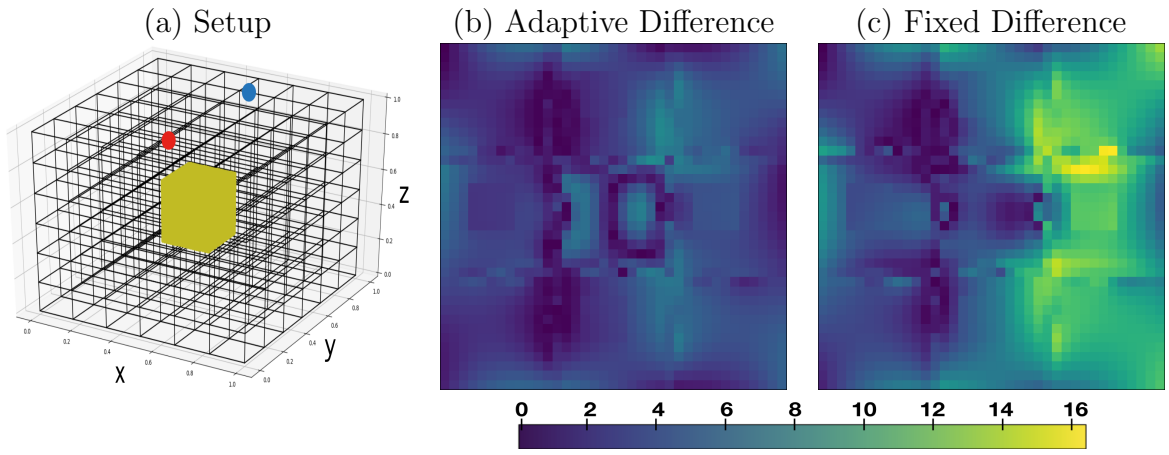


Figure 3.2: (a): Example of test-problem setup containing one dipole source (blue and red dot), and a conductive block (corresponding to the perturbation $\delta\mathbf{x}$) in the center of the domain. Here, we have 1369 receivers, which corresponds to having a receiver node in every fine-mesh node on the top surface of the domain. (b-c): Difference plot of fine-mesh and multiscale sensitivities on the surface of the domain. We obtain relative errors of 0.011 and 0.025 for the multiscale adaptive and fixed sensitivities, respectively. The color axis is chosen identically to allow for comparison.

$\mathbf{S}(\mathbf{x})$ is chosen in such a way that

$$\mathbf{P}^\top \mathbf{A}(\mathbf{x})^{-1}\mathbf{q} = \mathbf{P}^\top \mathbf{S}(\mathbf{x})\mathbf{A}_{\text{red}}(\mathbf{x})^{-1}\mathbf{S}(\mathbf{x})^\top \mathbf{q} \tag{3.31}$$

for every $\mathbf{x}$. In this case, it is clear that the dependence of $\mathbf{S}(\mathbf{x})$ on $\mathbf{x}$ does not affect the quality of the multiscale solution. We denote the error between the full and the reduced forward problem by

$$\mathbf{e}_k(\mathbf{x}) = \mathbf{P}^\top \left(\mathbf{A}(\mathbf{x})^{-1}\mathbf{q} - \mathbf{S}(\mathbf{x})\mathbf{A}_{\text{red}}(\mathbf{x})^{-1}\mathbf{S}(\mathbf{x})^\top \mathbf{q}\right), \tag{3.32}$$

and assume that for some $\epsilon > 0$, the reduced forward problem satisfies the relaxed version of (3.31)

$$\|\mathbf{e}_k(\mathbf{x})\| \leq \epsilon, \quad \forall \mathbf{x}. \tag{3.33}$$

From the Taylor expansion of $\mathbf{e}_k(\mathbf{x})$, we have that for any small perturbation $\delta\mathbf{x}$,

$$\mathbf{e}_k(\mathbf{x}+\delta\mathbf{x}) = \mathbf{e}_k(\mathbf{x}) + \mathbf{J}_{\mathbf{e}_k}\delta\mathbf{x} + \mathcal{O}(\|\delta\mathbf{x}\|^2), \tag{3.34}$$

where

$$\mathbf{J}_{\mathbf{e}_{\text{red}}} = \boldsymbol{\nabla}_{\mathbf{x}}\left(\mathbf{P}^\top\left(\mathbf{A}(\mathbf{x})^{-1}\mathbf{q} - \mathbf{S}(\mathbf{x})\mathbf{A}_{\text{red}}(\mathbf{x})^{-1}\mathbf{S}(\mathbf{x})^\top\mathbf{q}\right)\right)^\top = \mathbf{J} - \mathbf{J}_{\text{red}} \tag{3.35}$$

is the Jacobian of the error $\mathbf{e}_k$. This implies that

$$\begin{aligned}\|\mathbf{e}_k(\mathbf{x}+\delta\mathbf{x}) - \mathbf{e}_k(\mathbf{x})\| &= \|\mathbf{J}_{\mathbf{e}_k}\delta\mathbf{x} + \mathcal{O}(\|\delta\mathbf{x}\|^2)\| &(3.36)\\ &\leq \|(\mathbf{J}_{\text{red}} - \mathbf{J})\delta\mathbf{x}\| + \mathcal{O}(\|\delta\mathbf{x}\|^2) &(3.37)\\ &\leq 2\epsilon. &(3.38)\end{aligned}$$

Since the choice of $\delta\mathbf{x}$ is arbitrary, we obtain that the columns of $\mathbf{J}$ and $\mathbf{J}_{\text{red}}$ are at most 2ϵ away.

Now, consider the solution of the optimization problem where the reduced model yields an accurate approximation to the desired function. In this case, if the change in $\mathbf{x}$ is not too large, ignoring the dependence of $\mathbf{S}$ on $\mathbf{x}$ may not lead to any serious complications. Nonetheless, if the subspace approximation of the reduced model to the true model is not negligible, then ignoring it in the computation may lead to gross errors in the evaluation of the derivatives and to the lack of convergence of the optimization problem. As a small example, we compare the sensitivities computed using the fine-mesh discretization with those obtained for the fixed and adaptive multiscale method. We use a test problem with a fine mesh containing $36 \times 36 \times 12$ cells and a coarse mesh containing $12 \times 12 \times 12$ cells discretizing the domain $\Omega = (0, 1)^3$. We use one dipole source and 1369 receivers located at the top surface of the domain. Following the finite volume discretization described in [60], sources,

receivers, and fields, are discretized at the nodes of the mesh, whereas the model $\mathbf{x}$ is discretized at the cell centers. As a background model we use chose $m_{\text{ref}}(x) \equiv 10^{-2}$. To simulate a conductive block we add $\delta m(x) = 10^{-2}$ for all $x \in (.25, .5)^3$ and $\delta m(x) = 0$ elsewhere. We compare the multiscale fixed and adaptive sensitivities, $\mathbf{J}_{\text{red}}^{\text{fix}}(\mathbf{x})\mathbf{v}$ and $\mathbf{J}_{\text{red}}^{\text{adapt}}(\mathbf{x})\mathbf{v}$, with the fine-mesh sensitivities $\mathbf{J}(\mathbf{x})\mathbf{v}$. Here, $\mathbf{v} = [1, \ldots, 1]^\top$ is needed for the computation of the sensitivities since we do not compute $\mathbf{J}, \mathbf{J}_{\text{red}}^{\text{adapt}}$, and $\mathbf{J}_{\text{red}}^{\text{fix}}$ explicitly but rather their action on a vector. For $\mathbf{J}^{\text{fix}}(\mathbf{x})$, the matrix $\mathbf{S}$ in (3.22) is evaluated at $\mathbf{x}_{\text{ref}}$ and therefore ignores the dependence on $\mathbf{x}$, whereas for $\mathbf{J}_{\text{red}}^{\text{adapt}}(\mathbf{x})$, $\mathbf{S}$ is evaluated at $\mathbf{x}$ in (3.29). We obtain relative errors of

$$\frac{\|\mathbf{J}(\mathbf{x})\mathbf{v} - \mathbf{J}_{\text{red}}^{\text{adapt}}(\mathbf{x})\mathbf{v}\|_2^2}{\|\mathbf{J}(\mathbf{x})\mathbf{v}\|_2^2} \approx 0.011 \quad \text{and} \quad \frac{\|\mathbf{J}(\mathbf{x})\mathbf{v} - \mathbf{J}_{\text{red}}^{\text{fix}}(\mathbf{x})\mathbf{v}\|_2^2}{\|\mathbf{J}(\mathbf{x})\mathbf{v}\|_2^2} \approx 0.025. \tag{3.39}$$

As expected, ignoring the dependence of $\mathbf{S}$ on $\mathbf{x}$ may lead to large errors in the the sensitivity computations as can be seen in Figure 3.2.

Finally, Eq. (3.23) shows that when the coarse mesh approaches the fine mesh, we obtain the fine-mesh system (see also [80]). Hence, the reconstruction obtained using the multiscale approaches the fine-mesh reconstruction, and will be identical when $k = m$.

3.4.5 Local Sensitivity Computation

The directional derivatives $\mathcal{Y}(\mathbf{v}, \mathbf{x})$ and $\mathcal{X}(\mathbf{w}, \mathbf{x})$ are key components in the computation of the reduced adaptive sensitivities; see (3.29). The computations of $\mathcal{Y}(\mathbf{v}, \mathbf{x})$ and $\mathcal{X}(\mathbf{w}, \mathbf{x})$ require the multiscale basis $\mathbf{S}$ to be differentiable, which is ensured when using MSFV methods.

Let the discretized version of (3.18) in a particular coarse cell be given by

$$\mathbf{A}(\mathbf{x})\mathbf{s}_j(\mathbf{x}) = \mathbf{q}_j$$

for $j = 1, ..., k$, where $\mathbf{s}_j$ is the discretized version of the jth multiscale basis function. By construction of each multiscale basis function $\mathbf{s}_j$, $j = 1, \ldots, k$, this implies that

$$\mathbf{A}(\mathbf{x})\mathbf{S}(\mathbf{x})\mathbf{v} = \mathbf{Q}\mathbf{v}, \tag{3.40}$$

where as before, $\mathbf{S}(\mathbf{x}) = [\mathbf{s}_1(\mathbf{x})\ \mathbf{s}_2(\mathbf{x})\ \ldots\ \mathbf{s}_k(\mathbf{x})]$, $\mathbf{Q} = [\mathbf{q}_1\ \mathbf{q}_2\ \ldots\ \mathbf{q}_k]$, and $\mathbf{v} \in \mathbb{R}^k$ is the vector corresponding to the directional derivative $\mathcal{Y}(\mathbf{v}, \mathbf{x})$.

We exemplarily discuss the case of Dirichlet boundary conditions on the coarse block. Derivatives for local forcing terms can be computed along the same lines. We reduce the linear system (3.40) in the following manner. For ease of presentation, denote $\mathbf{S}(\mathbf{x})\mathbf{v}$ by $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^m$. Let N_I and N_B be the number of interior and boundary nodes in the current cell, respectively, and define $\mathbf{f}_I(\mathbf{x}) \in \mathbb{R}^{N_I}$ and $\mathbf{f}_B \in \mathbb{R}^{N_B}$ as the corresponding values of $\mathbf{f}(\mathbf{x})$ at the interior and boundary nodes of the chosen coarse cell (note that $\mathbf{x}_B$ is known and does not depend on the model). Similarly, let $\hat{\mathbf{q}} = \mathbf{Q}\mathbf{v}$, and define $\hat{\mathbf{q}}_I \in \mathbb{R}^{N_I}$ and $\hat{\mathbf{q}}_B \in \mathbb{R}^{N_B}$ as the entries of $\hat{\mathbf{q}}$ in the inner and boundary nodes, respectively. We can then rewrite (3.40) in terms of the interior nodes as

$$\mathbf{A}_{II}(\mathbf{x})\mathbf{f}_I(\mathbf{x}) = \hat{\mathbf{q}}_I - \mathbf{A}_{IB}(\mathbf{x})\mathbf{f}_B, \tag{3.41}$$

where we partition the matrix $\mathbf{A}$ as follows

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{II} & \mathbf{A}_{IB} \\ \mathbf{A}_{BI} & \mathbf{A}_{BB} \end{pmatrix}. \tag{3.42}$$

Differentiating both sides of (3.41) with respect to $\mathbf{x}$, and using the product rule, we can isolate the term $\boldsymbol{\nabla}_{\mathbf{x}} \mathbf{f}_I(\mathbf{x})$ to obtain

$$\boldsymbol{\nabla}_{\mathbf{x}} \mathbf{f}_I(\mathbf{x})^\top = -\mathbf{A}_{II}(\mathbf{x})^{-1}(\boldsymbol{\nabla}_{\mathbf{x}}(\mathbf{A}_{II}(\mathbf{x})\mathbf{f}_I) + \boldsymbol{\nabla}_{\mathbf{x}}(\mathbf{A}_{IB}(\mathbf{x})\mathbf{f}_B))^\top, \tag{3.43}$$

Algorithm 3.1: Construction of Local Sensitivities X and Y

given current model $\mathbf{x}$ and vectors $\mathbf{v}$ and $\mathbf{w}$:
for *each coarse cell:* **do**
    compute inner and boundary node indices I_{in}, I_{bc}
    obtain $\mathbf{x}$, $\mathbf{v}$ and $\mathbf{w}$ values on current coarse cell
    compute local basis $\mathbf{S}(\mathbf{x})$ (see Section 3.3)
    assemble local PDE operator $\mathbf{A}(\mathbf{x})$
    use I_{in} and I_{bc} to obtain $\mathbf{A}_{II}(\mathbf{x})$ and $\mathbf{A}_{IB}(\mathbf{x})$ (see Eq. (3.42))
    construct inner node basis $\mathbf{R}_I$
    solve for (3.45) and (3.44) to get $\mathcal{X}(\mathbf{w}, \mathbf{x})$ and $\mathcal{Y}(\mathbf{v}, \mathbf{x})$ on current cell
patch all local $\mathcal{X}(\mathbf{w}, \mathbf{x})$ and $\mathcal{Y}(\mathbf{v}, \mathbf{x})$ to form global directional derivatives

which can be rewritten as

$$\boldsymbol{\nabla}_{\mathbf{x}} \mathbf{f}(\mathbf{x})^{\top} = -\mathbf{R}_I \mathbf{A}_{II}(\mathbf{x})^{-1} \mathbf{R}_I{}^{\top} (\boldsymbol{\nabla}_{\mathbf{x}} (\mathbf{A}(\mathbf{x})\mathbf{f}))^{\top}, \tag{3.44}$$

where $\mathbf{R}_I \in \mathbb{R}^{m \times N_I}$ is the basis for the inner nodes constructed by eliminating the columns of the identity matrix $\mathbf{I} \in \mathbb{R}^{m \times m}$ corresponding to the indices of the boundary nodes. The above computation yields

$$\mathcal{Y}(\mathbf{v}, \mathbf{x}) = \boldsymbol{\nabla}_{\mathbf{x}} \mathbf{S}(\mathbf{x})\mathbf{v} = \boldsymbol{\nabla}_{\mathbf{x}} \mathbf{f}(\mathbf{x})$$

on a local coarse cell. This procedure is repeated for each coarse cell, and the solutions are merged together as it is done in the construction of $\mathbf{S}$. The construction of $\mathcal{X}(\mathbf{w}, \mathbf{x})$ is along the same lines and is given by

$$\mathcal{X}(\mathbf{w}, \mathbf{x}) = \boldsymbol{\nabla}_{\mathbf{x}} \mathbf{S}(\mathbf{x})^{\top} \mathbf{w} = -(\boldsymbol{\nabla}_{\mathbf{x}} (\mathbf{A}(\mathbf{x})\mathbf{S})^{\top} \mathbf{R}_I \mathbf{A}_{II}(\mathbf{x})^{-\top} \mathbf{R}_I{}^{\top} \mathbf{w}. \tag{3.45}$$

We present an outline of the algorithm for computing Y and X in Algorithm 3.1. We would like to emphasize that, similar to the construction of the multiscale basis $\mathbf{S}$, computing $\mathcal{Y}(\mathbf{v}, \mathbf{x})$ and $\mathcal{X}(\mathbf{w}, \mathbf{x})$ can be done *locally* and *independently* on each coarse cell, leading to a parallelizable implementation. With $\mathbf{S}(\mathbf{x})$, $\mathcal{Y}(\mathbf{v}, \mathbf{x})$, and $\mathcal{X}(\mathbf{w}, \mathbf{x})$, we

are thus able to solve the reduced adaptive optimization using gradient or Hessian-based methods.

3.5 Numerical Results

In this section, we demonstrate the potential of the multiscale MOR parameter estimation method for a DCR survey. We show that the multiscale inversion can reduce the time-to-solution compared to iterative PDE solvers, which are necessary for large-scale parameter estimation problems, with a moderate loss of reconstruction quality. We provide results for the fixed and adaptive multiscale basis; see Section 3.4.2 and Section 3.4.3, respectively. We experiment on two test data: a block model that consists of two conductive blocks and that is homogeneous in the rest of the domain for proof-of-concept, and a more realistic 3D SEG/EAGE model of a salt reservoir described in [3]. We also perform a strong scaling test for the construction of $\mathbf{S}(\mathbf{x})$, $\mathcal{Y}(\mathbf{v}, \mathbf{x})$, and $\mathcal{X}(\mathbf{w}, \mathbf{x})$ to show the parallel efficiency of our current implementation.

Our multiscale framework is implemented as an extension to `jInv` [136], an open-source package for PDE parameter estimation written in Julia [13]. For the discretization of the PDE operators, we use built-in methods in `jInv`, which are based on the mimetic finite volume method described in [60]. We use `jInv`'s methods for optimization, misfit functions, and regularizers. For brevity, we omit the term "multiscale" when referring to the multiscale adaptive and multiscale fixed inversions and refer to them as adaptive and fixed inversions instead.

3.5.1 Block Model Test Problem

We compare the reconstructions of the block model using the fine-mesh inversion on a mesh containing $36 \times 36 \times 12$ cells discretizing the domain $\Omega = (0, 1)^3$, and the multiscale inversion using fixed and adaptive bases for two different coarse meshes.

The first one contains 9 coarse cells with 12^3 fine cells per coarse cell, and the second one contains 72 coarse cells with 6^3 fine cells per coarse cell. The test problem features 25 sources and $1,369$ receivers located on the top surface. Following the finite volume discretization presented in [60], we use a cell-centered discretization of the model $\mathbf{x}$ and nodal discretizations of the sources, receivers, and fields.

To construct the multiscale basis, we use boundary conditions obtained from Lagrange polynomials which correspond to the standard multiscale finite element/volume method [39]. We augment the basis by adding 25 basis functions of the global skeleton, and 93 and 400 local basis boundary conditions (see Section 3.3) for the coarse meshes with the 12^3 and 6^3 coarsenings, respectively. In our case, we do not use the source basis functions (see Section 3.3) since the sources in our experiments are located on the boundary (top surface) of the domain. The construction of these bases are done using 0 boundary conditions and would therefore lead to the trivial $s \equiv 0$ bases in our experiment.

Constructing these boundary conditions required 25 fine-mesh PDE solves in an offline phase which took about 0.24 seconds. The overall number of basis functions is $k = 150$ and $k = 572$ for the 12^3 and 6^3 respective coarsening strategies. To solve the fine-mesh forward problem, we use MUMPS [2] and a block CG method [125] with at most 100 iterations and stopping tolerance of 10^{-6}. To make the block CG competitive, we use symmetric successive over-relaxation (SSOR) as a preconditioner and use the implementation from KrylovMethods [134] that uses direct BLAS [36] access for efficiency and that uses multithreaded matrix-vector products. We choose a stopping criteria that provides a good trade-off between efficiency and accuracy of the gradient; see Table 3.1. The reduced multiscale forward problems are all solved using MUMPS as we assume they are always small enough to be solved using a direct solver. For the inversions, we use 10 projected Gauss-Newton iterations with at most 15 CG iterations for each step. We add 1% noise to the data, and enforce smoothness

	k	time (seconds)	relative error
fine mesh (MUMPS)	17,797	26.7	0
fine mesh (block CG)	17,797	423.9	6.7e-2
MS fixed 6^3	572	26.5	3.8e-2
MS adaptive 6^3	572	306.6	2.1e-2
MS fixed 12^3	150	25.4	4.9e-2
MS adaptive 12^3	150	114.7	2.6e-2

Table 3.1: Relative errors and runtimes for the block model test problem in Section 3.5.1. Here, k corresponds to the total number of basis functions (and size of the PDE linear system), MS Fixed 6^3 and MS Adaptive 6^3 correspond to the multiscale inversions using coarse cells consisting of 6^3 fine-mesh cells per coarse cell, and MS Fixed 12^3 and MS Adaptive 12^3 correspond to the multiscale inversions using coarse cells of size 12^3 fine-mesh cells per coarse cell. The experiment is run on a standard Macbook air 2015 running macOS Sierra, with Intel core-i7 2.2 GHz CPU with 2 cores and 8 GB of RAM. The adaptive multiscale reconstruction is done in parallel using 2 processors. The fine-mesh forward problem is solved using MUMPS and block CG. The reconstruction using MUMPS is considered as the baseline and is therefore assigned an error of 0.

by using a diffusion regularizer with regularization parameter $\alpha = 10^{-8}$. Here, we choose a low regularization parameter as we use iterative regularization by stopping at iteration 10.

In Table 3.1, we show results for the small block model test problem. The fine-mesh reconstruction requires solving a $17,797 \times 17,797$ linear system for the forward problem, whereas in the coarse meshes consisting of 6^3 and 12^3 cells, we project the PDEs onto 572×572 and 150×150-dimensional subspaces, respectively. We see that the adaptive inversions have a slower runtime than the fine-mesh inversion when using MUMPS. Indeed, when the problem size is small enough to be solved using a direct solver, there is no need to use MOR. However, in the typical setting where MOR is used, iterative solvers are more commonly used to solve large linear systems [137], and as shown in Table 3.1, both adaptive and fixed inversions are faster than the fine-mesh inversion using the block CG algorithm as a linear solver.

The adaptive inversion is in general slower than the fixed inversion since the projection bases must be rebuilt in every Gauss-Newton iteration and the sensitivity computations are more involved. However, as can be seen in the relative errors

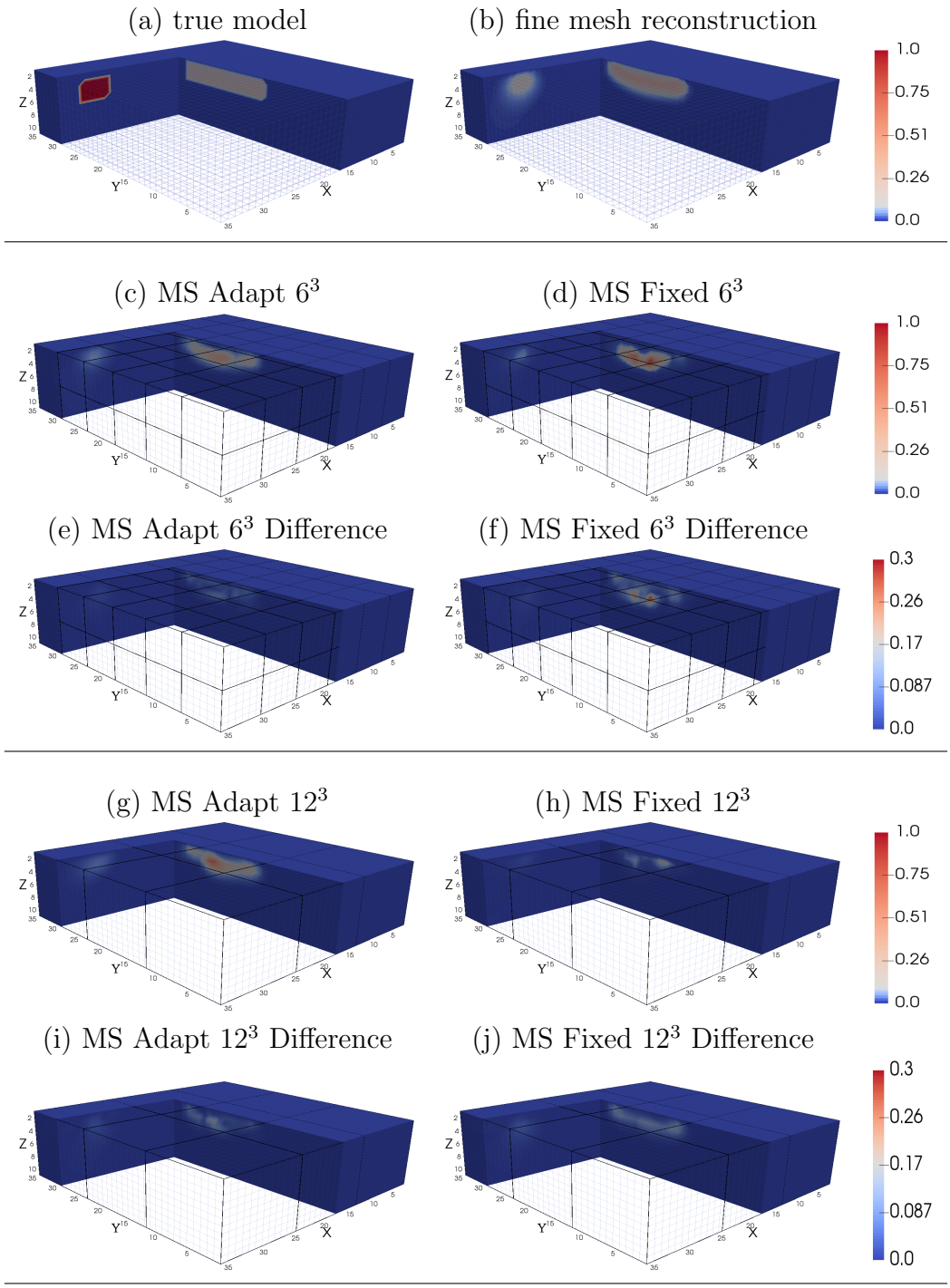


Figure 3.3: Model reconstructions for the block model test problem using the fine mesh, and the adaptive and fixed multiscale inversions for two different coarsenings: 6^3 and 12^3 fine-mesh cells per coarse cell, as well as their respective differences with the fine-mesh reconstruction. The figures were reproduced using the data visualization software Paraview [7].

and in Figure 3.3, we obtain superior reconstructions using the adaptive inversions. The relative error of the adaptive reconstructions are about two times lower than the relative errors of the fixed reconstructions. This coincides with the images in Figure 3.3, where for the aggressive 12^3 coarsening, the adaptive inversion manages to reconstruct the shape much better than the fixed reconstruction. For the more moderate 6^3 coarsening, we project to a much larger subspace, leading to a fair reconstruction of the shape in both fixed and adaptive inversions. However, the intensity values are more accurate in the adaptive reconstruction than in the fixed reconstruction.

	k	time (hours)	relative error
fine mesh (MUMPS)	139,425	0.46	0
fine mesh (block CG)	139,425	5.98	3.2e-2
MS Fixed 8^3	4415	0.25	7.0e-2
MS Adaptive 8^3	4415	3.08	4.0e-2
MS Fixed 16^3	1009	0.22	1.6e-1
MS Adaptive 16^3	1009	1.97	1.1e-1

Table 3.2: Relative errors and runtimes for the SEG test problem in Section 3.5.2. Here, k corresponds to the total number of basis functions (and size of the reduced discretized PDE), MS Fixed 8^3 and MS Adaptive 8^3 correspond to the multiscale inversions using coarse cells consisting of 8^3 fine-mesh cells per coarse cell, and MS Fixed 16^3 and MS Adaptive 16^3 correspond to the multiscale inversions using coarse-mesh cells of size 16^3 fine-mesh cells per coarse cell. The experiment is run on a shared memory computer operating Ubuntu 14.04 with 2 Intel Xeon E5-2670 v3 2.3 GHz CPUs using 12 cores each, and a total of 128 GB of RAM. Here, Julia is installed and compiled using Intel Math Kernel Library (MKL). The adaptive multiscale reconstruction is done in parallel using 16 processors. The fine-mesh forward problem is solved using MUMPS and block CG. The reconstruction using MUMPS is considered as the baseline and is therefore assigned an error of 0.

3.5.2 SEG/EAGE Test Problem

As a more realistic test problem, we consider the 3D SEG/EAGE model problem. The domain is of size 13.5 km $\times$ 13.5 km $\times$ 4.2 km, and is divided into $64 \times 64 \times 32$ equally-sized fine-mesh cells of size 211 m $\times$ 211 m $\times$ 131 m each. The DCR data is measured by 3698 receivers and generated by 72 dipole sources located on the top surface of the

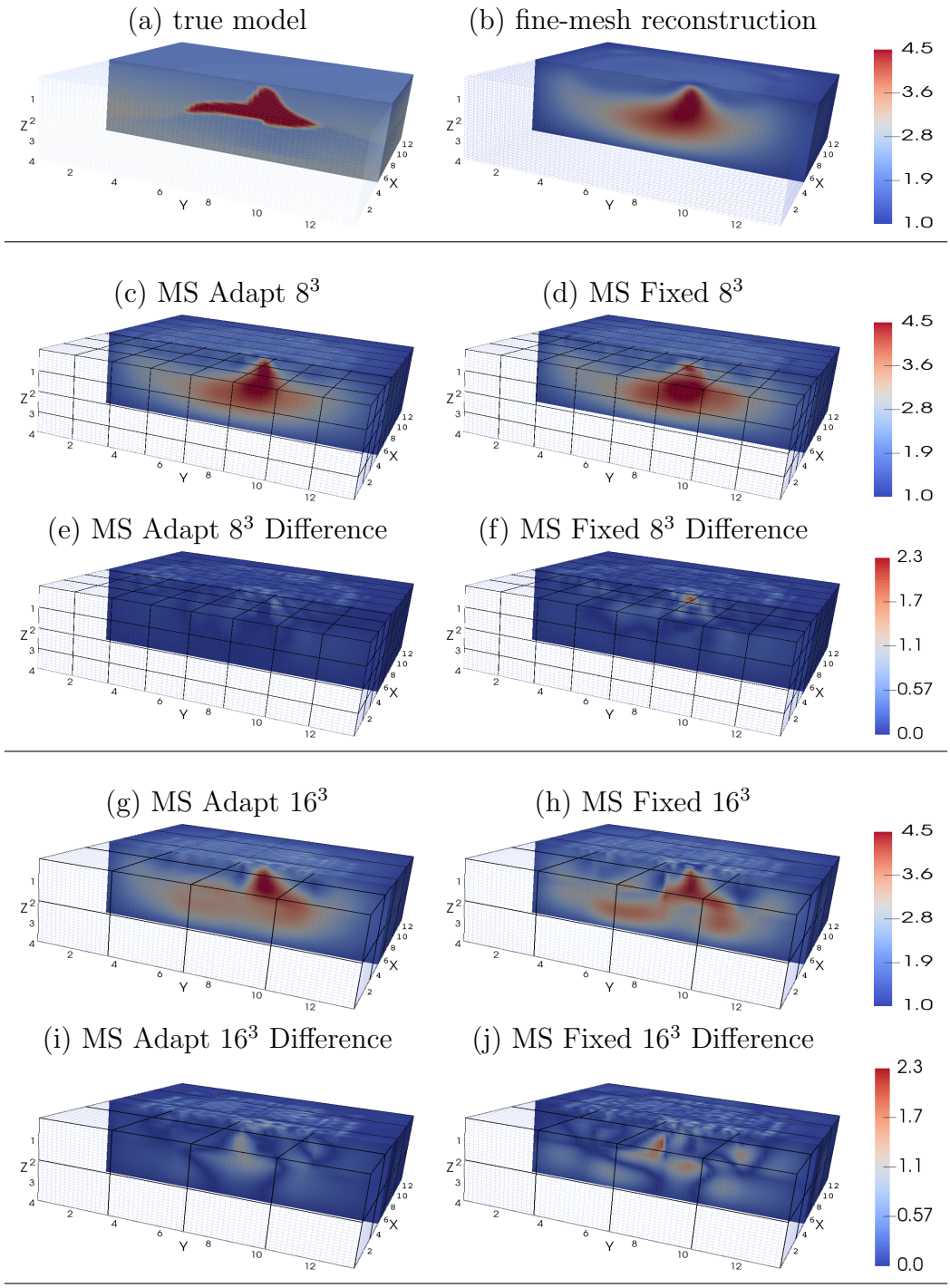


Figure 3.4: Model reconstructions for the SEG/EAGE test problem using the fine mesh, and the adaptive and fixed multiscale inversions for two different coarsenings: 6^3 and 12^3 fine-mesh cells per coarse cell, as well as their respective differences with the fine-mesh reconstruction. The figures were reproduced using the data visualization software Paraview [7].

domain. We compare the fine-mesh inversion with the fixed and adaptive inversions using two different coarse meshes: one containing 32 coarse cells with 16^3 fine cells per coarse cell and the other containing 256 coarse cells with 8^3 fine cells per coarse cell. As in the block model test problem, the model is discretized on the cell-centers whereas the sources and receivers are discretized on the nodes. The test data is generated in `jInv` using the fine mesh. We use Lagrange polynomials to construct the multiscale basis and augment the basis by adding 72 basis functions of the global skeleton. We also use 862 and 3938 local basis boundary conditions described in Section 3.3 for the coarse meshes, respectively. We do not use source bases since our sources are located on the boundary (top surface) of the domain as in the block model test problem. The construction of these boundary conditions required 72 fine-mesh PDE solves in an offline phase which took about 2.21 seconds. For this test problem, we have $k = 1009$ and 4415 basis functions for the 16^3 and 8^3 coarsenings, respectively. Similar to the block model test problem, we use MUMPS and the block CG method with at most 100 iterations and stopping tolerance of 10^{-6}. As in the previous experiment, we use SSOR as a preconditioner and use the implementation from KrylovMethods [134] and multithreaded matrix-vector products. We choose the stopping tolerance so that a good trade-off between efficiency and accuracy of the gradient is observed; see Table 3.2. For the inversions, we use 10 projected Gauss-Newton iterations with at most 15 CG iterations for each step. We add 1% noise to the data, and enforce smoothness by using a diffusion regularizer with regularization parameter $\alpha = 10^{-15}$ in both fine-mesh and multiscale inversions. Our inversion settings are the same as described in [136] to allow for comparison. The low optimization stopping tolerance is chosen to ensure that all methods have the same number of iterations, and the regularization parameter is also chosen to be low since we use iterative regularization by stopping at iteration 10.

In Table 3.2, we show results for the SEG model test problem. The fine-mesh

reconstruction requires solving a $139,425 \times 139,425$ linear system for the forward problem, whereas in the coarse meshes consisting of 8^3 and 16^3 cells per coarse block, we project the PDEs onto 4415×4415 and 1009×1009-dimensional subspaces, respectively. As in the previous experiment, the runtimes of the adaptive inversions are larger than the one for MUMPS but considerably smaller than the one obtained using the iterative block CG algorithm as a linear solver.

The relative errors and Figure 3.4 show that more accurate reconstructions are obtained using the adaptive inversion - this is particularly clear in the 16^3 coarsenings. In the 8^3 coarsenings, the coarse cells are fine enough so that the models are very similar; however, the adaptive reconstruction captures the peak of the fine-mesh model, whereas in the fixed reconstruction, there are discontinuities near the peak of the model. This is reflected in the relative errors in Table 3.2 and in the difference plots in Figure 3.4.

3.5.3 Parallel Efficiency

The computations of $\mathbf{S}(\mathbf{x}), \mathcal{Y}(\mathbf{v}, \mathbf{x}), \mathcal{X}(\mathbf{w}, \mathbf{x})$ require local PDE solves which can be performed independently on each coarse cell and provides an opportunity for parallel processing; see also Section 3.5.2. To demonstrate this, we test the behavior of a simple prototype implementation of these methods using Julia built-in methods [13]. We use a fixed problem size as we increase the number of workers from 1 to 8 as shown in Table 3.3. We use the same setup as in the example from Section 3.5.2, i.e., a mesh of size $64 \times 64 \times 32$ with a coarsening of 16^3 fine-mesh cells per coarse cell. Increasing the number of workers from 1 to 8 decreases the runtimes of $\mathbf{S}(\mathbf{x})$, $\mathcal{Y}(\mathbf{v}, \mathbf{x})$, and $\mathcal{X}(\mathbf{w}, \mathbf{x})$ from around $3.86, 2.74$, and 2.99 seconds to $1.41, 1.06$, and 1.38 seconds, respectively. We therefore get speedup factors of 2.73 for $\mathbf{S}(\mathbf{x})$, 2.57 for $\mathcal{Y}(\mathbf{v}, \mathbf{x})$, and 2.16 for $\mathcal{X}(\mathbf{w}, \mathbf{x})$ on the given machine, on which some resources such as caches are shared among workers. A summary of the runtimes can be found in Table 3.3.

	time (seconds)			speedup		
number of workers	$\mathbf{S}(\mathbf{x})$	$\mathcal{Y}(\mathbf{v},\mathbf{x})$	$\mathcal{X}(\mathbf{w},\mathbf{x})$	$\mathbf{S}$	$\mathcal{Y}(\mathbf{v},\mathbf{x})$	$\mathcal{X}(\mathbf{w},\mathbf{x})$
1	3.8595	2.7380	2.9933	1	1	1
2	2.7331	1.6775	1.8633	1.41	1.63	1.61
4	1.8164	1.0514	1.3548	2.12	2.60	2.21
8	1.4150	1.0658	1.3818	2.73	2.57	2.16

Table 3.3: Strong scaling tests for constructing $\mathbf{S}, \mathcal{Y}(\mathbf{v},\mathbf{x}), \mathcal{X}(\mathbf{w},\mathbf{x})$ using a coarsening of 16^3 fine cells per coarse cell, and with 72 sources and 3698 receivers. Computations are run on a Microway system that has four Intel Xeon E5-4627 CPUs with 40 cores and 1 TB of memory.

3.6 Discussion

We embed a multiscale finite volume (MSFV) methods into a PDE-constrained optimization framework and demonstrate its potential for solving high-dimensional parameter estimation problems. As usual in model order reduction (MOR) techniques, we reduce the computational costs associated with the PDE constraint by projecting the discrete PDEs onto a lower-dimensional subspace. Following the MSFV approach, we obtain a reduced version of the original fine-mesh PDE problem by projecting it onto a nested coarse mesh using an operator-dependent Galerkin projection. The key novelty of our method is the combination of MSFV and the numerical optimization scheme used for parameter estimation. Here, we exploit the fact that the multiscale basis is sensitive to the current PDE parameter and propose a reduced inverse problem featuring an adaptive projection that can be solved using derivative-based optimization. We outline the potential of our method using two inverse conductivity problems in 3D that are inspired by Direct Current Resistivity.

What sets our approach apart from existing works on MOR for PDE-constrained optimization is the choice of multiscale methods for obtaining the reduced problem. This choice is mainly motivated by two reasons. First, using multiscale methods as MOR techniques avoids the necessity of sampling the parameter space, which is problematic in high-dimensional spaces. Apart from optional boundary conditions, the method is fully online and does not require solving the fine-mesh problem. Sec-

ond, multiscale methods simplify the formulation and solution of adaptive inversion problems as the multiscale basis vary smoothly with respect to the PDE parameters. Similar to the recent work of [30], we show that computing derivatives is tractable by using local sensitivity computations; see Section 3.4.

Following a discretize-optimize strategy, we explicitly differentiate the solution of the discretized problem obtained with the MSFV solver with respect to the parameter to be estimated. Earlier works on differentiating multiscale solutions obtained approximate derivatives through interpolation [99], used adjoint-based approaches [48, 49], or required automatic differentiation [30]. Similar to other discretize-optimize approaches, our method produces an accurate gradient of the discrete objective problem irrespective of the mesh size and quality of the reduced order model. We also demonstrate that the involved local sensitivity computations can be performed in parallel providing additional opportunity for speedup.

Our numerical experiments show that the adaptive multiscale provides parameter estimates that feature details below the coarse-mesh resolution. As expected (see discussion in Section 3.4.2), our numerical experiments confirm that ignoring the dependence of $\mathbf{S}$ on the model parameter $\mathbf{m}$ also degrades reconstruction quality, especially when using aggressive coarsening. The computational complexity of the multiscale inversion grows linearly with respect to the number of coarse-mesh blocks and (if direct methods are used) cubic with respect to the number of fine-mesh cells in each coarse-mesh blocks. Therefore, additional computational benefits are expected for problems in which accurately solving the forward problem leads to linear systems that are too large to be handled by a direct solver. For finer mesh sizes, direct methods are currently infeasible and iterative methods such as the block CG method are being used. In our experiments, multiscale methods outperformed this iterative method (see Table 3.1 and 3.2).

We applied our method to a DCR survey, where the forward problem involves

solving the diffusion equation. We intend to further explore this method for other parameter estimation problems arising from electromagnetic, and gravity-based surveys which involve solving different PDEs [21, 155]. Multiscale Finite Volume methods provide more flexibility in building the reduced basis than used in this work. For example, additional basis functions representing sources and receivers can be added; see Section 3.3. Another approach to reduce the impact of boundary conditions on the multiscale basis is using oversampling [20, 39].

Chapter 4

Uncertainty-Weighted Asynchronous Optimization

The work in this chapter was done in collaboration with L. Ruthotto and is based on [50]. An additional numerical example on multinomial logistic regression is included in this chapter (see Section 4.3.2). In this chapter, we present our uncertainty-weighted asynchronous consensus ADMM method that aims to reduce the time-to-solution for large-scale parameter estimation. Our contribution is the introduction of uncertainty-based weights to the global variable consensus ADMM scheme. This chapter is organized as follows. We begin with background describing the typical communication and latency difficulties that common methods such as Gauss-Newton face in large-scale parameter estimation problems. We then present the uncertainty-weighted consensus ADMM scheme and show how to compute the uncertainty-based weights efficiently. We demonstrate the effectiveness of our scheme with a series of numerical experiments and conclude with a brief discussion.

4.1 Background

For problems in geophysics and machine learning, it is often the case that we have a sum of misfits, corresponding to, e.g., different training examples, or different sources. The MAP estimate computation can thus be formulated as

$$\mathbf{x}_{\text{MAP}} = \underset{\mathbf{x}}{\text{argmin}} \sum_{j=1}^{N} \Phi_j(\mathbf{x}) + \mathcal{R}(\mathbf{x}). \tag{4.1}$$

Standard Gauss-Newton algorithms can mainly be used to solve (4.1) for moderately sized parameter estimation problems [63,66,136], especially in cloud computing platforms, since they require a lot of memory and introduce severe communication overhead when parallelized. For instance, in the static-scheduling approach described in [136] for geophysical parameter estimation, the model and a number of meshes, sources, receivers, and forward problems are assigned to all the workers in the offline phase. Then, to evaluate the misfit, the gradient, and perform a Hessian matrix-vector product, each worker computes its corresponding batch of gradients and Hessian matrix-vector products, and communicates it to the main process. Consequently, every inner PCG iteration used to solve (2.16) requires sending the current iterate for the search direction to the workers and receiving the results from local matrix vector products. For large-scale problems this can result in a nontrivial amount of communication, especially when many PCG iterations are needed. Moreover, if the data is divided unevenly among the workers, the algorithm may lead to large latencies in each PCG iteration [136]. This motivates us to consider more scalable distributed algorithms, especially when the size and dimension of the problem is very large.

We consider the consensus alternating direction method of multipliers (ADMM) [16, 55, 83] as well as its asynchronous variant (async-ADMM) [162], which aims at reducing latencies and thereby reduce the time-to-solution. Consensus ADMM has previously been applied to high-dimensional inverse problems in data sciences [114,

119], statistical learning [16, 55, 148, 159], and imaging [54, 75, 88]. The algorithm tackles large-scale problems by partitioning the data into, say, N smaller batches that can be solved in parallel, and in some cases explicitly. This often leads to an improved ratio of local computation and communication. More specifically, each iteration of the algorithm breaks down into: 1) N subproblems using parts of the data that are solved locally and independently, 2) an averaging step that is performed once their corresponding processors have solved all N subproblems, and 3) an explicit update of the dual variable. The main change in the async-ADMM variant is that the averaging step is performed once $N_a < N$ subproblems have been solved, reducing the overall latency.

As we demonstrate in our numerical experiments, a straightforward implementation of consensus ADMM converges slowly in particular when the information contained in the split data sets is complementary and the number of batches, N, is large. One problem in these cases is that the averaging step in consensus ADMM gives equal weight to all the solutions corresponding to each batch, leading to an uninformed averaged reconstruction. In large-scale problems parameter estimation problems, this renders consensus ADMM prohibitive since often only a few iterations are affordable.

To increase the performance of consensus ADMM, particularly in early iterations, we introduce a novel weighting scheme that improves the convergence of consensus ADMM. The weights are obtained in a systematic and efficient way using the framework of uncertainty quantification (UQ) proposed in [46]. We demonstrate the effect of the weights on a collection of linear inverse problems. We also outline the potential of our method by comparing it to the Gauss-Newton method [60] and the nonlinear conjugate gradient (NLCG) method [67] on a single-physics PDE parameter estimation problem involving a travel time tomography survey, and a multiphysics parameter estimation problem involving Direct Current Resistivity (DCR) and travel time tomography [150] surveys.

4.2 Uncertainty-Weighted Consensus ADMM

In this section, we introduce our uncertainty-weighted ADMM method. First, we present the general formulation of the weighted ADMM, which involves rephrasing (4.1) as a global variable consensus problem [16], and review the asynchronous implementation presented in [162]. We propose a novel scheme for selecting the weights, which is based on approximate uncertainty information of the local subproblems that we obtain similarly to the framework in [46]. Finally, we use a numerical example to illustrate the intuition behind the weights.

4.2.1 Weighted Consensus ADMM

Motivated by the discussion in the previous section, we reformulate the optimization problem (2.7) as an equivalent weighted global variable consensus problem

$$\begin{aligned} \mathbf{x}_{\text{MAP}} = \underset{\mathbf{x}_1,\ldots,\mathbf{x}_N,\mathbf{z}}{\operatorname{argmin}} & \sum_{j=1}^{N} \left(\Phi_j(\mathbf{x}_j) + \mathcal{R}(\mathbf{x}_j)\right), \\ \text{s.t.} \quad & \mathbf{W}_j(\mathbf{x}_j - \mathbf{z}) = \mathbf{0}, \quad j = 1,\ldots,N, \end{aligned} \tag{4.2}$$

where in contrast to (4.1), the objective function is now separable and the coupling is enforced in the constraints. Here, $\mathbf{x}_j \in \mathbb{R}^n$ are the local variables that are brought into consensus via the global variable $\mathbf{z} \in \mathbb{R}^n$, and $\mathbf{W}_j \in \mathbb{R}^{n\times n}$ are nonsingular weight matrices. For ease of presentation and to obtain an efficient optimization scheme, this work uses diagonal weight matrices. In the standard global consensus formulation [16], the weight matrices are assigned as identity matrices. This reformulation allows each of the objective terms in (4.2) to be handled by its corresponding worker via the consensus ADMM algorithm.

Consider the augmented Lagrangian defined by

$$\mathcal{L}_\rho(\mathbf{x}_1,\dots,\mathbf{x}_N,\mathbf{u}_1,\dots,\mathbf{u}_N,\mathbf{z}) = \sum_{j=1}^{N}\left(\Phi_j(\mathbf{x}_j)+\mathcal{R}(\mathbf{x}_j)+\mathbf{u}_j^\top(\mathbf{W}_j(\mathbf{x}_j-\mathbf{z}))+\frac{\rho}{2}\|\mathbf{W}_j(\mathbf{x}_j-\mathbf{z})\|_2^2\right). \tag{4.3}$$

Consensus ADMM aims at solving problem (4.2) by finding a saddle point of $\mathcal{L}_\rho$ via the following iterations:

$$\begin{aligned}
\mathbf{x}_j^{(k+1)} &= \operatorname*{argmin}_{\mathbf{x}_j}\mathcal{L}_\rho(\mathbf{x}_1^{(k)},\dots,\mathbf{x}_{j-1}^{(k)},\mathbf{x}_j,\mathbf{x}_{j+1}^{(k)},\dots,\mathbf{x}_N^{(k)},\mathbf{u}_1^{(k)},\dots,\mathbf{u}_N^{(k)},\mathbf{z}^{(k)}) \\
&= \operatorname*{argmin}_{\mathbf{x}_j}\left(\Phi_j(\mathbf{x}_j)+\mathcal{R}(\mathbf{x}_j)+(\mathbf{u}_j^{(k)})^\top\mathbf{W}_j\mathbf{x}_j+\frac{\rho}{2}\|\mathbf{W}_j(\mathbf{x}_j-\mathbf{z}^{(k)})\|_2^2\right), \qquad (4.4)\\
& j=1,\dots,N,\\
\mathbf{z}^{(k+1)} &= \operatorname*{argmin}_{\mathbf{z}}\mathcal{L}_\rho(\mathbf{x}_1^{(k+1)},\dots,\mathbf{x}_N^{(k+1)},\mathbf{u}_1^{(k)},\dots,\mathbf{u}_N^{(k)},\mathbf{z}) \\
&= \left(\sum_{j=1}^{N}\mathbf{W}_j^\top\mathbf{W}_j\right)^{-1}\sum_{j=1}^{N}\left(\mathbf{W}_j^\top\mathbf{W}_j\mathbf{x}_j^{(k+1)}+(1/\rho)\mathbf{W}_j\mathbf{u}_j^{(k)}\right), \qquad (4.5)\\
\mathbf{u}_j^{(k+1)} &= \mathbf{u}_j^{(k)}+\rho\mathbf{W}_j(\mathbf{x}_j^{(k+1)}-\mathbf{z}^{(k+1)}), \quad j=1,\dots,N, \qquad (4.6)
\end{aligned}$$

where k denotes the current iteration, $\mathbf{u}_j\in\mathbb{R}^n$ are the dual variables, and $\rho>0$ is the penalty parameter associated with the augmented Lagrangian term. In the first two steps, we have simplified the augmented Lagrangian by dropping all terms that enter the subproblems are constants. We note that the last step is a dual ascent step.

Letting n_j be the number of forward models assigned to the j^{th} subproblem, the minimization steps in (4.4) require n_j PDE solves per function, gradient, and Hessian evaluations. Thus, they are the most computationally challenging part of the algorithm. However, they correspond to the local subproblems that are solved independently by each worker. Another advantage is that the local subproblem can be solved using any optimization algorithm, which provides an easy way to tailor

Algorithm 4.1: Consensus ADMM

- initialize $\mathbf{x}_j^{(0)}, \mathbf{z}^{(0)}$, and $\mathbf{u}_j^{(0)}$ for $j = 1, \ldots, N$
- for $k = 0, 1, 2, \ldots$ until (2.28) holds
  1. obtain $\mathbf{x}_j^{(k+1)}$ by solving local problems in (4.4) for $j = 1, \ldots, N$
  2. obtain $\mathbf{z}^{(k+1)}$ using the averaging step (4.5)
  3. obtain $\mathbf{u}_j^{(k+1)}$ through dual update (4.6) for $j = 1, \ldots, N$

the method to different subproblems, e.g., subproblems containing different PDEs for which highly-optimized algorithms already exist. Consequently, ADMM sits at a higher-level of abstraction from classical optimization algorithms such as those mentioned in Chapter 2. The global variable $\mathbf{z}$ attempts to bring the local variables $\mathbf{x}_j$ into consensus by averaging them in (4.5), and finally, the dual variables are updated via a gradient ascent step in (4.6). As for the stopping criteria, we follow the procedured outlined in Section 2.2.2.

Parallelization of consensus ADMM is much more straightforward than that of the Gauss-Newton-PCG described in Section 2.2.1. The amount of communication per outer iteration is reduced as we only communicate one set of models, $\mathbf{x}_1, \ldots, \mathbf{x}_N$ per outer ADMM iteration. In the synchronous parallel implementation, the master processor must wait for all the workers to finish solving their corresponding subproblems in (4.4) before performing the averaging step (4.5) per iteration, which may lead to high latencies when some of the workers are much slower than others. The async-ADMM method in [162] aims at reducing these latencies in star network topologies. Here, the global averaging step (4.5) is performed when $N_a < N$ workers report their results. A *bounded delay* condition is also enforced, where every worker has to report at least once every k_a iterations to ensure sufficient "freshness" of all updates. We note that here we have better control of the overall amount of communication and latency since we can administer how many forward problems to assign to any given

Algorithm 4.2: Consensus async-ADMM

- initialize $\mathbf{x}_j^{(0)}, \mathbf{z}^{(0)}$, and $\mathbf{u}_j^{(0)}$ for $j = 1, \ldots, N$
- initialize N_a and k_a
- while (2.28) not satisfied
  1. solve (4.4) locally
  2. perform averaging step (4.5) when N_a workers report their solutions
  3. update the corresponding N_a dual variables (4.6)

worker, and how accurately to solve each subproblem.

Convergence results have been established for the synchronous ADMM algorithm in the case where the local subproblems are convex. In this case, the algorithm converges regardless of the initial choice $\rho^{(0)}$ [38, 77]. Even when solving (4.4) inexactly, ADMM convergence can be shown [77, Section 4]. For the asynchronous case, convergence is ensured via the bounded delay condition. For non-convex subproblems, it has been shown that ADMM converges to a local minimum under some modest assumptions, most importantly requiring ρ to be sufficiently large, [78, 115, 157]. These assumptions ensure that the Hessian of the Lagrangian of (4.2) remains positive definite throughout the ADMM iterations.

4.2.2 Computing the Weights

We choose the weights to be approximately equal to the inverse of the diagonals of the posterior covariance $\Gamma_{j,\text{post}} \in \mathbb{R}^{n \times n}$ corresponding to the j^{th} objective term in (4.2). This is one way to assign higher weights to elements of $\mathbf{x}_j$ for which the j^{th} subproblem contains more information. It also reduces the impact of elements for which the data of the subproblem is uninformative. Clearly, there are other options to transform uncertainties into weights. Since we are mostly interested in encoding large differences in the uncertainties between subproblems, we do not compute the

uncertainties with high accuracy.

As seen in (2.8), construction of the posterior covariance may not be tractable, especially for large-scale PDE parameter estimation problems and when the forward model is nonlinear. As a result, we follow the works of [46] for approximating the posterior covariance of each objective term in a tractable way. This is done via a low-rank approximation of the approximate Hessian of the misfit Φ_j in the following manner:

1. We linearize the residual in Φ_j and obtain the Gauss-Newton approximation

$$\mathbf{H}_{j,\text{mis}} \approx \mathbf{J}_j^\top (\mathbf{\Gamma}_{j,\text{noise}}^{-1}) \mathbf{J}_j, \tag{4.7}$$

   where $\mathbf{J}_j \in \mathbb{R}^{m_j \times n}$ is the Jacobian matrix of $\mathcal{F}_j$ evaluated at some reference model parameter, e.g., $\mathbf{x}_{\text{ref}}$. We note that explicit construction of $\mathbf{H}_{j,\text{mis}}$ is not necessary as we only need the action of $\mathbf{J}_j$ and $\mathbf{J}_j^\top$ on a vector.

2. Denoting the prior-conditioned approximate Hessian by

$$\tilde{\mathbf{H}}_{j,\text{mis}} = \mathbf{\Gamma}_{\text{prior}}^{1/2} \mathbf{H}_{j,\text{mis}} \mathbf{\Gamma}_{\text{prior}}^{1/2},$$

   we rewrite the j^{th} posterior covariance in (2.8) as

$$\mathbf{\Gamma}_{j,\text{post}} = \mathbf{\Gamma}_{\text{prior}}^{1/2} \left(\tilde{\mathbf{H}}_{j,\text{mis}} + \mathbf{I} \right)^{-1} \mathbf{\Gamma}_{\text{prior}}^{1/2}. \tag{4.8}$$

3. We then construct a low-rank approximation of the prior-conditioned Hessian using, e.g., randomized SVD [138] or Lanczos bidiagonalization [57] to obtain

$$\tilde{\mathbf{H}}_{j,\text{mis}} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^\top \approx \mathbf{V}_r \mathbf{\Lambda}_r \mathbf{V}_r^\top, \tag{4.9}$$

   where $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^{n \times n}$ and $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_n] \in \mathbb{R}^{n \times n}$ denote

the matrix of eigenvalues and eigenvectors of $\tilde{\mathbf{H}}_{j,\text{mis}}$, respectively, and $\mathbf{\Lambda}_r = \text{diag}(\lambda_1, \ldots, \lambda_r) \in \mathbb{R}^{r\times r}$ and $\mathbf{V}_r = [\mathbf{v}_1, \ldots, \mathbf{v}_r] \in \mathbb{R}^{n\times r}$ are their corresponding truncations retaining only the r largest eigenvalues and eigenvectors.

4. We plug this approximation into (4.8) and use the Sherman-Morrison-Woodbury formula [141] to obtain an expression for the inverse term:

$$\left(\tilde{\mathbf{H}}_{j,\text{mis}} + \mathbf{I}\right)^{-1} \approx \mathbf{I} - \mathbf{V}_r\mathbf{D}_r\mathbf{V}_r^\top + \mathcal{O}\left(\sum_{i=r+1}^{n} \frac{\lambda_i}{\lambda_i + 1}\right), \tag{4.10}$$

where $\mathbf{D} \in \mathbb{R}^{r\times r} = \text{diag}(\lambda_1/(\lambda_1+1), \ldots, \lambda_r/(\lambda_r+1))$.

5. Finally, we obtain a manageable approximation of the posterior covariance that does not involve the inverse of the Hessian, but that instead involves the square root of the prior:

$$\mathbf{\Gamma}_{j,\text{post}} \approx \mathbf{\Gamma}_{\text{prior}}^{1/2}(\mathbf{I} - \mathbf{V}_r\mathbf{D}_r\mathbf{V}_r^\top)\mathbf{\Gamma}_{\text{prior}}^{1/2}. \tag{4.11}$$

We choose the weights to be the inverse of the diagonals of $\mathbf{\Gamma}_{j,\text{post}}$,

$$\begin{aligned}
\mathbf{W}_j &= \text{diag}(\mathbf{\Gamma}_{j,\text{post}})^{-1}, \\
&\approx \text{diag}\left(\mathbf{\Gamma}_{\text{prior}} - \mathbf{\Gamma}_{\text{prior}}^{1/2}\left(\mathbf{V}_r\mathbf{D}_r\mathbf{V}_r^\top\right)\mathbf{\Gamma}_{\text{prior}}^{1/2}\right)^{-1}, \\
&= \left[\text{diag}\left(\mathbf{\Gamma}_{\text{prior}}\right) - \text{diag}\left(\mathbf{\Gamma}_{\text{prior}}^{1/2}\left(\mathbf{V}_r\mathbf{D}_r\mathbf{V}_r^\top\right)\mathbf{\Gamma}_{\text{prior}}^{1/2}\right)\right]^{-1}, \quad j = 1, \ldots, N,
\end{aligned} \tag{4.12}$$

so that we get higher weights in parts of the model where we are more certain and vice-versa. We note that to compute the diagonals of $\mathbf{\Gamma}_{j,\text{post}}$, we need to be able to multiply by $\mathbf{\Gamma}_{\text{prior}}^{1/2}$ and compute the diagonals of $\mathbf{\Gamma}_{\text{prior}} = (\mathbf{L}^\top\mathbf{L})^{-1}$ efficiently. In our experiments, we are mainly concerned with the case when $\mathbf{L}^\top\mathbf{L}$ is either a diagonal, or a biharmonic operator. In the diagonal case, multiplying by $\mathbf{\Gamma}_{\text{prior}}^{1/2}$ and computing the diagonals of $\mathbf{\Gamma}_{\text{prior}}$ is trivial. In the case we use the biharmonic operator, we have access to the spectral decomposition of $\mathbf{L}^\top\mathbf{L}$ using Fourier transforms [73], which allows us

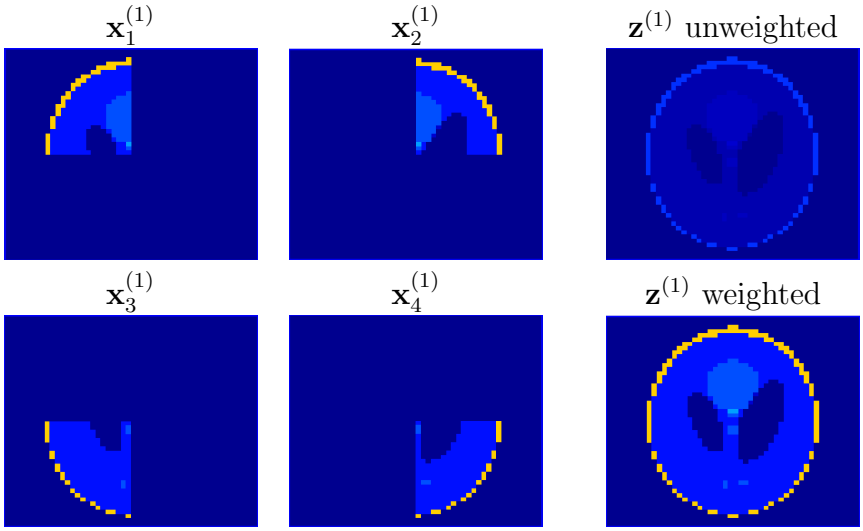


Figure 4.1: Averaging step of the weighted and unweighted consensus ADMM for Example 1. In this case, $\mathbf{W}_1$ assigns higher weights to the pixels in the upper-left quadrant of $\mathbf{x}_1$, $\mathbf{W}_2$ assigns higher weights to the upper-right quadrant of $\mathbf{x}_2$, etc. As a result, the weights educate the averaging step, leading to a better reconstruction of the image.

to efficiently multiply by $\mathbf{\Gamma}_{\text{prior}}^{1/2}$. We can also quickly estimate the diagonals of $\mathbf{\Gamma}_{\text{prior}}$ using probing methods [146], extrapolation methods [45], stochastic methods [10], and domain decomposition methods [106, 109, 145]. We may also update the weights throughout the ADMM scheme so that we instead employ local approximations of our posterior PDF [17]; however, convergence in this case is not guaranteed.

When the non-zero diagonal elements of $\mathbf{W}_j$ are equal to one, the weighted ADMM method corresponds to the standard unweighted ADMM scheme, which is known to converge slowly [16]. One reason is that the averaging step in (4.5) gives equal weight to all elements of $\mathbf{x}_j$ for all $j = 1, \ldots, N$, leading to poor reconstructions of $\mathbf{z}$, especially in the early iterations. We illustrate this in Example 1.

Example 1. *Consider solving the trivial linear system* $\mathbf{I}\mathbf{x} = \mathbf{d}$ *with the weighted and unweighted consensus ADMM with* $N = 4$ *splittings, where* $\mathbf{I} \in \mathbb{R}^{n\times n}$ *is the identity matrix, and* $\mathbf{x}, \mathbf{d} \in \mathbb{R}^n$ *are the model and the observed data, respectively. We formulate*

the least-squares problem as

$$\underset{\mathbf{x}_j,\mathbf{z}}{\operatorname{argmin}} \quad \sum_{j=1}^{4} \left(\frac{1}{2}\|\mathbf{I}_j\mathbf{x}_j - \mathbf{d}_j\|_2^2 + \frac{\alpha}{2}\|\mathbf{x}_j\|_2^2 \right) \tag{4.13}$$

$$\text{s.t.} \quad \mathbf{W}_j(\mathbf{x}_j - \mathbf{z}) = \mathbf{0}, \quad j = 1, \ldots, 4, \tag{4.14}$$

where $\alpha = 10^{-2}$*, and* $\mathbf{I}_j \in \mathbb{R}^{(n/4)\times n}$ *and* $\mathbf{d}_j \in \mathbb{R}^{n/4}$ *are subsets of the data obtained by partitioning the rows of* $\mathbf{I}$ *and* $\mathbf{d}$ *corresponding to the pixels in the top left, top right, bottom left, and bottom right quadrant of the domain as seen in Figure 4.1. We show the averaged reconstruction of both methods during the first iteration in Figure 4.1.*

We note that the forward model in Example 1 is separable and the weights can intuitively and easily be designed by hand. This example shows that we have a principled way to construct the weighted scheme that corresponds to manually choosing the weights in the cases that are as obvious as this example. In general, however, it is not always possible to manually design the weights, as the forward operators are not always separable.

4.3 Numerical Results

In this section, we outline the potential of the weighted scheme for consensus ADMM as well as its asynchronous variant on a series of linear and nonlinear inverse problems. We experiment on three classes of problems: linear least-squares, multinomial logistic regression, and 3D PDE parameter estimation problems. We first experiment on a deblurring and a tomography problem from `Regtools`, a MATLAB package containing discrete ill-posed inverse problems [72], as well as from a collection of linear least-squares problems from the UF Sparse Matrix Collection [29]. We then test our method on a multinomial logistic regression problem using the MNIST dataset [104]. Finally, we experiment on larger 3D PDE parameter estimation problems: a single-

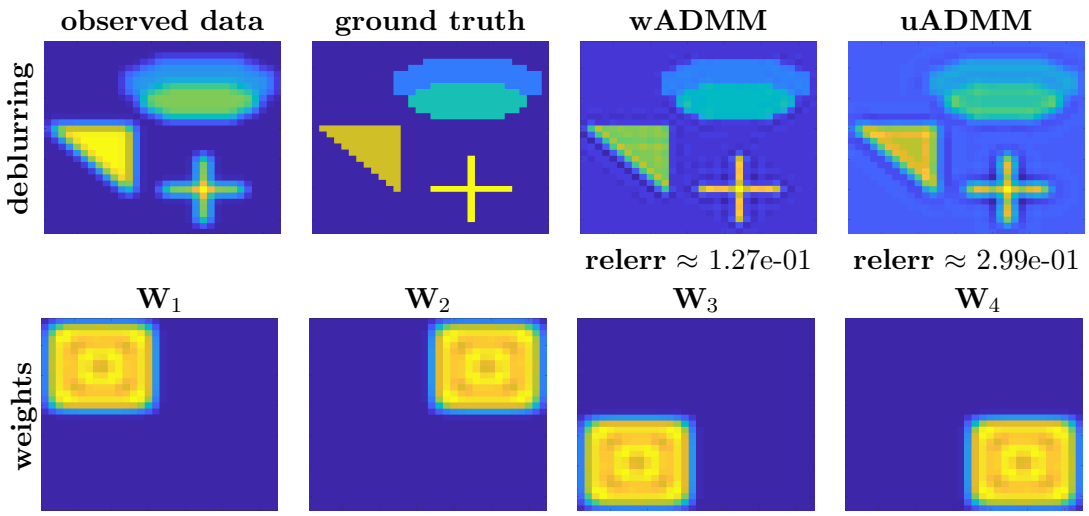


Figure 4.2: Observed data, ground truth, and reconstructions (first row) after 10 iterations and weights (second row) for the deblurring problem from `Regtools`.

physics parameter estimation problem involving a travel-time tomography survey, and a multiphysics parameter estimation problem involving DCR and travel-time tomography. We conclude this section with a comparison in the communication among the algorithms used for the multiphysics problem.

4.3.1 Least-Squares

We begin by comparing the weighted and unweighted consensus ADMM on a series of linear-least squares problems from `Regtools` [72] and the UF Library of Sparse Matrices [29]. For these problems, we use $N = 4$ splittings and solve

$$\begin{aligned} \underset{\mathbf{x}_j,\mathbf{z}}{\text{argmin}} \quad & \sum_{j=1}^{4} \left(\frac{1}{2}\|\mathbf{A}_j\mathbf{x}_j - \mathbf{d}_j\|_2^2 + \frac{\alpha}{2}\|\mathbf{x}_j\|_2^2 \right) \\ \text{s.t.} \quad & \mathbf{W}_j(\mathbf{x}_j - \mathbf{z}) = \mathbf{0}, \quad j = 1, \ldots, 4, \end{aligned} \tag{4.15}$$

where similar to Example 1, $\mathbf{A}_j \in \mathbb{R}^{(m/4)\times n}$ and $\mathbf{d}_j \in \mathbb{R}^{m/4}$, $j = 1, \ldots, 4$, are chosen by partitioning the rows of the original matrix and the data, $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{d} \in \mathbb{R}^m$, respectively. For the deblurring and tomography problems from `Regtools`, we use

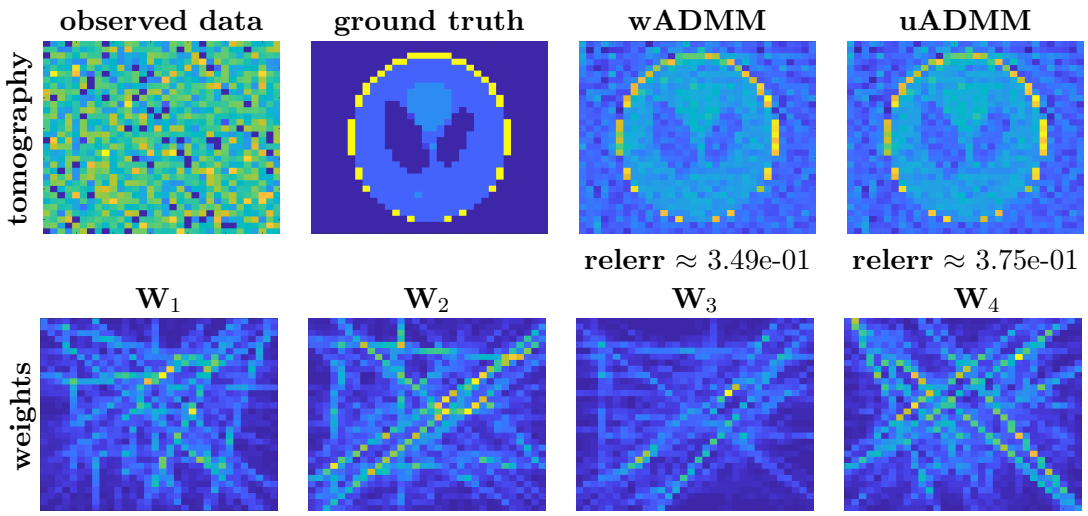


Figure 4.3: Observed data, ground truth, and reconstructions (first row) after 10 iterations and weights (second row) for the tomography problem from `Regtools`.

the same splittings as in Figure 4.1 where we split the rows corresponding to the different quadrants of the image. For the non-image based problems from the UF library, $\mathbf{A}_1$ and $\mathbf{d}_1$ correspond to the first $m/4$ rows of $\mathbf{A}$ and $\mathbf{d}$, respectively, $\mathbf{A}_2$ and $\mathbf{d}_2$ correspond to the second $m/4$ rows of $\mathbf{A}$ and $\mathbf{d}$, respectively, and so on. In the case that the number of rows, m, is not divisible by 4, we round accordingly.

We add a smallness regularization term with $\alpha = 10^{-2}$ since the splittings $\mathbf{A}_j$ in our experiments are underdetermined ($m/4 < n$), leading to rank-deficient coefficient matrices $\mathbf{A}_j^\top \mathbf{A}_j$ arising from the normal equations. We set the initial penalty parameter to be $\rho^{(0)} = 5$ and use the adaptive scheme described in (2.30). We run the unweighted and weighted consensus ADMM for ten iterations and show comparisons of the relative residuals and relative errors. To compute the weights, we follow the procedure in Section 4.2.2 and compute a rank-10 approximation of the Hessian of the misfits using MATLAB's `eigs` function.

We observe larger performance gains through the weighted ADMM in the deblurring problem as compared to the tomography problem. In the further, the weights are concentrated in different non-overlapping parts of the domain (see Figure 4.2), leading to more efficient averaging. For the tomography problem, however, the weights

UF Sparse Matrix Collection Results					
		unweighted ADMM		weighted ADMM	
matrix	**condition #**	**residual**	**rel. error**	**residual**	**rel. error**
bcspwr03	5.01e+02	4.47e-02	2.58e-01	1.97e-02	1.63e-01
bcsstk03	6.79e+06	6.67e-01	9.99e-01	5.82e-01	9.91e-01
bcsstk19	1.34e+11	2.81e-01	9.01e-01	6.30e-02	8.62e-01
bfwb782	1.81e+01	4.94e-02	1.04e-01	2.83e-01	1.38e-01
can_229	4.01e+17	5.13e-02	2.10e-01	1.81e-02	1.46e-01
cavity02	8.12e+04	6.07e-01	9.33e-01	2.46e-01	7.83e-01
cavity03	5.85e+05	5.69e-01	9.01e-01	1.90e-01	7.34e-01
ch5-5-b4	1.00e+00	1.21e-01	9.85e-01	5.01e-03	9.84e-01
dwt_307	2.35e+18	8.92e-02	1.82e-01	2.23e-02	9.20e-02
football	3.74e+02	5.90e-02	4.88e-01	2.44e-02	3.58e-01
fs_183_3	3.27e+13	7.33e-02	1.00e+00	2.32e-02	1.00e+00
G23	1.00e+04	2.27e-02	2.63e-01	1.88e-02	2.55e-01
GD98_c	9.87e+16	7.78e-02	3.70e-01	5.13e-02	2.62e-01
gre_115	4.97e+01	2.77e-01	4.70e-01	7.64e-02	2.67e-01
gre_343	1.12e+02	1.18e-01	1.77e-01	4.48e-02	6.90e-02
grid1_dual	3.35e+16	3.74e-02	3.67e-01	2.56e-02	3.20e-01
impcol_d	2.06e+03	3.50e-01	7.07e-01	9.74e-02	3.94e-01
jpwh_991	1.42e+02	1.89e-01	8.81e-01	1.61e-01	8.66e-01
lowThrust_1	Inf	4.40e-01	9.96e-01	2.84e-01	9.82e-01
lund_a	2.80e+06	1.06e-01	5.99e-01	4.05e-02	5.65e-01
nos3	3.77e+04	5.27e-01	9.88e-01	2.22e-01	9.67e-01
odepa400	2.26e+05	4.91e-01	9.99e-01	1.86e-01	9.97e-01
pde900	1.53e+02	5.98e-01	9.82e-01	2.90e-01	9.39e-01
poisson2D	1.33e+02	3.33e-01	7.44e-01	8.03e-02	6.64e-01
polbooks	7.20e+02	3.72e-02	2.97e-01	2.30e-02	2.51e-01
problem1	3.11e+16	4.08e-01	9.11e-01	1.43e-01	8.24e-01
rdb200l	1.33e+02	8.68e-02	1.44e-01	1.91e-02	1.05e-01
str˙0	2.74e+02	8.46e-01	9.37e-01	4.04e-01	7.02e-01
TF10	7.34e+02	2.63e-01	4.82e-01	5.21e-02	2.85e-01
young1c	4.15e+02	6.39e-01	9.40e-01	3.59e-01	6.97e-01

Table 4.1: Comparison of the accuracy obtained using the unweighted and weighted ADMM applied to least-squares problems from the UF sparse matrix collection [29].

look similar and contain a substantial amount of overlap, leading to averaged reconstructions that are similar to those of the unweighted ADMM (see Figure 4.3).

Finally, for the UF matrices, we randomly take 30 matrices with dimensions $100 \leq m, n \leq 1000$ from the library and compare both methods in Table 4.1 after ten iterations. We report their condition number, relative residuals, and relative errors. We obtain better results with the weighted ADMM after ten iterations. We refrain from solving these problems in parallel since they are small 2D problems and are mainly used as a proof-of-concept.

4.3.2 Multinomial Logistic Regression

Next, we experiment on a small multinomial logistic regression problem using the MNIST data set. We use 5000 images as training data and 1000 images as validation data. We split our data by class labels so that the first splitting consists of all images corresponding to the number zero, the second splittings consists of all images corresponding to the number one, and so on. This splitting allows us to avoid similar-looking weights in our splittings as seen in Figure 4.3. This results in 10 total splittings. In particular, we solve

$$\begin{aligned} \underset{\mathbf{X}_j,\mathbf{Z}}{\text{argmin}} \quad & \sum_{j=1}^{10} -\text{trace}(\mathbf{C}_j^\top \log(h_{\mathbf{X}_j}(\mathbf{D}_j))) + \|\mathbf{X}_j\|_F^2, \\ \text{s.t.} \quad & \mathbf{W}_j(\text{vec}(\mathbf{X}_j - \mathbf{Z})) = \mathbf{0}, \quad j = 1, \ldots, 10, \end{aligned} \tag{4.16}$$

where h is the softmax hypothesis function given in (2.11), and $\text{vec}(\mathbf{X})$ vectorizes a given matrix $\mathbf{X}$. Recall from Section 2.1.1 that $\mathbf{X}_j$ is a matrix with dimension $n_f \times n_c$ where n_f are the number of features and n_c are the number of classes. As a result, the weights $\mathbf{W}_j, j = 1, \ldots, 10$ are diagonal matrices with dimension $n_f n_c \times n_f n_c$. We compute the weights using a rank-20 approximation of the Hessian following Section 4.2.2.

For the MNIST data set, $n_f = 28^2$ and $n_c = 10$, where each image is of size 28×28. Thus, each $\mathbf{W}_j$ can be interpreted as containing $n_c = 10$ images along its diagonals, and we visualize them by plotting ten 28×28 images from each weight below.

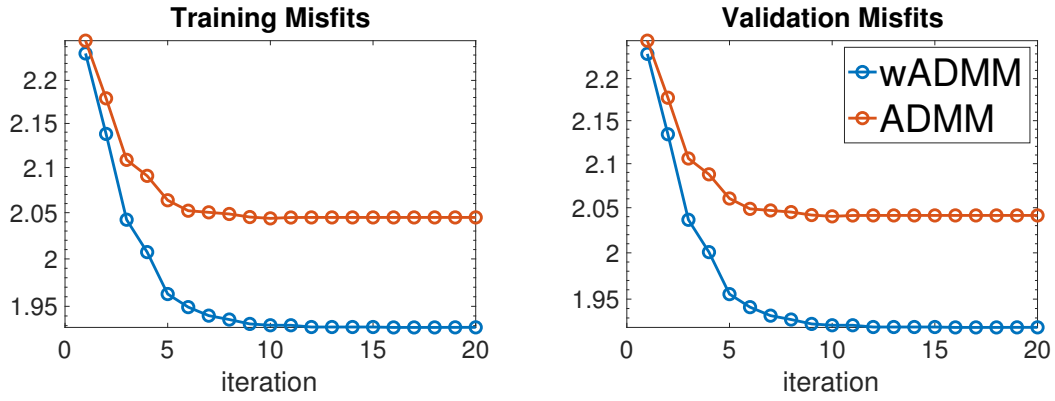


Figure 4.5: Training misfits and validation misfits for the weighted (wADMM) and unweighted ADMM (ADMM) for 20 iterations.

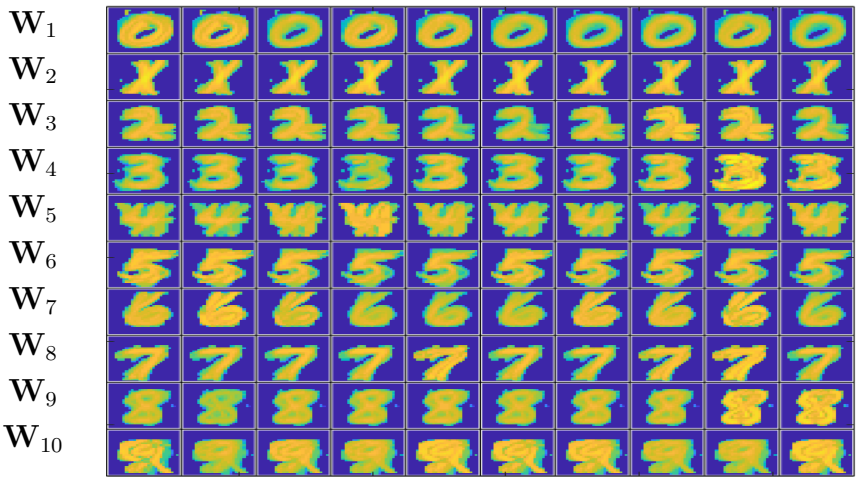


Figure 4.4: UQ-based weights for the MNIST dataset. The weights are plotted on a log-scale.

Each of the weights place more importance in the pixels that correspond to an averaged version of the images in each corresponding splitting. Since we split the data by classes, as we may intuitively expect, we obtain that $\mathbf{W}_1$ looks like averaged zeros, $\mathbf{W}_2$ looks like averaged ones, and so on. In Figure 4.5, we report the training and validation misfit histories after 20 iterations, and see the improved convergence due to the weights.

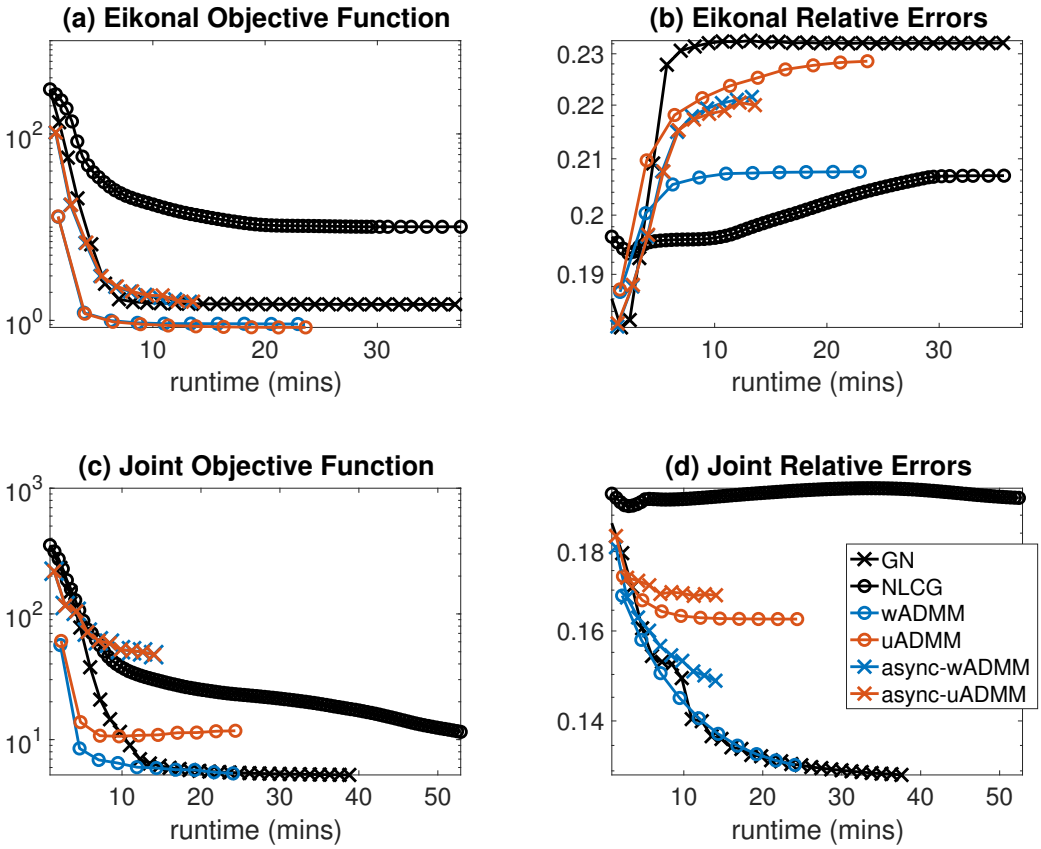


Figure 4.6: Objective function and relative errors for the Eikonal and joint inversions using 6 different algorithms: Gauss-Newton, NLCG, uADMM, wADMM, async-uADMM, and async-wADMM. Here, the x-axis represents runtime in minutes. The experiments were run on a shared memory computer operating Ubuntu 14.04 with 2 Intel Xeon E5-2670 v3 2.3 GHz CPUs using 12 cores each, and a total of 128 GB of RAM. Here, Julia is installed and compiled using Intel Math Kernel Library.

4.3.3 Single-physics Parameter Estimation

As more a more realistic test problem, we again consider the 3D SEG/EAGE model [3] also described in Section 3.5.2 as the ground truth (see Figure 4.7b) and test our method for a single-physics inversion involving the travel time tomography survey. The model contains a salt dome in which the velocity is significantly higher than in the background. The domain is of size 13.5 km $\times$ 13.5 km $\times$ 4.2 km and is divided into $64 \times 64 \times 32$ equally sized mesh cells of approximate size of 211m $\times$ 211m $\times$ 11m each. We implement our experiments in extension of `jInv` [136], an open-source package for PDE parameter estimation written in Julia [13]. For brevity, since the

travel time tomography problem is modeled by the Eikonal equation, we refer to it as the Eikonal problem for the remainder of the paper. We solve these problems in parallel and experiment on the effect of the asynchronous variant (async-ADMM) on the weighted and unweighted consensus ADMM.

The PDE involved in the forward problem is the Eikonal equation (2.32), and it is solved using the Factored Eikonal Fast Marching Algorithm [150]. We solve the inversion using 36 sources and 3600 receivers located on the top surface of the domain. We compare the weighted and unweighted ADMM (wADMM and uADMM), their asynchronous variants (async-wADMM and async-uADMM), Gauss-Newton (GN), and NLCG. For all 6 algorithms, we use a biharmonic regularization with regularization parameter $\alpha = 10^{-1}$ to enforce smoothness. We solve all inversions in parallel using 10 workers. Here, 6 workers solve forward problems containing 4 sources each, and the remaining 4 workers solve forward problems containing 3 sources each.

We run the Gauss-Newton inversion for a maximum of 30 outer iterations and use at most 10 PCG iterations with PCG stopping tolerance of 10^{-1} to solve the Gauss-Newton system. For the NLCG inversion, we set a maximum of 100 outer iterations since it is expected to take more iterations than Gauss-Newton to reach the same accuracy. In the ADMM inversions, we run a total of 10 outer iterations with 3 GN iterations used to solve the subproblems. This particular choice of inner GN and outer ADMM iterations aims to balance the runtime and computations performed with those of the Gauss-Newton inversion while avoiding solving the subproblems too inexactly, as this may lead to lack of convergence. In the ADMM subproblems, each GN iteration also uses at most 10 PCG iterations with PCG stopping tolerance of 10^{-1} as in the Gauss-Newton inversion.

For the penalty parameter, we use the scheme described in (2.30) to vary ρ and use a lower bound of 10^{-12}. As expected, the performance of ADMM depends crucially on the initial choice of ρ; therefore, we report the best results obtained from initial values

of $\rho^{(0)} \in [10^{-8}, 10^{2}]$. In our experiment, the optimal initial values are $\rho^{(0)} = 10^{-7}$ for uADMM and $\rho^{(0)} = 10^{-4}$ for wADMM. In the asynchronous case, we perform a global update whenever $N_a = 5$ workers report their solutions and enforce the bounded delay condition by requiring all workers to report results at least once every $k_a = 4$ iterations. To compute the weights, we follow the procedure described in Section 4.2.2, where we use the Lanczos bidiagonalization algorithm from `KrylovMethods` [134] to compute a rank-5 approximation of the approximate Hessians of the data misfits. To estimate the diagonal of the prior covariance, we perform 1000 iterations of the stochastic estimator proposed in [10] based on Hutchinson's technique for estimating the trace of a matrix. These iterations can be performed very efficiently as we have the spectral decomposition of the biharmonic operator. Again, we note that highly accurate uncertainties are not necessary in our case and a good guess is sufficient for our experiments. The computation of the weights took about 31 seconds.

In Figure 4.6(a-b), we show the relative errors and misfits for the Eikonal problem. Although the relative error does not improve during the iterations for any of the methods, we point out that the reconstructions visually become more similar to the ground truth; see Figure 4.7(c-h). The impact of communication and latency in the difference of runtimes between asynchronous ADMM variants, which ran for about 15 minutes, and the Gauss-Newton-PCG, which ran for about 38 minutes, is evident. For the NLCG, a total of 68 iterations were performed before a linesearch fail was reached. As expected, an iteration from the NLCG method is much quicker than an iteration from the remaining 5 methods since each NLCG iteration only requires explicit steps to update the model.

4.3.4 Multi-physics Parameter Estimation

We now add a second modality to Section 4.3.3, the DCR survey, which is modeled by the steady-state heterogeneous diffusion equation 2.31, and consider a multiphysics

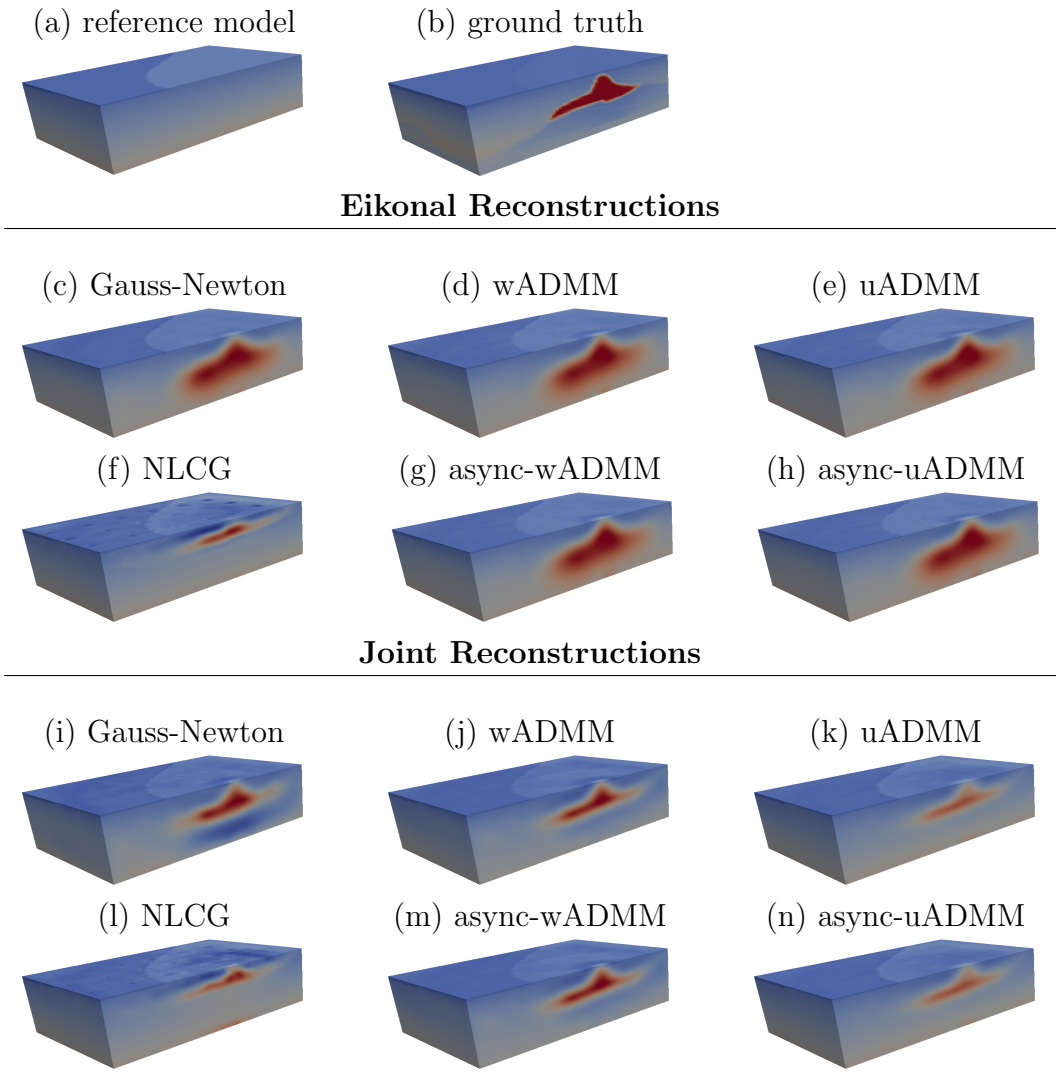


Figure 4.7: Reconstructions of SEG model with a single-physics and multiphysics experiment.

inversion. Here, we keep the same settings for the Eikonal problem and use 32 sources and 1682 receivers located on the top surface of the domain for the DCR survey. To solve the DCR forward problem, we use the finite volume method described in [60] to discretize the problem and solve the linear system using Julia's direct solver. For simplicity, we assume known petrophysics [140], which gives us an explicit relation between the ground conductivity σ and the wave velocity v given by

$$\sigma(v) = \left(2 - \frac{v}{c}\right)\left(\frac{b-a}{2}(\tanh(10(c-v+1)) + a\right). \tag{4.17}$$

Here, a and b are the conductivity values set to 0.1 and 1.0 respectively, and $c = 3.0$ is the velocity in which the contrast is centered. This setup was also used in [136].

As in Section 4.3.3, we compare six algorithms: wADMM, uADMM, async-wADMM, async-uADMM, Gauss-Newton-PCG, and NLCG. We solve all the inversions in parallel using ten workers. The PDE operator in the DCR experiment is small enough to be factorized in a single worker with a direct solver. In this case, it is not worth parallelizing the problem since communication takes longer than solving for all sources in one worker. For larger problems where the DCR problem must instead be solved iteratively, however, distributing the DCR sources among different workers will lead to faster time-to-solution. In contrast, the Eikonal problems are solved with a sequential fast marching scheme [150] and thus, we distribute the problems among the remaining nine workers. The nine workers in charge of the Eikonal problem solve forward problems containing 4 sources each. The inversion settings are also the same as in Section 4.3.3 except for the choice of initial penalty parameter, where we find the optimal initial values to be $\rho^{(0)} = 10^{-3}$ for uADMM and $\rho^{(0)} = 1.0$ for wADMM. We also follow the same procedure as in Section 4.3.3 to compute the weights for this setup, which took about 54 seconds.

We show the results for the relative errors and objective function values vs. runtime for the joint inversions in Figure 4.6 and the reconstructions in Figure 4.7. We see the weighted schemes improve the convergence and reconstruction quality of its unweighted counterparts. In fact, we obtain similar reconstruction accuracy between the Gauss-Newton scheme and the synchronous weighted ADMM (see Figure 4.6(d)); we highlight the considerable progress made by the ADMM method in the first few iterations as this is not very common for small problems that can be held in the memory of a few machines. We also obtain faster convergence using the asynchronous ADMM variants, with the weighted asynchronous ADMM leading to similar quality of reconstruction as can be seen in Figure 4.7. As expected, the joint inversions enhance

the quality of the reconstruction since the different physics involved capture different properties of the model [136].

4.3.5 Communication Costs

We use the multiphysics example to exemplify the differences in terms of communication costs for the Gauss-Newton, NLCG, and ADMM method. As discussed in Section 4.3.4, we assign worker 1 the DCR problem containing all of its 32 forward models, and assign the remaining nine workers $(2-10)$ four Eikonal forward models each. In the comparison below, the vectors communicated between the workers and the master process are of size $131,072 \times 1$.

Gauss-Newton. We use 30 GN iterations each of which involves up to 10 PCG iterations per GN iteration. In each GN iteration, the master process sends the current model to all ten workers. Worker 1 then returns the accumulated gradient vector corresponding to the 32 DCR forward problems, and workers 2-10 each return the accumulated gradient vector for the four local sources. This allows for the computation of the full gradient shown in (2.11).

Moreover, in each PCG iteration, the master process sends one vector to all workers. The workers then return the accumulated matrix-vector product of this vector with the approximated Hessians associated with the local subproblems (DCR for worker 1 and Eikonal for workers 2–10). This leads to a total of 20 vectors communicated between the master and the workers per PCG iteration. Overall, the Gauss-Newton method consists of

$$10 \text{ workers} \times (2 \text{ gradient vectors} + 20 \text{ PCG vectors}) \times 30 \text{ GN iterations} = 6,600$$

vectors communicated between the workers and the master process.

NLCG. We run NLCG for a maximum of 100 iterations. The communication pat-

tern is the same as in the Gauss-Newton method except that no inner PCG iterations are performed. Thus, the NLCG inversion consists of

$$10 \text{ workers} \times 2 \text{ gradient vectors} \times 100 \text{ NLCG iterations} = 2,000$$

vectors communicated between the workers and the master process.

sync-ADMM. We run a total of 10 outer ADMM iterations. In each ADMM iteration, the master process sends the current global variable $\mathbf{z}^{(k)}$ to all workers. Each worker then returns its corresponding local variable $\mathbf{x}_j^{(k+1)}$ to the main process. This leads to a total of two vectors communicated between each worker and the master process per ADMM iteration. The sync-ADMM inversion thus consists of

$$10 \text{ workers} \times 2 \text{ vectors} \times 10 \text{ ADMM iterations} = 200$$

total vectors communicated between the workers and the master process.

async-ADMM. As in sync-ADMM, we run a total of 10 ADMM iterations. However, we update the global variable whenever 4 workers report their solution. As a result, the async-ADMM inversion consists of

$$4 \text{ workers} \times 2 \text{ vectors} \times 10 \text{ ADMM iterations} = 80$$

total vectors communicated between the workers and the master process.

The communication comparison described above confirms that ADMM dramatically reduces the amount of communication in the inversion. This is also seen in the reduced runtimes in Figure 4.6. A similar comparison can be made for the single-physics parameter estimation problem in Section 4.3.3.

4.4 Discussion

We propose a weighted asynchronous consensus ADMM (async-wADMM) method for solving large-scale PDE parameter estimation problems in parallel. To this end, the data involved in the problem is divided among the available workers. Our scheme is geared toward applications such as PDE parameter estimation where only a few iterations can be afforded. Our proposed weighting scheme improves the convergence of the standard ADMM. Since our weights are informed by an approximate uncertainty quantification for the subproblems in (4.4), we formulate the parameter estimation problem in a Bayesian setting. It is important to note that our scheme can also be applied in the frequentist setting as long as weights are available. To obtain an overall efficient scheme, we follow the works of [46] to quantify the uncertainties in a tractable manner.

As test problems, we solve a collection of linear least-squares problems and a multinomial logistic regression problem for proof-of-concept as well as a more realistic single-physics involving the travel time tomography survey, and a multiphysics parameter estimation problem involving the DCR and travel time tomography survey. Our numerical results show that our method accelerates the convergence of consensus ADMM, particularly in the early iterations. The quality of the parameter reconstructions obtained by the weighted async-ADMM scheme are comparable to those of the Gauss-Newton-PCG method, however, the weighted async-ADMM method requires substantially less communication among workers and has smaller latencies, resulting in reduced inversion runtimes and less communication. Moreover, since we can choose any optimization scheme to solve the subproblems in async-ADMM, the method sits at a higher level of abstraction and provides additional flexibility. Each subproblem can, therefore, be solved with a tailored solver, making the weighted async-ADMM especially attractive for large-scale multiphysics PDE parameter estimation problems. For brevity, we do not show the case where the weights are computed in every itera-

tion; however, in this case, we obtain indiscernible reconstructions from those shown in Figure 4.7. We intend to further explore our method for the case where the weights are correlated (non-diagonal), and for large-scale problems where the Gauss-Newton-PCG method cannot be used as well as on computational environments with small communication bandwidth such as cloud computing platforms.

Chapter 5

Classification with Multinomial Logistic Regression

The work in this chapter is based on [52], and was done in collaboration with L. Ruthotto, S. Tyrväinen, and E. Haber. In this chapter, we present an efficient learning algorithm for solving large-scale classification problems. We consider a reformulation of the MLR problem into a constrained optimization problem whose objective function is separable along the examples and whose coupling is enforced by the constraints - we note that this is not the consensus formulation previously discussed in Chapter 4. We then apply ADMM to decompose the constrained problem into three subproblems that can be solved efficiently. We name this scheme ADMM-Softmax. In particular, each iteration of our scheme consists of 1) a weight update that involves solving a least-squares problem, 2) a global variable update that involves a cross-entropy problem that is separable along the examples, and 3) a trivial dual update variable. The least-squares problem arising from the weights can be efficiently solved using direct or iterative solvers [56, 137]. The convex and smooth softmax problem arising from the global variable update can also be solved efficiently since the separability along examples renders it highly parallelizable. Finally, the dual variable update is

a trivial step requiring only a matrix-vector product. As we will see in Section 5.3, the reformulation of the MLR problem in this case is different from that of the global variable consensus formulation previously described in Chapter 4.

5.1 Related Work

Multinomial regression is a central problem in many tasks, and has led to many numerical methods tailored to specific data structures. One class of methods is based on deriving separable upper bounds on the objective function. For example, in [15,58], an upper bound based on the first-order concavity property of the log-function is used. This approach gives a new optimization problem that is not convex, but where the objective is separable across the weights associated with the different classes. There are other possible upper-bounds for a softmax function, such as a quadratic upper bound and a product of sigmoids. Detailed comparison of these and analytical solutions in a Bayesian setting can be found in [15]. In [58], Gopal and Yang use the concavity bound to solve multinomial logistic regression in parallel, and prove that their iterative optimization of the bounded objective converges to the same optimal solution as the unbounded original model. Related to the concavity bound in [43] and [132] where the convex conjugate of negative log to reformulate the problem as a double-sum that can be solved iteratively with SGD.

Another group of methods are based on a splitting idea. An ADMM approach on the MLR problem can also be found in [58], where the problem is reformulated as a constrained optimization problem where linear and nonlinear terms of the objective function are solved separately. That is, the new global auxiliary parameter of ADMM is implanted only on the challenging log-sum term. The new optimization problem can be solved in parallel, but the authors state that this approach does not compare in computational efficiency to the principal method presented in their paper. The

approach is inspired by Boyd in [16] who solves the sparse logistic regression problem in parallel by splitting it across features with ADMM.

An approach for solving the non-convex problem of training neural networks using ADMM and Bregman iteration can be found in [147], where examples concentrate on binomial regression which allows them to use quadratic loss function and closed form solutions for each iteration steps. Another related approach to train neural networks is the method of auxiliary coordinates (MAC) [19]. In MAC new variables are introduced to decouple the problem. Unlike ADMM, however, MAC breaks the deep nesting, i.e. function compositions, in the objective function with the new parameters.

Sparse logistic regression has been proposed as a method for feature selection in large-scale classification problems as sparsity can help identify the most important features, which avoids over-fitting and can reduce the computing time. The sparsity is generally forced with ℓ_1-regularization, but the norm is non-differentiable, leading to difficulties in training. There are different approaches to solve this, e.g., interior point-methods [95], iterative shrinkage methods [71] and hybrid algorithms [142]. See, for example, [96] or the survey [161].

5.2 Mathematical Formulation

We now revisit the mathematical formulation for multinomial logistic regression (MLR) described in Section 2.1.1 in more detail. Recall that in this setting, we are given the feature matrix $\mathbf{D} = [\mathbf{d}_1\, \mathbf{d}_2 \ldots \mathbf{d}_{n_e}]^\top \in \mathbb{R}^{n_e \times n_f}$, the label (or class) matrix $\mathbf{C} = [\mathbf{c}_1\, \mathbf{c}_2 \ldots \mathbf{c}_{n_e}]^\top \in \mathbb{R}^{n_e \times n_c}$, and the softmax classifier, $h_{\mathbf{X}}(\mathbf{D})$, which can be found in (2.11). Here, n_e, n_f, and n_c are the number of examples, features, and class labels, respectively. The softmax function normalizes all values between zero and one so that all rows sum to one - this guarantees that each row of the predicted matrix

$\mathbf{C}_{\text{pred}}$ resembles a probability distribution, and the weight matrix $\mathbf{X} \in \mathbb{R}^{n_f \times n_c}$ is the parameter of interest.

To learn the weights $\mathbf{X}$, we use the cross-entropy loss function given in (2.10). In order to make the coupling of the weights along examples evident, we revisit and rewrite the cross-entropy function in (2.10) more explicitly as

$$\begin{aligned}
&\Phi_{\text{CE}}(\mathbf{X}) = -\text{trace}\left(\mathbf{C}^\top \log\left(\text{h}_{\mathbf{X}}(\mathbf{D})\right)\right) \\
&= -\text{trace}\left(\mathbf{C}^\top \log\left(\text{diag}\left(\frac{1}{\exp(\mathbf{D}\mathbf{X})\mathbf{e}_{\text{n}_\text{c}}}\right)\exp(\mathbf{D}\mathbf{X})\right)\right) \\
&= \sum_{j=1}^{n_e}\Big[-(\mathbf{c}_j \odot (\mathbf{X}^\top \mathbf{d}_j))^\top \mathbf{e}_{n_c} + \log(\exp(\mathbf{X}^\top \mathbf{d}_j)^\top \mathbf{e}_{n_c})\Big]
\end{aligned} \tag{5.1}$$

where $\mathbf{e}_{n_c} \in \mathbb{R}^{n_c}$, is a vector of all ones. The cross-entropy loss function quantifies the between the true probabilities $\mathbf{C}$ and the predicted probabilities $h_{\mathbf{X}}(\mathbf{D})$. Here $\odot$ denotes the Hadamard product, i.e., element-wise product. Learning the weights is thus equivalent to solving

$$\mathbf{x}_{\text{MAP}} = \underset{\mathbf{X}}{\text{argmin}} \ \Big(\Phi_{\text{CE}}(\mathbf{X}) + \mathcal{R}(\mathbf{X})\Big), \tag{5.2}$$

where $\mathcal{R}\colon \mathbb{R}^{n_f \times n_c} \mapsto \mathbb{R}$ is a regularization operator that helps avoid overfitting. From the term in (5.1), we can see that the weights are coupled for each example. This lack of separability leads to difficulties in the parallelization of classical optimization algorithms used to minimize the entropy loss function. The problem can thus be very challenging computationally for large amounts of high-dimensional data.

5.3 ADMM-Softmax

To circumvent the lack of separability described above, we derive a novel ADMM algorithm to solve (5.2). Our main idea, is to split the multinomial regression problem

into a regularized least-squares problem and a separable smooth convex optimization problem, noting that efficient solvers exist for both subproblems. For ease of presentation, we use ℓ_2-regularization since it allows us to have a closed-form least-squares solution when updating the weights, however, our method can easily be extended to other regularizers. In particular, we relax (5.2) by introducing a global auxiliary variable $\mathbf{Z} \in \mathbb{R}^{n_e \times n_c}$ and reformulate it as

$$\begin{aligned} \underset{\mathbf{X},\mathbf{Z}}{\operatorname{argmin}} \;\; & \Phi\left(\mathbf{C}, \mathbf{Z}\right) + \frac{\alpha}{2} \left\| \mathbf{L}\left(\mathbf{X} - \mathbf{X}_{\text{ref}}\right) \right\|_F^2 \\ \text{s.t.} \;\; & \mathbf{Z} - \mathbf{D}\mathbf{X} = \mathbf{0}. \end{aligned} \tag{5.3}$$

The second term is a Tikhonov regularizer [121,123,149] where $\mathbf{L}$ is the regularization operator.

To solve (5.3) using the ADMM algorithm we first consider the Lagrangian

$$\begin{aligned} \mathcal{L}_\rho\left(\mathbf{X}, \mathbf{Z}, \mathbf{Y}\right) = & \Phi\left(\mathbf{C}, \mathbf{Z}\right) + \mathbf{e}_{n_e}^\top \left(\mathbf{Y} \odot \left(\mathbf{Z} - \mathbf{D}\mathbf{X}\right)\right) \mathbf{e}_{n_c} \\ & + \frac{\rho}{2} \left\| \mathbf{Z} - \mathbf{D}\mathbf{X} \right\|_F^2 + \frac{\alpha}{2} \left\| \mathbf{L}\left(\mathbf{X} - \mathbf{X}_{\text{ref}}\right) \right\|_F^2, \end{aligned} \tag{5.4}$$

where $\mathbf{Y} \in \mathbb{R}^{n_e \times n_c}$ is the estimate of the Lagrange multiplier, $\rho > 0$ is the penalty parameter, and $\mathbf{e}_{n_e} \in \mathbb{R}^{n_e}, \mathbf{e}_{n_c} \in \mathbb{R}^{n_c}$ are vectors of all ones. The ADMM algorithm aims to find the saddle point of the Lagrangian via the following iterations:

$$\begin{aligned} \mathbf{X}^{(k+1)} &= \underset{\mathbf{X}}{\operatorname{argmin}} \;\; \mathcal{L}_\rho\left(\mathbf{X}, \mathbf{Z}^{(k)}\right), \\ \mathbf{Z}^{(k+1)} &= \underset{\mathbf{Z}}{\operatorname{argmin}} \;\; \mathcal{L}_\rho\left(\mathbf{X}^{(k+1)}, \mathbf{Z}\right), \\ \mathbf{Y}^{(k+1)} &= \mathbf{Y}^{(k)} + \rho\left(\mathbf{Z}^{(k+1)} - \mathbf{D}\mathbf{X}^{(k+1)}\right). \end{aligned} \tag{5.5}$$

These iterations can be written more explicitly using the scaled ADMM algorithm

(see [16]) as

$$\mathbf{X}^{(k+1)} = \underset{\mathbf{X}}{\operatorname{argmin}} \ \frac{\rho}{2}\left\|\mathbf{Z}^{(k)} - \mathbf{D}\mathbf{X} + \mathbf{U}^{(k)}\right\|_F^2 + \frac{\alpha}{2}\left\|\mathbf{L}\left(\mathbf{X} - \mathbf{X}_{\text{ref}}\right)\right\|_F^2 \tag{5.6}$$

$$\mathbf{Z}^{(k+1)} = \underset{\mathbf{Z}}{\operatorname{argmin}} \ \Phi\left(\mathbf{C}, \mathbf{Z}\right) + \frac{\rho}{2}\left\|\mathbf{Z} - \mathbf{D}\mathbf{X}^{(k+1)} + \mathbf{U}^{(k)}\right\|_F^2 \tag{5.7}$$

$$\mathbf{U}^{(k+1)} = \mathbf{U}^{(k)} + \left(\mathbf{Z}^{(k+1)} - \mathbf{D}\mathbf{X}^{(k+1)}\right), \tag{5.8}$$

where $\mathbf{U} = (1/\rho)\mathbf{Y}$ is the scaled Lagrange multiplier. The subproblem (5.6) involves a regularized least-squares, and a global variable step (5.7), which is convex and separable along the examples, leading to trivial parallelization (see Section 5.3.2).

We note that using a different regularization in the original optimization problem (5.2) would only impact the least-squares subproblem (5.6) and the global variable minimization step (5.7) would stay the same. There are many efficient ways to solve least-square problem with different types regularizations terms. A good example would be using ℓ_1-regularization, which renders (5.6) a lasso problem. In this case, there are many established solvers that efficiently solve the lasso problem (see Section 5.1) that can be used in our framework. As for the stopping criteria, we follow the procedured outlined in Section 2.2.2.

5.3.1 Solving the Least-Squares

Updating the weight matrix $\mathbf{X}$ requires solving (5.6). This is equivalent to solving normal equations

$$\left(\rho\mathbf{D}^\top\mathbf{D} + \alpha\mathbf{L}^\top\mathbf{L}\right)\mathbf{X} = \rho\mathbf{D}^\top\left(\mathbf{Z} + \mathbf{U}\right) + \alpha\mathbf{L}^\top\mathbf{L}\mathbf{X}_{\text{ref}}. \tag{5.9}$$

Note that the coefficient matrix is not iteration-dependent. Thus, depending on the number of features, the matrix can be factorized once and quickly applied.

In the case that coefficient matrix cannot be explicitly constructed, we can use iter-

ative methods, e.g., CGLS on the normal equations [56] with the incomplete Cholesky factorization [56] as a preconditioner. Another option is to use sample average approximation [93] to reduce the dimension of the problem and pre-compute a factorization in the offline phase, e.g. thin QR or Cholesky [56], leading to trivial solves throughout the optimization scheme. We note that these are only some approaches for solving (5.9), and that a myriad of additional options exist [56, 137].

5.3.2 Global Variable Update

The global variable minimization step (5.7) can be written as

$$\begin{aligned}\Psi(\mathbf{Z}) &= -\mathbf{e}_{n_e}^\top (\mathbf{C} \odot \mathbf{Z})\, \mathbf{e}_{n_c} + \mathbf{e}_{n_e}^\top \log(\exp(\mathbf{Z})\, \mathbf{e}_{n_c}) + \frac{\rho}{2} \|\mathbf{Z} - \mathbf{Z}_{\text{ref}}\|_F^2, \\ &= \sum_{i=1}^{n_e} \left(-\mathbf{c}_i^\top \mathbf{z}_i + \log(\exp(\mathbf{z}_i)\mathbf{e}_{n_c}) + \frac{\rho}{2} \|\mathbf{z}_i - \mathbf{z}_{\text{ref},i}\|_2^2 \right),\end{aligned} \tag{5.10}$$

where $\mathbf{Z}_{\text{ref}} = \mathbf{DX} - \mathbf{U}$, and the vectors $\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{n_e}\}$ correspond to the rows of the matrix $\mathbf{Z}$. For brevity, we maintain the matrix notation.

We can then write the gradient as

$$\begin{aligned}\nabla_{\mathbf{Z}} \Psi(\mathbf{Z}) = &- \mathbf{C} + \exp(\mathbf{Z}) \odot \left(\left(\frac{1}{\exp(\mathbf{Z})\, \mathbf{e}_{n_c}} \right) \mathbf{e}_{n_c}^\top \right) \\ &+ \rho (\mathbf{Z} - \mathbf{Z}_{\text{ref}}),\end{aligned} \tag{5.11}$$

and the product of the Hessian times a matrix $\mathbf{S} \in \mathbb{R}^{n_e \times n_c}$ as

$$\begin{aligned}\nabla_{\mathbf{Z}}^2 \Psi(\mathbf{Z})\, \mathbf{S} = &\left(\frac{\exp(\mathbf{Z})}{\exp(\mathbf{Z})\, \mathbf{e}_{n_c} \mathbf{e}_{n_c}^\top} \right) \odot \mathbf{S} \\ &- \left(\frac{\exp(\mathbf{Z})}{(\exp(\mathbf{Z})\, \mathbf{e}_{n_c})^2\, \mathbf{e}_{n_c}^\top} \right) \odot \left((\exp(\mathbf{Z}) \odot \mathbf{S})\, \mathbf{e}_{n_c} \mathbf{e}_{n_c}^\top \right) + \rho \mathbf{S},\end{aligned} \tag{5.12}$$

where the squaring in the denominator of the second term is applied componentwise. Unlike in (5.1), the objective function in (5.10) is separable along the examples,

allowing for easy parallelization. In particular, the gradients and Hessians can be computed locally and independently for different examples, leading to reduced run-times. Finally, ADMM-Softmax provides us with additional flexibility since we can solve (5.7) using any classical gradient-based optimization algorithm.

5.3.3 Computational Costs and Convergence

For the ADMM-Softmax formulation, the global variable update (5.7) can be solved using the Newton-CG method [157] (as we do in our experiments), and the Hessian matrix-matrix product used in (5.12) only requires performing a Hadamard product with $\exp(\mathbf{Z})$, which is in the order of $\mathcal{O}(n_e n_c)$. Assuming n_p workers are available, the cost for each Hessian matrix-matrix product seen in (5.12) is in the order of about $\mathcal{O}\left(\frac{n_e}{n_p} n_c\right)$ per worker, leading to very fast computations of the global variable update. The main computational bottleneck in ADMM-Softmax thus lies in solving the least-squares problem (5.6), for which there are ample options for solving efficiently [56]. When a factorization of the coefficient matrix can be performed in the off-line phase, e.g., Cholesky or thin QR, we can trivially solve the least-squares throughout the optimization.

Finally, it has been shown that the ADMM algorithm converges linearly for convex problems with existing solution regardless of the initial choice $\rho^{(0)}$ [38]. If (5.6) is solved inexactly, ADMM still converges under additional assumptions [77]. In our case, the cross-entropy loss function with the softmax classifier is convex, and thus we are guaranteed convergence with ADMM-Softmax.

5.4 Numerical Experiments

In this section, we demonstrate the potential of our proposed ADMM-Softmax on the MNIST and CIFAR-10 datasets. For both datasets, we compare the performance of

the proposed ADMM-Softmax, SGD, and Newton-PCG. Our experiments are coded in MATLAB using the Meganet deep learning package [158], and our ADMM-Softmax framework is coded as an extension of the package. We perform the experiments in this chapter using the same machine previously described in Chapters 3 and 4: a shared memory computer operating Ubuntu 14.04 with 2 Intel Xeon E5-2670 v3 2.3 GHz CPUs using 12 cores each, and a total of 128 GB of RAM.

5.4.1 Setup

MNIST

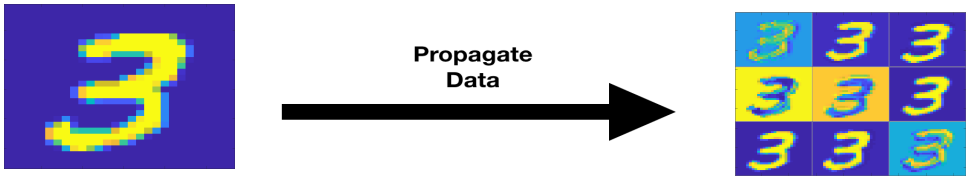
The MNIST database consists of 60,000 grey-scale hand-written images of digits ranging from 0 to 9. [103, 104] Here, we set 50,000 examples for training our digit-recognition system and the remaining 10,000 as validation data. Each digit is normalized with size 28×28, or with 784 pixel in total as the features.

Since the data is not linearly separable, we seek a hyperplane in an enlarged subspace obtained by a non-linear transformation to the original variables. Specifically, we propagate the data through a single fixed hidden layer where we apply a 3×3 random convolution filter with 9 channels:

$$\mathbf{D}_{\text{prop}} = \tanh(\mathbf{DK}), \tag{5.13}$$

where $\mathbf{K} \in \mathbb{R}^{n_f \times m}$ is a block-circulant matrix with circulant block (BCCB) convolution matrix [73] and m denotes the size of the new feature space. The transformed feature matrix has dimensions $50,000 \times 7057$, where each row now consists of 9 images. An illustration of the feature transformation is shown in Figure 5.1.

We compare three algorithms, our proposed ADMM-Softmax, SGD, and Newton-PCG. In SGD, we use Nesterov momentum with minibatch size 30, and learning rate $l_r = l_{r_0}/\sqrt{c_e}$, where c_e is the current epoch, $l_{r_0} = 10^{-2}$ is the initial learning rate.



original image propagated images

Figure 5.1: Feature transformation of a single image in the original feature matrix.

Here, we choose the initial learning rate and minibatch sizes by performing a grid-search on $[10^{-8}, 10^{3}]$ and $[1, 100]$, respectively. The initial learning rate grid-search is done logarithmically whereas the minibatch size grid-search is performed uniformly. In Newton-PCG, we set a maximum number of 20 inner CG iterations per Newton iteration, with CG tolerance of 10^{-2}. In the ADMM-Softmax, we use initial $\rho = 10^{-2}$, with absolute and relative tolerance described in (2.29) to be 10^{-3}. To solve the LS system, we compute a Cholesky factorization in the off-line phase, which for this experiment took about 0.3 seconds. To solve (5.7), we use the Newton-CG method from the Meganet package using a maximum 30 iterations and a gradient norm stopping tolerance of 10^{-1}. The inner Newton system is solved using the conjugate gradients (CG) with a maximum of 10 inner iterations and stopping tolerance of 10^{-1}.

For all three methods, we use the Laplace operator as the regularization operator $\mathbf{L}$ to enforce smoothness of the images, and set reference weights to $\mathbf{X}_{\text{ref}} = \mathbf{0}$. We set regularization parameters to $\alpha = 10^{-1}$ for SGD, $\alpha = 10^{-3}$ and $\alpha = 1$ for ADMM-Softmax, which were chosen based on the best results from performing a logarithmic grid-search on $[10^{-8}, 10^{3}]$.

CIFAR-10

The CIFAR-10 dataset [97] consists of 60,000 32×32 RGB-valued images in 10 classes. Here, we have 50,000 examples of data for training and 10,000 for validation. The images belong to one of the following 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

For this dataset, we increase the feature space by propagating our feature matrix through a pre-trained AlexNet [98] on the ImageNet dataset [33] from MATLAB's deep neural networks toolbox. In particular, we remove the last fully-connected layer and treat the rest AlexNet as a fixed feature extractor for the new dataset. In this case, the propagated feature matrix has dimensions $50,000 \times 4096$.

As for the optimization, we maintain the same setup as the MNIST dataset. We perform the same grid-search on the learning rate and minibatch sizes as in MNIST and report the parameters that led to the best results. In SGD, we obtain the best results with learning rate as $l_{r_0} = 10$ and minibatch size 40. In ADMM-Softmax, we choose the penalty parameter as $\rho = 10^{-1}$. Since the propagated features no longer correspond to images in this case, we use the identity as the regularization operator with $\alpha = 10^{-1}$ for SGD, $\alpha = 10^{-5}$ for Newton-PCG, and $\alpha = 1$. As before, we chose the α and ρ that led to the best results for each algorithm after performing a grid-search on $[10^{-8}, 10^{3}]$ for each algorithm, respectively.

5.4.2 Results

In Figure 5.2, we show the performance of both algorithms applied to MNIST and CIFAR-10. To make a fair comparison among all the algorithms, we compare the performance based on the runtime of each algorithm - this is because an iteration of ADMM-Softmax, SGD, and Newton-PCG require different amounts of computation. We let both algorithms run for a maximum of 500 seconds, and as can be seen, we obtain faster convergence with ADMM-Softmax. We note that we did not use any parallelization in any of these experiments, however, further speed ups are to be expected when the global variable step (5.7) is performed in parallel.

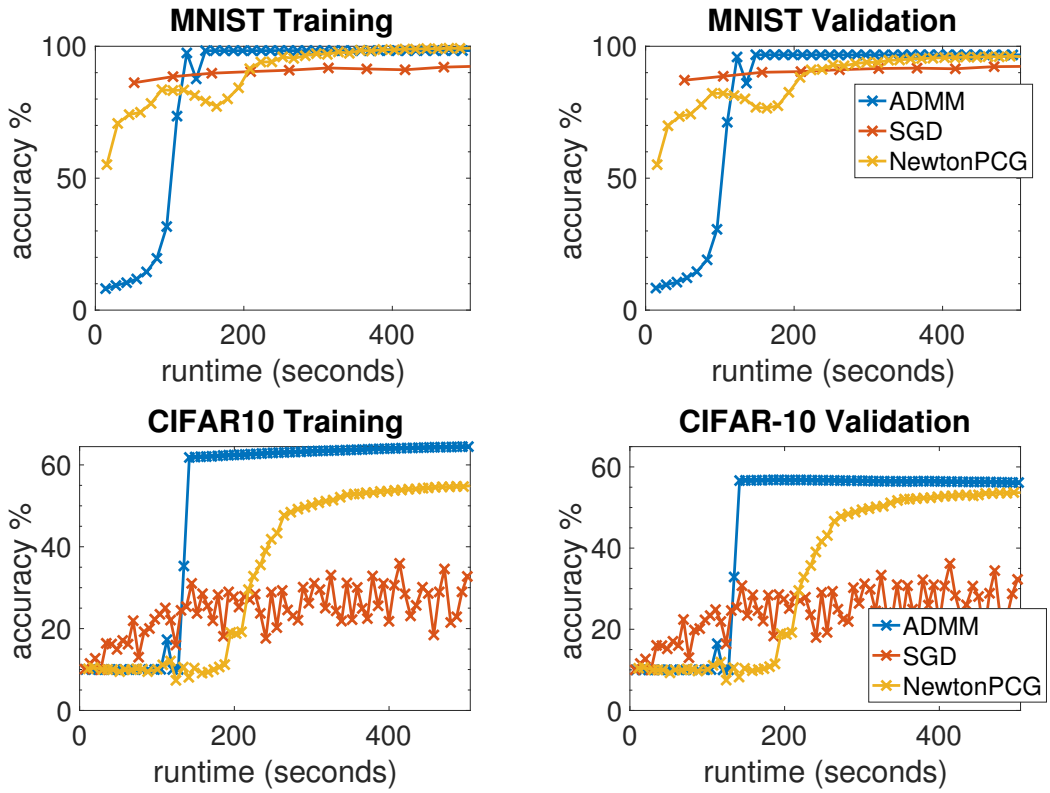


Figure 5.2: Training and validation accuracies for the MNIST and CIFAR-10 datasets.

5.5 Discussion

We propose a simple and efficient algorithm for solving large-scale image classification problems. To this end, we reformulate the traditional softmax regression problem consisting of an unconstrained coupled optimization into a constrained one where the objective function is decoupled and the coupling is enforced by the constraints. The new formulation is solved by the alternating direction method of multipliers (ADMM) which breaks down the problem into three simpler steps consisting of a least-squares (LS), a separable softmax problem, and a trivial dual variable update per outer iteration. ADMM-Softmax allows for plenty of flexibility since the resulting separable softmax problem can be solved using any classical optimization algorithm efficiently and in parallel (see Section 5.3).

Our numerical results show improved convergence when compared to SGD and

Newton-PCG for the MNIST and CIFAR-10 datasets. We refrain from solving these problems in parallel since the datasets were relatively small; however, since our proposed ADMM-Softmax contains a highly parallelizable step, further benefits are to be expected for large datasets where parallelization is necessary.

We note that better accuracies, especially for the CIFAR-10 dataset, could be achieved if we fine-tune the parameters of pre-trained AlexNet [160]. To this end, our method can accelerate block-coordinate algorithms that alternate between updating the network weights and the classifier. This is a direction of our future work.

Chapter 6

Summary and Outlook

The task of estimating parameters in the modern technological era, where large amounts of data is easily attainable, has become one of great computational challenge. One area of active research that attempts to tackle this challenge arises on the hardware side, e.g., supercomputing, quantum computing, etc. Access to these resources is unfortunately very limited, and quantum computing in particular is in its very early stages. The development of efficient optimization algorithms such as those presented in this thesis have therefore played an integral role in the transformational progress made in large-scale parameter estimation problems over the past few decades.

In this thesis, we presented three methods to reduce the computational burdens of estimating parameters in geophysics and machine learning applications. These methods fall under one of two general approaches: model order reduction, which aims to reduce the computational complexity of the model, and parallel/distributed optimization, which aims to lower the time-to-solution in parameter estimation. In Chapter 2, we presented the mathematical background relevant for this dissertation. The general formulation of MAP estimation and uncertainty quantification of parameter estimation problems as well as some numerical optimization techniques for

their computation was reviewed. We concluded this chapter with an overview of the applications used throughout our experiments.

In Chapter 3, we motivated the use of model reduction techniques with an overview of the discretized DCR forward and inverse problem. We then presented an adaptive multiscale model reduction technique based on MSFV, along with a tractable way for computing the projection bases and their derivatives. We compared our model reduction scheme using adaptive and fixed bases for the DCR survey.

Beginning in Chapter 4, we shifted the focus to parallel and distributed methods. We presented a weighted, asynchronous ADMM method to reduce the time-to-solution for these large-scale inverse problems. The weights are computed efficiently using iterative schemes, and are based on the uncertainty of the models. We tested this scheme on several linear least-squares problems, a multinomial logistic problem, and a single-physics parameter estimation problem and multi-physics problem consisting of travel-time tomography and DCR. We observed accelerated convergence for consensus ADMM when the uncertainty-based weights were included.

In Chapter 5, we presented another ADMM-based scheme (ADMM-Softmax) tailored to large-scale classification problems. This scheme is specific to the multinomial logistic regression problem, where we decoupled the objective function, and enforced the coupling as constraints. Applying ADMM to this reformulation allowed for easier parallelization of the problem. We tested our method on the MNIST and CIFAR-10 data set, and compared its performance to that of SGD and Newton-PCG.

The work in this thesis paves way for future work in parameter estimation, especially in the rapidly evolving and expanding field of machine learning. For instance, the adaptive MOR scheme could be used for different parameter estimation problems arising from electromagnetic, and gravity-based surveys which involve solving different PDEs [21, 155], as well as for PDE-based neural networks [135], where evaluating the forward model (or forward-propagation) is equivalent to solving a time-dependent

PDE. Multiscale finite volume methods provide more flexibility in building the reduced basis than used in this work. For example, additional basis functions representing sources and receivers can be added; see Section 3.3. Another extension to this work is to reduce the impact of boundary conditions on the multiscale basis is using oversampling [20, 39].

For the UQ-weighted ADMM, future work includes exploring our method for the case where the weights are correlated (non-diagonal). We also intend to explore the effectiveness of our method for large-scale problems where the Gauss-Newton-PCG method cannot be used as well as on computational environments with small communication bandwidth such as cloud computing platforms. Finally, to improve the accuracies for ADMM-Softmax, a natural extension involves fine-tuning the parameters of the pre-trained AlexNet [160]. To this end, ADMM-Softmax can accelerate block-coordinate algorithms that alternate between updating the network weights and the classifier.

Bibliography

[1] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[2] P. R. Amestoy, I. S. Duff, J.-Y. LExcellent, and J. Koster. Mumps: a general purpose distributed memory sparse solver. In *International Workshop on Applied Parallel Computing*, pages 121–130. Springer, 2000.

[3] F. Aminzadeh, B. Jean, and T. Kunz. *3-D salt and overthrust models*. Society of Exploration Geophysicists, 1997.

[4] D. Amsallem and C. Farhat. An online method for interpolating linear parametric reduced-order models. *SIAM Journal on Scientific Computing*, 33(5):2169–2198, 2011.

[5] S. R. Arridge. Optical tomography in medical imaging. *Inverse Problems*, 15(2):R41–R93, 1999.

[6] S. R. Arridge and J. C. Schotland. Optical tomography: forward and inverse problems. *Inverse Problems*, 25(12):123010–60, 2009.

[7] U. Ayachit. The paraview guide: a parallel visualization application. 2015.

[8] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera. An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667–672, 2004.

[9] U. Baur, C. Beattie, P. Benner, and S. Gugercin. Interpolatory Projection Methods for Parameterized Model Reduction. *SIAM Journal on Scientific Computing*, 33(5):2489–2518, 2011.

[10] C. Bekas, E. Kokiopoulou, and Y. Saad. An estimator for the diagonal of a matrix. *Applied numerical mathematics*, 57(11-12):1214–1229, 2007.

[11] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252, 1962.

[12] P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015.

[13] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[14] A. Borzi and V. Schulz. *Computational optimization of systems governed by partial differential equations*. SIAM, Philadelphia, 2011.

[15] G. Bouchard. Efficient bounds for the softmax function, applications to inference in hybrid models. In *NIPS*, Whistler, Canada, Dec 2007.

[16] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[17] T. Bui-Thanh, O. Ghattas, J. Martin, and G. Stadler. A computational framework for infinite-dimensional Bayesian inverse problems part i: The linearized case, with application to global seismic inversion. *SIAM Journal on Scientific Computing*, 35(6):A2494–A2523, 2013.

[18] T. Bui-Thanh, K. Willcox, and O. Ghattas. Model Reduction for Large-Scale Systems with High-Dimensional Parametric Input Space. *SIAM Journal on Scientific Computing*, 30(6):3270–3288, 2008.

[19] M. Á. Carreira-Perpiñán and W. Wang. Distributed optimization of deeply nested systems. In S. Kaski and J. Corander, editors, *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 10–19. PMLR, Reykjavik, Iceland, Apr 2014.

[20] L. Caudillo Mata, E. Haber, and C. Schwarzbach. An oversampling technique for multiscale finite volume method to simulate frequency-domain electromagnetic responses. In *SEG Technical Program Expanded Abstracts 2016*, pages 981–985. Society of Exploration Geophysicists, 2016.

[21] L. A. Caudillo-Mata, E. Haber, L. J. Heagy, and C. Schwarzbach. A framework for the upscaling of the electrical conductivity in the quasi-static maxwells equations. *Journal of Computational and Applied Mathematics*, 317:388–402, 2017.

[22] M. Cheney, D. Isaacson, and J. C. Newell. Electrical impedance tomography. *SIAM Review*, 41(1):85–101, 1999.

[23] E. T. Chung, Q. Du, and J. Zou. Convergence analysis of a finite volume method for maxwell's equations in nonhomogeneous media. *SIAM Journal on Numerical Analysis*, 41(1):37–63, 2003.

[24] E. T. Chung, Y. Efendiev, and R. L. Gibson Jr. An energy-conserving discontinuous multiscale finite element method for the wave equation in heterogeneous media. *Advances in Adaptive Data Analysis*, 3(01n02):251–268, 2011.

[25] E. T. Chung, Y. Efendiev, and C. S. Lee. Mixed generalized multiscale finite element methods and applications. *Multiscale Modeling & Simulation*, 13(1):338–366, 2015.

[26] E. T. Chung, Y. Efendiev, G. Li, and M. Vasilyeva. Generalized multiscale finite element methods for problems in perforated heterogeneous domains. *Applicable Analysis*, 95(10):2254–2279, 2016.

[27] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[28] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

[29] T. A. Davis and Y. Hu. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1, 2011.

[30] R. J. de Moraes, J. R. Rodrigues, H. Hajibeygi, and J. D. Jansen. Multiscale gradient computation for flow in heterogeneous porous media. *Journal of Computational Physics*, 336:644–663, 2017.

[31] E. De Sturler, S. Gugercin, M. E. Kilmer, S. Chaturantabut, C. Beattie, and M. O'Connell. Nonlinear parametric inversion using interpolatory model reduction. *SIAM Journal on Scientific Computing*, 37(3):B495–B517, 2015.

[32] O. S. Dean, A. C. Reynolds, and N. Liu. *Inverse Theory for Petroleum Reservoir Characterization and History Matching*. Cambridge University Press, Cambridge, 2008.

[33] J. Deng, W. Dong, R. Socher, L. jia Li, K. Li, and L. Fei-fei. Imagenet: A large-scale hierarchical image database. In *In CVPR*, 2009.

[34] A. Dey and H. Morrison. Resistivity modeling for arbitrarily shaped three dimensional structures. *Geophysics*, 44(4):753–780, 1979.

[35] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 647–655, Bejing, China, 22–24 Jun 2014. PMLR.

[36] J. J. Dongarra, J. Du Croz, S. Hammarling, and I. S. Duff. A set of level 3 basic linear algebra subprograms. *ACM Transactions on Mathematical Software (TOMS)*, 16(1):1–17, 1990.

[37] L. Durlofsky, Y. Efendiev, and V. Ginting. An adaptive local–global multiscale finite volume element method for two-phase flow simulations. *Advances in Water Resources*, 30(3):576–588, 2007.

[38] J. Eckstein and D. P. Bertsekas. On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3):293–318, 1992.

[39] Y. Efendiev and T. Y. Hou. *Multiscale finite element methods: theory and applications*, volume 4. Springer Science & Business Media, New York, 2009.

[40] Y. R. Efendiev, T. Y. Hou, and X.-H. Wu. Convergence of a nonconforming multiscale finite element method. *SIAM Journal on Numerical Analysis*, 37(3):888–910, 2000.

[41] H. C. Elman and Q. Liao. Reduced Basis Collocation Methods for Partial Differential Equations with Random Coefficients. *SIAM/ASA Journal on Uncertainty Quantification*, 1(1):192–217, 2013.

[42] I. Epanomeritakis, V. Akcelik, O. Ghattas, and J. Bielak. A newton-cg method for large-scale three-dimensional elastic full-waveform seismic inversion. *Inverse Problems*, 24(3):034015, 2008.

[43] F. Fagan and G. Iyengar. Unbiased scalable softmax optimization. volume abs/1803.08577, 2018.

[44] L. Feng and P. Benner. A robust algorithm for parametric model order reduction. *PAMM*, 7(1):1021501–1021502, 2007.

[45] P. Fika, M. Mitrouli, and P. Roupa. Estimating the diagonal of matrix functions. *Mathematical Methods in the Applied Sciences*, 41(3):1083–1088, 2018.

[46] H. P. Flath, L. C. Wilcox, V. Akçelik, J. Hill, B. van Bloemen Waanders, and O. Ghattas. Fast algorithms for Bayesian uncertainty quantification in large-scale linear inverse problems based on low-rank partial hessian approximations. *SIAM Journal on Scientific Computing*, 33(1):407–432, 2011.

[47] J. Friedman, T. Hastie, and R. Tibshirani. *The Elements of Statistical Learning*, volume 1. Springer.

[48] J. Fu, J. Caers, and H. A. Tchelepi. A multiscale method for subsurface inverse modeling: Single-phase transient flow. *Advances in water resources*, 34(8):967–979, 2011.

[49] J. Fu, H. A. Tchelepi, and J. Caers. A multiscale adjoint method to compute sensitivity coefficients for flow in heterogeneous porous media. *Advances in water resources*, 33(6):698–709, 2010.

[50] S. W. Fung and L. Ruthotto. An uncertainty-weighted asynchronous admm method for parallel pde parameter estimation. *arXiv preprint arXiv:1806.00192*, 2018.

[51] S. W. Fung and L. Ruthotto. A multiscale method for model order reduction in pde parameter estimation. *Journal of Computational and Applied Mathematics*, 350:19–34, 2019.

[52] S. W. Fung, S. Tyrväinen, L. Ruthotto, and E. Haber. Large-scale classification using multinomial regression and admm. *arXiv preprint arXiv:1901.09450*, 2019.

[53] R. Gamkrelidze. *Principles of optimal control theory*, volume 7. Springer Science & Business Media, New York, 2013.

[54] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk. Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences*, 7(3):1588–1623, 2014.

[55] T. Goldstein, G. Taylor, K. Barabin, and K. Sayre. Unwrapping ADMM: efficient distributed computing via transpose reduction. In *Artificial Intelligence and Statistics*, pages 1151–1158, 2016.

[56] G. H. Golub and C. F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[57] G. H. Golub and C. F. Van Loan. *Matrix Computations*, volume 3. JHU Press, Baltimore, 2012.

[58] S. Gopal and Y. Yang. Distributed training of large-scale logistic models. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 289–297, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[59] B. Haasdonk and M. Ohlberger. Reduced basis method for finite volume approximations of parametrized linear evolution equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 42(02):277–302, 2008.

[60] E. Haber. *Computational methods in geophysical electromagnetics*. SIAM, Philadelphia, 2014.

[61] E. Haber. Solving csem problems with massive number of sources and receivers. In *78th EAGE Conference and Exhibition 2016*, 2016.

[62] E. Haber, M. Chung, and F. Herrmann. An effective method for parameter estimation with PDE constraints with multiple right-hand sides. *SIAM Journal on Optimization*, 22(3):739–757, 2012.

[63] E. Haber, M. Chung, and F. Herrmann. An effective method for parameter estimation with pde constraints with multiple right-hand sides. *SIAM Journal on Optimization*, 22(3):739–757, 2012.

[64] E. Haber, E. Holtham, J. Granek, D. Marchant, D. Oldenburg, C. Schwarzbach, and R. Shekhtman. An adaptive mesh method for electromagnetic inverse problems. In *SEG Technical Program Expanded Abstracts 2012*, pages 1–6. Society of Exploration Geophysicists, 2012.

[65] E. Haber and L. Ruthotto. A multiscale finite volume method for maxwell's equations at low frequencies. *Geophysical Journal International*, 199(2):1268–1277, 2014.

[66] E. Haber and C. Schwarzbach. Parallel inversion of large-scale airborne time-domain electromagnetic data with multiple octree meshes. *Inverse Problems*, 30(5):055011, 2014.

[67] W. W. Hager and H. Zhang. A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM Journal on optimization*, 16(1):170–192, 2005.

[68] W. W. Hager and H. Zhang. A survey of nonlinear conjugate gradient methods. *Pacific journal of Optimization*, 2(1):35–58, 2006.

[69] H. Hajibeygi and P. Jenny. Adaptive iterative multiscale finite volume method. *Journal of Computational Physics*, 230(3):628–643, 2011.

[70] H. Hajibeygi, H. A. Tchelepi, et al. Compositional multiscale finite-volume formulation. *SPE Journal*, 19(02):316–326, 2014.

[71] E. T. Hale, W. Yin, and Y. Zhang. Fixed-point continuation for ℓ_1-minimization: Methodology and convergence. *SIAM J. Optim*, 19:1107–1130, Jun 2008.

[72] P. C. Hansen. Regularization tools: A matlab package for analysis and solution of discrete ill-posed problems. *Numerical algorithms*, 6(1):1–35, 1994.

[73] P. C. Hansen, J. G. Nagy, and D. P. O'leary. *Deblurring Images: Matrices, Spectra, and Filtering*, volume 3. Siam, 2006.

[74] B. He, H. Yang, and S. Wang. Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. *Journal of Optimization Theory and applications*, 106(2):337–356, 2000.

[75] J. Heredia-Juesas, A. Molaei, L. Tirado, W. Blackwell, and J. Á. Martínez-Lorenzo. Norm-1 regularized consensus-based ADMM for imaging with a compressive antenna. *IEEE Antennas and Wireless Propagation Letters*, 16:2362–2365, 2017.

[76] M. R. Hestenes. Multiplier and gradient methods. *Journal of optimization theory and applications*, 4(5):303–320, 1969.

[77] M. Hong and Z.-Q. Luo. On the linear convergence of the alternating direction method of multipliers. *Mathematical Programming*, 162(1-2):165–199, 2017.

[78] M. Hong, Z.-Q. Luo, and M. Razaviyayn. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM Journal on Optimization*, 26(1):337–364, 2016.

[79] L. Horesh and E. Haber. A Second Order Discretization of Maxwell's Equations in the Quasi-Static Regime on OcTree Grids. *SIAM Journal on Scientific Computing*, 33(5):2805–2822, 2011.

[80] T. Hou, X.-H. Wu, and Z. Cai. Convergence of a multiscale finite element method for elliptic problems with rapidly oscillating coefficients. *Mathematics of Computation of the American Mathematical Society*, 68(227):913–943, 1999.

[81] T. Y. Hou and X. H. Wu. A multiscale finite element method for elliptic problems in composite materials and porous media. *Journal of Computational Physics*, 134(1):169–189, 1997.

[82] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1):489–501, 2006. Neural Networks.

[83] K. Huang and N. D. Sidiropoulos. Consensus-ADMM for general quadratically constrained quadratic programming. *IEEE Transactions on Signal Processing*, 64(20):5297–5310, 2016.

[84] P. Jenny, S. Lee, and H. A. Tchelepi. Multi-scale finite-volume method for elliptic problems in subsurface flow simulation. *Journal of Computational Physics*, 187(1):47–67, 2003.

[85] P. Jenny, S. H. Lee, and H. A. Tchelepi. Adaptive multiscale finite-volume method for multiphase flow and transport in porous media. *Multiscale Modeling & Simulation*, 3(1):50–64, 2005.

[86] P. Jenny and I. Lunati. Modeling complex wells with the multi-scale finite-volume method. *Journal of Computational Physics*, 228(3):687–702, 2009.

[87] L. Jiang, Y. Efendiev, and V. Ginting. Multiscale methods for parabolic equations with continuum spatial scales. *Discrete and Continuous Dynamical Systems Series B*, 8(4):833, 2007.

[88] J. H. Juesas, G. Allan, A. Molaei, L. Tirado, W. Blackwell, and J. A. M. Lorenzo. Consensus-based imaging using ADMM for a compressive reflector antenna. In *Antennas and Propagation & USNC/URSI National Radio Science Meeting, 2015 IEEE International Symposium on*, pages 1304–1305. IEEE, 2015.

[89] J. Kaipio and E. Somersalo. *Statistical and Computational Inverse Problems*, volume 160 of *Applied Mathematical Sciences*. Springer Science & Business Media, New York, 2006.

[90] D. Z. Kalchev, C. S. Lee, U. Villa, and Y. Efendiev. Upscaling of mixed finite element discretization problems by the spectral AMGe method. *SIAM Journal on Scientific Computing*, 38(5):A2912–A2933, 2016.

[91] M. P. Kaleta, R. G. Hanea, A. W. Heemink, and J.-D. Jansen. Model-reduced gradient-based history matching. *Computational Geosciences*, 15(1):135–153, 2010.

[92] C. T. Kelley. *Iterative Methods for Optimization*, volume 18. SIAM, 1999.

[93] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

[94] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 2(12):479–502, 2002.

[95] K. Koh, S.-J. Kim, and S. Boyd. An interior-point method for large-scale l1-regularized logistic regression. *J. Mach. Learn. Res.*, 8:1519–1555, Dec. 2007.

[96] B. Krishnapuram, L. Carin, M. A. T. Figueiredo, and A. Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(6):957–968, June 2005.

[97] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Department of Computer Science, University of Toronto, 2009.

[98] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[99] S. Krogstad, V. L. Hauge, A. Gulbransen, et al. Adjoint multiscale mixed finite elements. *SPE Journal*, 16(01):162–171, 2011.

[100] K. Kunisch and S. Volkwein. Proper orthogonal decomposition for optimality systems. *ESAIM: Mathematical Modelling and Numerical Analysis*, 42(1):1–23, 2008.

[101] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 265–272, USA, 2011. Omnipress.

[102] T. H. Le, T. T. Tran, and L. K. Huynh. Identification of hindered internal rotational mode for complex chemical species: A data mining approach with multivariate logistic regression model. *Chemometrics and Intelligent Laboratory Systems*, 172:10 – 16, 2018.

[103] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[104] Y. Lecun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits. 1998.

[105] S. H. Lee, H. Zhou, and H. A. Tchelepi. Adaptive multiscale finite-volume method for nonlinear multiphase transport in heterogeneous formations. *Journal of Computational Physics*, 228(24):9036–9058, 2009.

[106] S. Li, S. Ahmed, G. Klimeck, and E. Darve. Computing entries of the inverse of a sparse matrix using the find algorithm. *Journal of Computational Physics*, 227(22):9408–9427, 2008.

[107] J. Liao and K.-V. Chin. Logistic regression for disease classification using microarray data: model selection in a large p and small n case. *Bioinformatics*, 23(15):1945–1951, 2007.

[108] C. Lieberman, K. Willcox, and O. Ghattas. Parameter and State Model Reduction for Large-Scale Statistical Inverse Problems. *SIAM Journal on Scientific Computing*, 32(5):2523–2542, 2010.

[109] L. Lin, J. Lu, L. Ying, R. Car, E. Weinan, et al. Fast algorithm for extracting the diagonal of the inverse matrix with application to the electronic structure analysis of metallic systems. *Communications in Mathematical Sciences*, 7(3):755–777, 2009.

[110] K. Lipnikov, J. Morel, and M. Shashkov. Mimetic finite difference methods for diffusion equations on non-orthogonal non-conformal meshes. *Journal of Computational Physics*, 199(2):589–597, 2004.

[111] A. Lipponen, A. Seppänen, and J. Kaipio. Electrical impedance tomography imaging with reduced-order model based on proper orthogonal decomposition. *Journal of Electronic Imaging*, 22(2):023008–16, 2013.

[112] B. Lohmann and R. Eid. Efficient order reduction of parametric and nonlinear models by superposition of locally reduced models. *Methoden und Anwendungen der Regelungstechnik*, 2007.

[113] I. Lunati and P. Jenny. Multiscale finite-volume method for density-driven flow in porous media. *Computational Geosciences*, 12(3):337–350, 2008.

[114] M. Ma, A. N. Nikolakopoulos, and G. B. Giannakis. Fast decentralized learning via hybrid consensus ADMM. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018.

[115] J. Macdonald and L. Ruthotto. Improved susceptibility artifact correction of echo-planar mri using the alternating direction method of multipliers. *Journal of Mathematical Imaging and Vision*, 60(2):268–282, Feb 2018.

[116] S. P. MacLachlan and J. D. Moulton. Multilevel upscaling through variational coarsening. *Water Resources Research*, 42(2):131–9, 2006.

[117] R. Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[118] P. R. McGillivray. *Forward modeling and inversion of DC resistivity and MMR data*. PhD thesis, University of British Columbia, 1992.

[119] H. Miao, X. Liu, B. Huang, and L. Getoor. A hypergraph-partitioned vertex programming approach for large-scale consensus optimization. In *Big Data, 2013 IEEE International Conference on*, pages 563–568. IEEE, 2013.

[120] O. Møyner and K.-A. Lie. A multiscale restriction-smoothed basis method for high contrast porous media represented on unstructured grids. *Journal of Computational Physics*, 304:46–71, 2016.

[121] J. L. Mueller and S. Siltanen. *Linear and Nonlinear Inverse Problems with Practical Applications*, volume 10. SIAM, 2012.

[122] F. Negri, G. Rozza, A. Manzoni, and A. Quarteroni. Reduced basis method for parametrized elliptic optimal control problems. *SIAM Journal on Scientific Computing*, 35(5):A2316–A2340, 2013.

[123] A. Y. Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM, 2004.

[124] M. O'Connell, M. E. Kilmer, and E. de Sturler. Computing Reduced Order Models via Inner-Outer Krylov Recycling in Diffuse Optical Tomography. *SIAM Journal on Scientific Computing*, 39(2):B272–B297, 2017.

[125] D. P. O'Leary. The block conjugate gradient algorithm and related methods. *Linear algebra and its applications*, 29:293–322, 1980.

[126] H. Panzer, J. Mohring, R. Eid, and B. Lohmann. Parametric Model Order Reduction by Matrix Interpolation. *at - Automatisierungstechnik*, 58(8), 2010.

[127] R. L. Parker. *Geophysical Inverse Theory*. Princeton University Press, Princeton NJ, 1994.

[128] L. Parra, C. Spence, , and P. Sajda. Higher-order statistical properties arising from the nonstationarity of natural signals. *In Advances in Neural Information Processing Systems*, 13:786–792, 2001.

[129] E. Parramore, M. G. Edwards, M. Pal, and S. Lamine. Multiscale finite-volume cvd-mpfa formulations on structured and unstructured grids. *Multiscale Modeling & Simulation*, 14(2):559–594, 2016.

[130] L. Y. Pratt. Discriminability-based transfer between neural networks. In *Advances in Neural Information Processing Systems*, pages 204–211, 1993.

[131] R. Pratt. Seismic waveform inversion in the frequency domain, part 1: Theory, and verification in a physical scale model. *Geophysics*, 64(3):888–901, 1999.

[132] P. Raman, S. Matsushima, X. Zhang, H. Yun, and S. V. N. Vishwanathan. DS-MLR: exploiting double separability for scaling up distributed multinomial logistic regression. *CoRR*, abs/1604.04706, 2016.

[133] H. Robbins and S. Monro. A Stochastic Approximation Method. *The annals of mathematical statistics*, 22(3):400–407, 1951.

[134] L. Ruthotto. *KrylovMethods.jl*. https://github.com/lruthotto/KrylovMethods.jl.

[135] L. Ruthotto and E. Haber. Deep neural networks motivated by partial differential equations. *arXiv preprint arXiv:1804.04272*, 2018.

[136] L. Ruthotto, E. Treister, and E. Haber. `jInv`–a flexible julia package for PDE parameter estimation. *SIAM Journal on Scientific Computing*, 39(5):S702–S722, 2017.

[137] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, Philadelphia, 2003.

[138] A. K. Saibaba, J. Lee, and P. K. Kitanidis. Randomized algorithms for generalized Hermitian eigenvalue problems with application to computing Karhunen–Loève expansion. *Numerical Linear Algebra with Applications*, 23(2):314–339, 2016.

[139] R. Sargent. Optimal control. *Journal of Computational and Applied Mathematics*, 124(1):361–371, 2000.

[140] J. H. Schön. *Physical Properties of Rocks: Fundamentals and Principles of Petrophysics*, volume 65. Elsevier, Amsterdam, 2015.

[141] J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950.

[142] J. Shi, W. Yin, S. Osher, and P. Sajda. A fast hybrid algorithm for large-scale l1-regularized logistic regression. *J. Mach. Learn. Res.*, 11:713–741, Mar 2010.

[143] A. M. Stuart. Inverse problems: a Bayesian perspective. *Acta Numerica*, 19:451–559, 2010.

[144] M. Taddy. Distributed multinomial regression. *The Annals of Applied Statistics*, 9(3):1394–1414, Sep 2015.

[145] J. M. Tang and Y. Saad. Domain-decomposition-type methods for computing the diagonal of a matrix inverse. *SIAM Journal on Scientific Computing*, 33(5):2823–2847, 2011.

[146] J. M. Tang and Y. Saad. A probing method for computing the diagonal of a matrix inverse. *Numerical Linear Algebra with Applications*, 19(3):485–501, 2012.

[147] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein. Training neural networks without gradients: A scalable admm approach. In *International Conference on Machine Learning*, pages 2722–2731, 2016.

[148] G. Taylor, Z. Xu, and T. Goldstein. Scalable classifiers with ADMM and transpose reduction. In *Association for the Advancement of Artificial Intelligence Workshops*, 2017. https://www.aaai.org/ocs/index.php/WS/AAAIW17/paper/view/15174.

[149] A. N. Tikhonov. On the solution of ill-posed problems and the method of regularization. In *Doklady Akademii Nauk*, volume 151, pages 501–504. Russian Academy of Sciences, 1963.

[150] E. Treister and E. Haber. A fast marching algorithm for the factored eikonal equation. *Journal of Computational Physics*, 324:210–225, 2016.

[151] Y. Tsuruoka, J. McNaught, J. Tsujii, and S. Ananiadou. Learning string similarity measures for gene/protein name dictionary look-up using logistic regression. *Bioinformatics*, 23(20):2768–2774, 2007.

[152] T. van Leeuwen and F. J. Herrmann. A penalty method for pde-constrained optimization in inverse problems. *Inverse Problems*, 32(1):015007, 2015.

[153] S. Wang and L. Liao. Decomposition method with a variable parameter for a class of monotone variational inequality problems. *Journal of optimization theory and applications*, 109(2):415–429, 2001.

[154] S. Ward and G. Hohmann. Electromagnetic theory for geophysical applications. *Electromagnetic Methods in Applied Geophysics*, 1:131–311, 1988. Soc. Expl. Geophys.

[155] W. Wilhelms, C. Schwarzbach, R.-U. Börner, and K. Spitzer. A fast 3d mt inversion–the forward operator behind. *Journal of Future Generation Computer Systems*, 20(3):475–487.

[156] K. Willcox and J. Peraire. Balanced Model Reduction via the Proper Orthogonal Decomposition. *AIAA Journal*, 40(11):2323–2330, 2002.

[157] S. Wright and J. Nocedal. Numerical optimization. *Springer Science*, 35:67–68, 1999.

[158] XtractOpen. Meganet. 2018. https://github.com/XtractOpen.

[159] Z. Xu, G. Taylor, H. Li, M. A. Figueiredo, X. Yuan, and T. Goldstein. Adaptive consensus ADMM for distributed optimization. In *International Conference on Machine Learning*, pages 3841–3850, 2017.

[160] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3320–3328, Cambridge, MA, USA, 2014. MIT Press.

[161] G.-X. Yuan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *J. Mach. Learn. Res.*, 11:3183–3234, Dec 2010.

[162] R. Zhang and J. Kwok. Asynchronous distributed ADMM for consensus optimization. In *International Conference on Machine Learning*, pages 1701–1709, 2014.

[163] T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 116–124, New York, NY, USA, 2004. ACM.