The Dissertation Committee for Nicholas Vieau Alger
certifies that this is the approved version of the following dissertation:

Data-Scalable Hessian Preconditioning for Distributed Parameter PDE-Constrained Inverse Problems

Committee:

Omar Ghattas, Supervisor

Tan Bui-Thanh, Co-Supervisor

George Biros

Inderjit Dhillon

Sergey Fomel

J. Tinsley Oden

Data-Scalable Hessian Preconditioning for Distributed Parameter PDE-Constrained Inverse Problems

by

Nicholas Vieau Alger

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2019

Dedicated to my parents, Jeff and Cindy, and my brothers, J.J. and Bob.

Acknowledgments

I thank my advisors Omar Ghattas and Tan Bui-Thanh. I am grateful to Omar and Tan for sharing their deep knowledge of inverse problems and optimization with me. Their intellectual courage—attempting to solve the right problems even if they are difficult—has inspired me. I also thank my committee, especially Sergey Fomel for our discussions about seismic imaging.

My thinking has been influenced by interactions with many friends and colleagues in the CCGO and CSEO research groups, as well as other graduate students and researchers at the University of Texas at Austin. Much of what I know about inverse problems was learned through conversations and nature walks with James Martin. My research was greatly aided by many conversations with Aaron Myers, Vishwas Rao, and Pearl Flath.

I credit Umberto Villa, Tan Bui-Thanh, and Omar Ghattas for their work on the augmented KKT preconditioner paper, of which they are co-authors. Additionally, I thank James Martin and Toby Isaac for helpful discussions regarding the augmented KKT preconditioner. The constant in the augmented KKT condition number bound was improved based on suggestions from an anonymous reviewer. I credit Vishwas Rao, Aaron Myers, Tan Bui-Thanh, and Omar Ghattas for their work on the adaptive product-convolution scheme paper, of which they are co-authors. Additionally, I thank J.J. Alger, Benjamin Babcock, Joe Bishop, Andrew Potter, Georg Stadler, Umberto Villa, and Hongyu Zhu for helpful discussions regarding the adaptive product-convolution scheme.

This dissertation has benefited from comments and editing suggestions from J.J. Alger, Jeff Alger, Tan Bui-Thanh, Josh Chen, Omar Ghattas, Trevor Heise, Aaron Myers, Tom O'Leary, and Vishwas Rao. I especially thank J.J. for our many conversations about this document during the writing process and for his editing suggestions, and Trevor for his extensive editing suggestions.

Data-Scalable Hessian Preconditioning for Distributed Parameter PDE-Constrained Inverse Problems

Publication No. _____

Nicholas Vieau Alger, Ph.D.
The University of Texas at Austin, 2019

Supervisors: Omar Ghattas
Tan Bui-Thanh

Hessian preconditioners are the key to efficient numerical solution of large-scale distributed parameter PDE-constrained inverse problems with highly informative data. Such inverse problems arise in many applications, yet solving them remains computationally costly. With existing methods, the computational cost depends on spectral properties of the Hessian which worsen as more informative data are used to reconstruct the unknown parameter field. The best case scenario from a scientific standpoint (lots of high-quality data) is therefore the worst case scenario from a computational standpoint (large computational cost).

In this dissertation, we argue that the best way to overcome this predicament is to build data-scalable Hessian/KKT preconditioners—preconditioners that perform well even if the data are highly informative about the parameter. We present a novel data-scalable KKT preconditioner for a diffusion inverse problem, a novel data-scalable Hessian preconditioner for an advection inverse problem, and a novel data-scalable domain decomposition preconditioner for an auxiliary operator that arises in connection with KKT preconditioning for a wave inverse problem. Our novel preconditioners outperform existing preconditioners in all three cases: they are robust to large numbers of observations in the diffusion inverse problem, large Peclet numbers in the advection inverse problem, and high wave frequencies in the wave inverse problem.

Table of Contents

Chapter 1

Introduction

In distributed parameter inverse problems governed by partial differential equations (PDEs), one seeks to reconstruct an unknown spatially varying parameter field from measurements ("data") of a state variable that depend on the parameter implicitly through the solution of a PDE.[1] Although such inverse problems arise in a wide variety of applications, solving them remains computationally costly since a large number of PDEs must be solved in the process.

- The number of PDE solves that must be performed depends on spectral properties of the Hessian in the inverse problem.
- These spectral properties of the Hessian worsen as more informative data are used to reconstruct the parameter.

By saying the spectral properties of the Hessian "worsen," we mean that the eigenvalues of the data misfit term in the Hessian become larger and less clustered. This forces existing methods for solving the inverse problem to perform more PDE solves. By saying one set of data is "more informative" than another set of data, we mean the more informative data could, in principle (ignoring computational cost), be used to reconstruct the parameter field with more certainty than the less informative data. The informativeness of the data would increase, for example, if we performed more measurements, or if we performed measurements with greater accuracy.

This leads to a predicament: the best case scenario from a scientific standpoint (lots of high-quality data) is the worst case scenario from a computational stand-

[1] This chapter contains content from [8] (Nick Alger, Umberto Villa, Tan Bui-Thanh, and Omar Ghattas. A data scalable augmented Lagrangian KKT preconditioner for large-scale inverse problems. SIAM Journal on Scientific Computing, 39(5):A2365–A2393, 2017.) and [7] (Nick Alger, Vishwas Rao, Aaron Myers, Tan Bui-Thanh, and Omar Ghattas. Scalable matrix-free adaptive product-convolution approximation for locally translation-invariant operators. arXiv preprint arXiv:1805.06018, 2018. Submitted.).

point (many PDE solves required). We overcome this predicament by building **data-scalable Hessian preconditioners**—preconditioners that perform well regardless of how informative the data are about the parameter. When data-scalable Hessian preconditioners are used within appropriate numerical methods, the cost to solve the inverse problem does not grow with the informativeness of the data.[2] This allows computationally tractable inverse problems with highly informative data to be solved faster, and offers a way to solve inverse problems with highly informative data that are currently intractable.

1.1 Model problems

We focus on inverse problems in which we have noisy observations,

$$y = Bu + \zeta,$$

of a state variable, u, which depends on an unknown spatially varying parameter, q, implicitly through the solution of a state equation,

$$A(q)u = f(q). \tag{1.1}$$

Here, B is a linear observation operator, ζ is unknown noise,[3] and (1.1) is a PDE or system of PDEs that is uniquely solvable for u given q. Although we work with state equations that are linear in u, the methods developed in this dissertation are equally applicable to smooth nonlinear state equations, because such state equations look linear locally. Given a candidate parameter, q, the forward problem is to generate predicted noise-free observations by solving (1.1) for u, then computing Bu. The goal of the inverse problem is to invert this process:

Inverse problem
Given y, infer q.

More detailed coverage of inverse problems can be found in [183]. The following model problems illustrate this inverse problem framework.

[2] We will discuss how Hessian preconditioners can speed up numerical methods for solving the inverse problem in Chapter 3 and Appendix A.

[3] In practice ζ will also contain model error, though we do not consider model error here.

1.1.1 Diffusion model problem

An object is heated non-uniformly until it reaches thermal equilibrium, then measurements of the temperature are performed at many points distributed throughout the object. The inverse problem is to use the temperature data to infer the object's unknown spatially varying log-conductivity field (see Figure 1.1). In this model problem, the parameter, q, is the log-conductivity field, the state variable, u, is the temperature, and the state equation is the inhomogeneous Poisson equation,

$$\underbrace{-\nabla \cdot e^q \nabla u = f}_{\text{“}A(q)u=f\text{”}}. \tag{1.2}$$

We write $A(q)u = f$ to denote the result of converting (1.2) to weak form, incorporating appropriate boundary conditions, and discretizing with the finite element method. The observation operator, B, extracts point measurements of the temperature,

$$(Bu)_i := u(x_i),$$

for a collection of points x_i.[4]

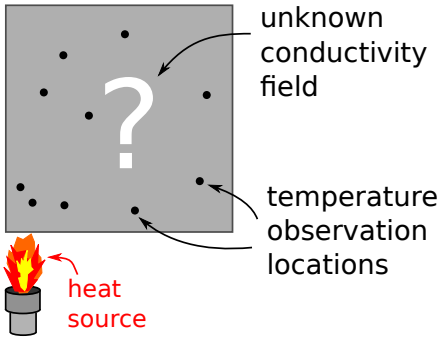


Figure 1.1: **(Diffusion model problem)** We are given point measurements of temperature at a collection of locations in a material (indicated by the black dots), and seek to infer the material's spatially varying log-conductivity field.

We also consider the source inversion simplification of this inverse problem, in which the conductivity is known and uniform, and the parameter being inverted for

[4]Defining the observation operator this way assumes that the x_i do not lie on points of discontinuity of u. In realistic scenarios, u will each be continuous almost everywhere. Alternatively, one may address regularity concerns by defining the measurements as averages over small balls surrounding the x_i, rather than point measurements at the x_i.

is the spatially varying heat source, $f(q) = q$. In this case the PDE becomes

$$\underbrace{-\Delta u = q}_{\text{“}Au=f(q)\text{”}}.$$

The source inversion problem shares the same diffusive character as the parameter inversion problem, and the Hessian for the source inversion problem has similar spectral properties as the Hessian for the parameter inverse problem, but in the source inversion problem the map from parameter to observations is linear.

1.1.2 Advection model problem

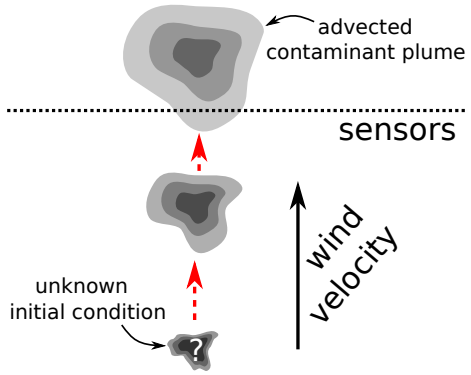


Figure 1.2: **(Advection model problem)** We seek to infer the initial concentration of a contaminant plume based on time series measurements of contaminant concentration as the plume passes through an array of sensors.

A contaminant is released into the atmosphere, where it is transported by the wind. Measurements of the contaminant concentration are performed as the contaminant passes through a sensor-containing geographic boundary. The inverse problem is to use the sensor data to infer the initial spatially varying concentration of the contaminant (see Figure 1.2). In this case, q is the initial concentration of the contaminant, u is the concentration of the contaminant at all times, the state equation is the advection-diffusion equation,

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{1}{\text{Pe}}\Delta u - b \cdot \nabla u, & t > 0, \\ u = q, & t = 0, \end{cases} \tag{1.3}$$

with wind velocity field b, and B performs time series measurements of the concentration of the contaminant along a surface Γ_{obs} for times $t \in (0, 1]$:

$$(Bu)(s,t) = u(s,t), \quad s \in \Gamma_{\text{obs}},\ t \in (0,1].$$

Advection inverse problems are discussed at greater length in [4].

1.1.3 Wave model problem

An oil company excites the earth by dropping a heavy weight on the ground, generating seismic waves that travel through the subsurface, reflecting off interfaces between different types of rock. These reflected waves are measured with an array of sensors on the surface. The inverse problem is to use these measurements to create a picture of the squared slowness field in the subsurface (See Figure 1.3).

To model this, we use the frequency domain seismic full-waveform inversion framework [165] with the acoustic approximation.[5] Seismic inversion uses data from many sources (the ground is excited many times at different locations). For each source, data are obtained for many frequencies. Let k be a linear index that runs over all source-frequency pairs, let ω_k be the associated frequency, and let f_k be the time-harmonic component of the associated source for that frequency. The time harmonic component of the pressure wave associated with this source-frequency pair, u_k, satisfies the Helmholtz equation,

$$\underbrace{\Delta u_k + \omega_k^2\, q\, u_k = f_k}_{\text{“}A_k(q)u_k = f_k\text{”}}. \tag{1.4}$$

Here, the parameter is the squared slowness, $q = 1/c^2$, where c is the sound speed. We write $A_k(q)u_k = f_k$ to denote the result of converting (1.4) to weak form, incorporating Dirichlet boundary conditions on the top boundary (earth's surface), PML absorbing boundary conditions [37] on the other boundaries, and discretizing with the finite element method.

The observation operator measures the following frequency-weighted normal derivative of the time harmonic pressure along a subset Γ_{obs} of the top surface:

$$(B_k u_k)(s) = \nu \cdot \omega_k^{-2} \nabla u_k(s) \quad \text{for } s \in \Gamma_{\text{obs}}.$$

[5] That is, we consider only pressure waves and ignore shear waves.

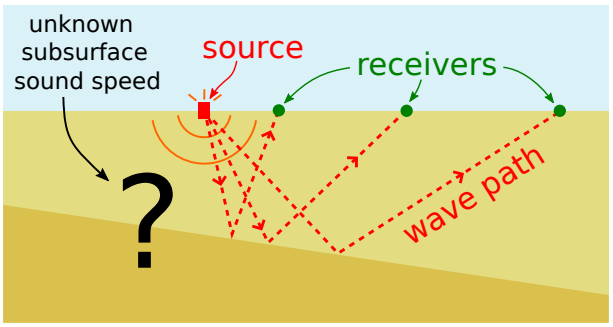


Figure 1.3: **(Wave model problem)** We seek to infer the subsurface sound speed field from measurements of reflected waves.

These measurements correspond, after some processing, to vertical accelerometer readings along Γ_{obs}.

When using multiple frequencies and sources, the overall state u is a concatenation of the state variables for each frequency-source pair, and satisfies the block-diagonal state equation:

$$\underbrace{\begin{bmatrix} A_1(q) & & & \\ & A_2(q) & & \\ & & A_3(q) & \\ & & & \ddots \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \end{bmatrix}}_{A(q)u = f}. \tag{1.5}$$

Each block row of (1.5) is a state equation of the form (1.4). Similarly, the observations for all frequencies and sources takes the form

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & B_3 & \\ & & & \ddots \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \end{bmatrix} + \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \\ \vdots \end{bmatrix}}_{y = Bu + \zeta},$$

where ζ_k is unknown noise corrupting the measurements associated with the k^{th} source-frequency pair.

1.1.4 Operators preconditioned

We contribute novel data-scalable preconditioners for the KKT operator—a sparse operator that is algebraically equivalent to the Hessian—in the source inversion vari-

ant of the diffusion inverse problem, for the Hessian in the advection inverse problem with uniform flow, and for the operator $B^*B + \rho A^*A$ in the single-frequency single source variant of the wave inverse problem (ρ is a penalty parameter). Our novel preconditioners outperform existing preconditioners[6] for these operators: they are robust to large numbers of observations in the diffusion inverse problem, large Peclet numbers in the advection inverse problem, and high wave frequencies in the wave inverse problem.

The complete diffusion inverse problem (not the source inversion variant), the advection-diffusion inverse problem with non-uniform flow, and the multiple frequency multiple source wave inverse problem are targets of future research. We expect that methods presented in this dissertation will be useful for preconditioning these more difficult problems.

1.2 Setting and notation

We use an abstract version of the "discretize-then-optimize" approach [107]. Vectors (e.g. q) are abstract elements of *finite dimensional* function spaces (e.g. finite element spaces) without reference to a basis. Operators (e.g., B) are abstract mappings between these spaces. Although we operate in a finite-dimensional setting, we always pay attention to the behavior of numerical methods and preconditioners as the mesh size goes to zero and the dimension of the finite dimensional function spaces goes to infinity (the continuum limit). We consider methods for solving the inverse problem to be unacceptable if their performance significantly degrades as the mesh used to discretize the problem is refined.

We use bold lettering (e.g., $\mathbf{q}$, $\mathbf{B}$) to denote arrays of numbers that represent vectors and operators with respect to concrete bases. These arrays of numbers are what are stored and operated on when numerically solving the inverse problem with a computer. In a concrete basis, all formulas remain essentially the same, with the exception that Gram matrices (e.g., mass matrices) and their inverses appear within formulas to account for the Riesz representation theorem for adjoints in a non-orthogonal basis.

[6] We will discuss existing preconditioners in Chapter 6.

Unless otherwise stated, we use L^2 norms, inner products, and adjoints. If M is positive and self-adjoint, we write $\|x\|_M := (x^* M x)^{1/2}$ to denote the norm induced by M. When we need to convert between between bilinear forms and their associated linear operators, we use the symbol $\simeq$. For example, if $c(\cdot, \cdot)$ is a bilinear form, we write $C \simeq c$ to say that $C : x \mapsto c(\cdot, x)$ is the linear operator associated with c.

1.3 The inverse problem

The inverse problem of inferring q from y is ill-posed. If many different q's are consistent with y, which q should we choose? Should we choose a single q at all? The two primary frameworks for addressing ill-posedness are the deterministic framework and the Bayesian framework. In Chapter 2, we will discuss ill-posedness in greater detail and derive these frameworks. We use the deterministic framework in computations, but we also present the Baysian framework since it is required for our information-theoretic analysis of the spectrum of the Hessian in Chapter 4.

Briefly, the simplest[7] q consistent with y is the solution to the optimization problem

$$\arg\min_q \mathcal{J}(q), \tag{1.6}$$

where $\mathcal{J}$ takes the form

$$\mathcal{J}(q) := \frac{1}{2} \|Bu(q) - y\|_Y^2 + \frac{1}{2} \|q - \overline{q}_0\|_R^2 . \tag{1.7}$$

Here, $u(q)$ denotes the solution of (1.1) as a function of q, and $\overline{q}_0$ is a parameter that we expect q to be similar to before the observations are taken into account. The operator Y weights the data to account for uncertainty in the measurements, and R is a symmetric positive definite operator that encodes our notion of simplicity. The smaller the second term is, the simpler q is. We call R (or a scaled version of R, in some contexts) the *regularization operator*. We will discuss regularization in greater detail in Chapter 2. In the deterministic framework, one computes a single estimate of q by solving optimization problem (1.6).

[7] under assumptions we will discuss in Section 2.2

Within $\mathcal{J}$, the data misfit term,

$$\mathcal{J}_d(q) := \frac{1}{2} \|Bu(q) - y\|_Y^2 ,$$

measures the discrepancy between the observed data (based on the unknown true parameter) and the predicted data (based on the candidate parameter q). The regularization term,

$$\mathcal{J}_R(q) := \frac{1}{2} \|q - \overline{q}_0\|_R^2 ,$$

measures how different q is from $\overline{q}_0$. To make $\mathcal{J}_d$ small, the predicted observations should match the true observations. To make $\mathcal{J}_R$ small, the parameter should look like what we expect it to look like. Minimizing $\mathcal{J}$, which is the sum of $\mathcal{J}_d$ and $\mathcal{J}_R$, requires balancing these competing interests.

Alternatively, under Gaussian assumptions[8] for the noise and prior, the Bayesian posterior probability distribution for q given y is

$$\pi(q|y) \propto \exp\left(-\mathcal{J}(q)\right), \tag{1.8}$$

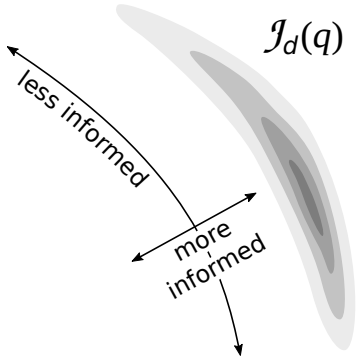
where $\propto$ denotes proportionality up to a normalizing constant. In this case $\mathcal{J}$ also takes the form (1.7), but here R is the inverse of the prior covariance, Y is the inverse of the noise covariance, and $\overline{q}_0$ is the prior mean.

1.4 Information, the Hessian, and the KKT operator

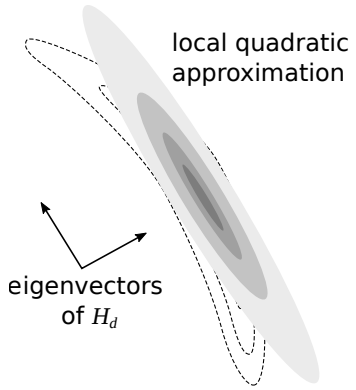
The directional scalings of the contours of $\mathcal{J}_d$ characterize how informative the data are about different components of q. Intuitively,

- directions in which the contours of $\mathcal{J}_d$ are closely spaced correspond to components of q which are well-informed, since perturbing q in such a direction would yield large changes to the data misfit, and
- directions in which the contours of $\mathcal{J}_d$ are widely spaced correspond to components of q that are poorly-informed, since perturbing q in such a direction would yield small changes to the data misfit.

[8] These Gaussian asumptions are not as restrictive as they may seem; see Section 2.5.



(a) The data misfit function, $\mathcal{J}_d$. Directions in which the contours of $\mathcal{J}_d$ are closely spaced are more informed by the data than directions in which the contours are widely spaced.



(b) Eigenvalues and eigenvectors of H_d characterize the informativeness of the data under a local quadratic approximation to $\mathcal{J}_d$.

Figure 1.4

This is illustrated in Figure 1.4. The directional scalings of the contours of $\mathcal{J}_d$ are locally quantified by the eigenstructure of the data misfit Hessian,

$$H_d \simeq \frac{d^2\mathcal{J}_d}{dq^2}.$$

The larger an eigenvalue of H_d, the more closely spaced the contours of $\mathcal{J}_d$ are in the associated eigenvector's direction, and the more informative the data are about that component of the parameter in the associated eigenvector's direction. In Chapter 4, we will use information theory to make precise the connection between the eigenvalues of H_d and the information contained in the data.

The Hessian,

$$H = H_d + R \simeq \frac{d^2\mathcal{J}}{dq^2},$$

is the data misfit Hessian plus the Hessian of the regularization or prior term, $R \simeq \frac{d^2\mathcal{J}_R}{dq^2}$. In Chapter 3 we will see that H is central to computational methods for solving the inverse problem. In particular, Newton-Krylov methods for solving the deterministic optimization problem, (1.6), require solving a linear system with H as the coefficient operator at every outer (Newton) iteration. These methods use an inner Krylov iteration to perform these linear solves. Hessian preconditioners allow

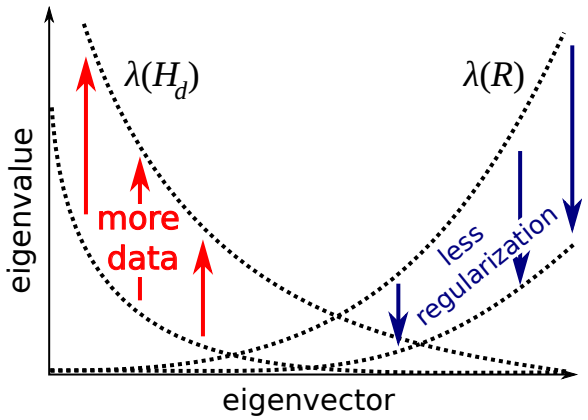


Figure 1.5: Eigenvalues for the generalized eigenvalue problem of H_d and R. Eigenvalues of R should be small where eigenvalues of H_d are large, and vice-versa. As more informative data are included in the inversion, the eigenvalues of H_d increase. As less regularization is used, the eigenvalues of R decrease.

us to solve the inverse problem faster by solving these Newton-Krylov linear systems faster.

Where it is convenient, we will precondition the KKT operator, K, instead of the Hessian, H. The KKT operator is the second derivative of the Lagrangian function for a formulation of optimization problem (1.6) in which the state equation constraint, (1.1), is enforced with Lagrange multipliers. We will define the Lagrangian function and the KKT operator in Section 3.2.2. The operators H and K are algebraically equivalent—if one can solve linear systems with K as the coefficient operator, then one can solve linear systems with H as the coefficient operator, and vice versa. We will summarize the connections between H and K in Chapter 5.

1.5 Regularization preconditioning is not good enough

Despite the importance of the Hessian, current techniques for Hessian preconditioning in PDE-constrained inverse problems scale poorly with increasingly informative data. Unlike operators in forward problems, access to the Hessian is *matrix-free*. We can apply the Hessian to vectors but cannot easily access individual entries of the Hessian's matrix representation. Additionally, the spectral behavior of the Hessian differs from that of forward operators because the Hessian consists of two terms (H_d and R) which ideally act in opposition to each other. In directions that H_d acts

strongly, R should act weakly, and vice versa (more on this in Section 2.7.1). Most preconditioning techniques that are highly effective for forward problems are therefore inapplicable to Hessians.

Regularization preconditioning (preconditioning H by R) and techniques based on low-rank approximation of the regularization preconditioned data misfit Hessian are currently the most effective techniques for many circumstances. These techniques are *mesh-scalable* in that the required number of PDE solves remains roughly constant as the mesh used to discretize the parameter is refined. But these techniques are not *data-scalable*. As the data become more informative or as the regularization lessens, the data misfit term increases in importance while the regularization term decreases in importance within the Hessian (see Figure 1.5). This results in worse performance of regularization or low-rank approximation based preconditioners as more informative data are included in the inverse problem. *For this dissertation we will not be satisfied with reducing the number of PDE solves to a mesh-independent constant. The goal is to also make this constant data-independent and as small as possible.*

1.6 Summary of the dissertation

This dissertation will:

1. provide background on distributed parameter PDE-constrained inverse problems, with emphasis on ill-posedness, the informativeness of the data, the spectrum of the Hessian, and how these factors affect the performance of solution methods, and

2. present our new preconditioners and demonstrate that they are data-scalable.

1.6.1 Background summary

Chapter 2 (Inverse Problem Frameworks) The fundamental characteristic that differentiates inverse problems from forward problems is ill-posedness, which is physically meaningful, not just a computational nuisance. In this chapter, we review ill-posedness, derive deterministic and Bayesian frameworks for the inverse problem from first principles, and show how these two frameworks are connected. We then

discuss over- and under-regularization, and provide practical and heuristic guidance on the choice of regularization or prior.

Chapter 3 (Deterministic Solution Methods) Solving the inverse problem within the deterministic framework requires solution of a large-scale nonlinear optimization problem. The primary source of difficulty in solving this optimization problem is ill-conditioning. To address ill-conditioning, we advocate using second-order Newton-Krylov-type methods, and building preconditioners for the linear systems—linear systems with the Hessian or other Hessian-like operators as the coefficient operator—that arise in these methods.

Chapter 4 (Spectrum of the Hessian and information) In this chapter we show that the spectrum of H_d characterizes how informative the data are about the parameter. The larger an eigenvalue of H_d is, the more informative the data are about the component of the parameter in the corresponding eigenvector's direction.

Chapter 5 (Hessian and KKT facts) Applying the Hessian to a vector can be done via a process that involves solving a sequence of two PDEs with the same form as the state equation. There are formulas for the other Hessian-like operators that we will use later when constructing preconditioners (i.e. the Gauss-Newton Hessian, the KKT operator, and the Gauss-Newton KKT operator). We show that these other Hessian-like operators are equivalent to the Hessian for the purposes of preconditioning: a solver or good preconditioner for one of the Hessian or Hessian-like operators can be used to build a good preconditioner for any of the others.

Chapter 6 (Existing Hessian Solvers, Preconditioners, and Approximations) Existing Hessian and KKT preconditioners, as well as other relevant solvers, preconditioners, and approximation schemes, are not satisfactory. Existing methods are either not data-scalable, or make unrealistic assumptions, or only apply to niche problems. New preconditioners are needed.

1.6.2 Hessian preconditioners summary

Chapter 7 (Augmented Lagrangian KKT preconditioner) We present an augmented Lagrangian-type preconditioner for the Gauss-Newton KKT operator based on a block diagonal approximation of the upper left block of an augmented version of the Gauss-Newton KKT operator.

The idea of the preconditioner is to form a Schur complement for the adjoint variable (instead of the parameter) within the KKT operator. Unfortunately, because the objective block of the KKT operator is singular for inverse problems with limited information, this adjoint Schur complement does not exist. So, we augment the Lagrangian with a quadratic penalty on the constraint. The objective block of the augmented KKT operator (the KKT operator associated with the augmented Lagrangian) is invertible, and the adjoint Schur complement of the augmented KKT operator exists.

The preconditioner requires solvers for two linear subproblems that arise in the augmented KKT operator, which are easier to precondition than the Hessian. Analysis of the spectrum of the preconditioned KKT operator indicates that the preconditioner is effective when the regularization neither over-penalizes highly informed parameter modes, nor under-penalizes uninformed modes.

We present a numerical study for the source inversion variant of the diffusion model problem, demonstrating the effectiveness and data-scalability of the preconditioner. See Figure 1.6 for a preview of the convergence results using this preconditioner. In this example, three MINRES iterations on the KKT system with our preconditioner results in a reconstruction with better accuracy than 50 iterations of CG on the Hessian system with regularization preconditioning.

Chapter 8 (Adaptive Product-Convolution Approximation) Hessians and interface operators that arise in PDE-constrained inverse problems often exhibit local translation invariance. That is, the impact of a point source centered at point p on a target at point q is similar to the impact of a point source centered at point $p+h$ on a target at point $q+h$, if h is not too large.

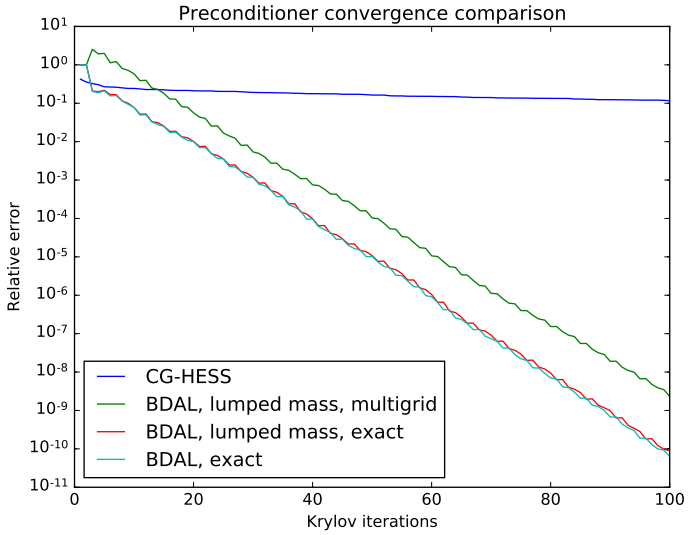


Figure 1.6: Preview of results from Chapter 7. Relative error in the parameter reconstruction for a big data Poisson source inversion problem, as a function of the number of Krylov iterations. "CG-HESS" indicates regularization preconditioning, whereas "BDAL" indicates variants of our new augmented Lagrangian preconditioner.

In this chapter, we develop an operator approximation scheme for operators that are locally translation-invariant, even if these operators are high-rank or full-rank. We present an adaptive grid matrix-free operator approximation scheme based on a "product-convolution" interpolation of convolution operators. Constructing this approximation requires applying the operator to point sources centered on nodes in an adaptively refined grid of sample points. A randomized, adjoint-based, a-posteriori error estimator drives this adaptivity. Once constructed, the approximation can be efficiently applied to vectors using the fast Fourier transform (FFT), and used as a surrogate model. It can also be efficiently converted to hierarchical matrix (H-matrix) format, then inverted or factorized using scalable H-matrix arithmetic. Using fewer sample points yields a less accurate approximation, which can be used as a preconditioner. We address issues related to boundaries, which plague existing product-convolution schemes, and prove that our scheme eliminates boundary artifacts.

We apply the scheme to the data misfit Hessian for our advection inverse problem. Numerical results show that the scheme is data-scalable—the number of sample points remains constant as the Peclet number, a proxy for the informativeness of the

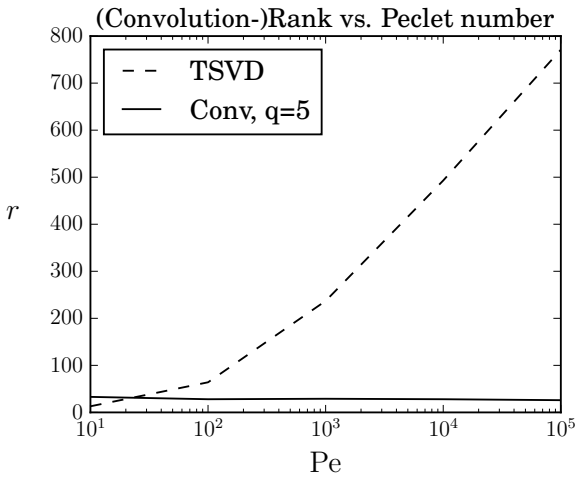


Figure 1.7: Preview of results from Chapter 8. Approximation of the data misfit Hessian for the advection inverse problem. The plot shows the (convolution-)rank, r, required to achieve a relative approximation error of 10% for a variety of Peclet numbers, Pe. "TSVD" indicates low-rank approximation via the truncated singular value decomposition. "Conv" indicates our new product-convolution approximation scheme.

data, increases. A preview of our numerical results is shown in Figure 1.7. The resulting preconditioner substantially outperforms regularization preconditioning. We also apply the scheme to other, non-Hessian, operators in Appendix E, and to an interface operator used within our preconditioner for the wave inverse problem in Chapter 9.

Chapter 9 (Domain Decomposition Wave preconditioner) The augmented Lagrangian KKT preconditioner from Chapter 7 may be applied to the wave inverse problem, but this requires preconditioners for two subproblems. Whereas these subproblems can each be effectively preconditioned by multigrid for the diffusion inverse problem, one of these subproblems—the subproblem with $B^*B + \rho A^*A$ as the coefficient operator—is not amenable to multigrid preconditioning in the wave inverse problem. We use Robin-Robin domain decomposition to develop a preconditioner for this subproblem.

We partition the domain into two subdomains: one small subdomain near the surface, which contains the observations, and one large subomain below the surface, which does not. The diagonal block of the operator associated with the observations subdomain is small and can be solved efficiently using hierarchical matrix methods. In

Table 1.1: Preview of results from Chapter 9. Number of iterations required to solve a linear system with $B^*B + \rho A^*A$ as the coefficient operator, for a variety of angular frequencies $\omega = 10, 20, \ldots, 100$ (Column 1) and relative error tolerances $\tau = 10^{-1}, 10^{-2}, \ldots, 10^{-6}$. We indicate our wave domain decomposition method with 'DDWAVE', and indicate ρA^*A as a preconditioner by 'AA'. Columns 3-8 show the number of Krylov iterations required to solve the linear system using our method to the desired tolerance. Columns 9-14 show the number of Krylov iterations for solving the linear system using ρA^*A as the preconditioner. Dashed entries ('—'), indicate that the method did not converge to the desired tolerance.

ω		DDWAVE						AA					
	r	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
10	5	3	6	7	9	10	12	118	119	119	119	119	120
20	8	3	5	7	9	11	13	360	429	435	438	439	439
30	12	3	5	8	10	13	15	284	581	759	956	1000	—
40	13	4	6	8	10	12	15	266	435	765	—	—	—
50	16	3	6	8	10	12	14	269	503	—	—	—	—
60	18	3	6	8	11	14	15	235	454	882	—	—	—
70	20	3	6	8	11	14	16	211	414	987	—	—	—
80	21	4	6	9	12	14	16	232	453	768	—	—	—
90	23	4	6	9	13	15	17	209	377	705	—	—	—
100	25	3	6	9	12	15	17	200	384	702	—	—	—

the diagonal block of the operator associated with the large subdomain, the forward and adjoint variables are partially decoupled, so linear systems with this block as the coefficient operator can be solved by performing two wave solves in sequence. Coupling these subdomains to solve the overall system requires approximating an impedance-to-impedance map at the interface between the subdomains. We do this with the product-convolution scheme from Chapter 8.

Numerical examples demonstrate that the preconditioner is frequency-scalable and penalty parameter scalable. As the frequency of waves used in the inversion increases, or as the penalty parameter ρ decreases, the number of Krylov iterations to solve a linear system with $B^*B + \rho A^*A$ as the coefficient operator remains constant. A preview of our numerical results is shown in Table 1.1.

Chapter 10 (Conclusion) We conclude the dissertation by reviewing the preconditioners presented and suggesting directions for future research.

1.6.3 Appendix summary

Appendix A (Bayesian Solution Methods) Although we solve inverse problems in the deterministic framework here, Hessian preconditioners are also relevant to the Bayesian framework. Solving the inverse problem in the Bayesian framework requires drawing samples from an implicitly defined, high-dimensional, often non-Gaussian, posterior probability distribution. The Hessian is a local approximation to the inverse covariance of the posterior, and is therefore central to state-of-the-art sampling methods. The Hessian operations that these sampling methods require can be significantly sped up if a good Hessian preconditioner is available.

Appendix B (Dimension dependence of sparse-direct methods) Sparse-direct factorization of the KKT matrix provides a reliable alternative method for solving inverse problems. However, sparse-direct methods can only be used if the resulting factor matrices can fit in memory, and if the time required to perform the factorization is not too long. In this appendix, we discuss how the spacetime dimension of the problem affects the performance of sparse-direct methods. Sparse-direct methods perform well in 1 spacetime dimension, acceptably in 2 spacetime dimensions, and poorly in 3 or more spacetime dimensions.

Appendix C (Additional Algorithms) This appendix provides a more detailed description of some algorithms used in Chapter 9.

Appendix D (Additional Proofs) This appendix provides proofs of several results that were stated but not proven in the body of the dissertation.

Appendix E (Additional Numerical Results) Although the primary purpose of the product-convolution scheme in Chapter 8 is Hessian preconditioning, the scheme is also useful for approximating operators that arise in other situations, particularly high-rank Schur complements. In this appendix we present numerical results showing that the product-convolution scheme improves upon state-of-the-art methods for approximating other, non-Hessian, operators.

Chapter 2

Inverse Problem Frameworks

Because of ill-posedness, inverse problems cannot be "solved" in the classical sense of uniquely determining the parameter from the observations (Section 2.1). But ill-posed inverse problems occur widely and are important. To make progress we must expand our conception of what it means to "solve" an inverse problem. Presently, the two primary frameworks for addressing ill-posedness are the deterministic framework (Section 2.2), and the Bayesian framework (Section 2.3). The deterministic framework applies Occam's razor to find the solution of the inverse problem as a single estimate for q that solves a regularized optimization problem. The Bayesian framework assumes the existence of a prior probability distribution for q, then applies Bayes' theorem to find the solution as a probability distribution over all possible parameters, q. The two frameworks are closely connected (Section 2.4). After identifying operators in the deterministic framework with corresponding operators in the Bayesian framework, the objective function in the deterministic optimization problem translates to the negative log posterior in the Bayesian framework. The solution to the deterministic optimization problem is the maximum a-posteriori probability (MAP) point in the Bayesian framework. In the deterministic framework, the Hessian characterizes the directional curvatures of a local quadratic approximation to the objective function, and in the Bayesian framework the inverse of the Hessian is the covariance operator for a local Gaussian approximation to the posterior (Section 2.6).

Because the parameter is an infinite-dimensional field, it is difficult to construct a rigorously justified regularization function for the deterministic framework or prior for the Bayesian framework. It is widespread practice, therefore, to use heuristics to choose the regularization or prior (Section 2.7). The most important heuristic is avoidance of over- and under-regularization (Section 2.7.1), which leads naturally to

smoothing priors/regularization, where the regularization operator is a differential operator (Section 2.7.2).

2.1 Ill-posedness

Given a candidate parameter q, one can generate predicted, noise-free observations by solving (1.1) for u, then computing Bu. We call this process the *parameter-to-observable map*, which we denote by $\mathcal{G}$. That is,

$$\mathcal{G}(q) := Bu(q),$$

where $u(q)$ denotes the solution of (1.1) as a function of q. Notice that

$$y = \mathcal{G}(q) + \zeta. \tag{2.1}$$

The classical approach to find q given y would be to set $\zeta = 0$ (ignore noise) and solve (2.1) for q. However, for this equation to be solvable, $\mathcal{G}$ must be injective and surjective, and for the solution to be stable with respect to noise, $\mathcal{G}^{-1}$ must be continuous. If one or more of these requirements fails, the inverse problem is ill-posed in the classical (Hadamard) sense:

(Type 1 ill-posedness) $\mathcal{G}$ is not injective.

(Type 2 ill-posedness) $\mathcal{G}$ is not surjective.

(Type 3 ill-posedness) $\mathcal{G}^{-1}$, where it exists, is not continuous.

Distributed parameter PDE-constrained inverse problems usually exhibit Type 1 and Type 3 ill-posedness, and sometimes exhibit Type 2 ill-posedness.

The meaning of Type 1 ill-posedness is that multiple parameters may generate exactly the same predicted observations. Degree-of-freedom counting shows that this is inevitable. Whereas parameter fields are functions on a continuum, and are therefore infinite-dimensional objects, the amount of data that can be gathered and stored is finite. Type 2 ill-posedness arises in situations where there exists a point in the observation space that does not correspond to any parameter. If the inverse problem

exhibits Type 2 ill-posedness then noise or model inadequacy may cause y to be outside of Image($\mathcal{G}$). In Type 3 ill-posedness, substantially different parameters lead to predicted observations that are arbitrarily close: for any $\epsilon > 0$, there exist q_1 and q_2 such that $\|q_1 - q_2\| \geq 1$ and $\|\mathcal{G}(q_1) - \mathcal{G}(q_2)\| < \epsilon$. If ϵ is equal to or smaller than the noise level, the data cannot be used to distinguish q_1 from q_2.

2.1.1 Ill-posedness examples

In the wave inverse problem, wave barriers (e.g., salt domes) block waves, creating "shadowed regions" (See Figure 2.1). Waves either do not pass through shadowed regions, or pass through these regions minimally. Thus, changing the parameter in shadowed regions would not affect the observations. In other words, $\mathcal{G}$ has a null-space (Type 1 ill-posedness) or near null-space (Type 3 ill-posedness) that consists of functions supported in shadowed regions.

In the advection inverse problem, we can only infer the initial contaminant concentration for portions of the contaminant plume that flow through the sensor locations while the sensors are collecting data (see Figure 2.2). We cannot infer much about the initial contaminant concentration in regions that are downwind of the sensors, because only a small amount of the downwind contaminant will pass through the sensors by diffusing upwind. Functions supported downwind of the sensors reside in the null-space or near null-space of $\mathcal{G}$, causing Type 1 or Type 3 ill-posedness.

Diffusion operators average out oscillatory components of functions that they act on, blurring small features of those functions (imagine a drop of colored dye diffusing into a cup of water). In the diffusion inverse problem, the PDE solve, A^{-1}, within $\mathcal{G}$ therefore diminishes the impact of highly oscillatory features of the parameter on the observations. This creates a near null-space for $\mathcal{G}$ consisting of highly oscillatory functions, causing Type 3 ill-posedness. We can only infer small features of the parameter in a region if we have many observations in that region, and if the noise is small. Having many observations causes an amplifying effect of B to counteract the diminishing effect of A^{-1} within $\mathcal{G}$. Having small noise decreases the threshold, ϵ, at which different observations become functionally indistinguishable.

Diffusion also causes ill-posedness in the wave and advection inverse problems.

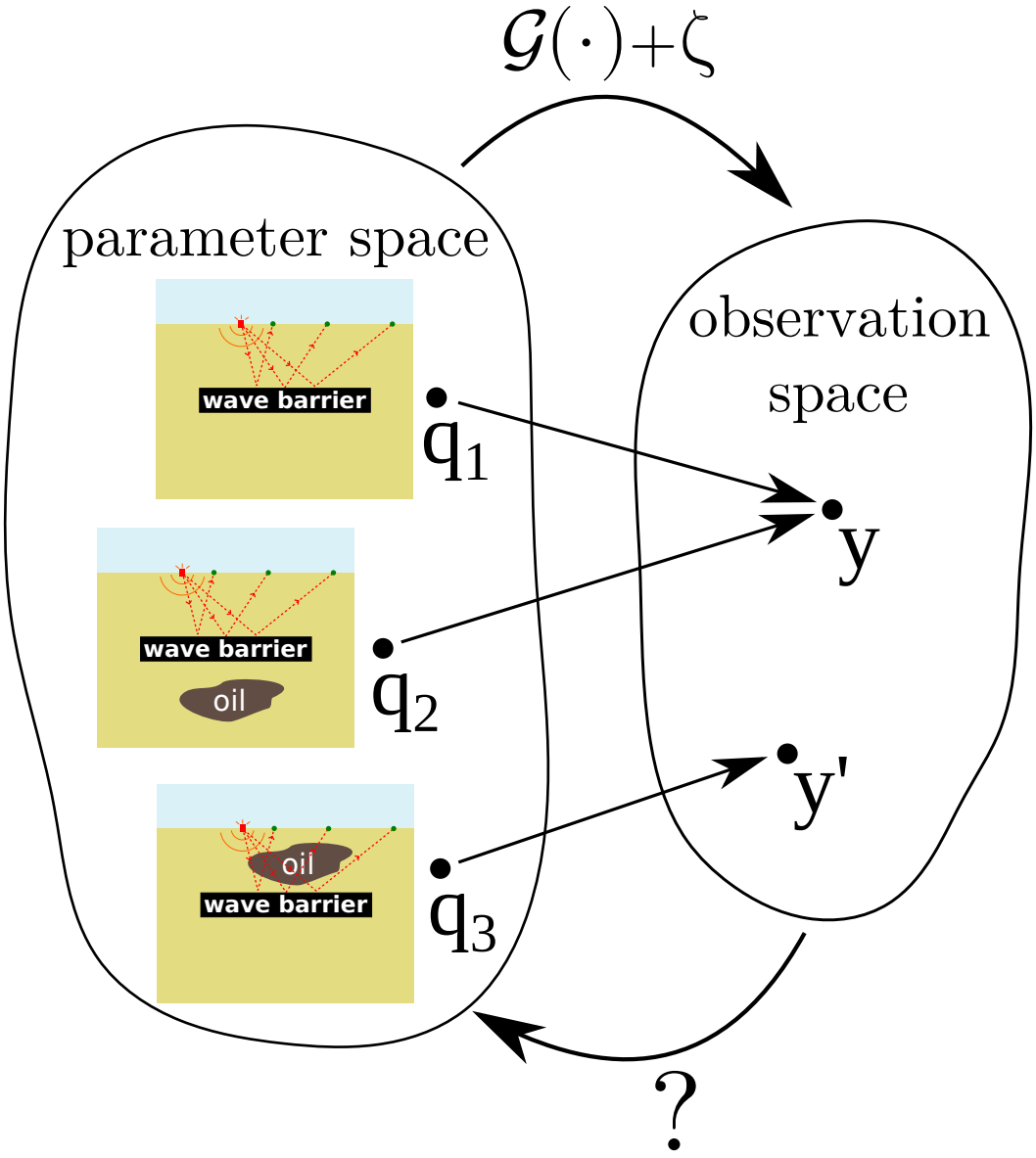


Figure 2.1: Is there oil contained within the rock behind a wave barrier or not? If the observed waves do not pass through a "shadowed" region of the domain, then we would get nearly the same observations regardless of what is in that region, so we cannot use the observations to determine whether or not oil is there.

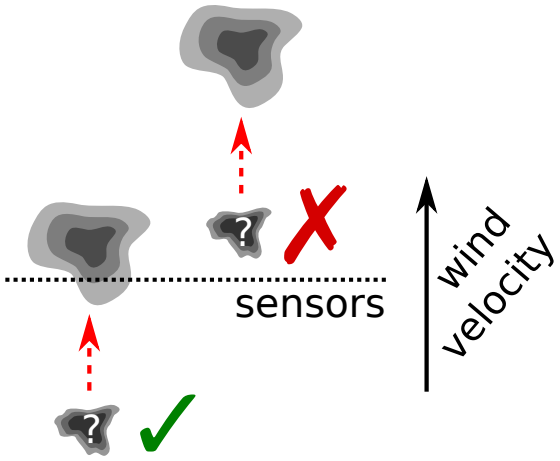


Figure 2.2: The initial condition of the contaminant plume cannot be inferred for regions of the domain that are downwind of the sensors.

Diffusion enters into the wave inverse problem since the Helmholtz operator acts like a diffusion operator on sub-wavelength-scales. The higher the wave frequency, the more oscillatory the parameter can be before diffusion becomes a significant factor. In the advection inverse problem, as the contaminant plume advects it also diffuses. The farther away the contaminant plume is from the sensors when it is released, the more it diffuses before observation, and the less we can infer about it. The Peclet number determines the ratio of advection to diffusion. As the Peclet number increases, the contaminant plume diffuses less before it is observed by the sensors, which allows smaller scale features of the initial condition to be distinguished from one another.

2.2 Deterministic framework: Occam's razor

If multiple q yield predicted observations that differ from y to within the noise, which q should we choose? **Occam's razor** suggests a resolution:

> *Choose the simplest q consistent with y.*

To use Occam's razor we need to define:

- What it means for q to be "simple".
- What it means for q to be "consistent with y."

Rather than maximize simplicity, it will be more convenient to instead minimize complexity. We assume a complexity function,

$$\text{complexity}(q) := \frac{1}{2}\left\|q - \overline{q}_0\right\|_{R_0}^2, \tag{2.2}$$

for some norm $\|\cdot\|_{R_0}$ induced by a positive self-adjoint operator R_0, and some vector $\overline{q}_0$. For example, if R_0 is the identity operator and $\overline{q}_0 = 0$, then (2.2) says that q is simpler than p if q has a smaller norm than p. If R_0 is a power of the Laplacian operator and $\overline{q}_0 = 0$, then (2.2) says that q is simpler than p if q is smoother than p.

To define consistency with the data, we assume that the noise is bounded:

$$\|\zeta\|_Y \leq \delta \tag{2.3}$$

for some $\delta > 0$ and norm $\|\cdot\|_Y$. A candidate parameter q, and observations vector y, are consistent if equality holds in (2.1) for some ζ satisfying (2.3). Hence the set of q's consistent with y are those such that

$$\|\mathcal{G}(q) - y\|_Y \leq \delta. \tag{2.4}$$

After quantifying complexity with (2.2) and consistency with (2.4), Occam's razor becomes a constrained optimization problem. The simplest q consistent with the data solves:

$$\begin{aligned} \min_q \quad & \frac{1}{2}\left\|q - \overline{q}_0\right\|_{R_0}^2, \\ \text{such that} \quad & \|\mathcal{G}(q) - y\|_Y \leq \delta. \end{aligned} \tag{2.5}$$

If equality holds in the constraint at the solution to (2.5) (i.e., the constraint is active), then standard optimization duality theory shows that local solutions of (2.5) are also local solutions of this regularized optimization problem:

$$\min_q \quad \frac{1}{2}\|\mathcal{G}(q) - y\|_Y^2 + \frac{\alpha}{2}\left\|q - \overline{q}_0\right\|_{R_0}^2, \tag{2.6}$$

where $1/\alpha$ is the Lagrange multiplier enforcing the consistency constraint in the form $\frac{1}{2}\|\mathcal{G}(q) - y\|^2 = \delta^2/2$. We call α the *regularization parameter*.[1]

[1] If the constraint is not active at the solution then proof by contradiction shows that the solution to the optimization problem, $q_{\min}$, satisfies $q_{\min} = \overline{q}_0$, so the problem is trivial.

If δ is known, one can solve a sequence of problems of the form (2.6) with different values of α until the solution satisfies the Morozov discrepancy principle,

$$(1-\tau)\delta \leq \|\mathcal{G}(q) - y\|_Y \leq (1+\tau)\delta, \tag{2.7}$$

for some small τ [17, 147]. But in practice, δ may be unknown, or known bounds for δ may be overly pessimistic. Additionally, R_0 is usually chosen based on heuristics and computational considerations, rather than careful reasoning about simplicity/complexity. It may thus be necessary or preferable to view α as a tunable parameter rather than a Lagrange multiplier, and to use other methods of choosing it (e.g., L-curves [119]). See [27, 28] for extensive analysis and comparison of the different methods for choosing α.

The unconstrained optimization problem, (2.6), can also be reframed in the following constrained form:

Deterministic optimization problem

$$\begin{aligned} \min_{q,u} \quad & \frac{1}{2}\|Bu - y\|_Y^2 + \frac{\alpha}{2}\|q - \overline{q}_0\|_{R_0}^2 \\ \text{such that} \quad & A(q)u = f(q). \end{aligned} \tag{2.8}$$

This follows from (2.6) after explicitly enforcing the state equation as a constraint, rather than eliminating it implicitly in the objective function. We call (2.6) the reduced space problem and (2.8) the full space problem.

2.3 Bayesian framework

Due to noise and ill-posedness, many different q could explain y, each with varying probabilities. **Bayes' rule**,

$$\pi(q|y) = \frac{\pi(y|q)\pi(q)}{\pi(y)}, \tag{2.9}$$

establishes a consistent framework for assigning these probabilities. In Bayes' rule,

- $\pi(q|y)$ is the *posterior* probability density of q, given y (from a probabilistic point of view, this is the "solution" to the inverse problem).
- $\pi(y|q)$ is the *likelihood* of observing y, given q.

- $\pi(q)$ is our *prior* probability density for q, before any observations have been made.

- $\pi(y)$ is the probability density of y (for our purposes it is a constant, since it does not depend on q).

Let us assume that both the prior and noise have normal distributions with means $\overline{q}_0$ and 0, respectively, and covariances C_{prior} and C_{noise}, respectively. That is,

$$\pi(q) \propto \exp\left(-\frac{1}{2}\left\|q-\overline{q}_0\right\|^2_{C^{-1}_{\text{prior}}}\right) \tag{2.10}$$

and

$$\pi(\zeta) \propto \exp\left(-\frac{1}{2}\left\|\zeta\right\|^2_{C^{-1}_{\text{noise}}}\right). \tag{2.11}$$

Because $y = \mathcal{G}(q) + \zeta$, (2.11) implies

$$\pi(y|q) \propto \exp\left(-\frac{1}{2}\left\|y-\mathcal{G}(q)\right\|^2_{C^{-1}_{\text{noise}}}\right). \tag{2.12}$$

Substituting (2.12) and (2.10) into (2.9) and simplifying yields the following posterior distribution for q:

Posterior distribution

$$\pi(q|y) \propto \exp\left(-\frac{1}{2}\left\|\mathcal{G}(q)-y\right\|^2_{C^{-1}_{\text{noise}}} - \frac{1}{2}\left\|q-\overline{q}_0\right\|^2_{C^{-1}_{\text{prior}}}\right). \tag{2.13}$$

2.4 Connections between the frameworks

Let

$$\mathcal{J}(q) := \frac{1}{2}\left\|\mathcal{G}(q)-y\right\|^2_Y + \frac{1}{2}\left\|q-\overline{q}_0\right\|^2_R.$$

If we set $R := \alpha R_0$, then the reduced space optimization problem in the deterministic framework, (2.6), takes the form

$$\min_q \ \mathcal{J}(q).$$

Similarly, if we set $Y := C^{-1}_{noise}$ and $R := C^{-1}_{\text{prior}}$, then the posterior probability distribution in the Bayesian framework, (2.13), is

$$\pi(q|y) \propto \exp\left(-\mathcal{J}(q)\right).$$

If $Y = C_{\text{noise}}^{-1}$ and $R = C_{\text{prior}}^{-1}$, then the objective function for the deterministic problem is the negative log posterior in the Bayesian problem (up to a constant), and the maximum a-posteriori probability point (MAP point) in the Bayesian problem is the solution to the deterministic problem.

2.5 Discussion of the quadratic/Gaussian assumptions

In the deterministic framework we assumed that the simplicity function and noise norm are quadratic. Correspondingly, in the Bayesian framework we assumed that the prior and the noise distributions are Gaussian. These assumptions are not as restrictive as they might seem. Even with these assumptions, the objective function/posterior can be non-quadratic/non-Gaussian due to nonlinearity of the parameter-to-observable map, $\mathcal{G}$. More importantly, even if these assumptions are violated, the Hessian preconditioning tools we develop will still be relevant, because non-quadratic/non-Gaussian functions look quadratic/Gaussian locally, and the best numerical methods for solving the inverse problem are based on making successive local approximations.

2.6 Local quadratic/Gaussian models

Because of ill-posedness, perturbations to the parameter in some directions may have little impact on $\mathcal{J}$, while perturbations in other directions may have substantial impact on $\mathcal{J}$. As a result, $\mathcal{J}$ will typically have widely varying directional scalings—in some directions $\mathcal{J}$ will be sharply curved, and in other directions $\mathcal{J}$ will be flat. Quadratic models of $\mathcal{J}$ based on truncated Taylor series allow us to characterize these directional scalings locally. These quadratic models also form the basis for the best numerical methods for solving the inverse problem, as we will discuss in Chapter 3 and Appendix A.

Let q_i be a fixed parameter and let p be a small perturbation to q_i. Truncating a Taylor series of $\mathcal{J}$ about q_i after the quadratic term yields the approximation

$$\mathcal{J}(q_i + p) \approx \mathcal{J}(q_i) + g^* p + \frac{1}{2} p^* H p,$$

where

$$g = \left(\frac{d\mathcal{J}}{dq}\right)^*$$

is the gradient of $\mathcal{J}$ (Riesz representor of $\frac{d\mathcal{J}}{dq}$) and

$$H \simeq \frac{d^2\mathcal{J}}{dq^2}$$

is the Hessian of $\mathcal{J}$ (the linear operator associated with the bilinear form $\frac{d^2\mathcal{J}}{dq^2}$), and these derivatives are evaluated at q_i. Hence, the eigenvalues and eigenvectors of H locally characterize the directional scalings of $\mathcal{J}$. If an eigenvalue of H is large, then locally $\mathcal{J}$ is sharply curved in the associated eigenvector direction. If an eigenvalue of H is small, then locally $\mathcal{J}$ is flat in the associated eigenvector direction.

Alternatively, we may form a slightly different local quadratic model of $\mathcal{J}$ by truncating the Taylor series for $\mathcal{G}$ after the linear term, and replacing $\mathcal{G}$ with this approximation within $\mathcal{J}$. We have

$$\mathcal{G}(q_i + p) \approx \mathcal{G}(q_k) + Gp,$$

where

$$G := \frac{d\mathcal{G}}{dq}$$

is the Jacobian of $\mathcal{G}$, evaluated at q_i. Replacing $\mathcal{G}$ with this approximation within $\mathcal{J}$ yields an approximate objective function,

$$\begin{aligned}\mathcal{J}(q_i + p) \approx& \frac{1}{2}\left\|Gp - (y - \mathcal{G}(q_i))\right\|_Y^2 + \frac{1}{2}\left\|p - (\bar{q}_0 - q_i)\right\|_R^2 \\ =& \mathcal{J}(q_i) + g^* p + \frac{1}{2} p^* H^{\mathrm{gn}} p,\end{aligned} \tag{2.14}$$

where we define the Gauss-Newton Hessian:

$$H^{\mathrm{gn}} := G^* Y G + R, \tag{2.15}$$

or $H^{\mathrm{gn}} = H_d^{\mathrm{gn}} + R$, where we define the data misfit Gauss-Newton Hessian $H_d^{\mathrm{gn}} := G^* Y G$. Hence, the eigenstructure of H^{gn} also characterizes the local directional scalings of $\mathcal{J}$, in the same manner as the eigenstructure of H. Although the quadratic approximation involving H^{gn} is asymptotically less accurate than that of H, H^{gn} is

positive definite while H may be indefinite away from the optimal point. But generally H and H^{gn} will be similar; we will discuss connections between these operators in Chapter 5.

Notice that (2.14) is the objective function that would arise if the deterministic framework from Section 2.2 were applied to a hypothetical inverse problem in which the parameter-to-observable map has been replaced by a local linearization. Intuitively, in this local approximation one assumes that the true parameter is a small perturbation from a known reference parameter, and seeks to determine the perturbation. If the true parameter is sufficiently close to the true parameter, and if high accuracies are not required, then this local approximation may replace the original model (see, e.g., [144] for an example of this in ocean acoustic inversion). If higher accuracies are needed, a sequence of successively better local models may be generated iteratively. One solves for the optimal perturbation based on the current reference model, then adds this perturbation to the reference model, creating a better reference model. This process repeats until convergence. We discuss this process in greater detail in Section 3.2.

Since $\mathcal{J}$ is the negative log posterior in the Bayesian framework, these local quadratic approximations of $\mathcal{J}$ yield directional-scaling-aware local Gaussian approximations of the posterior. We have

$$\begin{aligned}\pi(q|y) &\propto \exp\left(-\frac{1}{2}\left\|\mathcal{G}(q)-y\right\|_Y^2+\frac{1}{2}\left\|q-\overline{q}_0\right\|_R^2\right)\\ &\approx \exp(-g^*(q-q_i)-\frac{1}{2}(q-q_i)^*\widetilde{H}(q-q_i))\\ &\propto \exp\left(-\frac{1}{2}(q-\overline{q})^*\widetilde{H}(q-\overline{q})\right),\end{aligned}$$

where q_i is the point at which we are making the approximation, $\widetilde{H}$ is either H or H^{gn}, and

$$\overline{q} := q_i - \widetilde{H}^{-1}g.$$

As a result, near q_k the posterior is well-approximated by the Gaussian model

$$\widetilde{\pi}(q|y) \propto \exp\left(-\frac{1}{2}(q-\overline{q})^*\widetilde{H}(q-\overline{q})\right). \tag{2.16}$$

This approximation is only well-defined if $\widetilde{H}$ is positive definite. Otherwise, the right hand side of (2.16) is not normalizable. The Gauss-Newton Hessian, H^{gn}, is always positive definite, while the true Hessian, H, is positive definite near the MAP point, but may be indefinite far away from the MAP point. If $\widetilde{H} = H^{\text{gn}}$, then $\widetilde{\pi}(q|y)$ is the exact posterior for the approximate inverse problem in which the parameter-to-observable map is replaced by its linearization.

2.7 Choice of regularization/prior in practice

To use the deterministic framework, one must choose a regularization operator, and to use the Bayesian framework, one must choose a prior. In the deterministic framework, the regularization is supposed to represent one's notion of simplicity, and in the Bayesian framework, the prior is supposed to encode one's prior knowledge. Unfortunately, in practice people seldom take these perspectives seriously because deriving rigorously justified simplicity/complexity functions or prior probability distributions on function spaces is too hard. Perhaps a more rigorous framework will someday be developed for choosing the regularization/prior for distributed parameter inverse problems—this is an area of open research. Currently the following practical perspectives predominate:

- **Practical deterministic perspective.** The purpose of the regularization is to create curvature in the objective function of the deterministic optimization problem in directions that would otherwise be flat, thereby ensuring that the optimization problem has a unique solution that is stable with respect to noise or other small perturbations to the data. At the same time, the regularization should affect the objective function as little as possible in directions that are not flat, since regularization to ensure a stable solution is not needed in those directions.

- **Practical Bayesian perspective.** The purpose of the prior is to make the posterior a well-defined probability distribution by supplying information about the parameter that is not present in the likelihood [181]. Without a sufficiently

strong prior, the posterior would be an improper distribution in the uninformed directions.

The regularization or prior is then judged based on how well it works in practice—does it lead to a "good" reconstruction of the parameter? The quality of the reconstructed parameter may be evaluated, for example, by the subjective judgement of an expert, or by comparison with the true parameter for test problems in which the true parameter is known. Currently, smoothing regularization/priors (to be discussed in Section 2.7.2) have the most justification for distributed parameter inverse problems, and yield good results in most circumstances.

2.7.1 Over- and under-regularization

Both over- and under-regularization (regularization in which eigenvalues of R are too large or too small, respectively, for the problem at hand) lead to bad parameter reconstructions.[2] To avoid over- and under-regularization, the operators H_d and R in the Hessian must compete with each other—H_d should act strongly on vectors that R acts weakly on, and vice versa.

Consider the reduced space deterministic optimization problem, (2.6), in the special case where $Y = I$, $\overline{q}_0 = 0$, and the parameter-to-observable map is linear ($\mathcal{G}(q) := Gq$):

$$\min_q \quad \frac{1}{2} \|Gq - y\|^2 + \frac{1}{2} \|q\|_R^2 . \tag{2.17}$$

Let q_{true} denote the unknown true parameter that generated the observations:

$$y = Gq_{\text{true}} + \zeta. \tag{2.18}$$

The solution, q, to (2.17) is the solution to the normal equations,

$$(H_d + R)q = G^* y, \tag{2.19}$$

where $H_d = G^*G$. Substituting (2.18) into (2.19) and then subtracting the result from q_{true}, we see that the error in the reconstructed parameter, found by solving

[2] This section contains content from [8] (Nick Alger, Umberto Villa, Tan Bui-Thanh, and Omar Ghattas. A data scalable augmented Lagrangian KKT preconditioner for large-scale inverse problems. SIAM Journal on Scientific Computing, 39(5):A2365–A2393, 2017.)

(2.17), takes the form

$$q_{\text{true}} - q = e_\zeta + e_q,$$

which consists of a term

$$e_\zeta := -\left(G^*G + R\right)^{-1} G^*\zeta \tag{2.20}$$

which depends on the noise, and a term

$$e_q := \left(I - \left(G^*G + R\right)^{-1} G^*G\right) q_{\text{true}} \tag{2.21}$$

that does not. From the form of equations (2.20) and (2.21), a trade-off is evident: strengthening the regularization tends to reduce e_ζ at the expense of increasing e_q, and weakening the regularization tends to reduce e_q at the expense of increasing e_ζ. To achieve a good reconstruction of the parameter, it is desirable for both of these terms to be as small in magnitude as possible.

To investigate this trade-off in more detail, we express the errors in the bases of generalized singular vectors associated with the generalized singular value decomposition [188] of G with respect to R. From the generalized singular value decomposition, there exists a unitary operator U and a normalized but non-unitary basis Φ such that

$$\begin{cases} U^*G\Phi = \operatorname{diag}(g_k) \\ \Phi^*R\Phi = \operatorname{diag}(r_k), \end{cases} \tag{2.22}$$

where the vectors g_k are the generalized singular vectors of G and the scalars r_k are the generalized singular values of R associated with this generalized singular value problem.[3] In the bases of Φ and U, we can formulate expressions for the errors in the reconstruction in a per-component manner. Substituting the singular value decomposition factors from (2.22) into the error expressions from (2.20) and (2.21), and then performing some algebraic manipulations, yields

$$e_\zeta = -\Phi \operatorname{diag}\left(\frac{g_k}{g_k^2 + r_k}\right) U^*\zeta, \tag{2.23}$$

$$e_q = \Phi \operatorname{diag}\left(\frac{r_k}{g_k^2 + r_k}\right) \Phi^{-1} q_{\text{true}}. \tag{2.24}$$

[3]The idea here is to compute the singular value decomposition of G with respect to the R-inner product, then rescale the right singular vectors so that they have unit length in the standard (non-R) norm.

From (2.23), we see that the regularization should not be weak (small r_k) in directions, ϕ_k, to which the observations are insensitive (small g_k). Otherwise, the noise associated with components of the observations in those directions will be highly amplified, leading to large errors. In such a scenario we say that the problem is *under-regularized*.

But (2.24) also shows that strong regularization can lead to large errors. Some degree of error in e_q is to be expected—there is no hope to reconstruct the component of the parameter in directions for which the corresponding data are lacking, or are dominated by noise. However, if g_k is large then the observations are highly sensitive to changes in the parameter in direction ϕ_k, so it is likely that the observations associated with direction ϕ_k contain more signal than noise. That is, when g_k is large, it is likely that the component of the parameter q_{true} in direction ϕ_k can, in principle, be inferred from the data. Hence, if the regularization is strong (r_k is large) in directions for which the parameter-to-observable map is also strong (g_k is large), the reconstruction will contain substantial *unnecessary* error due to the regularization. In this scenario, we say that the problem is *over-regularized*. To avoid both under- and over-regularization, the regularization should be strong in directions where the parameter-to-observable map is weak, and weak in directions where the parameter-to-observable map is strong.

2.7.2 Motivation for smoothing regularization/priors

Smoothing regularization is regularization in which the regularization operator, R, is a differential operator, such as a power of the Laplacian (see, for example, [68]). In the Bayesian framework, the analog of smoothing regularization is a smoothing prior, in which the prior covariance operator, R^{-1}, acts like the inverse of a differential operator. With smoothing regularization/prior, the eigenvalues of R increase as the eigenvectors become more oscillatory, so smooth parameters are preferred over rough parameters. We use smoothing regularization/priors, and design preconditioners with this type of regularization/prior in mind. The main downside of smoothing regularization/priors is that they tend to smooth out sharp jumps in the parameter field, such as interfaces between different types of rocks. However, the upsides are

numerous:

Avoiding over- and under-regularization. In order to neither over- nor under-regularize, H_d and R should oppose each other (see Section 2.7.1). In Chapter 5, we will see that H_d contains A^{-1} and A^{-*}, and therefore acts like the inverse of a differential operator. Thus R should act like a differential operator.

Existence of a continuum limit. Although the inverse problem becomes finite-dimensional when discretized, we must ensure that the infinite-dimensional inverse problem is well posed in order to guarantee that the finite-dimensional problem converges to a well-defined limit as the mesh is refined.

Constructing prior distributions on infinite-dimensional spaces is challenging. In infinite-dimensions, priors that appear intuitively reasonable may fail to be normalizable, leading to a posterior that is not a probability distribution. Priors with covariance operators given by sufficiently strong powers of the inverse Laplacian have been proven to yield existence and uniqueness of posterior probability distributions in the infinite-dimensional limit [181].

Correlation vs. distance between points. Smoothing priors follow from the assumption that nearby points in the parameter field are more strongly correlated than far away points. This is reasonable: rock in the subsurface is more likely to be similar to nearby rock than far away rock.

Many popular covariance models in geostatistics fall in the Matérn class, a class of covariance models that characterize the decay rate of the covariance between two points as they grow farther apart. Gaussian random fields using covariance models in the Matérn class are equivalent to inverse Laplacian-type priors [131]. The power that the inverse Laplacian is raised to depends on the parameters in the Matérn class.

Chapter 3

Deterministic Solution Methods

Solving the inverse problem in the deterministic framework amounts to solving a large-scale nonlinear optimization problem of the form (2.6) or (2.8). Two difficulties make these optimization problems hard to solve:

- Non-convexity.
- Ill-conditioning.

Non-convexity causes optimization algorithms to converge to local minima rather than global minima, while ill-conditioning slows the local convergence. In theory, non-convexity can also slow convergence, and it is possible to construct optimization problems where non-convexity makes convergence slow (e.g., imagine an objective function that looks like a maze). In practice, convergence speed is overwhelmingly governed by ill-conditioning, as long as appropriate globalization safeguards are used. In this dissertation our focus is ill-conditioning, which we address by building preconditioners for the linear systems that must be solved at each iteration of second-order optimization algorithms. Although we do not address difficulties stemming from non-convexity in this dissertation, our preconditioners are also relevant to that task because they can be used to speed up homotopy continuation methods, e.g., [196].

Optimization algorithms may be classified by how many derivatives of the objective function they use. Ill-conditioning prevents first-order algorithms (algorithms that use only first derivative/gradient information) from being data-scalable. This is because inverse problems with highly informative data tend to be ill-conditioned, and first order algorithms converge slowly for ill-conditioned problems (Section 3.1).

Zeroth-order (derivative-free) algorithms usually converge extremely slowly for high-dimensional problems. Methods involving third- or higher-order derivatives are

rarely used, though optimization using higher-order derivatives is an under-developed field worthy of further research.

Second-order (Newton-type) algorithms (Section 3.2) require low numbers of iterations to converge regardless of the conditioning. Of course, as the problem becomes more ill-conditioned, the linear systems that must be solved at each iteration become more ill-conditioned, and therefore harder to solve. This shifts the computational burden from the nonlinear optimizer, where little can be done to address ill-conditioning,[1] to the linear algebra solver, where we can address ill-conditioning directly by building a preconditioner.

For this reason, we believe that the best algorithms for solving (2.6) or (2.8) are Newton-Krylov-type algorithms [75, 157], where the optimization algorithm is a second-order Newton-type method, and the linear systems that must be solved at each outer Newton iteration are solved with a preconditioned inner Krylov iteration. Using Newton-Krylov methods requires first and second derivative information: one must be able to compute the gradient, and apply the Hessian to vectors. We can perform both of these tasks efficiently using adjoint-based techniques (see Section 5.2). Constructing and factorizing the (dense) Hessian is not required.

3.1 Ill-conditioning hobbles first-order methods

Gradient descent minimizes $\mathcal{J}$ by iteratively moving downward in the steepest direction from the current point. The convergence rate of gradient descent is linear, and depends on the condition number of the Hessian. Suppose we choose the gradient descent step length by exact line-search. By "exact line-search," we mean choosing the step length that minimizes the objective function along the ray originating at the current point and traveling in the negative gradient direction. Then near the solution to the optimization problem, $q_{\min}$, we have

$$\mathcal{J}(q_i) - \mathcal{J}(q_{\min}) \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^{2i} \left(\mathcal{J}(q_0) - \mathcal{J}(q_{\min})\right), \tag{3.1}$$

where q_i is the i^{th} iterate, q_0 is the initial guess, and $\kappa := \lambda_{\max}(H)/\lambda_{\min}(H)$ is the condition number of the Hessian at $q_{\min}$. For details on gradient descent and line-

[1] In theory, one can do nonlinear preconditioning.

search, we recommend [157]. For large-scale big-data inverse problems, convergence rate (3.1) is unacceptable since κ will be large. In the problems we face $\kappa = 10^6$ is not uncommon.

More advanced first-order algorithms use a linear combination of gradients from multiple previous steps as the descent direction. Algorithms in this class include:

- Nesterov methods [152]: a class of algorithms that include "momentum" in the descent algorithm (imagine a ball rolling down a valley).
- Nonlinear conjugate gradient [116]: a nonlinear version of the Krylov conjugate gradient method which attempts to find the best solution to the problem in a sequence of successively larger subspaces spanned by previous gradients.
- BFGS and L-BFGS [132]: quasi-Newton algorithms that build and continually update Hessian approximations based on how the gradient changes from one iteration to the next.[2]

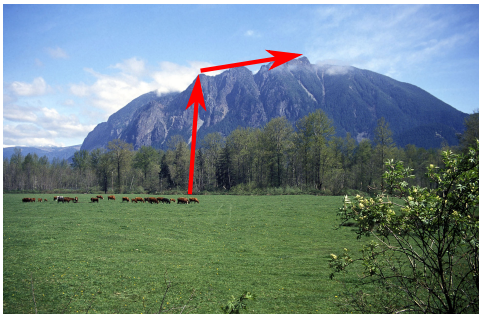
By including information from previous gradients, these more advanced algorithms partially account for how the gradient is changing, leading to faster convergence. However, these more advanced algorithms still perform poorly on ill-conditioned problems (albeit, not as poorly as gradient descent).

3.1.1 Climbing an N-dimensional mountain: "hypercube paths"

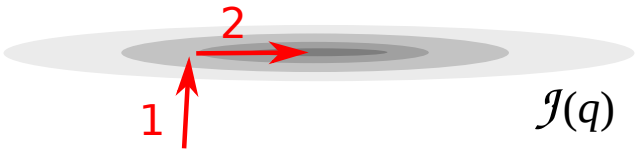
Intuitively, since the gradient always points in the steepest direction, any method based on linear combinations of previously computed gradients cannot effectively explore a given direction in parameter space until the method has eliminated all other directions in parameter space for which the objective function is steeper.[3] Imagine a mountaineer climbing up an oblong mountain using the method of steepest ascent.

[2] Although BFGS and L-BFGS mimic Newton's method, they do not include true second derivative information and are therefore properly viewed as a first-order method. On quadratic problems with optimal line search and infinite precision arithmetic, conjugate gradient and BFGS generate the same sequence of iterates [151].

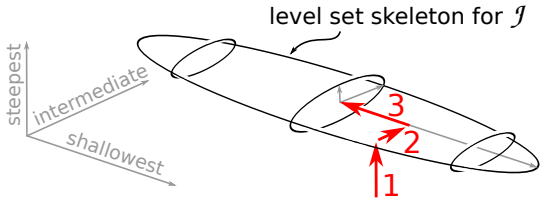
[3] This section contains content from [6] (Nick Alger. Relating condition number of Hessian to the rate of convergence. Mathematics Stack Exchange, 2018. https://math.stackexchange.com/q/2847190.).



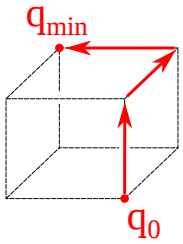
(a) Climbing Mt. Si via gradient ascent.



(b) 2-D gradient ascent path. This is what the mountain climbing path in 3.1a looks like when viewed from above the mountain.



(c) 3-D gradient ascent path.



(d) The ill-conditioned gradient path approximately follows edges on a box.

Figure 3.1: **(Hypercube path)** When climbing a mountain with the method of steepest ascent, one first climbs to a ridge, then follows the ridge to the peak. On an N-dimensional mountain, one first climbs to the $(N-1)$-dimensional "ridge" perpendicular to the steepest direction, then to the $(N-2)$-dimensional "ridge" perpendicular to both the first and second steepest directions, and so forth. The path resembles a sequence of edges on an N-dimensional box connecting one corner of the box to the diametrically opposite corner. Mountain picture by Mountains to Sound Greenway Trust, © 2004. See also: Figure 3.3.

They will not walk straight towards the peak. Rather, they willl climb up the steepest face of the mountain until they reach a ridge, then follow the ridge to the top, making a path that resembles a connected set of two edges of a rectangle (see Figure 3.1). Were it possible to climb an N-dimensional mountain, the path of steepest ascent would be to first climb to the top of the $(N-1)$-dimensional ridge perpendicular to the steepest direction, then to the top of the $(N-2)$-dimensional ridge perpendicular to the first two steepest directions, and so on, making a path that resembles a connected set of N edges on a hypercube. If N is large, this would require a lot of climbing! For our inverse problem, the unknown is a discretization of a continuum field, so N is effectively infinite. When climbing to a ridge, only minor progress will be made in the direction of future, less steep, ridges. This minor progress is what convergence bound (3.1) relies on. The more well-separated the eigenvalues of the Hessian are, the smaller this minor progress will be (see Figure 3.3).

3.1.2 Zig-zag paths vs. hypercube paths

Ill-conditioning also causes overshooting, leading to "zig-zag paths" with sharp turns (see Figure 3.2). But "zig-zagging" is a separate issue from hypercube paths. In zig-zagging, subsequent steps backtrack in directions previously explored, correcting for overshooting errors. In the mountain analogy, one step takes the mountaineer past the ridge, the next step takes them past the same ridge the other way, and so on. In contrast, in hypercube paths each step is roughly perpendicular to *all* previous steps. The issue in the hypercube path case is that there are many different perpendicular directions to explore, and one can only explore them roughly one at a time.

Many people in the optimization community erroneously conflate these issues. The standard textbook explanation of gradient descent convergence (see, e.g., Section 3.3 in [157], or Section 4.3.2.2 in [104]) goes like this:

1. A mathematical proof of convergence bound (3.1) (or similar) is presented, showing that convergence is slow for ill-conditioned problems.

2. A picture of zig-zagging resembling Figure 3.2 is presented.

The reader is left to conclude:

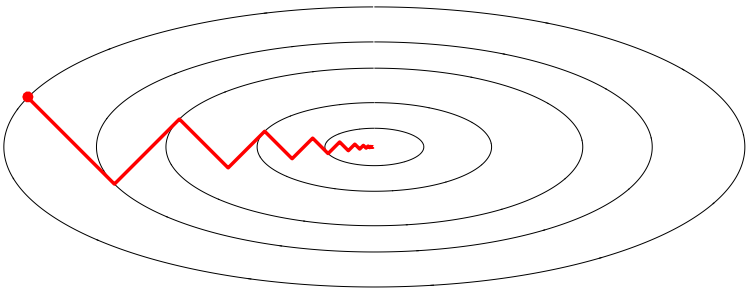


Figure 3.2: **(Zig-zag path)** Gradient descent on ill-conditioned problems continually overshoots, creating "zig-zag" paths. This is different from the hypercube path problem illustrated in Figure 3.1. Black ellipses indicate level sets of $\mathcal{J}$, and the red zig-zag path indicates the gradient descent path, where the step length at each iteration is chosen to minimize $\mathcal{J}$ for that step. The red dot is the initial guess.

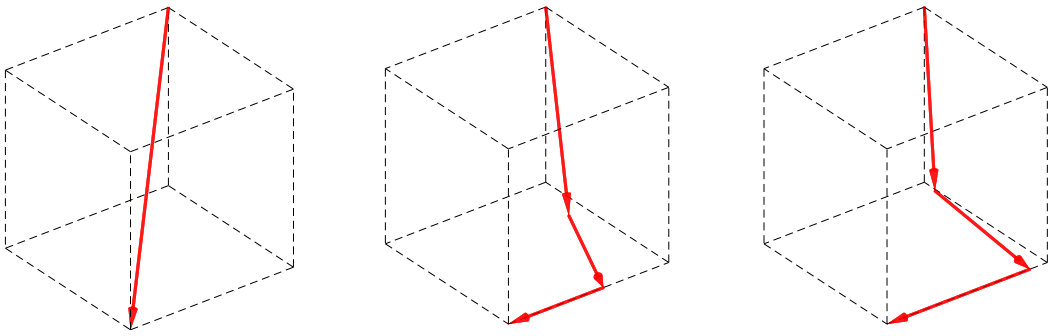
3. Zig-zagging is the fundamental cause of slow convergence. If we eliminated zig-zagging, then convergence would be fast. (this is wrong)

But 3 does not follow from 1 and 2. Even if we eliminated zig-zagging, convergence would still be slow because of the hypercube path issue. In Theorem 1 we will prove that if one hypothetically chose "ridgeline" step lengths so that there would be no zig-zagging, the convergence rate of gradient descent would actually be slightly worse than (3.1) in the realistic regime where the number of gradient descent iterations is less than N. By "ridgeline" step length, we mean the step length that eliminates the error in the largest active eigenvector's direction (resulting in a point exactly on the ridgeline of the mountain). This step length is different from the step length generated by exact line search, which will overshoot the ridge slightly. Of course, choosing ridgeline step lengths is computationally infeasible in practice. We consider it only as a thought experiment to see what would happen if there were no zig-zagging.

Consider a quadratic minimization problem of the form

$$\min_q \quad \frac{1}{2} q^* H q + b^* q, \tag{3.2}$$

where H is self-adjoint and positive-definite. Since non-quadratic problems look quadratic near their minima, analysis for the quadratic case also applies asymptotically in the non-quadratic case.



(a) $H = \mathrm{diag}(1, 1, 1)$ (b) $H = \mathrm{diag}(1, 3, 9)$ (c) $H = \mathrm{diag}(1, 10, 100)$

Figure 3.3: **(Hypercube path)** Gradient descent path with ridgeline steplengths for minimizing $\frac{1}{2}q^T Hq$, starting at $q_0 = (1, 1, 1)$ and ending at $q_3 = (0, 0, 0)$, where $H = \mathrm{diag}(\lambda_3, \lambda_2, \lambda_1)$. The more well-separated the eigenvalues of H are, the more the path resembles edges on a hypercube. See also: Figure 3.1.

Let the eigenvalues, λ_k, of H be written in descending order,

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N > 0,$$

let ϕ_k denote the corresponding eigenvectors, let $e_i := q_i - q_{\min}$ denote the error at the i^{th} iteration of a gradient-descent algorithm for solving (3.2), where $q_{\min}$ is the unknown minimizer of (3.2), and let $e_i^{(k)}$ denote the component of e_i in the direction of the k^{th} eigenvector. That is,

$$e_i = \sum_{k=1}^{N} e_i^{(k)} \phi_k.$$

Now consider one gradient descent step,

$$q_{i+1} = q_i - \beta g,$$

where $g = Hq_i + b$ is the gradient of the objective function at q_i, and $\beta > 0$ is a parameter that controls the step length. Simple algebra yields an update formula for the components of the error after the gradient step:

$$e_i^{(k)} = (1 - \beta\lambda_i) e_{i-1}^{(k)}. \tag{3.3}$$

From (3.3), we see that, to take the ridgeline step length for the i^{th} iteration—the step length that would make $e_i^{(i)} = 0$—we should set $\beta = 1/\lambda_i$. Gradient descent

with ridgeline step lengths is illustrated in Figure 3.3. With this step length, other eigenvector components of the error decrease as follows for $k > i$ in (3.3):

$$e_i^{(k)} = (1 - \lambda_k/\lambda_i) e_{i-1}^{(k)}.$$

Iterating this process, the components of the error have the following exact formula after i iterations:

$$e_i^{(k)} = \begin{cases} 0, & k \leq i, \\ \left(1 - \frac{\lambda_k}{\lambda_1}\right)\left(1 - \frac{\lambda_k}{\lambda_2}\right)\dots\left(1 - \frac{\lambda_k}{\lambda_i}\right) e_0^{(k)}, & k > i. \end{cases} \tag{3.4}$$

Theorem 1. *If the i^{th} step length is chosen such that $e_i^{(i)} = 0$ (i.e., ridgeline step length), then the error bound*

$$\mathcal{J}(q_i) - \mathcal{J}(q_{min}) \leq \left(\frac{\kappa - 1}{\kappa}\right)^{2i} \left(\mathcal{J}(q_0) - \mathcal{J}(q_{min})\right), \tag{3.5}$$

holds for gradient descent on the quadratic objective function $\mathcal{J}(q) := \frac{1}{2} q^* H q + q^* b$. This bound is sharp for $i < N$, in the sense that there exist H, b, and q_0 such that the inequality in (3.5) may be made arbitrarily close to equality.

Proof. Applying the arithmetic-geometric mean inequality to (3.4) and using the ordering of the eigenvalues yields the component-wise error bound

$$\begin{aligned} |e_i^{(k)}| &= \left(1 - \frac{\lambda_k}{\lambda_1}\right)\left(1 - \frac{\lambda_k}{\lambda_2}\right)\dots\left(1 - \frac{\lambda_k}{\lambda_i}\right)|e_0^{(k)}| \\ &\leq \left(\frac{1 - \frac{\lambda_k}{\lambda_1} + 1 - \frac{\lambda_k}{\lambda_2} \pm \dots + 1 - \frac{\lambda_k}{\lambda_i}}{i}\right)^i |e_0^{(k)}| \\ &= \left(1 - \lambda_k \left(\frac{1}{i}\sum_{l=1}^{i} \frac{1}{\lambda_l}\right)\right)^i |e_0^{(k)}| \\ &\leq \left(1 - \frac{\lambda_k}{\lambda_1}\right)^i |e_0^{(k)}| \\ &\leq \left(1 - \frac{1}{\kappa}\right)^i |e_0^{(k)}| \\ &= \left(\frac{\kappa - 1}{\kappa}\right)^i |e_0^{(k)}| \end{aligned}$$

for $k > i$ (and by construction, $e_i^{(k)} = 0$ for $k \leq i$).

Using the definitions of $\mathcal{J}$ and e_i, the fact that $Hq_{\min} = -b$, and the inequality from the previous paragraph, we have the bound

$$\begin{aligned}
\mathcal{J}(q_i) - \mathcal{J}(q_{\min}) &= \frac{1}{2} e_i^* H e_i \\
&= \sum_{k=1}^{N} \lambda_k |e_i^{(k)}|^2 \\
&\leq \left(\frac{\kappa - 1}{\kappa}\right)^{2i} \sum_{k=1}^{N} \lambda_k |e_0^{(k)}|^2 \qquad (3.6) \\
&= \left(\frac{\kappa - 1}{\kappa}\right)^{2i} |\mathcal{J}(q_0) - \mathcal{J}(q_{\min})|,
\end{aligned}$$

as required.

To construct a problem for which this bound is sharp, let the first $N-1$ eigenvalues be distinct but nearly equal positive numbers that differ from λ_1 by at most some small number $\epsilon > 0$. As $\epsilon \to 0$, we have

$$\left(1 - \frac{\lambda_N}{\lambda_1}\right)\left(1 - \frac{\lambda_N}{\lambda_2}\right) \cdots \left(1 - \frac{\lambda_N}{\lambda_i}\right) \to \left(1 - \frac{1}{\kappa}\right)^i,$$

which implies

$$|e_i^{(N)}| \to \left(\frac{\kappa - 1}{\kappa}\right)^i |e_0^{(N)}|.$$

Hence the inequality in (3.6), and consequently the inequality in (3.5), can be made arbitrarily close to equality by taking $b = 0$ and

$$q_0^{(k)} = e_0^{(k)} := \begin{cases} \epsilon, & k < N \\ 1, & k = N, \end{cases}$$

and letting $\epsilon \to 0$. □

This shows that the effectiveness of first-order algorithms is fundamentally limited for high-dimensional ill-conditioned problems. Perhaps some future algorithm could eliminate zig-zagging (indeed, CG, BFGS and Nesterov methods greatly reduce zig-zagging), but no first order method can eliminate hypercube paths. First order algorithms can only explore in the affine subspace consisting of the initial guess plus the span of the previously computed gradients, and this affine subspace cannot contain shallow directions without also first containing all steeper directions (unless the initial guess is lucky and does not contain any steep components to begin with).

3.2 Second-order methods

In order to eliminate dependence of the convergence rate on the conditioning of the problem, optimization algorithms must account for the directional scalings of the objective function in all (or almost all) directions, for all (or almost all) iterations. Second-order methods do this by:

1. Making a local quadratic model of the optimization problem based on a Taylor series.

2. Solving a linear system to find the solution of the quadratic model.

3. Moving towards the solution of the model, then repeating these steps until convergence.

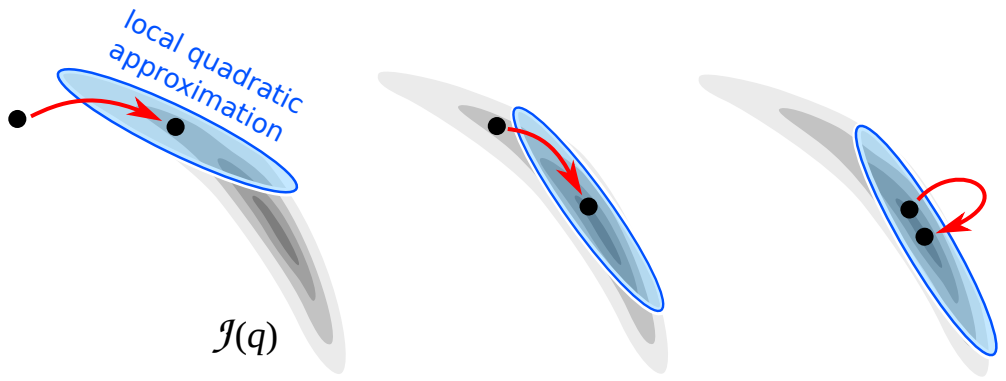


Figure 3.4: Newton and Gauss-Newton methods construct a local quadratic model of $\mathcal{J}$ about the current point, move towards the minimizer of the quadratic model, then repeat. Illustration depicts "complete" Newton steps that jump to the minimizer of the model (step length parameter, $\beta = 1$).

Reduced space methods form a local quadratic model of the objective function, then find the minimizer of the model (see Figure 3.4). Full space methods form a local quadratic model of the Lagrangian function, then find the saddle point of the model.

Standard versions of these methods build the local model by taking a Taylor series of the overall function (objective or Lagrangian), while Gauss-Newton and sequential quadratic programming (SQP) variants take Taylor series of intermediate quantities. Below we will discuss all four methods: reduced space vs. full space and standard vs.

Method	Seeking	Linear system	Based on Taylor series of
Reduced space Newton	Minimum of $\mathcal{J}$	$Hp = -g$	$\mathcal{J}$
Reduced space Gauss-Newton	Minimum of $\mathcal{J}$	$H^{\text{gn}}p = -g$	$\mathcal{G}$
Full space Newton	Saddle point of $\mathcal{L}$	$Kw = -l$	$\mathcal{L}$
Full space Gauss-Newton	Saddle point of $\mathcal{L}$	$K^{\text{gn}}w = -l$	$A(q)u - f(q)$

Table 3.1: Summary of second-order methods for solving the deterministic inverse problem.

Gauss-Newton. These methods are summarized in Table 3.1. In Chapter 5, we will see that an effective preconditioner for the linear system associated with any one of these methods can be used to build effective preconditioners for the linear systems associated with all of the other methods.

For the standard reduced space Newton's method, we have the asymptotic convergence rate

$$\|g(q_{i+1})\| \leq C \|g(q_i)\|^2 ,$$

where $g(q_i)$ and $g(q_{i+1})$ are the gradients of $\mathcal{J}$ at the i^{th} and $(i+1)^{\text{th}}$ iterations, respectively, and C is a constant that does not depend on the condition number of the Hessian. So, Newton's method improves on first order methods in that convergence is quadratic rather than linear (the number of correct digits roughly doubles each iteration), and in that *the convergence rate does not depend on the conditioning of the Hessian*. Convergence of the full space Newton method is also quadratic and independent of the conditioning of the problem. Convergence of the Gauss-Newton variants is fast, but not quadratic in general.

Here we present only the basic idea of the methods. For in depth discussion about how to choose step length parameters, when to terminate linear solves, how to deal with potential indefiniteness of the Hessian, proofs of convergence, and other important details, we highly recommend [157]. A discussion of these methods in the context of PDE-constrained optimization can be found in [5]. With minor modifications, ev-

erything we present can also be adapted for use in trust-region methods, since these methods give rise to similar linear systems that must be solved at each iteration.

3.2.1 Reduced space

In reduced space methods, at each iteration we minimize one of the local quadratic models for $\mathcal{J}$ derived in Section 2.6. Given the current point, q_i, we minimize:

$$\min_p \quad \mathcal{J}(q_i) + g^* p + \frac{1}{2} p^* \widetilde{H} p, \tag{3.7}$$

where $\widetilde{H}$ is either H or H^{gn}, and $\widetilde{H}$ and g are evaluated at q_i. If we use H, we call this the reduced space Newton method, and if we use H^{gn}, we call this the reduced space Gauss-Newton method. Since the model is quadratic, solving this minimization problem reduces to solving the linear system

$$\widetilde{H} p = -g.$$

Once this linear system is solved, one updates

$$q_{i+1} = q_i + \beta p, \tag{3.8}$$

where β is a step length parameter. Then the process repeats at q_{i+1}, and at q_{i+2}, and so on, continuing until convergence.

3.2.2 Full space

From the theory of Lagrange multipliers, the solution to the full-space problem, (2.8), is a saddle point of the following Lagrangian function:

$$\mathcal{L}(q, u, \lambda) := \frac{1}{2} \|Bu - y\|_Y^2 + \frac{1}{2} \|q - \overline{q}_0\|_R^2 + \lambda^* \left(A(q)u - f(q)\right). \tag{3.9}$$

In this framework, u is viewed as a free variable—away from the saddle point the pair of variables q and u may fail to satisfy the state equation. Let $z_i = \begin{bmatrix} q_i & u_i & \lambda_i \end{bmatrix}^T$ denote an instance of all variables in the Lagrangian formulation. Second-order methods for the full space problem mirror second-order methods for the reduced space problem, but we make a local quadratic approximation of $\mathcal{L}$ instead of $\mathcal{J}$, and seek the saddle point of the quadratic approximation rather than its minimum.

Truncating the Taylor series of $\mathcal{L}$ after the quadratic term yields the quadratic model

$$\mathcal{L}(z_i + w) \approx \mathcal{L}(z_i) + l^* w + \frac{1}{2} w^* K w,$$

where

$$l \simeq \frac{\partial \mathcal{L}}{\partial z}$$

and

$$K \simeq \frac{\partial^2 \mathcal{L}}{\partial z^2}$$

is the KKT operator.

Alternatively, replacing the state equation constraint with its linearization about q_i and u_i in (2.8) yields the quadratic program:

$$\begin{aligned} \min_{p,v} \quad & \frac{1}{2} \left\| Bv - (y - Bu_i) \right\|_Y^2 + \frac{1}{2} \left\| p - (\overline{q}_0 - q_i) \right\|_R^2, \\ \text{such that} \quad & A(q_i)v + Tp = A(q_i)u_i - f(q_i), \end{aligned} \tag{3.10}$$

for perturbations p to q_i and v to u_i. Here, T is the sensitivity of $A(q)u - f(q)$ to changes in q, i.e.,

$$Tp := \left(\frac{\partial A}{\partial q}(q_i)p \right) u_i - \frac{\partial f}{\partial q}(q_i)p.$$

The solution to (3.10) is the saddle point of the quadratic Gauss-Newton Lagrangian:

$$\begin{aligned} \mathcal{L}^{\mathrm{gn}}(p, v, \gamma) :=& \frac{1}{2} \left\| Bv - (y - Bu_i) \right\|_Y^2 + \frac{1}{2} \left\| p - (\overline{q}_0 - q_i) \right\|_R^2 \\ & + \gamma^* \left(Av + Tp - (A(q_i)u_i - f(q_i)) \right) \qquad (3.11) \\ =& \mathcal{L}^{\mathrm{gn}}(z_i) + l^* w + w^* K^{\mathrm{gn}} w, \qquad (3.12) \end{aligned}$$

where we define the Gauss-Newton KKT operator

$$K^{\mathrm{gn}} \simeq \frac{\partial^2 \mathcal{L}^{\mathrm{gn}}}{\partial w^2},$$

and the perturbation variable $w := \begin{bmatrix} p & v & w \end{bmatrix}^T$.

In either case, the saddle point of the quadratic model is the solution to the linear system

$$\widetilde{K} w = -l, \tag{3.13}$$

where $\widetilde{K}$ is either K or K^{gn}. If we use K, we call this the full space Newton method, and if we use K^{gn}, we call this the full space Gauss-Newton method. After solving (3.13) for w, we update $z_{i+1} = z_i + \beta w$ and repeat the process until convergence.

3.3 Krylov methods prefer clustered eigenvalues

Each iteration of the second-order optimization methods discussed in Section 3.2 requires solving a linear system of the form

$$Mx = b, \tag{3.14}$$

where M is either H, H^{gn}, K, or K^{gn}. Krylov methods [187] are often the best matrix-free methods for solving these linear systems. The performance of Krylov methods depends on the spectral properties of M. The more clustered the eigenvalues of M, the faster the convergence [82, 192].

We illustrate the convergence of Krylov methods with MINRES [161] applied to a self-adjoint linear system of the form (3.14). The j^{th} iterate of MINRES, x_j, is defined as the vector with minimum residual within a subspace generated by repeatedly applying M to b. Specifically,

$$x_j := \arg\min_{z \in \mathcal{K}_j} \|b - Mz\|,$$

where $\mathcal{K}_j$ is the Krylov subspace

$$\mathcal{K}_j = \text{span}\left(b, Mb, M^2b, \ldots, M^{j-1}b\right).$$

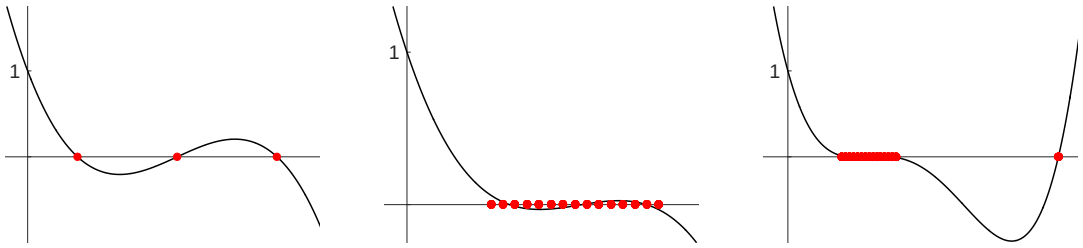
The following well-known result characterizes the convergence of MINRES in terms of the result of a hypothetical polynomial approximation problem involving the spectrum of M.

Theorem 2 (Convergence of MINRES)**.** *Let x_j be the j^{th} MINRES iterate for solving $Mx = b$, and let $\mathcal{Q}_j$ denote the set of all j^{th} order polynomials Q satisfying $Q(0) = 1$. We have*

$$\|b - Mx_j\|^2 = \min_{Q \in \mathcal{Q}_j} \sum_{k=1}^{N} Q(\lambda_k)^2 b_k^2,$$

where λ_k are the eigenvalues of M, and b_k are the components of b in the basis of eigenvectors of M.

Proof. See Appendix D. □



(a) Isolated eigenvalues, 3rd order polynomial.

(b) Clustered eigenvalues, 3rd order polynomial.

(c) Cluster plus one isolated eigenvalue, 4th order polynomial.

Figure 3.5: MINRES polynomials $Q(s)$ for different eigenvalue distributions. Red dots indicate the eigenvalue locations. Equal importance assigned to each eigenvalue ($b_k = 1$, for $k = 1 \dots N$).

For in-depth analysis and discussion of the convergence of Krylov methods, see [173]. Results analogous to Theorem 2 are derived there for a variety of Krylov methods, including GMRES [172], a generalized version of MINRES that does not require M to be symmetric.

From Theorem 2, we see that MINRES implicitly finds a j^{th} order polynomial $Q(s)$ such that:

1. $Q(0) = 1$.
2. $|Q(\lambda_k)|$ is as small as possible for all eigenvalues λ_k of M.[4]

The performance of MINRES is determined by how well 2 can be achieved while satisfying 1. This leads to the following consequences:

- A polynomial of degree j can take on any set of j roots. Thus if M has j distinct eigenvalues, MINRES will converge in at most j iterations.[5] See Figure 3.5a.
- It is easier to force a polynomial to be small on a small interval than on a large interval. If the eigenvalues of M reside within one cluster (see Figure 3.5b), the convergence rate of MINRES will depend on the size and location of the

[4]The relative magnitude of b_k^2 compared to b_j^2 determines the relative importance of making $Q(\lambda_k)$ small compared to $Q(\lambda_j)$.

[5]This is ignoring the impact of numerical rounding errors, which may lead to more required iterations when j is large.

cluster. The smaller and further away from the origin the cluster is, the better. In particular, a minimax argument shows that the convergence rate depends on the ratio of the largest eigenvalue in the cluster to the smallest eigenvalue in the cluster. This ratio is the condition number of M.

- If the eigenvalues of M consist of multiple clusters, or a combination of clusters and isolated eigenvalues, then MINRES will converge fast if those clusters are small and well-separated from each other and the origin. See Figure 3.5c.

Thus we seek a preconditioner, P, that compresses the spectrum of $P^{-1}M$ into a small number of well-conditioned clusters. The application of a Krylov method to solve a preconditioned version of (3.14) will then yield an accurate solution in a small number of iterations.

3.4 Summary

We use second-order methods since first-order methods are ineffective for high-dimensional ill-conditioned problems. Second-order methods require solving a large ill-conditioned linear system at each iteration, where the coefficient operator for the linear system is either H, H^{gn}, K, or K^{gn}. We use Krylov methods to solve these linear systems. Krylov methods can be sped up by using a preconditioner that clusters the spectrum of the coefficient operator.

Chapter 4

Spectrum of the Hessian and Information

The spectrum of the data misfit Gauss-Newton Hessian, H_d^{gn}, characterizes how informative the data are about the parameter. The larger an eigenvalue of H_d^{gn} is, the more informative the data are about the component of the parameter in the associated eigenvector direction.

We make this precise by carrying out the following plan: In Section 4.1, we use the generalized eigenvalue problem of H_d^{gn} and R to simultaneously factor the prior and a Gaussian approximation to the posterior into products of *independent* one-dimensional Gaussian random variables. These one-dimensional random variables are the components of the parameter in the normalized generalized eigenvector basis. Then in Section 4.2, we use the information theory concept of mutual information, along with a limiting argument, to quantify how much more information one can learn from the data about the component of the parameter in the k^{th} generalized eigenvector direction, as compared to the component of the parameter in the j^{th} generalized eigenvector direction. We show (Theorem 4) that this difference in information is given by

$$\log d_k^{1/2} - \log d_j^{1/2},$$

where d_k and d_j are the corresponding generalized eigenvalues of H_d^{gn}.

4.1 Simultaneously factoring the prior and posterior

Let Φ be the matrix of normalized[1] generalized eigenvectors, ϕ_k, for the generalized eigenvalue problem of H_d^{gn} and R, let d_i be the generalized eigenvalues associated with

[1] Often in the literature the length of ϕ_k is not set to 1, but rather is scaled such that $r_k = 1$. We do not use this convention, since our natural notion of length is captured by the underlying norm of the space $\|\cdot\|$, rather than the norm induced by R.

H_d^{gn}, and let r_k be the generalized eigenvalues associated with R. That is,

$$\begin{cases} \Phi^* H_d^{\text{gn}} \Phi = \text{diag}(d_k) \\ \Phi^* R \Phi = \text{diag}(r_k), \end{cases}$$

with columns ϕ_k of Φ satisfying

$$\|\phi_k\| = 1.$$

Recall from Section 2.3 and Section 2.4 that our prior, $\pi(q)$, is normally distributed, with covariance $C_{\text{prior}} = R^{-1}$. Also recall from Section 2.6 that the posterior can be locally approximated by a normal distribution, $\widetilde{\pi}(q|y)$, with covariance $(H_d^{\text{gn}} + R)^{-1}$. This posterior corresponds to the hypothetical inverse problem in which the parameter-to-observable map is replaced by its linearization.

Theorem 3. *Both $\pi(q)$ and $\widetilde{\pi}(q|y)$ separate into products of independent 1D normally distributed random variables in the coefficients $q^{(k)}$. Specifically, let*

$$q = \sum_k q^{(k)} \phi_k$$

be the expansion of q in the generalized eigenvector basis. Then

$$\pi(q) = \prod_k \pi(q^{(k)}), \quad \textit{and} \quad \widetilde{\pi}(q|y) = \prod_k \widetilde{\pi}(q^{(k)}|y),$$

where $\pi(q^{(k)})$ are normal distributions with variance r_k^{-1}, and $\widetilde{\pi}(q^{(k)}|y)$ are normal distributions with variance $(d_k + r_k)^{-1}$.

Proof. Let $\overline{q}_0^{(k)}$ denote the component of $\overline{q}_0$ in direction ϕ_k. Since Φ diagonalizes R, the prior separates into an independent product of 1D Gaussians as follows:

$$\begin{aligned} \pi(q) &\propto \exp\left(-\frac{1}{2}(q - \overline{q}_0)^* R (q - \overline{q}_0)\right) \\ &= \exp\left(-\frac{1}{2}\left(\sum_k (q^{(k)} - \overline{q}_0^{(k)})\phi_k\right)^* R \left(\sum_k (q^{(k)} - \overline{q}_0^{(k)})\phi_k\right)\right) \\ &= \exp\left(-\sum_k \frac{\left(q^{(k)} - \overline{q}_0^{(k)}\right)^2}{2r_k^{-1}}\right) \\ &= \prod_k \exp\left(-\frac{(q^{(k)} - \overline{q}_0^{(k)})^2}{2r_k^{-1}}\right), \end{aligned}$$

from which the results regarding $\pi(q)$ directly follow.

Since Φ simultaneously diagonalizes both R and H_d, it also diagonalizes the posterior covariance, $(H_d + R)^{-1}$. Hence the results regarding $\widetilde{\pi}(q|y)$ follow from the same argument, except with the posterior covariance and mean replacing the prior covariance and mean, respectively. □

4.2 Informativeness of the data

The mutual information between random variables X and Z is

$$I(X;Z) = \int\int \pi(x,z) \log\left(\frac{\pi(x,z)}{\pi(x)\pi(z)}\right) \, dx \, dz,$$

where $\pi(x,z)$ is the joint distribution between X and Z, $\pi(x)$ is the marginal distribution in X, and $\pi(z)$ is the marginal distribution in Z. Mutual information quantifies the expected amount of information we would gain about one of the random variables if we learned the value of the other variable. For more details on mutual information, see [69].

Let $Q^{(k)}$ denote the random variable associated with $q^{(k)}$ for the approximate inverse problem in which $\widetilde{\pi}(q|y)$ is the posterior.

Proposition 1. *The mutual information between the data and a single component of the parameter takes the form:*

$$I(Q^{(k)};Y) = \frac{1}{2}\log\left(1 + d_k/r_k\right). \tag{4.1}$$

Proof. Direct calculation shows that if X and Z are random variables, and if X is normally distributed with variance σ^2_{prior} and $X|Z$ is normally distributed with variance σ^2_{post}, then

$$I(X;Z) = \frac{1}{2}\log\left(\frac{\sigma^2_{\text{prior}}}{\sigma^2_{\text{post}}}\right). \tag{4.2}$$

The desired result follows from substituting $\sigma^2_{\text{prior}} = r_k^{-1}$ and $\sigma^2_{\text{post}} = (d_k + r_k)^{-1}$ into (4.2) and performing algebraic manipulations. □

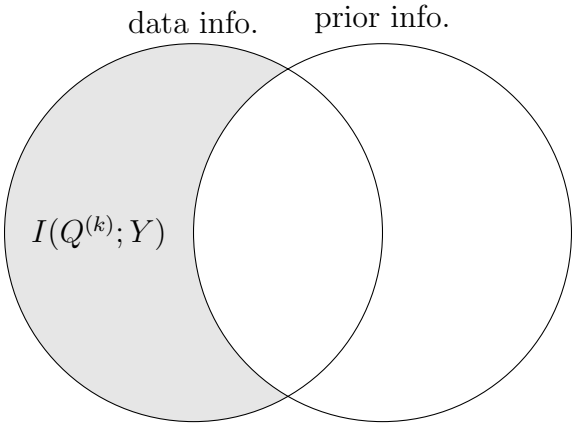


Figure 4.1: Venn diagram illustrating both the information contained in the data, and the prior information. The information contained in the data consists of both the information learned by incorporating the data (the mutual information $I(Q^{(k)};Y)$, the shaded region), as well as the information contained in the data that is not learned because it is already known (the overlapping region). Caution: the mutual information is not the overlapping region in this diagram.

Mutual information quantifies the expected amount of information a Bayesian analyst would gain about q_k if they learned the value of y, taking into account their prior knowledge (notice the presence of r_k in (4.1)). But in our case, we want to know how informative y is about q_k, independent of the prior knowledge of the person performing the analysis. For example, if one is totally certain about the parameter beforehand (Dirac prior), then the mutual information is zero. Yet this does not mean that the data are uninformative. Intuitively, the data may contain information that the Bayesian analyst cannot "learn" because they already know it, and information they already know is excluded from the mutual information (see Figure 4.1). That information is useless to them, but perhaps could be useful to someone else who has different prior knowledge.

A natural approach to characterizing the informativeness of the data (independent of the prior) is to maximize the mutual information over all possible priors in a given class, thereby determining the theoretical maximum amount of information that could be learned by any Bayesian analyst. Unfortunately, this maximization procedure fails for the class of Gaussian priors because it yields infinite mutual information: as the variance of the prior grows to infinity, the mutual information diverges. To remedy

this, we instead compare the mutual information associated with one component, $q^{(k)}$, with that of another component, $q^{(j)}$. In this case we will see that the divergent terms cancel out and the difference in mutual information converges to a finite limit.

Imagine we start with the same prior uncertainty for $q^{(k)}$ as for $q^{(j)}$, and then learn the value of y. Would we, on average, gain more information about $q^{(k)}$ or $q^{(j)}$? How much more? Note that we are not saying that the priors for $q^{(k)}$ and $q^{(j)}$ will always be equal.[2] We are merely considering a hypothetical scenario where they happen to be equal in order to make a fair comparison of the information gained. Quantitatively, this question reduces to evaluating the mutual information difference

$$I(Q^{(k)}, Y) - I(Q^{(j)}, Y), \tag{4.3}$$

under the hypothetical scenario where our priors for both $Q^{(k)}$ and $Q^{(j)}$ are normally distributed with the same variance, $r_k^{-1} = r_j^{-1} = c$. If this quantity is positive, the data are more informative about $q^{(k)}$, and if negative, the data are more informative about $q^{(j)}$.[3] Although $I(Q^{(k)}, Y)$ and $I(Q^{(j)}, Y)$ both diverge as $c \to \infty$ (uninformative prior limit) and therefore cannot be maximized over the class of all Gaussian priors, the mutual information difference, (4.3), converges to a finite value as $c \to \infty$.

Theorem 4. *Suppose that the parameter-to-observable map is affine.*[4] *Also suppose the priors for $q^{(k)}$ and $q^{(j)}$ have equal variance, $r_k^{-1} = r_j^{-1} = c$. Then for $d_k \neq 0$ and $d_j \neq 0$, we have*

$$\lim_{c \to \infty} \quad I(Q^{(k)}, Y) - I(Q^{(j)}, Y) = \log d_k^{1/2} - \log d_j^{1/2}. \tag{4.4}$$

Proof. Substituting (4.1) into (4.4) and performing algebraic manipulations yields

$$\begin{aligned} I(Q^{(k)}, Y) - I(Q^{(j)}, Y) =& \frac{1}{2} \log\left(1 + d_k/r_k\right) - \frac{1}{2} \log\left(1 + d_j/r_j\right) \\ =& \frac{1}{2} \log\left(\frac{1 + d_k c}{1 + d_j c}\right). \end{aligned}$$

[2] Indeed, if the priors for all modes were equal then the prior would not be well-defined in the limit where q is infinite-dimensional.

[3] One can interpret this as an information-theoretic analogue of the likelihood principle.

[4] Generally the map will be nonlinear; this theorem applies to local linearizations.

As $c \to \infty$, the constant 1 becomes negligible in both the numerator and the denominator, so

$$\begin{aligned}\lim_{c\to\infty} \; I(Q^{(k)}, Y) - I(Q^{(j)}, Y) &= \lim_{c\to\infty} \; \frac{1}{2}\log\left(\frac{d_k c}{d_j c}\right) \\ &= \frac{1}{2}\log\left(\frac{d_k}{d_j}\right) = \log d_k^{1/2} - \log d_j^{1/2},\end{aligned}$$

as required. □

These results from Theorem 4 relate the informativeness of the data to the eigenvalues of H_d. The more informative the data are about the component of the parameter in an eigenvector's direction, the larger the associated eigenvalue. As a result, when more informative data are included in the inversion, small eigenvalues become large, and large eigenvalues become larger. Numerical methods and preconditioners that perform worse as more eigenvalues of H_d become large (almost all existing methods and preconditioners, as we will see in Chapter 6) therefore cannot be data-scalable.

Chapter 5

Hessian and KKT Facts

This chapter takes a step back from the main argument of the dissertation to present a few well-known facts that will be required in future chapters. Primarily, we discuss facts related to the Hessian and KKT operators, and connections among these operators.

In Section 5.2, we will show how to compute the objective function, $\mathcal{J}$, and the gradient, g, and show how to apply the Hessian, H, to vectors. The procedure for applying H to vectors is matrix-free; it does not require building the Hessian's (dense) matrix representation. We summarize the procedures for performing these tasks in Table 5.1. These procedures will be used in solving the inverse problem with methods described in previous chapters.

In Section 5.3, we present formulas for the Hessian-like operators H^{gn}, K, and K^{gn}, and other relevant operators. These formulas provide insight into the structure of H^{gn}, K, and K^{gn}, and will be required in subsequent chapters when we review the literature and build our preconditioners.

In Section 5.4, we will show that H and H^{gn} are Schur complements of K and K^{gn} (respectively) for q, and that H and K converge to H^{gn} and K^{gn} (respectively) in the small data misfit limit. These connections are summarized in Figure 5.1. They imply that it does not matter which Hessian or KKT operator we choose to build a preconditioner for, because the ability to efficiently precondition or solve linear systems with any one lets us efficiently precondition any of the others. This justifies our choice, in subsequent chapters, to build preconditioners for the Gauss-Newton operators H^{gn} and K^{gn}.

The proofs of the results in this chapter follow from application of standard techinques from multivariate calculus and linear algebra, and are not particularly enlightening. Thus we relegate proofs for results in this chapter to Appendix D.

5.1 Prerequisite partial derivative operators

Before we present the main points of this chapter, we must define a few partial derivative operators. These operators appear in the derivative computation process in Section 5.2, in the formulas in Section 5.3, in the proofs in Section 5.4, and in other places throughout the dissertation.

Definition 1 (Prerequisite partial derivative operators)**.** *We define the operators T, Θ, and Ξ, to be the linear operators with the following actions:*

$$\begin{aligned} T : p &\mapsto \left(\frac{\partial A}{\partial q} p\right) u - \frac{\partial f}{\partial q} p \\ \Theta : p &\mapsto \left(\frac{\partial A}{\partial q} p\right)^* \lambda \\ p_1^* \;\Xi\; p_2 &:= \lambda^* \left(\frac{\partial^2 A}{\partial q^2} p_1 p_2\right) u - \lambda^* \left(\frac{\partial^2 f}{\partial q^2} p_1 p_2\right). \end{aligned}$$

The operators T, Θ, and Ξ are sparse when A is sparse, and may be efficiently computed and stored. For example, if one term in the weak form associated with A is

$$(u, v) \mapsto \int_\Omega e^q \nabla u \cdot \nabla v,$$

then the corresponding term in the bilinear form associated with T is

$$(p, v) \mapsto \int_\Omega p e^q \nabla u \cdot \nabla v.$$

If finite element discretization is used, the operator T can be assembled into a matrix $\mathbf{T}$ using procedures similar to those used to assemble $\mathbf{A}$ from A.

5.2 Computing derivatives of $\mathcal{J}$

In Table 5.1 we see the following results:

1. Computing $\mathcal{J}$ requires computing u, which requires the solution of the state equation.

2. Computing g requires computing an adjoint variable λ by solving a linear system which takes a form adjoint to the form of the state equation.

	Equation/Formula	Action to take
State equation	$Au = f$	Solve for u
Objective function	$\mathcal{J} = \frac{1}{2}\left\|Bu - y\right\|_Y^2 + \frac{\alpha}{2}\left\|q - \overline{q}_0\right\|_R^2$	Compute $\mathcal{J}$
Adjoint equation	$A^*\lambda = -B^*Y(Bu - y)$	Solve for λ
Gradient	$g = R(q - \overline{q}_0) + T^*\lambda$	Compute g
Incremental forward equation	$A\eta = -Tp$	Solve for η
Incremental adjoint equation	$A^*\xi = -\Theta p - B^*YB\eta$	Solve for ξ
Hessian-vector product	$Hp = Rp + \Xi p + \Theta^*\eta + T^*\xi$	Compute Hp

Table 5.1: The processes for computing the objective function, computing the gradient, and applying the Hessian to a vector (we prove the correctness of this table in Theorem 5).

3. Applying H to a vector requires computing an incremental forward variable,

$$\eta := \frac{du}{dq}p,$$

and an incremental adjoint variable,

$$\xi := \frac{d\lambda}{dq}p.$$

Computing η requires solving a linear system which takes the same form as the state equation, and computing ξ requires solving a linear system which takes the same form as the adjoint equation.

Theorem 5. *The procedure described in Table 5.1 correctly computes the objective function, $\mathcal{J}$, the gradient, g, and the action of the Hessian on an arbitrary vector, Hp.*

Proof. See Appendix D. □

5.3 Formulas for other operators

The operators G, H_d^{gn}, H_d, l, K, and K^{gn} can be expressed in terms of formulas involving the original operators in the problem and the sparse computable partial derivatives of the original operators defined in Definition 1.

Theorem 6 (Formulas for H_d^{gn}, H^{gn}, L, K, and K^{gn})**.** *The following formulas hold:*

$$\frac{d\mathcal{G}}{dq} = G = -BA^{-1}T \tag{5.1}$$

$$\frac{d^2\mathcal{J}_d^{gn}}{dp^2} \simeq H_d^{gn} = T^*A^{-*}B^*YBA^{-1}T \tag{5.2}$$

$$\frac{d^2\mathcal{J}^{gn}}{dp^2} \simeq H^{gn} = H_d^{gn} + R \tag{5.3}$$

$$\left(\frac{\partial\mathcal{L}}{\partial z}\right)^* = l = \begin{bmatrix} R(q-\overline{q}_0) + T^*\lambda \\ B^*Y(Bu-y) + A^*\lambda \\ Au - f \end{bmatrix} \tag{5.4}$$

$$\frac{\partial^2\mathcal{L}}{\partial z^2} \simeq K = \begin{bmatrix} R+\Xi & \Theta^* & T^* \\ \Theta & B^*YB & A^* \\ T & A & \end{bmatrix} \tag{5.5}$$

$$\frac{\partial^2\mathcal{L}^{gn}}{\partial z^2} \simeq K^{gn} = \begin{bmatrix} R & & T^* \\ & B^*YB & A^* \\ T & A & \end{bmatrix}. \tag{5.6}$$

Proof. See Appendix D. □

5.4 Connections among operators

Since H is the Schur complement of K for q (Theorem 7), the abilty to perform solves with K can be used to perform solves with H, and vice versa (Corollary 1 and Corollary 2). Preconditioners for H can therefore be exploited to build preconditioners for K (see, for example, [39, 40, 111]), and vice versa. This applies to the standard versions of these operators, as well as to their Gauss-Newton variants, H^{gn} and K^{gn}.

The Gauss-Newton variants of these operators well-approximate the standard versions if the data misfit, $\|Bu - y\|_Y$, is small (Theorem 8). Per the Morozov discrepancy principle, this will be the case for properly regularized inverse problems with

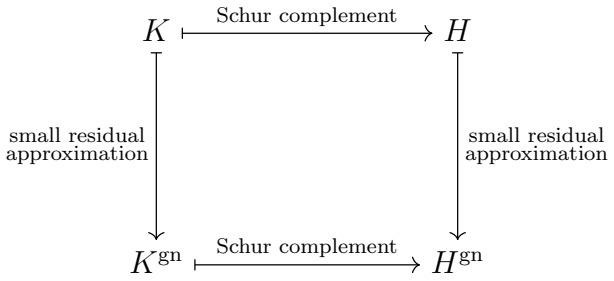


Figure 5.1: Commutative diagram relating The Hessian, the KKT operator, and their Gauss-Newton variants.

low noise. Even when $\|Bu - y\|_Y$ is large, algorithms that use the Gauss-Newton operators often outperform algorithms that use the standard operators. No one knows why Gauss-Newton methods often perform well when the noise is large, though several explanations have been proposed [62]. One reason for Gauss-Newton methods' surprisingly good performance may be the fact that H^{gn} is always positive whereas H may be indefinite away from the optimal point.

Connections among H, H^{gn}, K, and K^{gn} are summarized in Figure 5.1.

Theorem 7 ($K \to H$)**.** *H is the Schur complement of K for q:*

$$H = K_{qq} - \begin{bmatrix} K_{qu} & K_{q\lambda} \end{bmatrix} \begin{bmatrix} K_{uu} & K_{u\lambda} \\ K_{\lambda u} & K_{\lambda\lambda} \end{bmatrix}^{-1} \begin{bmatrix} K_{uq} \\ K_{\lambda q} \end{bmatrix}, \tag{5.7}$$

where we denote the blocks of K with subscripts (e.g., K_{qu} is the linear operator associated with $\frac{\partial^2 \mathcal{L}}{\partial q \partial u}$). The result also holds if H is replaced by H^{gn} and K is replaced by K^{gn}.

Proof. See Appendix D. □

Corollary 1 (K solver $\to$ H solver)**.** *The following implication holds:*

$$K \begin{bmatrix} p \\ \eta \\ \xi \end{bmatrix} = \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix} \qquad \Longrightarrow \qquad Hp = b.$$

The implication also holds if K is replaced by K^{gn} and H is replaced by H^{gn}.

Corollary 2 (H solver $\to$ K solver)**.** *The following implication holds:*

$$\begin{cases} Av = d \\ A^*\gamma = c - B^*YBv \\ Hp = b - \Theta^* v - T^*\gamma \\ A\eta = d - Tp \\ A^*\xi = c - \Theta p - B^*YB\xi \end{cases} \qquad \Longrightarrow \qquad K \begin{bmatrix} p \\ \eta \\ \xi \end{bmatrix} = \begin{bmatrix} b \\ c \\ d \end{bmatrix}. \tag{5.8}$$

The implication also holds if H is replaced by H^{gn}, K is replaced by K^{gn}, and the terms containing Θ are replaced by zero.

Proof. See Appendix D. □

Theorem 8 ($H, K \to H^{\text{gn}}, K^{\text{gn}}$)**.** *We have*

$$H = H^{gn} + O(\|\mathcal{G}(q) - y\|_Y), \tag{5.9}$$

where the Hessians are evaluated at any point, and

$$K = K^{gn} + O(\|\mathcal{G}(q) - y\|_Y), \tag{5.10}$$

where the KKT operators are evaluated at the solution to the full-space optimization problem.

Proof. See Appendix D. □

Chapter 6

Existing Hessian Solvers, Preconditioners, and Approximations

In previous chapters, we saw that the computational bottleneck for solving large-scale inverse problems with highly informative data is the solution of a linear system of the form

$$Mx = b,$$

where M is either the Hessian, the Gauss-Newton Hessian, the KKT operator, or the Gauss-Newton KKT operator.[1] In this chapter we review *solvers, preconditioners, and approximations* (SPAs) for these linear systems. Solvers solve the linear system directly, preconditioners speed up the solution of the linear system when used in Krylov methods, and approximations can be used to build preconditioners. SPAs for Hessians, KKT operators, or other similar operators have been developed in many different contexts, including parameter estimation, optimal control, PDE-constrained optimization, optimal design, and saddle point systems arising in mixed discretizations of forward problems [34, 67, 139]. However, existing SPAs are not satisfactory. New SPAs are needed.

6.1 Desired properties for SPAs

To evaluate the quality of a SPA for a Hessian or KKT system, one should consider the performance of the SPA with respect to the following desired properties:

(a) **Problem generality:** An SPA has problem generality if it may be used to

[1] This chapter contains content from [8] (Nick Alger, Umberto Villa, Tan Bui-Thanh, and Omar Ghattas. A data scalable augmented Lagrangian KKT preconditioner for large-scale inverse problems. SIAM Journal on Scientific Computing, 39(5):A2365–A2393, 2017.) and [7] (Nick Alger, Vishwas Rao, Aaron Myers, Tan Bui-Thanh, and Omar Ghattas. Scalable matrix-free adaptive product-convolution approximation for locally translation-invariant operators. arXiv preprint arXiv:1805.06018, 2018. Submitted.).

solve a wide variety of inverse problems.

(b) **Efficient solvers for subproblems:** If using the SPA involves solving subproblems, it is desirable for us to have efficient solvers for those subproblems.

(c) **Mesh-scalability:** SPAs are *mesh-scalable* if their effectiveness does not degrade substantially as the meshes used to discretize the problem are refined.

(d) **Regularization robustness:** SPAs are *regularization robust* if their effectiveness does not degrade substantially as the regularization is weakened (e.g., as the regularization parameter α is made smaller).

(e) **Data-scalability:** SPAs are *data-scalable* if their effectiveness does not degrade substantially as more informative data are included in the inverse problem.

For inverse problems, we view the goal of robustness to arbitrarily-chosen values of the regularization parameter, (d), to be unwarranted and unnecessarily restrictive. As we saw in Section 2.7, for properly regularized inverse problems the regularization operator and regularization parameter are not arbitrary. Rather, they are chosen in response to the data available in the problem. They should constrain parameter modes that are not informed by the data, while minimally modifying components of the parameter that are informed by the data. A preconditioner should perform well as the informativeness of the data increases while the strength of the regularization decreases correspondingly, but it need not perform well in an under-regularized regime where the regularization is weak but the data are uninformative. In an under-regularized regime, a good preconditioner would simply accelerate convergence to noise, i.e., more rapid solution of the wrong optimization problem. We advocate designing preconditioners that perform well with increasingly informative data, (e), for which the regularization parameter is considered a dependent parameter—chosen so that the inverse problem is neither substantially over- nor under-regularized. This extra flexibility permits design of the preconditioner to better achieve the entire set of

desired properties (a)–(e). Currently there is no known SPA that achieves all of these desired properties. The preconditioners we present in later chapters do not achieve all of these properties either, but they do improve upon existing SPAs, particularly with respect to data-scalability.

6.2 Dense factorization of the Hessian

The Hessian is dense since it contains the dense operators A^{-1} and A^{-*}. If the number of degrees of freedom in the parameter discretization is N, then a matrix representation of the Hessian would contain N^2 entries. So for moderate-scale or large-scale problems, matrix representations of the Hessian are too large be built, stored, or factorized.

For example, there are 27 million degrees of freedom for a parameter field discretized on a $300 \times 300 \times 300$ grid with one degree of freedom per gridpoint. In this case storing a matrix representation of the Hessian would require $(27 \text{ million})^2 \approx 2.9 \cdot 10^{15}$ bytes of memory in single precision floating point format. This is more than the memory capacity of the worlds most powerful supercomputer as of March 2018 (Sunway TaihuLight, $1.31 \cdot 10^{15}$ bytes [99]). Even if we could store a matrix representation of the Hessian, building it would require solving $O(N)$ PDEs (two PDE solves to build each column). This translates to solving 54 million PDEs, an infeasible task with current computing power. Furthermore, even if we could build and store a matrix representation of the Hessian, using this matrix representation to solve linear systems would require factorizing it, which is even more costly than construction and storage.

6.3 Sparse-direct factorization of the KKT matrix

Since the KKT matrix is typically sparse[2], one can solve linear systems with the KKT matrix as the coefficient matrix by factorizing the KKT matrix with sparse direct methods. For comprehensive coverage of sparse-direct methods, see [72]. How-

[2]If matrix representations of the operators A, B, R, T, Y, Θ, and Ξ are sparse, then the KKT matrix, which contains only these operators (and not their inverses) within its blocks, is sparse as well. The matrix representations of these operators are typically sparse because they arise from finite element discretization of differential operators. Differential operators act locally, and finite element discretization of an operator that acts locally yields a sparse matrix.

ever, sparse-direct methods are only mesh-scalable in terms of memory in spacetime dimension $d = 1$ or $d = 2$, and only mesh-scalable in terms of operations in spacetime dimension $d = 1$. We discuss the relationship between the spacetime dimension and the performance of sparse-direct methods in Appendix B.

6.4 Multigrid

Existing multigrid methods for distributed parameter inverse problems [46] are not data-scalable. These multigrid methods are classically categorized into three main categories: (1) speeding up or preconditioning forward and adjoint solves, (2) using multigrid to precondition the Hessian, and (3) collective smoothing.

Methods in category (1) do not use multigrid to address the fundamental difficulties stemming from highly data informed inverse problems because speeding up the forward (and adjoint) solves does not address the challenge of creating a preconditioner that is data-scalable. The required number of forward/adjoint solves scales with the informativeness of the data.

The big difficulty with category (2) is that, when the regularization is chosen appropriately, the regularization and data misfit terms of the reduced Hessian compete with each other (see Section 2.7.1). Thus smoothers for the regularization term tend to be roughers for the data misfit term, and vice versa. So multigrid methods belonging to the second category tend to be restricted to the case $R \approx I$. We note papers [1, 2, 4, 79, 80], on elliptic, parabolic, and Stokes source inversion problems with this restriction. Effective smoothers for the Hessian in an elliptic boundary data optimal control problem and a shape-optimization problem have been constructed using Fourier/pseudo-differential symbol analysis [10, 13].

In collective smoothing, category (3), one designs multigrid smoothers for the entire KKT system (parameter, forward, and adjoint) at once [44, 45]. Collective smoothers also tend to either require $R \approx I$, e.g., [182], or substantially degrade in performance as the regularization parameter decreases, e.g., [16].

6.5 Low-rank approximation and regularization preconditioning

Regularization preconditioning and methods based on low-rank approximation of the (regularization preconditioned) data misfit Hessian are robust and widely used, but not data-scalable. This is because they well-approximate the regularization term, but rely on brute-force to deal with the data misfit term, and the data misfit term increases in relative importance compared to the regularization term as more informative data are included in the inversion. These methods rely on the fact that H_d is usually a compact operator [52, 53, 54, 191], and therefore has arbitrarily accurate finite-rank approximations. The physical meaning of this compactness is that the observations inform only a finite number of the components of the parameter (see Chapter 4 for a discussion of the connection between the spectrum of H_d and the informativeness of the data).

Low-rank approximation To take advantage of the data misfit Hessian's compactness, one can build a low-rank approximation of the data misfit Hessian, or of the regularization preconditioned data misfit Hessian, and use this low-rank approximation along with the Sherman-Morrison-Woodbury formula to solve linear systems involving the overall Hessian [55, 59, 71, 96, 164, 178]. These low-rank approximation can be constructed with classical methods such as Lanczos or Arnoldi iterations, or modern randomized SVD [118]. These methods require only application of the data misfit Hessian to vectors. The number of required Hessian applications is of the order of the rank of the desired approximation. However, even when the prior preconditioned data misfit Hessian is low-rank in the sense that the rank, r, is much less than the parameter dimension, N, the cost of computing the low-rank approximation may be prohibitive. As seen in Section 5.2, applying the data misfit Hessian to a vector involves multiple PDE solves, so low-rank approximation of the data misfit Hessian requires $O(r)$ linearized forward/adjoint PDE solves. For large-scale problems with e.g. N of order 10^6, even a compression of 0.1% still means that thousands of forward solves are needed, which is often computationally expensive [51, 64, 126]. Moreover, since r grows with increasingly informative data, these methods are not data-scalable.

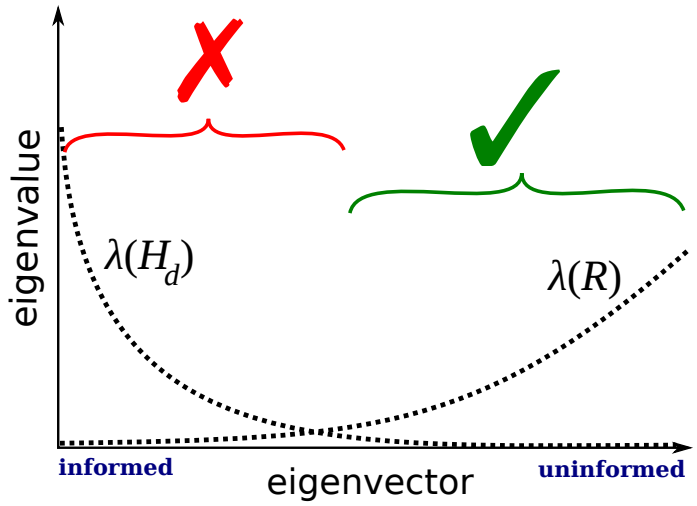


Figure 6.1: Regularization preconditioning and low-rank approximation of the regularization-preconditioned Hessian deal with all of the uninformed modes of the parameter simultaneously, but require brute-force to deal with the informed modes.

Regularization preconditioning Alternatively, one can use Krylov methods to solve linear systems with the Hessian as the coefficient operator, and use the regularization operator as the preconditioner. With this *regularization preconditioning*, the preconditioned Hessian is the identity plus a compact operator, so Krylov methods applied to the preconditioned Hessian will converge at mesh independent, superlinear rates[3] [19, 98, 122]. Intuitively, regularization preconditioning deals with the infinite set of uninformed parameter modes simultaneously, but requires many Krylov iterations to deal with the leftover finite set of informed parameter modes (see Figure 6.1, and also our discussion of the convergence of Krylov methods in Section 3.3). Since the regularization is usually an elliptic differential operator (see Section 2.7.2), applying the inverse of the regularization to a vector requires an elliptic PDE solve, which can be done with multigrid and is considered cheap in the inverse problem context. However, since the eigenvalues of the regularization preconditioned Hessian corresponding to informed modes are typically well-separated, each informed component of the parameter will require roughly one Krylov iteration, so Hessian solves using regularization preconditioning will require large numbers of Krylov iterations

[3]Here, by *superlinear*, we mean that the norm of the error asymptotically decays superlinearly with respect to the number of Krylov iterations.

on problems with highly informative data [5].

6.6 Adjoint Schur complement and block scaling preconditioners

Rather than precondition the Hessian, which is the Schur complement of the KKT operator for the parameter (see Theorem 5.7), one may instead build preconditioners based on block factorizations of the KKT operator that eliminate the parameter and state, resulting in a Schur complement for the adjoint variable [22, 35, 121, 167, 168, 169, 175, 176, 177, 180]. This approach requires one to design preconditioners for the objective block (the 2×2 block corresponding to q and u in K) and for the Schur complement associated with the adjoint variable. Several preconditioners in this class achieve mesh-scalability and regularization robustness, e.g., [26, 162, 163]. The drawbacks to preconditioners in this class include that they only apply to specific problems, or that they make restrictive assumptions about the B, R, and T operators. The restrictions $B \approx I$ and/or $R \approx I$ are common. The restriction $B \approx I$ rules out inverse problems with limited observations, and the restriction $R \approx I$ rules out smoothing regularization, which is well-suited for distributed parameter inverse problems (see Section 2.7.2).

Preconditioners based on the adjoint Schur complement are closely related to abstract "block-scaling" approaches in which one searches for optimal block diagonal preconditioners for saddle point/KKT systems [199, 154, 155]. In certain circumstances, these approaches achieve regularization robustness, but typically these preconditioners either require observations everywhere ($B \approx I$), or other restrictive assumptions. The block diagonal preconditioner

$$\begin{bmatrix} \alpha I & & \\ & B^*B + \alpha\widehat{A^*A} & \\ & & \frac{1}{\alpha}I \end{bmatrix} \tag{6.1}$$

was recently proposed to overcome the observations everywhere limitation [138]. Here $\widehat{A^*A}$ is a 4th order elliptic operator that is spectrally equivalent to A^*A. This preconditioner was proven to be mesh-scalable and regularization robust for a specific

source inversion problem with L^2 regularization (L^2, $R \approx I$, $T \approx -I$). Despite substantial differences in motivation and analysis, our proposed augmented Lagrangian KKT preconditioner presented in Chapter 7 could be considered as a generalization of this work to more general operators R and T.

6.7 Convolution interpolation

Since the linear operator that performs a convolution may be numerically full-rank (e.g., convolution with a delta function, the identity operator) or high-rank (e.g., convolution with a Gaussian with a small standard deviation), interpolation of convolution operators can, where applicable, be used to approximate dense operators with far fewer terms than the rank of the operator. The Hessian preconditioner we present in Chapter 8 is based on interpolation of convolution operators.

Convolution interpolation schemes fall into two categories: product-convolution schemes where the element-wise products with weighting functions are performed before the convolutions, and convolution-product schemes where these operations are performed in the opposite order[4]. That is,

$$\underbrace{f \mapsto \sum_{k=1}^{r} \varphi_k * (w_k \cdot f)}_{\text{product-convolution}} \qquad \text{vs.} \qquad \underbrace{f \mapsto \sum_{k=1}^{r} w_k \cdot (\varphi_k * f)}_{\text{convolution-product}}. \tag{6.2}$$

Convolution interpolation schemes have been used in many fields including image restoration and deblurring [88, 86, 94, 149, 150, 185, 184, 3, 97, 115, 170, 166], wireless communication signal processing [125], ultrasound imaging [153], systems biology [103], and Hessian approximation in seismic inversion [198].[5] Convolution interpolation schemes differ in how they construct the functions w_k and φ_k. For a comprehensive overview of existing schemes for constructing these functions, we refer the reader to the excellent summaries in [77, 87, 100].

Broadly, existing schemes can be categorized by whether the span of the functions w_k is fixed, or the span of the functions φ_k is fixed, or both of the spans are fixed,

[4] Reader beware: in some papers this naming convention (product-convolution vs. convolution-product) is reversed!

[5] In many of these applications, the impulse response is known as the point spread function (PSF), as it corresponds to the spreading of a point source of light as it passes through an optical system.

or neither of the spans are fixed. Schemes then attempt to find the remaining (not fixed) functions so that the error in the operator approximation is small. Established choices for the span of the functions φ_k include the span of impulse responses of A to a collection of delta function sources at fixed locations (we do this), subspaces of this span, and the span of functions with known analytic forms (e.g., Gaussians, spherical harmonics). Established choices for the span of the functions w_k include spans of Fourier modes, piecewise polynomials on a regular grid (e.g., piecewise constants, piecewise linear functions, B-splines), wavelets, radial basis functions [38], and functions based on Kriging.

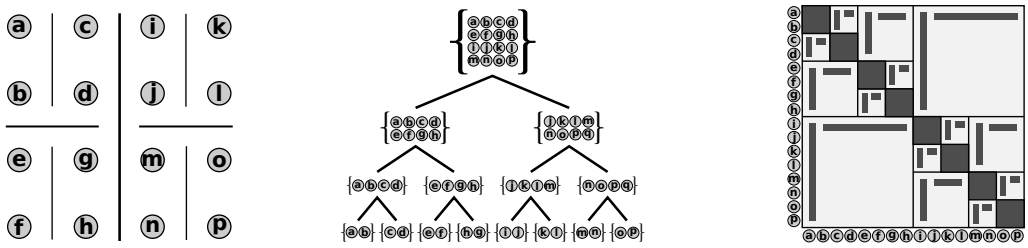
On one hand, existing schemes where the functions w_k are not fixed[6] require more access to A than merely the ability to apply it to vectors. On the other hand, existing schemes where the functions w_k are fixed do not permit spatial adaptivity (with one exception). This includes existing sectioning approaches which partition the domain into pieces on a regular grid, then use different functions φ_k for each piece [149]. The exception is [24], which proposes partitioning the domain with an adaptively refined grid. However, [24] only proposes the concept; [24] suggests that the reader develop application-specific algorithms to perform the adaptivity in practice. The adaptive product-convolution scheme we present in Chapter 8 provides a fully specified, general purpose framework for performing adaptive refinement. The scheme we present also improves upon existing convolution interpolation schemes in the way it overcomes issues related to boundaries.

6.8 Hierarchical matrices

Hierarchical matrix (H-matrix) methods are theoretically asymptotically scalable for many Hessians, but in practice are computationally costly. Also, the cost scales with wave frequency for Hessians and KKT operators in wave inverse problems, so H-matrix methods are not data-scalable for these problems.

Hierarchical matrices [112] are matrices that may be full-rank, but the blocks of

[6]The terminology for this is potentially confusing: in the literature, computed (rather than fixed) functions w_k are known as "adaptive" weighting functions, but this is unrelated to our "adaptive grid" weighting functions



(a) Clusters. (b) Cluster tree. (c) HODLR H-matrix.

Figure 6.2: Clusters (6.2a), cluster tree (6.2b), and block structure (6.2c) for a HODLR H-matrix, with dense blocks on the diagonal and low-rank blocks off the diagonal.

the matrix associated with clusters of degrees of freedom that are far away from each other (or satisfy some other admissibility condition) are low-rank. This structure allows for compressed storage and fast (nearly linear) matrix arithmetic, including matrix inversion and factorization. Special subclasses of H-matrices allow for greater compression and faster matrix arithmetic (e.g., H^2-matrices [114] and closely related HSS matrices [61, 194], among others). For an overview of H- and H^2-matrices, see [42, 113].

Classical H-matrix construction techniques assume that the original matrix is stored in memory in dense or sparse format, and convert it to H-matrix format by computing low-rank factorizations of its low-rank blocks using standard numerical linear algebra. If the matrix is not stored in memory but individual matrix entries can be computed efficiently, the matrix can be efficiently converted to H-matrix format by performing CUR factorizations for each of its low-rank blocks [29, 43, 186]. Since we can access the Hessian only by applying it and its adjoint to vectors, we cannot use these H-matrix construction techniques.

More recently, an asymptotically scalable matrix-free H-matrix construction technique was proposed based on a recursive "peeling process" that involves computing low-rank factorizations of blocks at the coarsest level of the H-matrix block cluster hierarchy, then the using the low-rank factorizations at the coarsest level to help compute low-rank factorizations at next coarsest level, and so on all the way to the finest level [130]. The method was improved in [141, 142] for hierarchically off-diagonal low-rank (HODLR) matrices and hierarchically block separable (HBS) [143] matrices

(restrictive special cases of H and H^2 matrices). Yet it has several subtle limitations:

- Although asymptotically scalable in theory, the peeling process must apply the original operator to a large number of vectors in practice.
- Constructing a less accurate approximation by applying the original operator to fewer vectors is dangerous. Errors at any step of the "peeling process" compound during subsequent steps.
- The approximation procedure is purely algebraic and does not take into account any other properties of the underlying operator. This makes the method more general, at the cost of potentially being less efficient than specialized schemes that do take advantage of additional properties.

There has been recent progress on improving the peeling process to address these issues [47].

The KKT operator is typically sparse. Thus the KKT operator can be converted to H-matrix format, then inverted, using classical H-matrix techniques. However, this process may be inefficient since blocks of the inverse of the KKT operator may have a much larger hierarchical rank than the Hessian. In particular, when the forward problem involves high-frequency wave propagation (such as the wave model problem), complex long-range interactions cause the solution operator for the forward problem, A^{-1}, to have poor H-matrix approximations (and since A is contained within a block of K, this causes K^{-1} to have poor H-matrix approximations).

We remember that the Hessian can be formed as a Schur complement through algebraic combinations of blocks of the KKT operator (see Theorem 7), so in principle one can construct an H-matrix approximation of H by converting blocks of K into a H-matrices, then algebraically combining these blocks into the desired form using H-matrix arithmetic. However, the Hessian often operates on functions defined on boundaries, time slices, or other lower dimensional domains than the domains of functions the intermediate blocks of the KKT matrix operates on. Furthermore, although H-matrix arithmetic and inversion is asymptotically scalable, it is still computationally expensive.

6.9 Other methods

Preconditioning techniques based on sparse approximation, such as incomplete factorization, sparse approximate factorization, sparse approximate inverses, or banded approximations [31, 32, 33, 36, 73, 81, 90, 93, 135, 136, 171, 193], are not useful to us since the relevant matrices (the inverse, LU factors, or Cholesky/LDLt factors of H or K) have poor sparse approximations.

Preconditioners built by reusing gradient or Krylov information from previous Newton iterations [85, 91, 92, 145, 146] cannot be data-scalable because, at best, they can reduce cost of performing many Newton steps to the cost of performing one Newton step. Yet even one Newton step can be arbitrarily computationally expensive if the data in the inverse problem are highly informative about the parameter.

A variety of problem-specific Hessian solvers and preconditioners have been developed using a diverse set of techniques, including analysis of the pseudo-differential symbol [23] of the reduced Hessian [11, 14, 15], matrix probing [66, 74], approximate sparsity in wavelet frames [120], and analytic expressions derived for model problems [2, 12, 95, 96, 102, 190].

Chapter 7

Augmented Lagrangian KKT preconditioner

In this chapter we propose clustering the spectrum of the Gauss-Newton KKT operator,

$$K^{\text{gn}} = \begin{bmatrix} \alpha R_0 & & T^* \\ & B^*YB & A^* \\ T & A & \end{bmatrix}, \tag{7.1}$$

by using the following block diagonal preconditioner,

$$P := \begin{bmatrix} \alpha R_0 + \rho T^*T & & \\ & B^*YB + \rho A^*A & \\ & & \frac{1}{\rho} I \end{bmatrix}, \tag{7.2}$$

where I denotes the identity map associated with the appropriate inner product (in the computations, a mass matrix).[1] We further propose choosing $\rho = \sqrt{\alpha}$ based on theoretical results and numerical evidence (recall α is the regularization parameter in the deterministic framework). We prove that, using our preconditioner (7.2), the symmetrically preconditioned KKT operator satisfies the condition number bound

$$\text{cond}\left(P^{-1/2} K^{\text{gn}} P^{-1/2}\right) \leq \frac{3}{(1-\beta)\delta}, \tag{7.3}$$

where $\text{cond}(\cdot)$ denotes the condition number, and δ and β are bounds on the eigenvalues of the arithmetic and geometric means of certain damped projectors. Based on the nature of the damped projectors, we expect these eigenvalue bounds to be satisfied with good constants δ and β if the inverse problem is neither over- nor under-regularized.

[1] This chapter contains content from, and is primarily based on, [8] (Nick Alger, Umberto Villa, Tan Bui-Thanh, and Omar Ghattas. A data scalable augmented Lagrangian KKT preconditioner for large-scale inverse problems. SIAM Journal on Scientific Computing, 39(5):A2365–A2393, 2017.). *Required contribution and copyright statement:* Nick Alger framed the problem, developed the numerical algorithms, wrote the code, performed the theoretical analysis, and wrote the paper. The other coauthors engaged in regular helpful discussions about the work and helped edit the paper. Nick Alger holds the copyright on the paper.

In our theory and numerical experiments we assume that A and R_0 are invertible maps. Although the application of preconditioner (7.2) and the abstract theory we present do not depend on invertibility of T, much of the intuition behind the assumptions of the theory is lacking in the case where T is non-invertible. Nevertheless, there are many inverse problems characterized by invertible T operators. Remedies for the case where T is not invertible are the subject of ongoing research. In addition to source inversion problems (addressed in Section 7.6 and Section 7.7), coefficient inverse problems in which the state and parameter share the same discretization often give rise to invertible T. While existing data-scalable KKT preconditioners usually require regularization operators R_0 that are spectrally equivalent to the identity (see Chapter 6), our preconditioner (7.2) performs well even if R_0 is a discretization of an unbounded operator (e.g., Laplacian regularization).

7.1 Overview

In Section 7.4.2 we prove the condition number bound (7.3). In Section 7.6 we derive quantitative bounds on δ and β for the special case of source inversion problems with spectral filtering regularization. When the regularization is chosen appropriately, these bounds are independent of the mesh size and of the information content in the data.

In Section 7.7 we numerically demonstrate the effectiveness of the preconditioner on a Poisson source inversion problem with highly informative data and Laplacian regularization. Preconditioning the KKT system with our preconditioner results in greater accuracy in three MINRES iterations than the widely-used regularization preconditioning on the Hessian system achieves in 50 conjugate gradient iterations. Even though the regularization is not a spectral filter, our preconditioner still exhibits mesh independence and good scalability with respect to a decrease in the regularization parameter by 10 orders of magnitude. As suggested by our theory, we see that the performance of the preconditioner in the small regularization regime actually improves as more data are included in the inversion.

7.2 Commentary on solving the preconditioner subsystems

Applying our preconditioner (7.2) requires the solution of two subsystems with coefficient operators

$$\alpha R_0 + \rho T^* T \tag{7.4}$$

and

$$B^* Y B + \rho A^* A, \tag{7.5}$$

respectively. This can be a challenge. However, Hessian preconditioning and KKT preconditioning for large-scale inverse problems with highly informative data are fundamentally difficult endeavors, and the operators (7.4) and (7.5) have many advantages over the alternatives.

To begin with, we typically have easy access to the entries of the concrete matrix representations of these operators.[2] Thus we have at our disposal the entire arsenal of symmetric positive definite sparse preconditioning techniques that deal with matrix entries; e.g., incomplete factorizations, factorized sparse approximate inverses [93], and modern multilevel techniques including algebraic multigrid and hierarchical interpolative factorizations [124]. This stands in direct contrast to the Hessian, which is dense owing to the inverses of the forward and adjoint operators within it, and as such may be accessed only via matrix-vector multiplies.

Additionally, the data misfit Hessian (which often acts as a compact operator) and the regularization operator (which often acts as a differential operator) tend to act in opposition to each other by construction (we discussed this in Section 2.7.1). Since the reduced Hessian is the sum of these operators, it is difficult to design preconditioners that are effective for both terms in the reduced Hessian at the same time. In contrast, the different terms in our subsystems tend not to act in opposition to each other.

In typical applications R_0 is chosen to be an elliptic differential operator, and T is either identity-like, or acts like a differential operator. Thus there is good reason to believe that multilevel techniques will be effective on the system $\alpha R_0 + \rho T^* T$ in

[2] Although (dense) inverses of mass matrices can arise in concrete representations of these subsystems due to the adjoint operation, these inverse mass matrices can typically be replaced with spectrally equivalent sparse lumped mass approximations.

situations of practical interest. A similar argument applies to $B^*YB+\rho A^*A$ whenever the forward operator A is amenable to multilevel techniques. In the numerical results section (Section 7.7), we see that for a source inversion problem with an elliptic PDE constraint, replacing the two subsystem solves with a few algebraic multigrid V-cycles results in nearly the same convergence rate as performing the solves exactly.

Of course, the operators in our subsystems are squared, and such squaring should always done with caution. However, subsystems involving squared operators are also present in state of the art preconditioners that have been proposed in the literature (see Section 6.6). In particular, a matrix spectrally equivalent to $B^*YB+\rho A^*A$ shows up in the preconditioner proposed in [138].

7.3 Derivation of the preconditioner

The preconditioner in (7.2) is derived from a block diagonal approximation to the KKT operator associated with an augmented Lagrangian formulation of the quadratic optimization problem (3.10). That is, the optimization problem that arises at each iteration of the full space Newton method for solving the deterministic framework. In the following derivation, it will be convenient to group the parameter and state variables into a single vector $x := \begin{bmatrix} q \\ u \end{bmatrix}$. With this grouping, optimization problem (3.10) takes the following standard quadratic programming form,

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^* M x - b^* x \\ \text{such that} \quad & Cx = r, \end{aligned} \tag{7.6}$$

where

$$r = A(q_k)u_k - f(q_k), \quad b := \begin{bmatrix} \alpha R_0(q_k - \bar{q}_0) \\ B^*Y(Bu_k - y) \end{bmatrix}, \quad C := \begin{bmatrix} T & A \end{bmatrix},$$

and M is the (generally singular) operator

$$M := \begin{bmatrix} \alpha R_0 & \\ & B^*B \end{bmatrix}.$$

The KKT operator from equation (7.1) then becomes,

$$K^{\text{gn}} := \begin{bmatrix} \alpha R_0 & & T^* \\ & B^*B & A^* \\ T & A & \end{bmatrix} = \begin{bmatrix} M & C^* \\ C & \end{bmatrix}. \tag{7.7}$$

For non-singular M, it is well-established [148] that the following positive definite block diagonal preconditioner,

$$\begin{bmatrix} M & \\ & CM^{-1}C^* \end{bmatrix}, \tag{7.8}$$

clusters the eigenvalues of the preconditioned operator onto at most three distinct values. Note that the positive operator $CM^{-1}C^*$ is the negative Schur complement for the adjoint variable. Since the objective block M is singular whenever B is not full rank (i.e., in the case of limited observations), we cannot directly use this result. However, (7.6) has the same solution as the following augmented optimization problem,

$$\begin{aligned} \min_x \quad & \frac{1}{2}x^*Mx - b^*x + \frac{\rho}{2}\|Cx - f\|^2 \\ \text{such that} \quad & Cx = f, \end{aligned}$$

where the constraint is enforced strictly, but an additional quadratic penalty term is added to the objective function to further penalize constraint violations when an iterate is away from the optimal point. The KKT operator for this augmented optimization problem is

$$\begin{bmatrix} M + \rho C^*C & C^* \\ C & \end{bmatrix}. \tag{7.9}$$

With this augmentation, the objective block is now nonsingular provided that M is coercive on the null space of C (i.e., the optimization problem is well-posed).

The positive definite block diagonal preconditioner analogous to (7.8) but based on the augmented KKT operator (7.9) is

$$\begin{bmatrix} M + \rho C^*C & \\ & C(M + \rho C^*C)^{-1}C^* \end{bmatrix}. \tag{7.10}$$

This preconditioner clusters the spectrum of the original (non-augmented) KKT operator onto the union of two well-conditioned intervals [106]. However, this preconditioner is not practical since it is computationally difficult to perform solves $(M + \rho C^*C)^{-1}$, as well as apply the Schur complement $C(M + \rho C^*C)^{-1}C^*$ and its inverse. Thus we construct the preconditioner in (7.2) by replacing these blocks with cheaper approximations.

Intuitively, when ρ is large, constraint violations are more strongly penalized by the objective, so the adjoint variable does not need to "work as hard" to enforce the constraint. This manifests in better conditioning of the Schur complement for the adjoint, $C(M + \rho C^*C)^{-1}C^*$. Indeed, it is easy to see that $C(M + \rho C^*C)^{-1}C^* \to \frac{1}{\rho}I$ as $\rho \to \infty$. To this end, we expect the approximate preconditioner

$$\begin{bmatrix} M + \rho C^*C & \\ & \frac{1}{\rho}I \end{bmatrix}, \tag{7.11}$$

to perform well when ρ is large. The preconditioner (7.11) is, essentially, a mechanism for using an unconstrained penalty method to precondition a constrained optimization problem.

The augmented objective block, $M + \rho C^*C$, takes the form

$$M + \rho C^*C = \begin{bmatrix} \alpha R_0 + \rho T^*T & \rho T^*A \\ \rho A^*T & B^*YB + \rho A^*A \end{bmatrix}.$$

Since this 2×2 block operator is difficult to solve, we cannot use preconditioner (7.11) directly, and must make further approximations. In particular, the off-diagonal blocks are scaled by ρ, so when ρ is small we expect the relative importance of these blocks to be reduced. Dropping the off-diagonal blocks in $M + \rho C^*C$ and then substituting the result into (7.11) yields our overall 3×3 block diagonal preconditioner (7.2),

$$P := \begin{bmatrix} \alpha R_0 + \rho T^*T & & \\ & B^*YB + \rho A^*A & \\ & & \frac{1}{\rho}I \end{bmatrix}.$$

One hopes that it is possible to choose ρ large enough that the Schur complement is well approximated by $\frac{1}{\rho}I$, but at the same time small enough that the objective block is well-preconditioned by the block diagonal approximation. Our theory and numerical results in subsequent sections suggest that these competing interests can be balanced by choosing $\rho = \sqrt{\alpha}$, provided that the inverse problem is appropriately regularized. In the next section we provide an abstract theoretical analysis of the preconditioner without making any assumptions about the value of ρ. A more specific analysis for source inversion problems with spectral filtering regularization, which motivates our choice of ρ, is performed in Section 7.6.

7.4 Abstract analysis of the preconditioner

In this section we analyze the preconditioned KKT operator, showing that it is well-conditioned if bounds on the arithmetic and geometric means of certain damped projectors are satisfied. First, we highlight the structure of the preconditioned KKT operator, state the necessary arithmetic and geometric mean bounds, and recall a prerequisite result from Brezzi theory. Then we prove bounds on the condition number of the preconditioned KKT operator based on the arithmetic and geometric mean bounds.

7.4.1 Prerequisites

7.4.1.1 Preconditioned KKT operator

Let E denote the symmetrically preconditioned KKT operator,

$$E := P^{-1/2} K^{\text{gn}} P^{-1/2},$$

with P and K^{gn} defined in (7.2) and (7.7), respectively. Direct calculation shows that the symmetrically preconditioned KKT operator has the following block structure,

$$E = \begin{bmatrix} I - Q^*Q & & Q^* \\ & I - U^*U & U^* \\ Q & U & \end{bmatrix}, \tag{7.12}$$

where the operators Q and U are defined as

$$Q := T\left(\frac{\alpha}{\rho} R_0 + T^*T\right)^{-1/2}, \quad U := A\left(\frac{1}{\rho} B^*Y B + A^*A\right)^{-1/2}.$$

For convenience, we further denote the objective and constraint blocks of the preconditioned system by X and Z, respectively, where

$$X := \begin{bmatrix} I - Q^*Q & \\ & I - U^*U \end{bmatrix}, \quad Z := \begin{bmatrix} Q & U \end{bmatrix}, \tag{7.13}$$

so that the preconditioned KKT operator takes the form

$$E = \begin{bmatrix} X & Z^* \\ Z & \end{bmatrix}. \tag{7.14}$$

7.4.1.2 Arithmetic and geometric mean assumptions

The quality of the preconditioner depends on the arithmetic and geometric means of the following two *damped projectors*,[3]

$$\Pi_R := QQ^* = T\left(\frac{\alpha}{\rho}R_0 + T^*T\right)^{-1}T^*, \tag{7.15}$$

and

$$\Pi_d := UU^* = A\left(\frac{1}{\rho}B^*YB + A^*A\right)^{-1}A^*.$$

Note that if T is invertible, we have

$$\Pi_d = T\left(\frac{1}{\rho}H_d^{\text{gn}} + T^*T\right)^{-1}T^*, \tag{7.16}$$

where we recall the formula $H_d^{\text{gn}} := -T^*A^{-*}B^*YBA^{-1}T$ for the data misfit Gauss-Newton Hessian.

As damped projectors, it is easy to show that the eigenvalues of Π_R and Π_d are bounded between 0 and 1. The degree to which the eigenvalues of Π_R are damped below 1 is controlled by the strength of the damping term $\frac{\alpha}{\rho}R_0$ and its interaction with the eigenstructure of T. Similarly, the degree of damping of the eigenvalues of Π_d is controlled by the strength of the damping term $\frac{1}{\rho}H_d^{\text{gn}}$ and its interaction with the eigenstructure of T (or the interaction of the damping term $\frac{1}{\rho}B^*YB$ with the eigenstructure of A, when T is not invertible).

Assumption 1 (Damped projector AM-GM bounds)**.** *We assume there exist constants β, δ such that the following bounds on the spectrum of the arithmetic and geometric means of the damped projectors hold:*

a) $\quad 0 < \delta \leq \frac{1}{2}\lambda_{min}\left(\Pi_R + \Pi_d\right),$

b) $\quad \lambda_{max}\left(\Pi_R\Pi_d\right)^{1/2} \leq \beta < 1,$

[3]Recall that $T(\gamma I + T^*T)^{-1}T^*$ approximates the orthogonal projector onto the column space of T for small γ. With this in mind, one can view an operator of the form $T(D + T^*T)^{-1}T^*$ as an approximate projector onto the column space of T, damped by the operator D. We call such operators *damped projectors*.

where $\lambda_{min}(X)$ and $\lambda_{max}(X)$ denote the smallest and largest eigenvalues of an operator X, respectively.

Theorem 10 will establish that the larger δ is and the smaller β is, the more effective preconditioner (7.2) is.

Qualitatively, if T is invertible and the regularization is chosen to act in opposition to the data misfit, as desired for the problem to be properly regularized based on the discussion in in Section 2.7.1, then αR_0 will act strongly on vectors that G acts weakly on, and vice versa. Thus we expect the damping in Π_R to be strong where the damping in Π_d is weak, and vice versa. Consequently, it is reasonable to hypothesize that Assumption 1 will be satisfied with good constants for inverse problems that are properly regularized. Making this intuition precise requires careful analysis of the interaction between the eigenstructures of R_0, H_d^{gn}, and T, which must be done on a case-by-case basis. We perform this analysis for the special case of source inversion problems with spectral filtering regularization in Section 7.6, and expect similar behavior to hold in more general situations.

7.4.1.3 Brezzi theory for well posedness of saddle point systems

The proof of the coercivity bound for our preconditioned KKT operator invokes Brezzi theory for saddle point systems [49, 76, 195]. In particular, we use a recently discovered bound in [128], which is slightly sharper than bounds derived from the classical theory. Here we state the prerequisite theorem (without proof), and refer the reader to [128] for more details. This theory can be stated in much greater generality than what we present here.

Theorem 9 (Krendl, Simoncini, and Zulehner)**.** *Let E be the saddle point system*

$$E = \begin{bmatrix} X & Z^* \\ Z & \end{bmatrix},$$

where X is self-adjoint and positive semidefinite. Further suppose that

- *X is coercive on the kernel of Z, i.e.,*

$$0 < a \leq \inf_{\substack{x \in Ker(Z) \\ x \neq 0}} \frac{x^* X x}{\|x\|^2}.$$

- *X is bounded, i.e., $\|X\| < b$.*
- *The singular values of Z are bounded from below, i.e.,*

$$0 < c \leq \sigma_{min}(Z).$$

Then the minimum singular value of E, $\sigma_{min}(E)$, is bounded from below, with the bound

$$\frac{a}{1 + \left(\frac{b}{c}\right)^2} \leq \sigma_{min}(E). \tag{7.17}$$

7.4.2 Bound on the condition number of the preconditioned KKT operator

To apply Brezzi theory (Theorem 9) to our problem, we need a coercivity bound for X on the kernel of Z, a continuity bound for X on the whole space, and a coercivity bound on Z, where the constants for these bounds are denoted a, b, and c, respectively. We use the particular structure of the KKT operator (7.7), along with Assumption 1, to derive these bounds in Section 7.4.2.1. In Proposition 2 we derive bounds for a and b, and then in Proposition 3 we derive a bound for c.

In Section 7.4.2.2 we derive well posedness and continuity bounds on the preconditioned KKT operator, E, and then combine these bounds to provide an upper bound on the condition number of E. Well posedness of E is proven in Proposition 4, using Brezzi theory in the form of Theorem 9. Continuity of E is proven directly in Proposition 5. Finally, the overall condition number bound for E is given in Theorem 10.

7.4.2.1 Bounds on X and Z

Proposition 2 (Bounds a, b for X)**.** *The eigenvalues of X restricted to the kernel of Z are bounded below by $1 - \beta$, where β is defined in Assumption 1. That is,*

$$0 < 1 - \beta \leq \inf_{\substack{x \in Ker(Z) \\ x \neq 0}} \frac{x^* X x}{\|x\|^2}.$$

Additionally,

$$\|X\| \leq 1.$$

Proof. For vectors $z \in \mathrm{Ker}(Z)$, we have,

$$x^* X x = x^*(X + Z^* Z)x \geq \lambda_{\min}(X + Z^* Z)||x||^2. \tag{7.18}$$

This augmented operator has the following block structure,

$$X + Z^* Z = \begin{bmatrix} I - Q^* Q & \\ & I - U^* U \end{bmatrix} + \begin{bmatrix} Q^* \\ U^* \end{bmatrix} \begin{bmatrix} Q & U \end{bmatrix} = \begin{bmatrix} I & Q^* U \\ U^* Q & I \end{bmatrix}.$$

Thus the eigenvalues λ of $X + Z^* Z$ satisfy,

$$\begin{bmatrix} I & Q^* U \\ U^* Q & I \end{bmatrix} \begin{bmatrix} v \\ \xi \end{bmatrix} = \lambda \begin{bmatrix} v \\ \xi \end{bmatrix},$$

or,

$$\begin{bmatrix} & Q^* U \\ U^* Q & \end{bmatrix} \begin{bmatrix} v \\ \xi \end{bmatrix} = (\lambda - 1) \begin{bmatrix} v \\ \xi \end{bmatrix}. \tag{7.19}$$

Solving for v from the block equation associated with the first row block of (7.19) and substituting into the second yields,

$$U^* Q Q^* U \xi = (\lambda - 1)^2 \xi.$$

Thus, the magnitudes of the shifted eigenvalues, $|\lambda - 1|$, are the square roots of the eigenvalues of $U^* Q Q^* U$. By a similarity transform, the eigenvalues of $U^* Q Q^* U$ are the same as the eigenvalues of the operator $Q Q^* U U^*$, and by the second part of Assumption 1, we know that these eigenvalues are bounded above by β. Thus,

$$|\lambda - 1| \leq \lambda_{\max}(Q Q^* U U^*)^{1/2} \leq \beta.$$

which implies,

$$1 - \beta \leq \lambda,$$

so that,

$$x^* X x \geq (1 - \beta) \|x\|^2,$$

from which the inf-sup bound directly follows.

Since $Q Q^*$ and $U U^*$ are damped projectors, their eigenvalues reside in the interval $[0, 1]$, as do the eigenvalues of $Q^* Q$ and $U^* U$. Using the definition of X in (7.13), this implies that the singular values of X reside in the interval $[0, 1]$, and so we have the upper bound $||X|| \leq 1$. □

Proposition 3 (Bound c for Z)**.** *The singular values of the preconditioned constraint are bounded below, with bound,*

$$0 < \sqrt{2\delta} \leq \sigma_{min}(Z).$$

Proof. Since U is invertible, $Z = \begin{bmatrix} Q & U \end{bmatrix}$ has full row rank. Thus the singular values of Z are the square roots of the eigenvalues of

$$ZZ^* = QQ^* + UU^*.$$

Recalling the arithmetic mean assumption (Assumption 1a), we have

$$0 < \delta \leq \frac{1}{2}\lambda_{\min}\left(QQ^* + UU^*\right) = \frac{1}{2}\lambda_{\min}\left(ZZ^*\right),$$

or

$$0 < \sqrt{2\delta} \leq \sigma_{\min}(Z).$$

□

7.4.2.2 Well posedness, continuity, and conditioning of the preconditioned KKT operator, E

Proposition 4 (Well posedness of E)**.** *The singular values of E have the following lower bound:*

$$0 < \frac{2}{3}(1-\beta)\delta \leq \sigma_{min}(E).$$

Proof. Based on the results of Proposition 2 and Proposition 3, and the block structure of E from (7.14), we can apply bound (7.17) from Theorem 9 to E with $a = 1-\beta$, $b = 1$, and $c^2 = 2\delta$. Doing this and then using the fact that $0 < \delta \leq 1$, we get the desired lower bound on the minimum singular value:

$$\sigma_{\min}(E) \geq \frac{1-\beta}{1+\frac{1}{2\delta}} = \frac{2(1-\beta)\delta}{1+2\delta} \geq \frac{2}{3}(1-\beta)\delta.$$

□

Proposition 5 (Continuity of E)**.** *The singular values of E are bounded above by* 2*. I.e.,*

$$\sigma_{max}(E) \leq 2.$$

Proof. To prove the upper bound, we directly estimate the quantity $|w_1^* E w_2|$ for arbitrary w_1, w_2. Denote the blocks of w_1 and w_2 by,

$$w_1 = \begin{bmatrix} p_1 \\ v_1 \\ \xi_1 \end{bmatrix}, \quad w_2 = \begin{bmatrix} p_2 \\ v_2 \\ \xi_2 \end{bmatrix}.$$

Recalling the blockwise definition of E from (7.12) and using the triangle inequality, we have

$$\begin{aligned} |w_1^* E w_2| &= \left| \begin{bmatrix} p_1^* & v_1^* & \xi_1^* \end{bmatrix} \begin{bmatrix} I - Q^*Q & & Q^* \\ & I - U^*U & U^* \\ Q & U & \end{bmatrix} \begin{bmatrix} p_2 \\ v_2 \\ \xi_2 \end{bmatrix} \right| \\ &= |p_1^*(I - Q^*Q)p_2 + p_1^* Q^* \xi_2 + v_1^*(I - U^*U)v_2 + v_1^* U^* \xi_2 + \xi_1^* Q p_2 + \xi_1^* U v_2| \\ &\leq |p_1^*(I - Q^*Q)p_2| + |p_1^* Q^* \xi_2| + |v_1^*(I - U^*U)v_2| + |v_1^* U^* \xi_2| + |\xi_1^* Q p_2| + |\xi_1^* U v_2|. \end{aligned} \tag{7.20}$$

Since the operators Q and U have singular values between zero and one, we can eliminate all of the intermediate operators in (7.20), yielding

$$|w_1^* E w_2| \leq \|p_1\| \|p_2\| + \|p_1\| \|\xi_2\| + \|v_1\| \|v_2\| + \|v_1\| \|\xi_2\| + \|\xi_1\| \|p_2\| + \|\xi_1\| \|v_2\|. \tag{7.21}$$

By Cauchy-Schwarz, three of the terms on the right hand side of (7.21) can be estimated as follows:

$$\begin{aligned} \|p_1\| \|p_2\| + \|v_1\| \|\xi_2\| + \|\xi_1\| \|v_2\| &\leq \left(\|p_1\|^2 + \|v_1\|^2 + \|\xi_1\|^2\right)^{1/2} \left(\|p_2\|^2 + \|v_2\|^2 + \|\xi_2\|^2\right)^{1/2} \\ &= \|w_1\| \|w_2\|. \end{aligned}$$

The other three terms can be estimated similarly:

$$\|p_1\| \|\xi_2\| + \|v_1\| \|v_2\| + \|\xi_1\| \|p_2\| \leq \|w_1\| \|w_2\|.$$

Thus we have the overall estimate

$$|w_1^* E w_2| \leq 2 \|w_1\| \|w_2\|,$$

which implies $\sigma_{\max}(E) \leq 2$, as required. □

Theorem 10 (Conditioning of E)**.**

$$\operatorname{cond}(E) \leq \frac{3}{(1-\beta)\delta}.$$

Proof. Divide the upper bound from Proposition 5 by the lower bound from Proposition 4. □

7.5 Spectral filtering and appropriate regularization assumptions

To better characterize the constants δ and β in the condition number bound in Theorem 10, in this section we propose *appropriate regularization assumptions* (Assumption 2) that limit the degree to which the inverse problem can be over- or under-regularized. These assumptions are motivated by an analysis of the error in the reconstruction of the parameter (Section 2.7.1), and apply to spectral filtering regularization operators (Definition 2). Since one part of Assumption 2 (specifically, Assumption 2b) is novel, we discuss that part in greater detail.

Since construction of spectral filtering regularization operators is too expensive for large-scale inverse problems with highly informative data, Assumption 2 is used for theoretical analysis only. In Section 7.6 we will prove that satisfying Assumption 2 implies the existence of good constants δ and β for source inversion problems, thereby guaranteeing that our preconditioner will perform well on these problems.

7.5.1 Spectral filtering regularization

Definition 2. *An operator R_0 is a spectral filtering regularization operator for a linear inverse problem with data misfit Hessian H_d^{gn} if R_0 and H_d^{gn} share a common basis of eigenvectors ϕ_k. We denote the eigenvalue of H_d^{gn} corresponding to ϕ_k by d_k, and the eigenvalue of R_0 corresponding to ϕ_k by r_k.*

By convention we order d_k in descending order ($d_k \geq d_{k+1}$). Note that the descending order for d_k forces an order (possibly non-monotone) for r_k.

Spectral filtering regularization is ideally suited for inverse problems—by manipulating the regularization singular values r_k, one can selectively filter out undesirable components of the parameter from the reconstruction without affecting the reconstruction of the desirable components. The larger r_k, the more component ϕ_k is penalized, and vice versa. Limiting cases of spectral filtering regularization include:

- identity regularization ($R = I$), where all singular vectors are penalized equally, and

- truncated SVD, where singular vectors ϕ_k are not penalized at all if d_k is above a given threshold, but are penalized infinitely[4] otherwise.

Spectral filtering regularization is routinely used for small to moderate sized inverse problems, and for large inverse problems that admit low-rank approximations to the parameter-to-observable map. However, aside from identity regularization, spectral filtering regularization is generally computationally infeasible for large-scale inverse problems with highly informative data. In fact, spectral filtering regularization requires computing the dominant singular vectors and singular values of G in order to construct R, and the number of dominant singular vectors of G scales with the informativeness of the data. Thus we view spectral filtering as an idealized form of regularization that practical regularization operators attempt to approximate. For a more comprehensive discussion of spectral filtering and its relation to other regularizations, we refer the reader to the classic monograph [83].

7.5.2 Appropriate regularization assumptions

In light of the discussion of over- and under-regularization in Section 2.7.1, we propose the following *appropriate regularization assumptions* for spectral filtering regularization operators:

Assumption 2 (Appropriate regularization)**.** *There exist constants μ and ν such that,*

a) $0 < \mu \leq d_k + \alpha r_k,$

b) $(d_k r_k)^{1/2} \leq \nu < \infty,$

for all k.

Assumption 2a is already required for the quadratic optimization problem (3.10) to be well-posed. It says that the regularization cannot be arbitrarily small in basis directions ϕ_k to which the observations are insensitive, but allows the regularization

[4]That is, the reconstruction of the component of q in the direction ϕ_k is set to zero.

to be arbitrarily small in directions ϕ_k to which the observations are sensitive. In contrast, Assumption 2b prevents the regularization from being large in basis directions ϕ_k to which the observations are sensitive, but still allows the regularization singular values to diverge ($r_k \to \infty$ as $k \to \infty$), as long as the sensitivity of the observations to changes to the parameter, d_k, goes to zero in the inverse manner. Informally, Assumption 2a says that the problem is not under-regularized, and Assumption 2b says that the problem is not over-regularized.

Since Assumption 2a is standard, we do not discuss it further. The motivation for Assumption 2b is less obvious, so we provide a more in-depth discussion of it. To begin with, the multiplicative nature of Assumption 2b makes it a relatively weak assumption compared to other possible candidates for preventing over-regularization. In particular, observe that the eigenvalues of $R_0^{-1}H^{\text{gn}}$, are $d_k/r_k + \alpha$. Thus situations in which the strength of R_0 on a mode is inversely proportional to how informed that mode is (i.e., $r_k \approx \frac{1}{d_k}$) can lead to arbitrarily poor conditioning of the regularization preconditioned Hessian while still satisfying Assumption 2b with a constant of order one.

An instructive model problem that illustrates Assumption 2b is the Poisson source inversion problem on a rectangular domain, with Laplacian regularization, zero Dirichlet boundary conditions for both A and R_0, and distributed observations of the first n_{obs} Fourier modes of the state variable in the domain. That is,

- $T = I$ and $Y = I$.
- $A = R_0^{1/2} = \Delta_D$, where Δ_D is the Laplacian operator with zero Dirichlet boundary conditions, and
- B is a wide rectangular operator with Fourier modes as right singular vectors (the same as A and R_0), but with singular values $\sigma_k = 1$, $k = 1, \ldots, n_{\text{obs}}$.

Substituting these operators into formula $H_d^{\text{gn}} = -T^* A^{-*} B^* Y B A^{-1} T$, and working in the basis of Fourier modes, we see that

$$d_k = \begin{cases} 1/\lambda_k^2, & k = 1, \ldots, n_{\text{obs}}, \\ 0, & k > n_{\text{obs}}, \end{cases}$$

where λ_k is the kth eigenvalue of Δ_D. At the same time, the singular values of R_0 are $r_k = \lambda_k^2$. Thus $d_k r_k = 1$ for $k = 1, \ldots, n_{\text{obs}}$ and $d_k r_k = 0$ for $k > n_{\text{obs}}$, so Assumption 2b holds with constant $\nu = 1$, regardless of the number of observations, n_{obs}.

7.6 Analysis of the source inversion problem with spectral filtering regularization

In Section 7.4.1.2 we hypothesized that the damped projector arithmetic and geometric mean assumptions (Assumption 1) are satisfied with good constants δ and β whenever an inverse problem is properly regularized. Then in Section 7.5 we formulated another assumption (Assumption 2) that quantifies the concept of proper regularization for spectral filtering regularization operators. Here we show that Assumption 2 implies Assumption 1 for the source inversion problem. Specifically, in Theorem 11 and Corollary 3 we prove quantitative bounds on the constants δ and β for source inversion problems that are neither over- nor under-regularized in the manner made precise by Assumption 2. The more appropriate to the problem the regularization is, the better the bounds.

Definition 3. *An inverse problem is a source inversion problem if the parameter q being inverted for is the right-hand-side of the state equation. That is, $T = -I$, and state equation takes the simplified form,*

$$Au = q,$$

where A does not depend on q.

Theorem 11. *Let R_0 be a spectral filtering regularization operator for a source inversion problem (see Definition 2 and Definition 3). If R_0 satisfies appropriate regularization Assumption 2 with constants μ and ν, then Assumption 1 is also satisfied, with constants*

$$\delta = \frac{1}{2}\left(1 + \frac{\alpha}{\rho^2}\nu^2\right)^{-1} \quad \text{and} \quad \beta = \left(1 + \frac{1}{\rho}\mu\right)^{-1/2}.$$

Proof. For δ, we seek a lower bound on the eigenvalues of the arithmetic mean of the damped projectors Π_R and Π_d (as defined in (7.15) and (7.16), respectively), while for

β we seek an upper bound on their geometric mean. For source inversion problems these damped projectors take the form

$$\Pi_R = \left(\frac{\alpha}{\rho} R_0 + I\right)^{-1} \quad \text{and} \quad \Pi_d = \left(\frac{1}{\rho} H_d^{\text{gn}} + I\right)^{-1}.$$

Furthermore, for spectral filtering regularization, R_0 and H_d^{gn} share the same eigenvectors, and have eigenvalues r_k and d_k, respectively. Thus the eigenvalues δ_k of the arithmetic mean $\frac{1}{2}(\Pi_R + \Pi_d)$ can be estimated as

$$\delta_k = \frac{1}{2}\left(\frac{1}{\frac{\alpha}{\rho} r_k + 1} + \frac{1}{\frac{1}{\rho} d_k + 1}\right) \geq \frac{1}{2}\left(1 + \frac{\alpha}{\rho^2} d_k r_k\right)^{-1} \geq \frac{1}{2}\left(1 + \frac{\alpha}{\rho^2}\nu^2\right)^{-1}.$$

In the first inequality we have combined fractions, and used the non-negativity of r_k, d_k and monotonicity of the function $f(x) = x/(a + x)$. In the second inequality we have used Assumption 2b.

Similarly, we use the Assumption 2a to bound the eigenvalues β_k of the geometric mean $(\Pi_R \Pi_d)^{1/2}$ as

$$\beta_k = \left(\frac{1}{\frac{\alpha}{\rho} r_k + 1} \cdot \frac{1}{\frac{1}{\rho} d_k + 1}\right)^{1/2} \leq \left(1 + \frac{\alpha}{\rho} r_k + \frac{1}{\rho} d_k\right)^{-1/2} \leq \left(1 + \frac{1}{\rho}\mu\right)^{-1/2}.$$

□

The following corollary of Theorem 11 shows that the preconditioner will be effective in the low to moderate regularization regime ($\alpha \leq 1$) if we choose $\rho = \sqrt{\alpha}$.

Corollary 3. *If the conditions of Theorem 11 are satisfied, and $\alpha \leq 1$, and the regularization parameter is chosen as $\rho = \sqrt{\alpha}$, then Assumption 1 is satisfied, with constants*

$$\delta = \frac{1}{2}\left(1 + \nu^2\right)^{-1} \quad \textit{and} \quad \beta = (1 + \mu)^{-1/2}.$$

Proof. Substituting in $\rho = \sqrt{\alpha}$ into the results of Theorem 11, we immediately have the desired lower bound on the arithmetic mean of damped projectors with constant $\delta = \frac{1}{2}(1 + \nu^2)^{-1}$. For the geometric mean, Theorem 11 implies

$$\lambda_{\max}\left(\Pi_R \Pi_d\right)^{1/2} \leq \left(1 + \alpha^{-1/2}\mu\right)^{-1/2}.$$

But note that for $\alpha \leq 1$ we have

$$\left(1 + \alpha^{-1/2}\mu\right)^{-1/2} \leq (1 + \mu)^{-1/2}, \tag{7.22}$$

and so we get the desired upper bound with $\beta = (1 + \mu)^{-1/2}$. □

7.7 Numerical results

We apply our method to a Poisson source inversion problem with pointwise observations randomly distributed throughout a rectangular domain $\Omega = [0, 1.45] \times [0, 1]$, using Laplacian regularization. Specifically, we take q, u, and v to reside in the space of continuous piecewise linear functions on a uniform triangular mesh with mesh size parameter h, with the L^2 inner product. The state equation

$$Au := \Delta_D u = q,$$

is the Poisson equation discretized by the finite element method, with homogeneous Dirichlet boundary conditions enforced by the symmetric Nitsche method [156]. Pointwise observations of the form

$$y_k = (Bu)_k = u(x_k),$$

are taken for a collection of points $\{x_k \in \Omega\}_{k=1}^{n_{\text{obs}}}$, shown in Figure 7.1. Noise is not included in the inverse problem since we are interested in preconditioners for the low noise, big data, small regularization limit. The regularization operator is defined by

$$R_0 := \Delta_N + tI,$$

where Δ_N is the Laplacian operator with Neumann boundary conditions discretized by the finite element method, and $t = 1/10$.

The true source field, q_{true}, used to generate the observations, y_k, is a grayscale image of the Peter O'Donnell Jr. building at the University of Texas at Austin, scaled to contain values in $[0, 1]$, and shown in Figure 7.1. The combination of sharp edges and smooth features in this image make this an ideal test case for highly informative data and small regularization.

Figure 7.1: Left: True source field q_{true} used for all inversions. Center: Reconstruction q for the case of $n_{\text{obs}} = 2000$ observations with regularization parameter $\alpha = 10^{-8}$ and mesh size $h = \sqrt{2} \cdot 10^{-2}$. Right: Observation locations x_k, denoted by dots.

Abstract vectors q, u, η are represented concretely by lists of nodal degrees of freedom $\mathbf{q}, \mathbf{u}, \boldsymbol{\eta}$, respectively. The norm of a concrete vector, e.g., $\|\mathbf{q}\|$, is the Euclidean norm (square root of the sum of the squares of the entries). Since we use uniform meshes and present only relative errors, this is spectrally equivalent to using the function space L^2 norm on the underlying function being represented by the concrete vector. We use the FEniCS [134] package to assemble concrete matrix representations of A, R_0, T, and I, which are denoted $\mathbf{A}$, $\mathbf{R}_0$, $\mathbf{T}$, and $\mathbf{W}$, respectively. The diagonal lumped mass matrix is denoted $\mathbf{W}_L$, with diagonal entries given by row sums of the mass matrix: $(\mathbf{W}_L)_{ii} = \sum_j \mathbf{W}_{ij}$. The concrete sparse matrix representation of the observation operator is denoted $\mathbf{B}$. Its (i, j) entry, $\mathbf{B}_{ij}$, equals the evaluation of the jth basis function at the ith observation location.

In a concrete basis, the KKT operator (7.1) becomes,

$$\begin{bmatrix} \alpha\mathbf{R}_0 & & -\mathbf{W} \\ & \mathbf{B}^T\mathbf{B} & \mathbf{A}^T \\ -\mathbf{W} & \mathbf{A} & \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \mathbf{u} \\ \boldsymbol{\eta} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{B}^T y \\ 0 \end{bmatrix}. \tag{7.23}$$

The reconstructed function q based on the exact[5] solution of this KKT system with regularization parameter $\alpha = 10^{-8}$ is shown in Figure 7.1.

In a concrete basis the preconditioner (7.2) becomes

$$\mathbf{P} = \begin{bmatrix} \alpha\mathbf{R}_0 + \rho\mathbf{W} & & \\ & \mathbf{B}^T\mathbf{B} + \rho\mathbf{A}^T\mathbf{W}^{-1}\mathbf{A} & \\ & & \frac{1}{\rho}\mathbf{W} \end{bmatrix}. \tag{7.24}$$

[5]By "exact," we mean that the result of a computation is accurate to tolerance 10^{-12} or smaller.

In our numerical experiments, we consider three variants of this preconditioner.

- **BDAL, exact**: all solves in preconditioner (7.24) are performed exactly.
- **BDAL, lumped mass, exact**: the mass matrix $\mathbf{W}$ is replaced with the lumped mass matrix $\mathbf{W}_L$, but preconditioner solves are performed exactly with this replacement.
- **BDAL, lumped mass, multigrid**: the mass matrix is replaced by the lumped mass matrix, and the solves for $\alpha\mathbf{R}_0+\rho\mathbf{W}_L$ and $\mathbf{B}^T\mathbf{B}+\rho\mathbf{A}^T\mathbf{W}_L^{-1}\mathbf{A}$ are replaced by a small number of algebraic multigrid V-cycles.

For algebraic multigrid we use the root-node smoothed aggregation [160, 189] method implemented in PyAMG [30], with the default settings. One V-cycle is used for $\alpha\mathbf{R}_0 + \rho\mathbf{W}_L$, and three V-cycles are used for $\mathbf{B}^T\mathbf{B} + \rho\mathbf{A}^T\mathbf{W}_L^{-1}\mathbf{A}$.

7.7.1 Convergence comparison

In Figure 7.2, we show a convergence comparison between between MINRES on the KKT system preconditioned by our block diagonal augmented Lagrangian preconditioner, and conjugate gradient on the Hessian preconditioned by the regularization term (CG-HESS). For our block diagonal augmented Lagrangian preconditioner, we also show results for lumped mass and algebraic multigrid approximations to the subsystems being solved. The regularization, forward, and adjoint solves used for the reduced Hessian solve are all performed exactly. The mesh size is $h = \sqrt{2} \cdot 10^{-2}$, the number of observations is 2000, and the regularization parameter is $\alpha = 10^{-8}$. Error is measured with respect to the converged solution to the linear system (7.23), i.e., $\|\mathbf{q} - \mathbf{q}_k\| / \|\mathbf{q}\|$. This allows us to make a fair comparison between the reduced and full space methods.

In terms of Krylov iteration count, our preconditioner far outperforms regularization preconditioning on the Hessian. The error in our method after three iterations is much less than the error after 50 iterations of regularization preconditioning on the reduced Hessian. Performance with the lumped mass approximation is almost identical to performance with exact solves. In the case with the multigrid approximation,

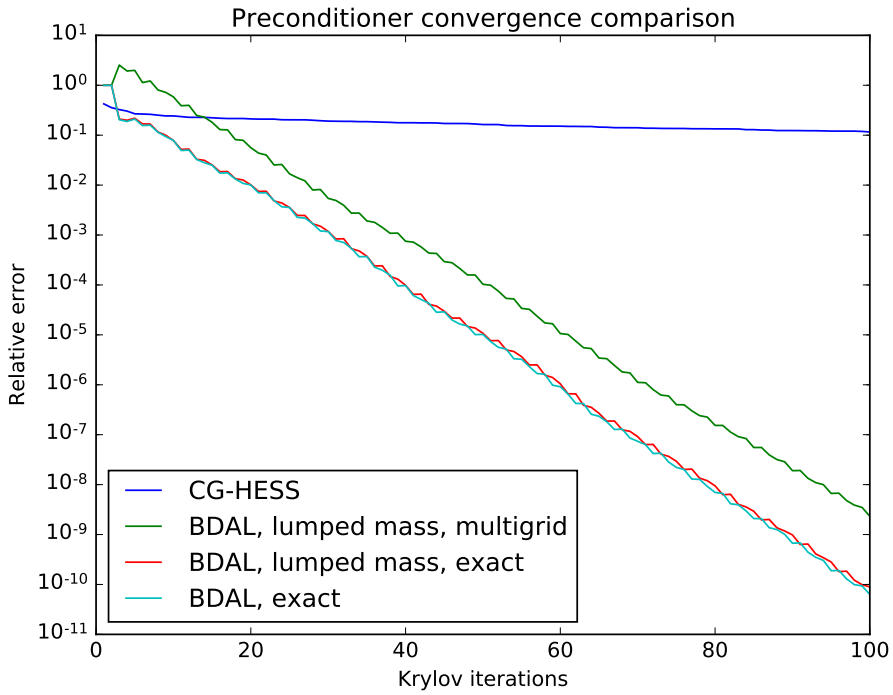


Figure 7.2: Relative error in the parameter, $\|\mathbf{q} - \mathbf{q}_k\| / \|\mathbf{q}\|$, for the big data Poisson source inversion problem, as a function of the number of Krylov iterations. The observation locations, regularization parameter, and mesh size are the same as in Figure 7.1 ($n_{\text{obs}} = 2000$, $\alpha = 10^{-8}$, $h = \sqrt{2} \cdot 10^{-2}$).

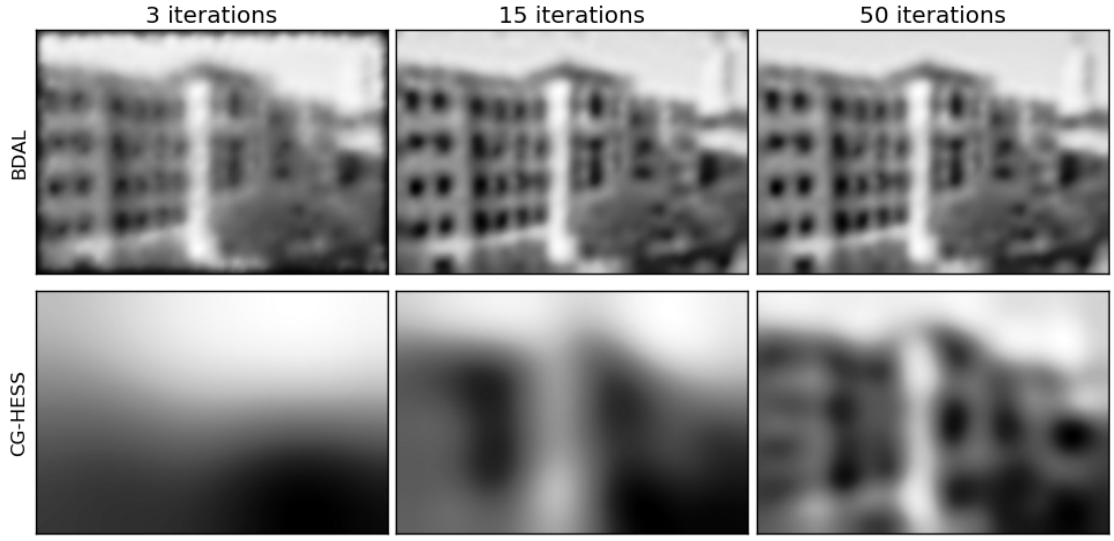


Figure 7.3: Visual comparison of the 3rd, 15th, and 50th Krylov iterates ($n_{\text{obs}} = 2000$, $\alpha = 10^{-8}$, $h = \sqrt{2} \cdot 10^{-2}$). Top row: reconstruction using MINRES on the KKT system with our "BDAL, lumped mass, exact" preconditioner. Bottom row: reconstruction using CG on the Hessian with regularization preconditioning.

Table 7.1: Mesh scalability study for our "BDAL, lumped mass, exact" preconditioner over a range of meshes. The table shows the number of MINRES iterations required to achieve parameter convergence to relative error 10^{-5}. The number of observations is $n_{\text{obs}} = 2000$, and the regularization parameter is $\alpha = 10^{-8}$. The observation locations x_k are the same for all mesh sizes.

h	# triangles	MINRES iterations
5.68e-02	1800	51
2.84e-02	7200	50
1.89e-02	16200	51
1.41e-02	29000	51
1.13e-02	45250	51
9.44e-03	65100	51
8.09e-03	88550	51
7.07e-03	116000	51
6.29e-03	146700	51
5.66e-03	181000	51

we see roughly the same asymptotic convergence rate as the exact solve, but with a lag of 10 to 20 iterations. In our numerical experiments we also observed that MINRES with our "BDAL, lumped mass, multigrid" preconditioner takes considerably less time per iteration than CG on the Hessian, which is expected since applying the Hessian requires solving the forward and adjoint equations to a high tolerance within each CG iteration.

In Figure 7.3, we see that the reconstruction using the Hessian starts off smooth, then slowly includes information from successively higher frequency parameter modes as the CG iterations progress. In contrast, our preconditioner applied to the KKT system reconstructs low and high frequency information simultaneously.

7.7.2 Mesh scalability

To test mesh scalability, we solve the Poisson source inversion problem on a sequence of progressively finer meshes using MINRES with our block diagonal augmented Lagrangian preconditioner. The same regularization parameter, $\alpha = 10^{-8}$, and observation locations, $\{x_k\}_{k=1}^{2000}$, are used for all meshes. The numbers of iterations k required to achieve a relative error of $\|\mathbf{q} - \mathbf{q}_k\| / \|\mathbf{q}\| < 10^{-5}$ are shown in Table 7.1. All meshes are uniform triangular meshes. The coarsest mesh has size

$h = 5.7 \cdot 10^{-2}$ with $1,800$ triangles, and the finest mesh has $h = 5.7 \cdot 10^{-3}$ with $181,000$ triangles. To quantify the error, the exact solution $\mathbf{q}$ was computed for each mesh using a sparse factorization of the KKT matrix. All results are based on the lumped mass approximation for mass matrices within the preconditioner.

The results clearly demonstrate mesh independence. The number of MINRES iterations required remains essentially constant over a two orders of magnitude increase in problem size, differing by at most one iteration across all mesh sizes.

7.7.3 Regularization and data scalability

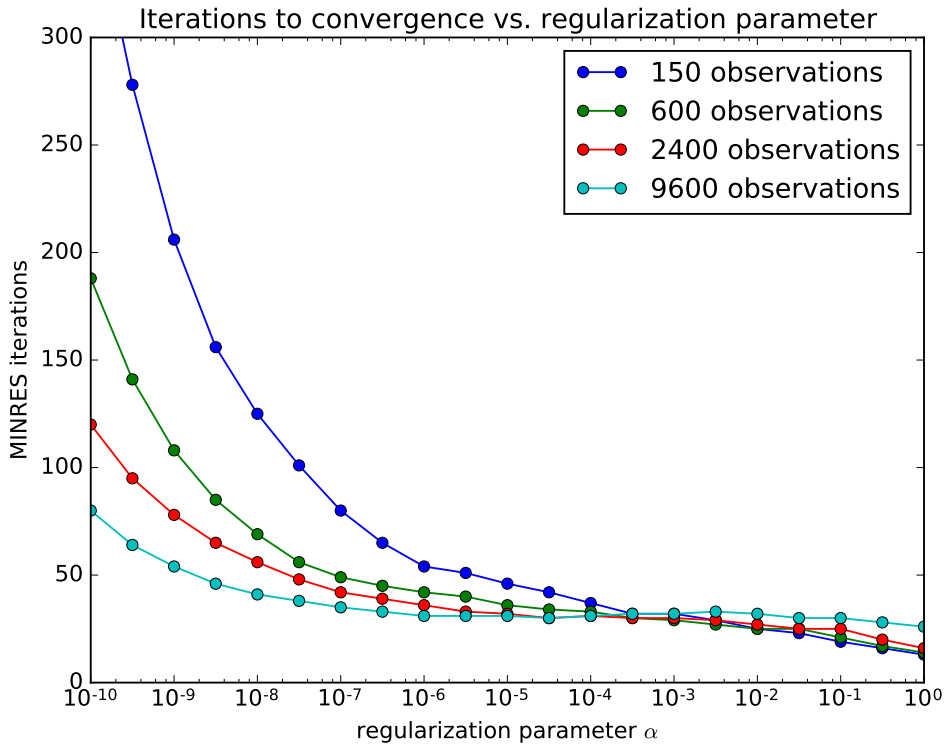


Figure 7.4: Regularization and data scalability study for our "BDAL, lumped mass, exact" preconditioner. Plot shows the number of MINRES iterations k required to achieve relative error $\|\mathbf{q} - \mathbf{q}_k\| \,/\, \|\mathbf{q}\| < 1e-5$.

A data and regularization robustness study is shown in Figure 7.4. The number of MINRES iterations k required for the method to converge to an error $\|\mathbf{q} - \mathbf{q}_k\| \,/\, \|\mathbf{q}\| < 10^{-5}$ is plotted for values of the regularization parameter in the range $\alpha \in [10^{-10}, 1.0]$, and number of observations $n_{\text{obs}} \in \{150, 600, 2400, 9600\}$. The mesh size is fixed at

$h = \sqrt{2} \cdot 10^{-2}$, and for each value of n_{obs}, the observation locations, x_k, are fixed as the regularization parameter varies.

The overall performance of the preconditioner is relatively steady over a broad range of values of α and n_{obs}. The performance of the method does decrease as the regularization parameter goes to zero for a fixed number of observations (upper left, Figure 7.4). However, the combination of small regularization parameter and small number of observations corresponds to the under-regularized regime, which we would not find ourselves in for an appropriately regularized problem. As the number of observations increases, the performance of the method improves in the small regularization regime while slightly worsening in the large regularization (over-regularized) regime, as suggested by our theory. This behavior is consistent with a data-scalable method: one can take small values for the regularization parameter if that choice is supported by the data available in the problem.

Chapter 8

Adaptive Product-Convolution Approximation

We present an adaptive product-convolution scheme for approximating locally translation-invariant operators.[1] That is, operators $A : l^2(\Omega) \to l^2(\Omega)$ satisfying

$$A[y, x] \approx A[y - x + p, p] \tag{8.1}$$

whenever x is not too far from p (see Figure 8.1). In this chapter, 'A' denotes a generic locally translation-invariant operator to be approximated (unlike the rest of the dissertation, in which A denotes the state operator). We consider the case in which Ω is a box[2] in $\mathbb{Z}^d$.

In this chapter, we detail the scheme and use it to build a preconditioner for the Hessian for the advection model problem. In Chapter 9 we will use the scheme to approximate a Schur complement that arises within a domain decomposition method for preconditioning the Hessian in the wave inverse problem. Our scheme is also well-suited for approximating or preconditioning operators that arise in Schur complement techniques [129, 174] for solving partial differential equations (PDEs), integral operators, covariance operators with spatially varying kernels, and Dirichlet-to-Neumann maps or other Poincaré–Steklov operators in multiphysics problems. These operators are typically dense and implicitly defined, and often do not admit a global low-rank approximation, making them difficult to approximate with standard techniques. We present numerical results using our adaptive product-convolution scheme to approximate other, non-Hessian, operators in Appendix E.

[1] This chapter contains content from, and is primarily based on, [7] (Nick Alger, Vishwas Rao, Aaron Myers, Tan Bui-Thanh, and Omar Ghattas. Scalable matrix-free adaptive product-convolution approximation for locally translation-invariant operators. arXiv preprint arXiv:1805.06018, 2018. Submitted.). *Required contribution and copyright statement:* Nick Alger framed the problem, developed the numerical algorithms, wrote the code, performed the theoretical analysis, and wrote the paper. The other coauthors engaged in regular helpful discussions about the

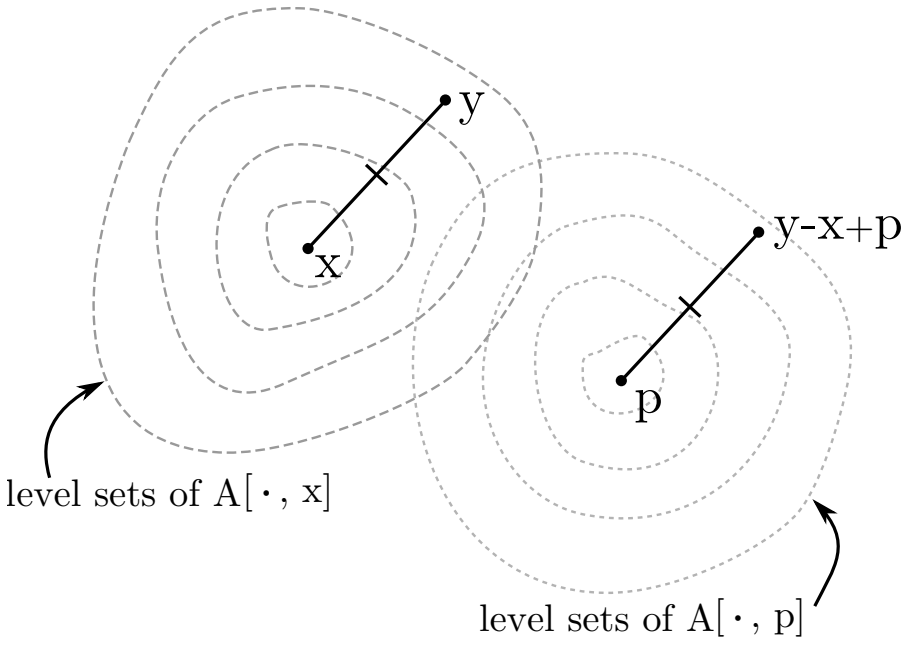


Figure 8.1: Our convolution-product scheme is suitable for operators that are locally approximately translation invariant. That is, operators for which the response at y to a point source at x is similar to the response at $y - x + p$ given a source of equal magnitude at p.

Let φ_p be the impulse response of A at p, i.e., the function created by applying A to a point source centered at point p, then translating the result to recenter it at 0:

$$\varphi_p[z] = (A\delta_p)[z + p], \quad z \in \Omega - p. \tag{8.2}$$

By "point source," δ_p, we mean the Kronecker delta that contains the value 1 at location p and zeros elsewhere. If A were translation-invariant (i.e., if (8.1) held with equality for all x, y), then A would be the convolution operator $A : f \mapsto \varphi_p * f$. To approximate operators that are only locally translation-invariant, we patch together a collection of convolution operators, each of which well-approximates A locally. Our approximation of A, denoted $\widetilde{A}$, takes the following form:

$$Af \approx \widetilde{A}f \coloneqq \sum_{k=1}^{r} \varphi_k^E * (w_k \cdot f), \tag{8.3}$$

where the w_k are locally supported weighting functions that overlap and form a partition of unity, '$\cdot$' denotes pointwise multiplication of functions, $*$ denotes convolution

work and helped edit the paper. Nick Alger holds the copyright on the paper.

[2]One can use our scheme in more general settings by mapping the domain to a box and interpolating functions onto a regular mesh.

(see Section 8.1.2 for more details on notation), and the functions φ_k^E are modified[3] versions of the (translated, recentered) impulse responses φ_{p_k} associated with a collection of sample points, p_k. Each point p_k is contained within the support of the associated weighting function w_k.

The basic form of (8.3) is known as a product-convolution approximation, and is well-established in the literature (see Section 6.7). Here we improve upon existing schemes by:

- Adaptively and automatically choosing the sample points p_k.
- Addressing issues related to boundaries.

In Section 8.1 we overview our results and present prerequisite background material. In Section 8.2 we derive our scheme, explain how we choose p_k, and detail the process for constructing w_k and φ_k^E. In Section 8.3 we detail how $\widetilde{A}$ can be used once constructed, including how to efficiently convert it to hierarchical matrix (H-matrix) format. In Section 8.4 we perform an a-priori error analysis of our scheme. We demonstrate our scheme numerically on the advection-diffusion Hessian in Section 8.5.

8.1 Background

8.1.1 Overview of results

The scheme we present is matrix-free in the sense that constructing $\widetilde{A}$ only requires the ability to apply A and its adjoint, A^*, to vectors. Access to the matrix representation of A is not needed. Once constructed, we can compute any matrix entry of $\widetilde{A}$ in $O(1)$ work. We can apply $\widetilde{A}$ and $\widetilde{A}^*$ to vectors in nearly linear work using the fast Fourier transform (FFT). Blocks of $\widetilde{A}$ and $\widetilde{A}^*$ can be applied to vectors in work that is nearly linear in the size of the block.

Often the ultimate goal is to solve linear systems with A as the coefficient operator. Krylov methods can be used to solve these systems [58]. However, the convergence of Krylov methods depends heavily on the spectral structure of the coefficient operator,

[3] To address issues with boundary artifacts, we construct φ_k^E by extending the function φ_{p_k} outside of $\Omega - p_k$ using information from neighboring functions, φ_{p_j} (more on this in Section 8.2.5).

leading to slow convergence when A is ill-conditioned. To address this, we explain how $\widetilde{A}$ can be efficiently converted to H-matrix format. Once in H-matrix format, $\widetilde{A}$ can be efficiently factorized or inverted using H-matrix arithmetic, then used as a preconditioner. Alternatively, one can build circulant preconditioners from $\widetilde{A}$ [60, 149].

We choose the sample points, p_k, in an adaptive grid: in regions where the error is large, we refine the grid. The effect of this refinement process is to place more sample points in regions where A is less translation-invariant, and fewer sample points in regions where A is more translation-invariant. The adaptivity is performed using a randomized a-posteriori error estimator.

Boundaries introduce two difficulties for product-convolution schemes:

1. **Boundary artifacts:** The impulse response associated with p_k is naturally defined on $\Omega - p_k$, but the product-convolution scheme (8.3) requires it to be defined on a larger set. The three standard extension techniques—extending the impulse response by zero, reflecting it across the boundary, or replicating it periodically—all create boundary artifacts wherever artificial data are used in place of undefined data.

2. **Boundary effects:** The underlying operator may fail to be translation-invariant near boundaries due to boundary conditions or other physically meaningful effects.

To overcome 1, we extend the support of the impulse responses using information from neighboring impulse responses. To overcome 2, we use anisotropic adaptivity. Our adaptive refinement scheme senses the coordinate direction in which A is least translation-invariant within a cell, and preferentially subdivides the cell in that direction. This allows the scheme to efficiently approximate operators that are not translation-invariant in directions perpendicular to boundaries, but are translation-invariant in directions parallel to boundaries. Boundary effects due to boundary conditions typically exhibit this direction-dependent form of translation-invariance (regardless of the type of boundary condition).

In Theorem 12, we prove that the error in our scheme is controlled by the local failure of translation-invariance in A. This, together with adaptivity, implies convergence: our scheme will continue to add new sample points until it achieves the desired error tolerance. The more translation-invariant A is, the fewer sample points will be used. Additionally, Theorem 12 implies that our approximation scheme will not introduce boundary artifacts. Without our impulse response extension procedure, the bound in Theorem 12 would fail near the boundary.

We demonstrate the scheme on a spatially varying blur operator, on the non-local component of an interface Schur complement for the Poisson operator, and on the data misfit Hessian for an advection dominated advection-diffusion inverse problem. Our scheme outperforms existing methods:

- Our scheme converges much faster than non-adaptive product-convolution approximation for the spatially varying blur operator.
- The number of sample points required to approximate the non-local component of the Poisson Schur complement is independent of the mesh size.
- Approximation using a small number of sample points yields a high quality preconditioner for the Poisson Schur complement.
- The number of sample points required to approximate the advection-diffusion Hessian is independent of the Peclet number, a proxy for the informativeness of the data in the inverse problem.
- A Hessian preconditioner that results from using our approximation performs well even if the Peclet number is large.

We also find that the randomized a-posteriori error estimator performs much better than standard theory predicts: we see that it performs almost as well with 5 random samples as it does with 100.

Although our scheme will eventually converge to any desired error tolerance, it is most useful for computing moderately accurate approximations (say, 80% to 99% accurate) of "difficult" operators that are poorly approximated by standard techniques.

In our numerical tests, we observe that the convergence slows beyond this accuracy. Moderate accuracy approximation is sufficient for many engineering applications, and is ideal for building preconditioners.

8.1.2 Setting and notation

In this chapter we work in l^2 spaces on $\mathbb{Z}^d$ or subsets of $\mathbb{Z}^d$; these spaces arise when one discretizes a function on a continuous domain using a regular grid. We write $\|\cdot\|_{\text{Fro}}$ to denote the Frobenius norm, and $\|\cdot\|_{\text{Fro}(X)}$to denote the square root of the sum of squares of all entries of an operator corresponding to indices in X. So, for example, $\|A\|_{\text{Fro}} = \|A\|_{\text{Fro}(\Omega\times\Omega)}$.

We routinely encounter Cartesian products of intervals, which we call *boxes* and denote with a capital letter in sans-serif font, e.g., C. Boxes are characterized by their *minimum point* and *maximum point*: the points in the box that are component-wise less than or equal to all other points in the box, or greater than or equal to all other points in the box, respectively. We denote the minimum and maximum points of a box with the same letter as the box, but lower-case, and with the subscripts "min" and "max", respectively. For example, $\mathsf{C} = \bigtimes_{i=1}^{d} [c^i_{\text{min}}, c^i_{\text{max}}]$, where $\bigtimes$ is the Cartesian product of sets. We write $\text{corners}(\mathsf{C}) \coloneqq \bigtimes_{i=1}^{d} \{c^i_{\text{min}}, c^i_{\text{max}}\}$ to denote the set of corners of C. The (approximate) midpoint, c_{mid}, of the box C is the integer vector closest to the real vector $(c_{\text{max}} + c_{\text{min}})/2$. The *linear dimension* of a box is the sum of all the dimensions of the box: $\sum_{i=1}^{d} c^i_{\text{max}} - c^i_{\text{min}}$.

Minkowski set arithmetic is used for addition and subtraction of one set with another set, negation of a set, and addition and subtraction of a set with a point:

$$X + Y = \{x + y : x \in X, y \in Y\}, \qquad X - Y = \{x - y : x \in X, y \in Y\},$$

and similar for negation of a set, and addition and subtraction of a point from a set. The number of elements in a set X is denoted $|X|$. We reserve N for the total number of points in the domain: $N \coloneqq |\Omega|$.

The evaluation of f at x is denoted $f[x]$, and $f[\mathsf{C}] \in l^2(\mathsf{C} - c_{\text{min}})$, with $(f[\mathsf{C}])[x] \coloneqq f[x + c_{\text{min}}]$. Likewise, $A[y, x]$ is the (y, x) "matrix entry" of A, and $A[\mathsf{T}, \mathsf{S}] \in l^2((\mathsf{T} - t_{\text{min}}) \times (\mathsf{S} - s_{\text{min}}))$ with $(A[\mathsf{T}, \mathsf{S}])[y, x] \coloneqq A[y + t_{\text{min}}, x + s_{\text{min}}]$. That is, $A[\mathsf{T}, \mathsf{S}]$

is the T, S "block" of A. A dot within indexing brackets, as in $A[\mathsf{C}, \ \cdot\ ]$ or $A[\ \cdot\ , \mathsf{C}]$, indicates the matrix of all columns or rows of A corresponding to points in C, respectively. The action of a linear operator A on a vector f is denoted Af. We write A^* to denote the adjoint of A. That is, $A^*[y, x] = \overline{A[x, y]}$, where the over-line indicates the complex conjugate.

A dot between two functions denotes *pointwise multiplication* of those functions:

$$(f \cdot g)[x] \coloneqq f[x]\, g[x]\,.$$

An asterisk between two functions denotes *convolution* of those functions:

$$(\psi * f)[y] \coloneqq \sum_{x \in \mathbb{Z}^d} f[x]\, \psi[y - x]\,. \tag{8.4}$$

If the domains of functions f, ψ are only subsets of $\mathbb{Z}^d$, we define their convolution to be the result of extending f, ψ by zero so that they are defined on all of $\mathbb{Z}^d$, then convolving them using formula (8.4). We use the term "convolution rank" to denote the number of terms in a weighted sum of convolution operators (e.g., r in (8.3)).

We define the functions

$$\delta_p[x] := \begin{cases} 1, & x = p, \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad \mathbb{1}_X := \begin{cases} 1, & x \in X, \\ 0, & \text{otherwise.} \end{cases}$$

We denote the support of a function f by $\operatorname{supp}(f)$. By the "support" of a function, we mean the largest set on which the function could, in principle, be non-zero (independent of whether the numerical value of the function happens to be zero). We call a function of N *nearly linear* if it scales as $O(N \log^a N)$ for $N \to \infty$, where a is some small non-negative integer (say $a \in \{0, 1, 2\}$).

8.1.3 Product-convolution vs. convolution-product

Product-convolution schemes perform element-wise products with weighting functions first and convolutions second, while convolution-product schemes reverse this order:

$$Af \approx \underbrace{\sum_{k=1}^{r} \psi_k * (\omega_k \cdot f)}_{\text{product-convolution}} \quad \text{vs.} \quad \underbrace{\sum_{k=1}^{r} \omega_k \cdot (\psi_k * f)}_{\text{convolution-product}}\,. \tag{8.5}$$

Since the entries of a convolution operator $L : f \mapsto \psi * f$ are $L[y, x] = \psi[y - x]$, product-convolution and convolution-product approximations have the following (y, x) matrix entries:

$$A[y,x] \approx \underbrace{\sum_{k=1}^{r} \omega_k[x]\,\psi_k[y-x]}_{\text{product-convolution}} \quad \text{vs.} \quad \underbrace{\sum_{k=1}^{r} \omega_k[y]\,\psi_k[y-x]}_{\text{convolution-product}}. \tag{8.6}$$

Both schemes are non-symmetric, but the adjoint of a product-convolution operator is a convolution-product operator, and vice versa. The operators defined by the following actions are adjoints of each other:

$$\underbrace{\sum_{k=1}^{r} \psi_k * (\omega_k \cdot f)}_{\widetilde{A}f} \quad \overset{\text{adjoint}}{\longleftrightarrow} \quad \underbrace{\sum_{k=1}^{r} \overline{\omega}_k \cdot \left(\text{flip}\left(\overline{\psi_k}\right) * f\right)}_{\widetilde{A}^* f}, \tag{8.7}$$

where $\text{flip}(\psi)[x] \coloneqq \psi[-x]$, and the over-line indicates the complex conjugate. Here we use a product-convolution scheme.

8.2 The adaptive product-convolution approximation

If A were translation-invariant (i.e., if (8.1) held with equality for all $x, y \in \mathbb{Z}^d$), then A would be the convolution operator defined by the action $Af = \varphi_p * f$, where φ_p is the impulse response of A at p, as defined in (8.2). For example, the solution operator for a homogeneous PDE on an unbounded domain is translation-invariant, and φ_p is the Green's function for the PDE. Of course, translation-invariant operators are rare in practice. It is more common for A to only be *approximately* translation-invariant (see Figure 8.1), and for the approximate translation-invariance to be valid only *locally*. That is,

$$A[p + y - x, p] \approx A[y, x] \quad \text{when } x \in U \tag{8.8}$$

for some neighborhood U consisting of points "near" p. We will provide a rigorous analysis of approximation errors in Section 8.4; for now we leave the exact nature of this approximate equality ($\approx$) intentionally vague. Just as translation-invariance of A implies that A is a convolution operator, local approximate translation-invariance of

A implies that A can be locally approximated by a convolution operator. Specifically, (8.8) implies

$$Ag \approx \varphi_p * g \quad \text{when } \operatorname{supp}(g) \subset U. \tag{8.9}$$

In order to approximate the action of A on functions f supported on a larger region of interest, we patch together local convolution operator approximations. Let $\{U_k\}_{k=1}^r$ be a collection of sets covering $\operatorname{supp}(f)$, let $\{w_k\}_{k=1}^r$ be a partition of unity subordinate to this cover, let $p_k \in U_k$ for $k = 1, \ldots, r$, and define $\varphi_k \coloneqq \varphi_{p_k}$. If the following local approximations hold:

$$Ag \approx \varphi_k * g \quad \text{when } \operatorname{supp}(g) \subset U_k, \quad k = 1, \ldots, r, \tag{8.10}$$

then A can be globally approximated as follows:

$$Af = A \sum_{k=1}^{r} w_k \cdot f = \sum_{k=1}^{r} A(w_k \cdot f) \approx \sum_{k=1}^{r} \varphi_k * (w_k \cdot f). \tag{8.11}$$

The first equality follows from the partition unity property of the functions w_k, the second follows from the linearity of A, and the approximate equality follows from the local approximation property (8.10) and the fact that $\operatorname{supp}(w_k \cdot f) \subset U_k$.

8.2.1 Overview of the approximation

The previous derivation leads us to approximate A with the following *product-convolution approximation*:

$$\widetilde{A}f \coloneqq \sum_{k=1}^{r} \varphi_k^E * (w_k \cdot f), \tag{8.12}$$

where

- $\left\{\varphi_k^E\right\}_{k=1}^r$ are modified ("extended") versions of the impulse responses

  $$\varphi_k[z] = (A\delta_{p_k})[z + p_k], \quad z \in \Omega - p_k, \tag{8.13}$$

  for a collection of *sample points* $\{p_k\}_{k=1}^r$.

- The sample points $\{p_k\}_{k=1}^r$ reside in a collection of overlapping sets $\{U_k\}_{k=1}^r$ that cover Ω:

  $$p_k \in U_k \text{ for } k = 1, \ldots, r \quad \text{and} \quad \Omega \subset \bigcup_{k=1}^{r} U_k.$$

- $\{w_k\}_{k=1}^r$ is a partition of unity subordinate to the cover:

$$\text{supp}(w_k) \subset U_k \text{ for } k = 1, \ldots, r \quad \text{and} \quad \sum_{k=1}^{r} w_k[x] = 1 \quad \text{for all } x \in \Omega.$$

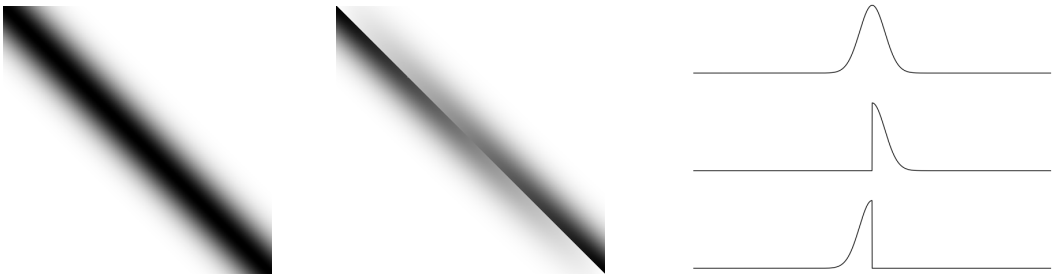
Our scheme is defined by the points p_k, the sets U_k, the partition of unity weighting functions w_k, and the extended impulse response functions φ_k^E.

In general, translation-invariance varies spatially. By this, we mean that the size of the neighborhood U on which the error in (8.8) is sufficiently small depends on the location of U. To fix ideas, suppose that A is the solution operator for an inhomogeneous elliptic PDE. In this case, the size of U will typically be small if the coefficient in the PDE varies over short length scales within U, and large if the coefficient varies over large length scales within U. In order to capture such spatial variations in translation-invariance while minimizing the number of sample points used, we choose p_k and U_k adaptively (Section 8.2.2 and Section 8.2.3). A randomized adjoint based a-posteriori error estimator (Section 8.2.6) drives the adaptivity.

Due to boundary effects, translation-invariance typically fails in directions perpendicular to a boundary, but holds in directions parallel to that boundary. For example, let φ_p be the Green's function at p for a homogeneous PDE on an infinite half-space. Although φ_p changes as p approaches the boundary, by symmetry it does not change as p moves parallel to the boundary. In order to address this direction-dependent translation-invariance, we refine anisotropically, subdividing preferentially in directions that φ_p changes the most as a function of p (Section 8.2.7).

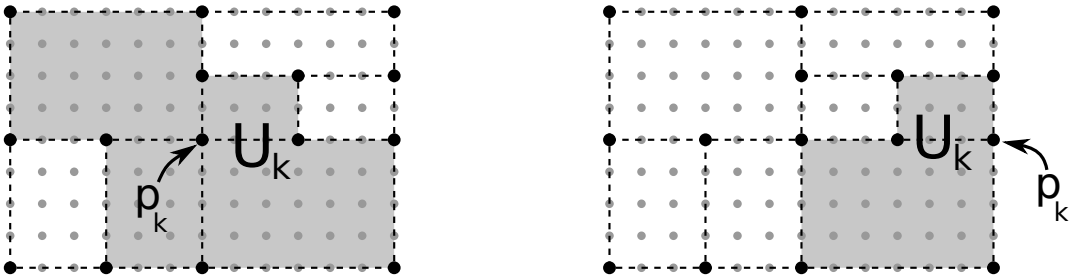
The adaptive refinement procedure creates unusually shaped neighborhoods U_k. We construct harmonic weighting functions, w_k, on these sets by solving local Laplace problems (Section 8.2.4).

Because of boundaries, the domains of definition of the functions φ_k are not large enough for the convolutions in the naive product-convolution formula, $\sum_{k=1}^r \varphi_k * (w_k \cdot f)$, to be well-defined. Extending functions by zero as needed makes these convolutions well-defined, but this leads to boundary artifacts wherever zeros are used in place of undefined data. These boundary artifacts are purely a side effect of the scheme and are unrelated to real boundary effects present in the underlying



(a) A (b) $\widetilde{A}$ (c) Top: φ. Mid: φ_{left}. Bot: φ_{right}.

Figure 8.2: Extending impulse responses by zero leads to boundary artifacts even if A is, itself, a convolution operator. Here A (8.2a) takes a function defined on $[1, N]$, extends it by zero to $\mathbb{Z}$, convolves it with a Gaussian φ (8.2c), then restricts the result to $[1, N]$. The approximation, $\widetilde{A}$ (8.2b), linearly interpolates between convolution with φ_{left} at 1 and φ_{right} at N, where φ_{left} and φ_{right} (8.2c) are the impulse responses of A to point sources centered at 1 and N, respectively, with extension by zero used as needed. Black indicates value 1, and white indicates value 0 in 8.2a and 8.2b.



(a) Blocky neighborhood U_k associated with an interior sample point p_k.

(b) Blocky neighborhood U_k associated with a boundary sample point p_k.

Figure 8.3: Sample points p_k (black points) form an adaptively refined grid within Ω (all gray and black points). The blocky neighborhood U_k associated with sample point p_k (shaded light gray region) is the union of all leaf cells that contain p_k.

operator A; they occur even in the case where A is, itself, a convolution operator (see Figure 8.2). To eliminate such boundary artifacts, we extend the functions φ_k outside of their natural support by using information from neighboring functions φ_j to create "extended" impulse response functions φ_k^E (Section 8.2.5).

8.2.2 Adaptive grid structure

We will choose the sample points, p_k, so that they form an adaptively refined rectilinear grid (for example, see Figure 8.3). This section defines the structure of the adaptive grid; the procedure for constructing it will be explained in Section 8.2.3.

We organize the domain Ω into a binary tree, $\mathcal{T}$, of boxes $\mathsf{C} \subset \Omega$ which we call *cells*. The root of $\mathcal{T}$ is the whole domain Ω. Cells may be either refined or not refined; refined cells are internal nodes in $\mathcal{T}$ and unrefined cells are leaves of $\mathcal{T}$. We denote the set of all leaves of the tree by leaves($\mathcal{T}$). Refined cells C are subdivided in a chosen direction into a set of two child cells that share an internal facet (more about how we choose the subdivision direction in Section 8.2.7). We denote the set of children of C by children(C). The corners of all cells form the set of sample points:

$$\{p_k\}_{k=1}^{r} = \bigcup_{\mathsf{C} \in \mathcal{T}} \text{corners}(\mathsf{C}).$$

Since the cells share facets, typically more than one cell contains a given sample point. We write

$$\text{cells}(p_k) \coloneqq \{\mathsf{C} : \mathsf{C} \in \text{leaves}(\mathcal{T}), p_k \in \mathsf{C}\}$$

to denote the set of all leaf cells containing p_k. We define the *blocky neighborhood*, U_k, associated with a sample point p_k as the union of all leaf cells containing p_k:

$$U_k \coloneqq \bigcup_{\mathsf{C}_i \in \text{cells}(p_k)} \mathsf{C}_i.$$

Sample points p_k and p_j are *neighbors* if they share a common leaf cell. That is, there exists a leaf cell C such that $p_k \in \mathsf{C}$ and $p_j \in \mathsf{C}$. Note that under this definition p_k is neighbors with itself. We write $\text{nbrs}(k) \subset \{1, \ldots, r\}$ to denote the set of indices of sample points that are neighbors of p_k, including p_k itself. In other words, $j \in \text{nbrs}(k)$ if p_k and p_j are neighbors.

8.2.3 Adaptive refinement algorithm

Starting with Ω subdivided once in all directions, we repeatedly estimate the error in all cells in leaves ($\mathcal{T}$) using an a-posteriori error estimator, then refine the leaf cell with the largest error. The refinement process continues until either (a) the desired error in the approximation is achieved, or (b) a predetermined maximum number of sample points p_k is reached. At each step of the refinement process we construct or modify the functions w_k and φ_k^E using methods that will be described in Section 8.2.4, Section 8.2.5, and Section 8.2.8. We perform the a-posteriori error estimation

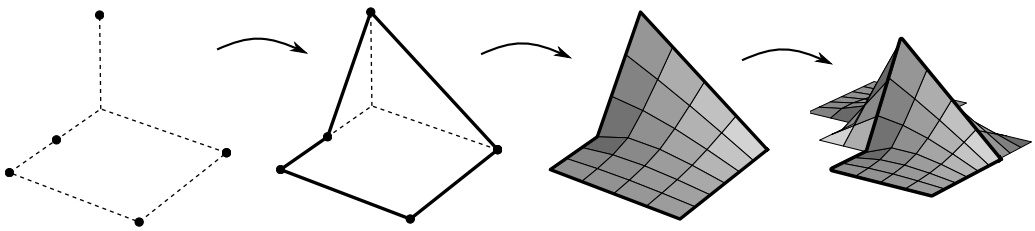


Figure 8.4: Construction of w_k for $d = 2$. For each box in U_k we assign w_k the value 1 at sample point p_k and 0 at all other sample points. For edges between sample points, we compute the values of w_k by solving the discrete 1-D Laplace equation, using the previously assigned values at sample points as Dirichlet boundary conditions. For faces, we compute the values of w_k by solving the discrete 2-D Laplace equation, using the previously computed edge values as Dirichlet boundary conditions. Finally, we form w_k on U_k by combining its constituent pieces on each box.

with a randomized method that will be described in Section 8.2.6. We choose which direction to subdivide cells in using a method that will be described in Section 8.2.7. The complete algorithm is summarized in Algorithm 1.

8.2.4 Harmonic weighting functions

We construct harmonic partition of unity weighting functions, w_k, by solving discrete local Laplace (diffusion) problems recursively on subsets of U_k. This process is equivalent to the construction of harmonic basis functions in finite element methods [41], and also shares conceptual ties with partition of unity finite element methods [20] and the construction of coarse basis functions in agglomerated element algebraic multigrid [127].

The blocky neighborhood U_k is a union of d-dimensional boxes. The boundary of each d-dimensional box is a union of $(d-1)$-dimensional facets, each of which is a box. There are $2d$ facets, corresponding to either the front or the back of the box in each coordinate direction. Facets that contain hanging nodes ("broken facets") are the union of several smaller $(d-1)$-dimensional boxes. Hence the boundary of each d-dimensional box can be expressed as the union of $(d-1)$-dimensional boxes, where we exclude broken facets in favor of their constituent smaller boxes. In the same way, the boundary of each $(d-1)$-dimensional box is a union of $(d-2)$-dimensional boxes, and so forth all the way down until we reach a set of 0-dimensional sample points. We

build harmonic weighting functions by solving the Laplace equation ($-\Delta w_k = 0$) on these boxes recursively in dimension, using the values from lower-dimensional boxes as Dirichlet boundary conditions for higher-dimensional boxes. For sample points p_j (the lowest level), we assign $w_k[p_k] = 1$ and $w_k[p_j] = 0$ for $j \neq k$. Figure 8.4 illustrates this process for $d = 2$. Linearity, the maximum principle, and induction on boxes of increasing dimension show that the functions w_k form a partition of unity on Ω.

For the discrete Laplace equation we use the (positive definite) discrete graph Laplacian; this is equivalent to discretizing the continuous Laplacian using a standard Kronecker sum finite difference approximation on a regular grid. The local Laplace problems can be solved efficiently (in time proportional to the number of unknowns) with multigrid [21, 48].

8.2.5 Extended impulse response functions

To construct φ_k^E, we first compute the impulse responses φ_k of A at the points p_k by applying A to point sources, then translating the results (see (8.13)). To eliminate boundary artifacts, we create φ_k^E by extending the support of φ_k, using data from neighboring functions φ_j to fill in regions outside of $\operatorname{supp}(\varphi_k)$.

1. For z within $\operatorname{supp}(\varphi_k)$, we set $\varphi_k^E[z] \coloneqq \varphi_k[z]$.

2. For z outside $\operatorname{supp}(\varphi_k)$ but within $\operatorname{supp}(\varphi_j)$ for at least one neighboring φ_j, we define $\varphi_k^E[z]$ as the average of all neighboring $\varphi_j[z]$ whose support contains z.

3. For z outside $\operatorname{supp}(\varphi_k)$ and outside $\operatorname{supp}(\varphi_j)$ for all neighboring φ_j, we set $\varphi_k^E[z] \coloneqq 0$.

Figure 8.5 illustrates this procedure for a 1-dimensional example. Our theory still holds if we use any weighted average of neighboring $\varphi_j[z]$ in Step 2, provided the weights are non-negative and sum to one. We use the average since it simplifies the implementation and the explanation, and since more elaborate schemes are likely to yield only minimal improvements. The fact that we set some entries of $\varphi_k^E[z]$ to zero

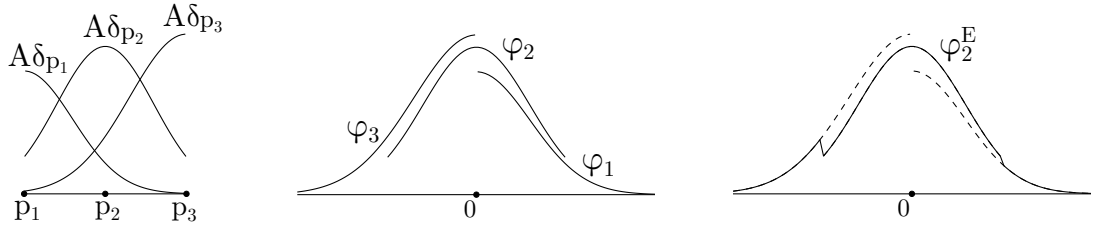


Figure 8.5: Illustration of impulse response extension procedure in 1 dimension. To construct φ_2^E, we extend the support of φ_2 by filling in regions where φ_2 is undefined with values from φ_1 and φ_3.

in Step 3 is irrelevant since our scheme never accesses these entries (this will follow from Proposition 6).

In preparation for the theory in Section 8.4, we now describe the process of constructing φ_k^E more precisely. First, we construct the following counting functions:

$$c_k \coloneqq \mathbb{1}_{\Omega - p_k} + \sum_{\substack{j \in \text{nbrs}(k) \\ j \neq k}} \mathbb{1}_{(\Omega - p_j) \setminus (\Omega - p_k)}.$$

Since $\text{supp}(\varphi_j) = \Omega - p_j$, $c_k[z]$ counts how many φ_j will contribute to $\varphi_k^E[z]$. Next we compute

$$v_k[z] \coloneqq \begin{cases} 1/c_k[z], & z \in \text{supp}(c_k) \\ 0, & \text{otherwise,} \end{cases}$$

and define

$$v_k^{(j)} \coloneqq \begin{cases} v_k \cdot \mathbb{1}_{\Omega - p_k}, & j = k \\ v_k \cdot \mathbb{1}_{(\Omega - p_j) \setminus (\Omega - p_k)}, & j \in \text{nbrs}(k),\ j \neq k, \end{cases} \tag{8.14}$$

The function $v_k^{(j)}[z]$ is the weight given to neighboring impulse response φ_j at point z when constructing φ_k^E. Finally, we construct φ_k^E:

$$\varphi_k^E \coloneqq \sum_{j \in \text{nbrs}(k)} v_k^{(j)} \cdot \varphi_j. \tag{8.15}$$

8.2.6 Randomized a-posteriori error estimator

In order to decide which cells to refine, we wish to compute the error in the approximation,

$$e_{\mathsf{C}} \coloneqq \left\| \left(\widetilde{A} - A \right) [\Omega, \mathsf{C}] \right\|, \tag{8.16}$$

for all cells $\mathsf{C} \in \text{leaves}(T)$. Computing these norms is prohibitively expensive, so instead we estimate them. If M is any matrix with N columns, then the following sample average approximation estimates the square of its Frobenius norm:

$$||M||^2 = \mathbb{E}\left(||M\zeta||^2\right) \approx \frac{1}{m}\sum_{i=1}^{m} ||M\zeta_i||^2 = \frac{1}{m}\left\|MZ\right\|^2, \tag{8.17}$$

where $\zeta, \zeta_i \sim N(0,1)^N$, are independent and identically distributed (i.i.d.) Gaussian random vectors, $\mathbb{E}$ is the expected value, m is the number of samples used in the sample average approximation, and $Z \sim N(0,1)^{N\times m}$ is an i.i.d. Gaussian random matrix (the matrix with columns ζ_i) [18]. Hence we can form an estimator, $\eta_{\mathsf{C}} \approx e_{\mathsf{C}}$, by forming a random matrix $Z \sim N(0,1)^{N\times m}$, computing

$$Y = A^*Z \quad \text{and} \quad \widetilde{Y} = \widetilde{A}^*Z,$$

then extracting blocks of the results, and taking norms:

$$\eta_{\mathsf{C}} \coloneqq \frac{1}{\sqrt{m}}\|\widetilde{Y}\left[\mathsf{C},\ \cdot\ \right] - Y\left[\mathsf{C},\ \cdot\ \right]\|. \tag{8.18}$$

By performing the randomized sample average approximation with the adjoints A^* and $\widetilde{A}^*$, we apply these operators once per sample, then post process the results to get estimators for all cells. Using the original operators A and $\widetilde{A}$ instead would require us to apply these operators to new random vectors for each cell.

It is straightforward to adapt the Chernoff bound in [18] to get an upper bound on the number of samples required. However, this bound is overly pessimistic; in practice we find the estimator is effective with only a handful of samples.

8.2.7 Anisotropic refinement: choosing the subdivision direction

We refine anisotropically by estimating the direction that φ_p changes the most as a function of p, then subdividing in that direction. This allows us to capture changes to φ_p in directions where translation-invariance fails, without refining the grid in directions where translation-invariance holds.

Let C be a cell that we have chosen to subdivide based on the randomized a-posteriori error estimator described in Section 8.2.6. For each coordinate direction i

in which C is big enough to be refined ($c^i_{\max} - c^i_{\min} > 2$), we partition the functions φ^E_k associated with the corners of C into two groups. One group is the set of φ^E_k associated with corners in the "front" of the cell ($+$) in the ith coordinate direction, and the other group is the set of φ^E_k associated with the "back" of the cell ($-$) in the ith coordinate direction:

$$\Psi^{i+} \coloneqq \{\varphi^E_k : p_k \in \text{corners}(\mathsf{C}),\ p^i_k = c^i_{\max}\},$$
$$\Psi^{i-} \coloneqq \{\varphi^E_k : p_k \in \text{corners}(\mathsf{C}),\ p^i_k = c^i_{\min}\}.$$

Next, we construct "average" φ^E_k functions for the front and back of the cell, respectively:

$$\varphi^{i+} \coloneqq \frac{1}{2^{d-1}} \sum_{\varphi^E \in \Psi^{i+}} \varphi^E \quad \text{and} \quad \varphi^{i-} \coloneqq \frac{1}{2^{d-1}} \sum_{\varphi^E \in \Psi^{i-}} \varphi^E.$$

Then we determine how much these average impulse responses change from the front to the back in direction i by computing $\|\varphi^{i+} - \varphi^{i-}\|_{l^2(\Omega - c_{\text{mid}})}$. Finally, we subdivide C in the coordinate direction i in which the average impulse response changes the most.

8.2.8 Construction cost

Algorithm 1 shows the complete algorithm for constructing $\widetilde{A}$. Updating $\widetilde{A}$ after refining a cell requires us to apply A to point sources centered at the new sample points created during the refinement. Hence the entire refinement process requires us to apply A to r vectors, where r is the total number of sample points in the final product-convolution approximation.

The dominant cost in the error estimation process is the cost of computing A^*Z and $\widetilde{A}^*Z$ for a random matrix Z with q columns. Since A^*Z is constant throughout the refinement process, we compute it once at the beginning.

Although $\widetilde{A}$ changes every time we refine a cell, after performing a refinement we do not have to recompute $\widetilde{A}^*Z$ from scratch. To see this, recall from (8.7) that the adjoint of our product-convolution operator is a convolution-product operator with the convolution functions reflected about the origin and complex conjugated. In order to recompute $\widetilde{A}^*Z$ after refining cells, we only need to compute the convolutions $\text{flip}\left(\overline{\varphi^E_k}\right) * \zeta_i$ for each column, ζ_i, in Z, and each sample point, p_k, that is *new* or has

Algorithm 1 Construction of $\widetilde{A}$

Input: $v \mapsto Av$, $v \mapsto A^*v$, Ω, τ, q
Output: $\left(w_k, \varphi_k^E\right)_{k=1}^r$

Draw random matrix $Z \sim N(0,1)^{N \times q}$
Compute $Y = A^*Z$ ▷ Cost: q applications of A^*
Initialize $\mathcal{T}$ with Ω as its root
Refine $\mathcal{T}$ by subdividing Ω once in each coordinate direction
Construct blocky neighborhoods U_k
Construct harmonic weighting function w_k
Compute impulse response functions $\varphi_k = A\delta_{p_k}$ ▷ Cost: 3^d applications of A
Construct extended impulse response functions φ_k^E
Compute $\widetilde{Y} = \widetilde{A}^*Z$ ▷ Cost: $q \times 3^d$ convolutions
Form local error estimators η_{C}
Form overall error estimator η_Ω
while $\eta_\Omega > \tau$ **do**
    Find cell $\mathsf{C} \in \text{leaves}\,\mathcal{T}$ with the largest η_{C}
    Determine subdivision direction, i, for C
    Subdivide C in direction i
    Construct U_k that are new or modified by the refinement
    Construct w_k for new or modified U_k
    Compute $\varphi_k = A\delta_{p_k}$ for all new p_k ▷ Cost: 1 application of A per new p_k
    Construct new or modified φ_k^E
    Update $\widetilde{Y}$ ▷ Cost: $O(q)$ convolutions per new p_k
    Form new or modified local error estimators η_{C}
    Form overall error estimator η_Ω

a *new neighbor*.[4] The convolutions for old sample points without new neighbors have been computed previously and can be re-used within (8.7). Thus the error estimation process requires computing $O(rm)$ convolutions. As we will discuss in Section 8.3.2, each of these convolutions can be done with the FFT in $O(N \log N)$ work. Updating the functions w_k can be done locally. This requires negligible work compared to the other costs already discussed. Putting all these pieces together, constructing $\widetilde{A}$ requires

$$O\left(rC + mC_* + rmN \log N\right),$$

work, where C and C_* are the costs to apply A and A^* to one vector, respectively.

8.3 Using the product-convolution approximation

The product-convolution format allows us to perform useful operations with $\widetilde{A}$ that we cannot perform with A.

8.3.1 Computing matrix entries of $\widetilde{A}$

Our approximation $\widetilde{A}$ is a product-convolution scheme and therefore (as seen in (8.6)) has the following matrix entries:

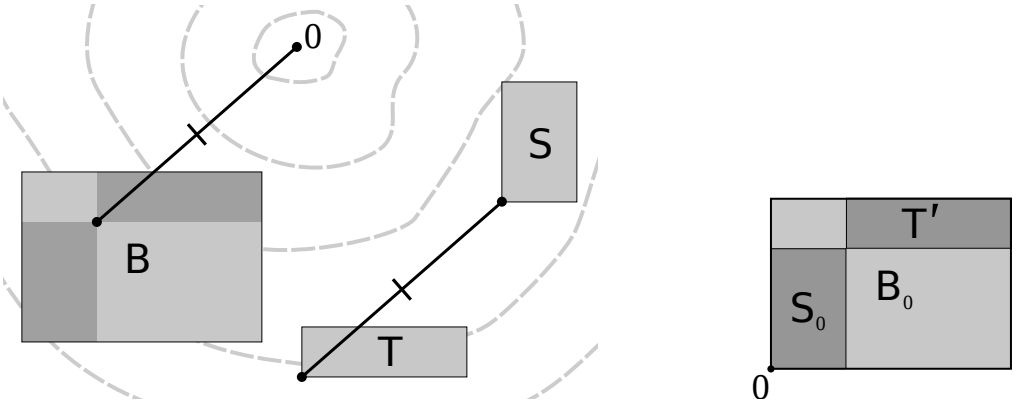
$$\widetilde{A}\left[y, x\right] = \sum_{k=1}^{r} w_k\left[x\right] \varphi_k^E\left[y - x\right] = \sum_{k : x \in U_k} w_k\left[x\right] \varphi_k^E\left[y - x\right]. \tag{8.19}$$

Using (8.19) we can compute individual matrix entries of $\widetilde{A}$ in $O(1)$ time even though $\widetilde{A}$ is not stored in memory in the conventional sense.

8.3.2 Applying $\widetilde{A}$ or $\widetilde{A}^*$ to vectors

Applying $\widetilde{A}$ or $\widetilde{A}^*$ to a vector requires computing r convolutions, r pointwise vector multiplications, and some vector additions (see equations (8.12) or (8.7), respectively). Out of these operations, the r convolutions are the most computationally expensive. Since the convolution theorem allows us to compute each of these convolutions using the FFT (after appropriate zero padding) [123] at $O(N \log N)$ cost, the cost of applying $\widetilde{A}$ or $\widetilde{A}^*$ to a vector is $O(rN \log N)$.

[4] The function φ_k^E depends on neighboring impulse responses due to the extension procedure.



(a) Boxes of interest for convolution (b) Translated boxes

Figure 8.6: Boxes of interest for computing $(\varphi * f)[\mathsf{T}]$. S: support of f. T: target region of interest. B: region of φ that influences the result. $\mathsf{S}_0, \mathsf{T}', \mathsf{B}_0$: translated boxes used in computation.

8.3.3 Applying blocks of $\widetilde{A}$ or $\widetilde{A}^*$ to vectors

One can implicitly apply a convolution operator to a function that is supported in a source box S then restrict the results to another target box T, by performing a convolution between a function supported on a box with the same shape as S and a function supported on a box with the same shape as $\mathsf{T} - \mathsf{S}$, then translating the results. Specifically, a change of variables shows that if f is supported on S, then

$$(\varphi * f)[\mathsf{T}] = (\varphi_0 * f_0)[\mathsf{T}'],$$

where

$$f_0[x_0] \coloneqq \begin{cases} f[x_0 + s_{\min}], & x_0 \in \mathsf{S}_0, \\ 0 & \text{else}, \end{cases} \qquad \varphi_0[z_0] \coloneqq \begin{cases} \varphi[z_0 + b_{\min}], & z_0 \in \mathsf{B}_0, \\ 0 & \text{else} \end{cases}$$

with $\mathsf{S}_0 \coloneqq \mathsf{S} - s_{\min}$, $\mathsf{T}' \coloneqq \mathsf{T} - t_{\min} + s_{\max} - s_{\min}$, $\mathsf{B} \coloneqq \mathsf{T} - \mathsf{S}$, and $\mathsf{B}_0 \coloneqq \mathsf{B} - b_{\min}$. This is illustrated in Figure 8.6. Thus one can apply a block of a convolution operator to a vector in work that scales nearly linearly with the linear dimensions of the block: $O(\sigma \log \sigma)$ where $\sigma = |\mathsf{S}| + |\mathsf{T}|$. To apply $\widetilde{A}[\mathsf{T}, \mathsf{S}]$ or $\widetilde{A}^*[\mathsf{T}, \mathsf{S}]$ to a vector, we use this method for each convolution in the sums ((8.12) and (8.7)) defining $\widetilde{A}$ or $\widetilde{A}^*$, respectively, that could be non-zero. Since the functions w_k are supported on the sets U_k, the terms in these sums that could be non-zero correspond to sets U_k that

intersect S when multiplying $\widetilde{A}[\mathsf{T},\mathsf{S}]$ with a vector, and T when multiplying with $\widetilde{A}^*[\mathsf{T},\mathsf{S}]$ with a vector. As a result, it costs

$$\underbrace{O(r_{\mathsf{S}}\ \sigma \log \sigma)}_{f \mapsto \widetilde{A}[\mathsf{T},\mathsf{S}]f} \qquad \text{and} \qquad \underbrace{O(r_{\mathsf{T}}\ \sigma \log \sigma)}_{f \mapsto \widetilde{A}^*[\mathsf{T},\mathsf{S}]f} \tag{8.20}$$

work to apply $\widetilde{A}[\mathsf{T},\mathsf{S}]$ and $\widetilde{A}^*[\mathsf{T},\mathsf{S}]$ to vectors, respectively. Here r_{S} and r_{T} are the number of sets U_k that intersect S and T, respectively.

8.3.4 Conversion to hierarchical matrix format

Construction of a hierarchical matrix proceeds in the following steps:

1. The degrees of freedom are partitioned hierarchically into a cluster tree.
2. The matrix entries are partitioned hierarchically into a block cluster tree.
3. A low-rank approximation is constructed for each block of the matrix that is marked as low-rank (i.e., admissible) within the block cluster tree.

The H-matrix construction process is scalable if we can construct low-rank approximations of the low-rank blocks (Step 3) in work that scales nearly linearly with the dimensions of the block. The method for efficiently applying blocks of $\widetilde{A}$ and $\widetilde{A}^*$ to vectors, outlined in Section 8.3.3, allows us to do this using Krylov methods or randomized SVD [118]. Whenever the Krylov method or randomized SVD requires the application of a block or its adjoint to a vector, we perform this computation using the method in Section 8.3.3. Alternatively, formula (8.19) for the matrix entries of $\widetilde{A}$ allows us to construct a low-rank approximation of a block by forming a CUR approximation [65, 137, 186], as is done in [29, 43, 186]. Whenever the CUR approximation algorithm requires a row, column, or entry of the block, we access it using (8.19).

Since applying the block $\widetilde{A}[\mathsf{T},\mathsf{S}]$ to a vector costs $O(\sigma \log \sigma)$ work, where $\sigma = |\mathsf{S}| + |\mathsf{T}|$, whereas accessing a row or column costs $O(\sigma)$ work, the CUR approach is asymptotically more scalable than the Krylov or randomized SVD approaches by a log factor. However, the CUR approach is less robust, and typically has poorer

dependence on the rank of the blocks. In either case the overall cost of constructing the H-matrix scales nearly linearly with N. Moreover, the construction process only uses the approximation, $\widetilde{A}$. It does not require expensive application of A.

8.4 Theory

Here we show that the error in $\widetilde{A}$ is controlled by the failure of A to be locally translation-invariant with respect to a locally expanded cover, $\{U_k^E\}_{k=1}^r$, created by unioning each U_k with its neighbors:

$$U_k^E \coloneqq \bigcup_{j \in \text{nbrs}(k)} U_j.$$

This provides an a-priori error estimate for the approximation, and shows that the approximation will not contain boundary artifacts.

Let F_k be the following functions that measure how much the impulse response of A at p_k fails to represent the impulse response of A at x (see Figure 8.1):

$$F_k[y, x] \coloneqq A[y - x + p_k, p_k] - A[y, x]. \tag{8.21}$$

We aggregate these F_k to form a function F which measures, pointwise, how much A fails to be locally translation-invariant with respect to the cover $\{U_k^E\}_{k=1}^r$. Specifically, we define

$$F[y, x] \coloneqq \max_{k:(y,x) \in \mu_k^E} |F_k[y, x]|, \tag{8.22}$$

where the sets

$$\mu_k^E \coloneqq \{(y, x) : x \in U_k^E, y \in \Omega, y - x + p_k \in \Omega\} \tag{8.23}$$

are defined to be all $(y, x) \subset \Omega \times \Omega$ such that $x \in U_k^E$, and $F_k[y, x]$ is well-defined without resorting to extension by zero. In Theorem 12 we will show that

$$\|\widetilde{A} - A\| \leq \|F\|. \tag{8.24}$$

If we instead maximized over $k : x \in U_k^E$ rather than $k : (y, x) \in \mu_k^E$ in (8.22), then the right hand side of bound (8.24) would be undefined, because evaluating $\|F\|$ requires evaluating $A[y - x + p_k, p_k]$, and $y - x + p_k$ may be outside of Ω even if x, y, and

p_k are in Ω. Extending A by zero would make $\|F\|$ well-defined, and would make the theory simple, but then the bound would be unnecessarily large due to boundary artifacts. Achieving bound (8.24) while maximizing over $k : (y, x) \in \mu_k^E$ in (8.22) requires the boundary extension procedure of Section 8.2.5, and is the reason why proving bound (8.24) will require several pages rather than a few lines.

A multi-step path leads to Theorem 12. In Proposition 6 we show that $\widetilde{A}$ can be reinterpreted as a weighted sum involving the original (not extended) impulse response functions φ_k, but with weighting functions that form a partition of unity on $\Omega \times \Omega$, and are supported in the sets μ_k^E. Proposition 6 relies on a lemma about the functions $v_k^{(j)}$ used in our impulse response extension procedure (Lemma 2), which in turn relies on a lemma about Minkowski sums of boxes (Lemma 1). After establishing these prerequisites, in Proposition 7 we show that $\widetilde{A} - A$ can be represented as a weighted sum of the F_k functions, with the same weighting functions as in Proposition 6. Finally, we use Proposition 7 and the properties of these weighting functions to prove bound (8.24) in Theorem 12.

Lemma 1. *If* S *and* T *are boxes, and* S *is at least as large as* T *in the sense that* $s^i_{max} - s^i_{min} \geq t^i_{max} - t^i_{min}$ *for* $i = 1, \ldots, d$, *then* $\mathsf{S} + \mathsf{T} = \mathsf{S} + \text{corners}(\mathsf{T})$.

Lemma 2. *We have*

$$\sum_{j \in \text{nbrs}(k)} v_k^{(j)}[z] = \begin{cases} 1, & z \in \Omega - U_k, \\ 0, & \textit{otherwise.} \end{cases} \tag{8.25}$$

Proof. By construction,

$$\sum_{j \in \text{nbrs}(k)} v_k^{(j)}[z] = \begin{cases} 1, & z \in \text{supp}(c_k), \\ 0, & \text{otherwise,} \end{cases}$$

and $\text{supp}(c_k) = \bigcup_{j \in \text{nbrs}(k)} (\Omega - p_j)$. We now show that $\Omega - U_k = \bigcup_{j \in \text{nbrs}(k)} (\Omega - p_j)$. To that end, recall that U_k is the union of leaf boxes C_i that contain p_k. Thus

$$\Omega - U_k = \Omega - \bigcup_{\mathsf{C}_i \in \text{cells}(p_k)} \mathsf{C}_i = \bigcup_{\mathsf{C}_i \in \text{cells}(p_k)} (\Omega - \mathsf{C}_i).$$

Since $\mathsf{C}_i \subset \Omega$, we see that Ω is at least as large as $-\mathsf{C}_i$ (in the sense of Lemma 1). Applying Lemma 1 to $\Omega - \mathsf{C}_i$ and performing algebraic manipulations yields:

$$\bigcup_{\mathsf{C}_i \in \text{cells}(p_k)} (\Omega - \mathsf{C}_i) = \bigcup_{\mathsf{C}_i \in \text{cells}(p_k)} (\Omega - \text{corners}\,(\mathsf{C}_i)) = \Omega - \bigcup_{\mathsf{C}_i \in \text{cells}(p_k)} \text{corners}\,(\mathsf{C}_i)\,.$$

Furthermore, by definition the union of all corners of leaf cells containing a point is the union of all neighboring points, so we have

$$\Omega - \bigcup_{\mathsf{C}_i \in \text{cells}(p_k)} \text{corners}\,(\mathsf{C}_i) = \Omega - \bigcup_{j \in \text{nbrs}(k)} p_j = \bigcup_{j \in \text{nbrs}(k)} (\Omega - p_j)\,,$$

which, with the chain of set equalities in previous lines, implies the desired result. □

Proposition 6. *Let*

$$W_k\,[y, x] \coloneqq \sum_{j \in \text{nbrs}(k)} w_j\,[x]\, v_j^{(k)}\,[y - x]\,. \tag{8.26}$$

1. *The entries of $\widetilde{A}$ can be written as:*

$$\widetilde{A}\,[y, x] = \sum_{k=1}^{r} W_k\,[y, x]\, \varphi_k\,[y - x]\,.$$

2. *The functions $\{W_k\}_{k=1}^r$ form a partition of unity:*

$$\sum_{k=1}^{r} W_k\,[y, x] = 1 \quad \textit{for all } (y, x) \in \Omega \times \Omega$$

3. *The partition of unity is subordinate to the cover $\{\mu_k^E\}_{k=1}^r$:*

$$\text{supp}(W_k) \subset \mu_k^E.$$

Proof.

1 Substituting the definition of φ_k^E from (8.15) into the definition of $\widetilde{A}$ from (8.12) then performing algebraic manipulations, we have:

$$\begin{aligned}
\widetilde{A}\,[y, x] &= \sum_{k=1}^{r} w_k\,[x] \sum_{j \in \text{nbrs}(k)} v_k^{(j)}\,[y - x]\, \varphi_j\,[y - x] \\
&= \sum_{k=1}^{r} \sum_{j \in \text{nbrs}(k)} w_k\,[x]\, v_k^{(j)}\,[y - x]\, \varphi_j\,[y - x] \\
&= \sum_{j=1}^{r} \sum_{k \in \text{nbrs}(j)} w_k\,[x]\, v_k^{(j)}\,[y - x]\, \varphi_j\,[y - x] = \sum_{j=1}^{r} W_j\,[y, x]\, \varphi_j\,[y - x]\,.
\end{aligned}$$

Going from the second to the third line we used the fact that

$$\sum_{a\in X}\sum_{\{b:b\in X,b\sim a\}} f(a,b) = \sum_{b\in X}\sum_{\{a:a\in X,a\sim b\}} f(a,b)$$

for any symmetric relation $\sim$. Note the switch of k and j.

2 Using the definition of W_k in (8.26), we have

$$\begin{aligned}\sum_{k=1}^{r} W_k\left[y,x\right] &= \sum_{k=1}^{r}\sum_{j\in\text{nbrs}(k)} w_j\left[x\right] v_j^{(k)}\left[y-x\right] \\ &= \sum_{j=1}^{r}\sum_{k\in\text{nbrs}(j)} w_j\left[x\right] v_j^{(k)}\left[y-x\right] = \sum_{j=1}^{r} w_j\left[x\right]\left(\sum_{k\in\text{nbrs}(j)} v_j^{(k)}\left[y-x\right]\right).\end{aligned}$$

If $x\in U_j$ and $y\in\Omega$, then Minkowski set arithmetic implies $y-x\in\Omega-U_j$, so (8.25) in Lemma 2 implies

$$\sum_{k\in\text{nbrs}(j)} v_j^{(k)}\left[y-x\right] = 1.$$

Since $\text{supp}(w_j)\subset U_j$, this implies

$$\sum_{j=1}^{r} w_j\left[x\right]\left(\sum_{k\in\text{nbrs}(j)} v_j^{(k)}\left[y-x\right]\right) = \sum_{j=1}^{r} w_j\left[x\right] = 1.$$

Thus $\sum_{k=1}^{r} W_k\left[y,x\right] = 1$ as required.

3 From the definition of $v_k^{(j)}$ in (8.14), either $\text{supp}(v_k^{(j)}) = (\Omega-p_j)\setminus(\Omega-p_k)$ when $k\neq j$, or $\text{supp}(v_k^{(j)}) = \Omega-p_j$ when $k=j$. In either case $\text{supp}(v_k^{(j)})\subset\Omega-p_j$. Thus

$$(y-x\notin\Omega-p_j) \implies \left(v_k^{(j)}\left[y-x\right]=0\right),$$

which is equivalent to the statement

$$(y-x+p_j\notin\Omega) \implies \left(v_k^{(j)}\left[y-x\right]=0\right). \tag{8.27}$$

Since W_k consists of a sum of terms, each term containing $v_j^{(k)}\left[y-x\right]$, statement (8.27) implies (note the swap of k,j):

$$(y-x+p_k\notin\Omega) \implies \left(W_k\left[y,x\right]=0\right). \tag{8.28}$$

Additionally, since each w_j in the sum defining W_k is supported in the blocky neighborhood U_j, and since the union of these blocky neighborhoods U_j is U_k^E, we have

$$\left(x \notin U_k^E\right) \implies \left(W_k\left[y,x\right]=0\right). \tag{8.29}$$

Altogether, (8.28), (8.29), and the definition of μ_k^E in (8.23) imply $\operatorname{supp}(W_k) \subset \mu_k^E$. □

Proposition 7. *The pointwise error in our product-convolution approximation takes the following form:*

$$\widetilde{A}\left[y,x\right]-A\left[y,x\right]=\sum_{k:(y,x)\in\mu_k^E} W_k\left[y,x\right]F_k\left[y,x\right]. \tag{8.30}$$

Proof. From Proposition 6 and the fact that $\varphi_k\left[z\right]=A\left[z+p_k,p_k\right]$, we know that

$$\widetilde{A}\left[y,x\right]=\sum_{k=1}^{r} W_k\left[y,x\right]A\left[y-x+p_k,p_k\right],$$

Hence the pointwise error in the approximation takes the following form:

$$\begin{aligned}
\widetilde{A}\left[y,x\right]-A\left[y,x\right] &= \sum_{k=1}^{r} W_k\left[y,x\right]A\left[y-x+p_k,p_k\right]-A\left[y,x\right] \\
&= \sum_{k=1}^{r} W_k\left[y,x\right]\left(A\left[y-x+p_k,p_k\right]-A\left[y,x\right]\right) \\
&= \sum_{k=1}^{r} W_k\left[y,x\right]F_k\left[y,x\right] = \sum_{k:(y,x)\in\mu_k^E} W_k\left[y,x\right]F_k\left[y,x\right]
\end{aligned}$$

Going from the first line to the second line we used the partition of unity property of W_k from Proposition 6. Going from the second to the third line we used the definition of F_k. In the last equality on the third line we used the fact that $\operatorname{supp}\left(W_k\right) \subset \mu_k^E$. □

Theorem 12. *We have*

$$\|\widetilde{A}-A\| \leq \|F\|. \tag{8.31}$$

Proof. Using the result of Proposition 7, the fact that W_k form a partition of unity, and the definition of F yields the pointwise error bound

$$\begin{aligned}
\left|\widetilde{A}\left[y,x\right]-A\left[y,x\right]\right| &= \left|\sum_{k:(y,x)\in\mu_k^E} W_k\left[y,x\right]F_k\left[y,x\right]\right| \\
&\leq \max_{k:(y,x)\in\mu_k^E}\left|F_k\left[y,x\right]\right| = F\left[y,x\right].
\end{aligned}$$

The overall bound, (8.31), follows directly from the definition of the norm and this pointwise bound. □

Remark 1. *Let*

$$T[y, x] := A[y + x, x]$$

be the spatially varying impulse response function (see, e.g., [38] for a more in-depth discussion of the SVIR). Under the change of variables $h := p - x$, $\xi := y - x$, we may express the failure of local translation invariance in terms of the SVIR as follows:

$$A[y - x + p, p] - A[y, x] = -\left(T[\xi, p + h] - T[\xi, p]\right).$$

If x is near p, then h is small, so

$$T[\xi, p + h] - T[\xi, p] \approx \frac{dT}{dp}(\xi, p)h.$$

Hence, if our scheme is applied to a discretization of a continuous operator, the smoother the function $x \mapsto T[y, x]$ is, the better our scheme will perform.

8.5 Numerical Results

Here we numerically test our scheme on the Hessian for the advection-diffusion model inverse problem (Section 1.1.2). We will see that our scheme requires roughly the same convolution rank to achieve a desired error tolerance regardless of how large the Peclet number is. The Peclet number controls the ratio of advection to diffusion. The larger the Peclet number, the less information about the parameter is destroyed by diffusion. Hence the Peclet number serves as a proxy for the informativeness of the data about the unknown parameter. While regularization preconditioning performs poorly when the Peclet number is large, a preconditioner constructed using our scheme performs well when the Peclet number is large.

Additionally, the randomized a-posteriori error estimator achieves good performance with only a handful of random samples: our scheme performs almost as well with $m = 5$ as it does with $m = 100$.

Problem setup We use our adaptive product-convolution scheme to approximate the data misfit Hessian,

$$A := H_d,$$

and then use this approximation, in combination with H-matrix methods, to build a preconditioner for the overall Hessian, $H = H_d + R$, for the advection-diffusion model inverse problem. Recall that in this inverse problem one inverts for the initial concentration, q, of a contaminant plume, u, based on timeseries measurements, y, of the contaminant plume as it flows past sensors. Specifically, we consider the following PDE:

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{1}{\mathrm{Pe}}\Delta u - \binom{0}{1} \cdot \nabla u, & t \in [0,1], \\ u = q, & t = 0, \end{cases} \tag{8.32}$$

where Pe is the Peclet number. The region of interest and support of q is the unit square, $\Omega = [0,1]^2$, and the desired unbounded domain for the PDE is $\mathbb{R}^2$. To simulate the effect of having an unbounded domain, we extend the computational domain beyond $[0,1]^2$ on all sides and use Neumann boundary conditions on the outer, larger, domain. We use y to denote the known noisy time series observations of u on the top boundary: $y(x,t) = u(x,t)+\zeta,\ x \in \Gamma,\ t \in (0,1]$, where $\Gamma := [0,1]\times\{1\}$ and ζ is 1% independent and identically distributed Gaussian noise.

We use Laplacian regularization, $R = \alpha\Delta$, where Δ is a discretization of the Laplacian operator with zero Dirichlet boundary conditions, and $\alpha = 10^{-3}$ is the regularization parameter. This value of α was chosen since it satisfies the Morozov discrepancy principle to within a 5% tolerance for all Peclet numbers considered. For discretization, we use piecewise linear finite elements defined on a regular rectilinear 100×100 mesh of triangles, with 100 time steps. We use backward Euler time stepping and SUPG stabilization [50].

We use an image of the University of Texas "Hook'em Horns" logo as the initial concentration, m. The sharp edges in this image are computationally expensive to recover using existing methods. The solutions to the inverse problem for Peclet numbers in the range 10^0 to 10^5 are shown in Figure 8.7.

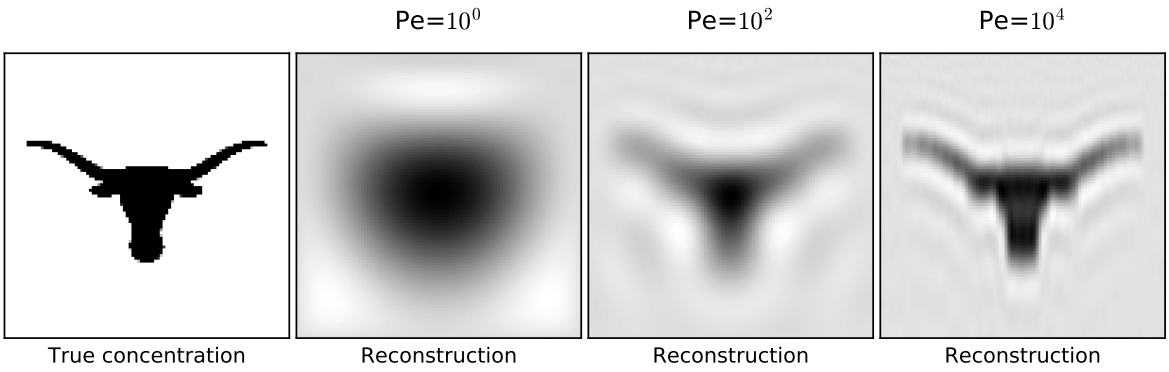


Figure 8.7: **Advection-diffusion inverse problem Hessian:** True initial concentration m (left image) and optimal reconstructions for Peclet numbers 10^0, 10^2, and 10^4 (all other images, left to right). The regularization parameter used for the reconstruction is $\alpha = 10^{-3}$, which satisfies the Morozov discrepancy principle within a 5% tolerance for all Peclet numbers considered.

Preconditioning the Hessian Good general purpose preconditioners for the Hessian in the advection-diffusion inverse problem with a large Peclet number have not been available (see Chapter 6 and [5] for a discussion of these issues). But now our convolution-product scheme allows us to build a good preconditioner as follows: first we form a product-convolution approximation of H_d, then convert it to H-matrix format, then symmetrize it, then add a small amount of identity regularization, then combine it with R, then finally invert the combined H-matrix with fast H-matrix arithmetic. In detail, we form the following approximation to the inverse of the Hessian, which we use as a preconditioner:

$$P^{-1} := \left((\widetilde{H_d} + \widetilde{H_d}^T)/2 + \tau \, \|H_d\| \, I + R \right)^{-1} \approx H^{-1}. \tag{8.33}$$

Here $\tau \, \|H_d\| \, I$ is a small amount of additional regularization (I is the identity matrix). We use $\tau = 0.0025$. Matrix addition, scaling, and inversion in (8.33) are performed with H-matrix arithmetic. For H-matrices, we use the standard coordinate splitting nested-bisection binary cluster tree[5], and the standard diameter-less-than-distance

[5]Degrees of freedom are split into two equally-sized clusters by a hyperplane normal to widest coordinate direction for that cluster. Then each cluster is split into two smaller clusters in the same way, and so on, recursively. The splitting continues until the number of degrees of freedom in a cluster is less than 32.

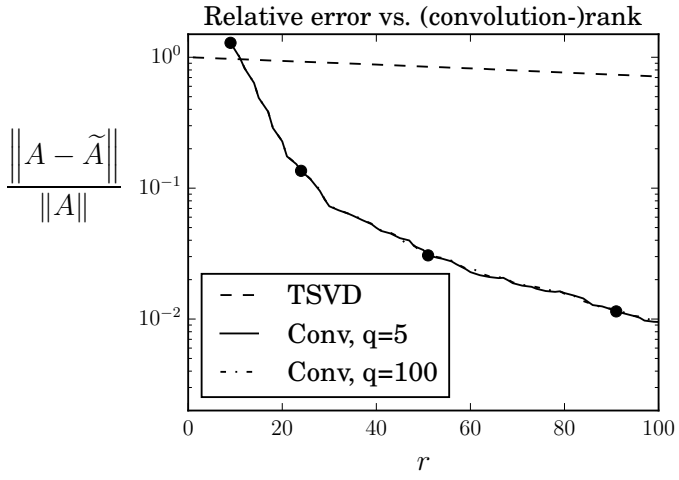


Figure 8.8: **Advection-diffusion inverse problem Hessian:** Relative error in the truncated SVD ('TSVD') low-rank approximation compared to our product-convolution approximation ('Conv') as the (convolution) rank, r, changes. We show convergence curves for our scheme using both $q = 5$ and $q = 100$ random samples for the a-posteriori error estimator. Black dots correspond to the adaptive grids visualized in Figure 8.9.

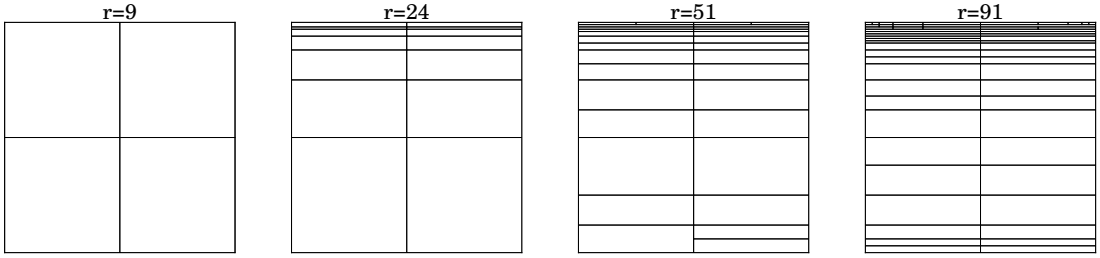


Figure 8.9: **Advection-diffusion inverse problem Hessian:** Intermediate stages of adaptive grid refinement corresponding to black dots in Figure 8.8.

admissibilty condition[6]. Here, we use a fixed rank of 20 for the low-rank approximations performed during H-matrix construction and arithmetic.

Results Figure 8.8 compares the convergence of our product-convolution scheme ('CONV') to truncated SVD low rank approximation ('TSVD') when $\mathrm{Pe} = 10^4$. Our scheme performs better than TSVD: at $r = 100$ our scheme has less than 1% error whereas TSVD has approximately 71% error. Like the Poisson problem, the convergence curve for $q = 5$ is almost identical to the convergence curve for $q = 100$. Figure 8.9 shows the adaptive meshes from four different stages of the adaptive refinement

[6]We mark a block of the matrix as low rank (admissible) if the distance between the degree of freedom cluster associated with the rows of the block and the diameter of the degree of freedom cluster associated with the columns of the block is less than or equal to the diameter of the smaller of the two degree of freedom clusters.

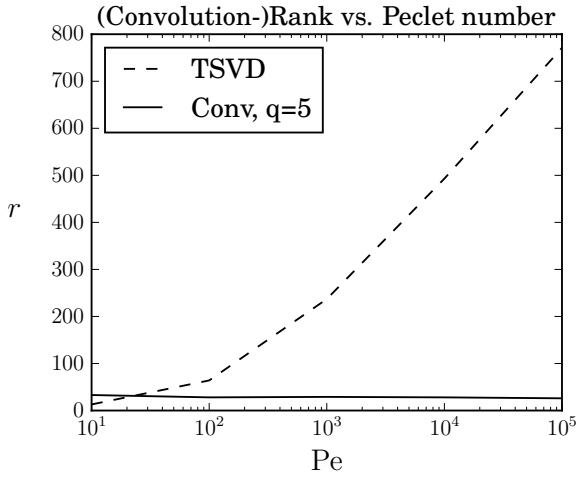


Figure 8.10: **Advection-diffusion inverse problem Hessian:** The (convolution) rank, r, required to achieve a relative approximation error of 10% for a variety of Peclet numbers, Pe. 'TSVD' indicates truncated SVD low rank approximation, and 'Conv' indicates our product-convolution scheme.

process from Figure 8.8. Our scheme chooses to adaptively refine in the direction of the vertical flow, prioritizing refinement near the top surface. We expect similar results would hold for inverse problems involving non-vertical, non-uniform flow if the convolution grid were aligned with the streamlines of the flow.

Figure 8.10 compares our scheme to TSVD for a sequence of increasing Peclet numbers, from $\text{Pe} = 10^1$ to $\text{Pe} = 10^5$. The curves show the (convolution) rank, r, required to achieve a relative error tolerance of 10% (estimated using $q = 5$ random adjoint samples). Whereas the required rank for TSVD grows dramatically as Pe increases, the required convolution rank for our scheme remains constant.

Figure 8.11 shows the convergence of Krylov methods for solving the Hessian linear system using GMRES with our preconditioner ('GMRES-CONV'), compared to conjugate gradient with regularization preconditioning ('CG-REG'), for $\text{Pe} = 10^4$. Here the product-convolution approximation is computed to a 5% relative error tolerance. Our preconditioner substantially outperforms regularization preconditioning, coverging rapidly even though the Peclet number is large. In Figure 8.12, we show intermediate reconstructions associated with 1, 5, and 50 Krylov iterations, for both GMRES-CONV and CG-REG. CG-REG first reconstructs large-scale features of m, then medium-scale features, then small-scale features, while GMRES-CONV reconstructs features of m at all scales simultaneously. Even one iteration of GMRES-CONV yields a visually reasonable reconstruction.

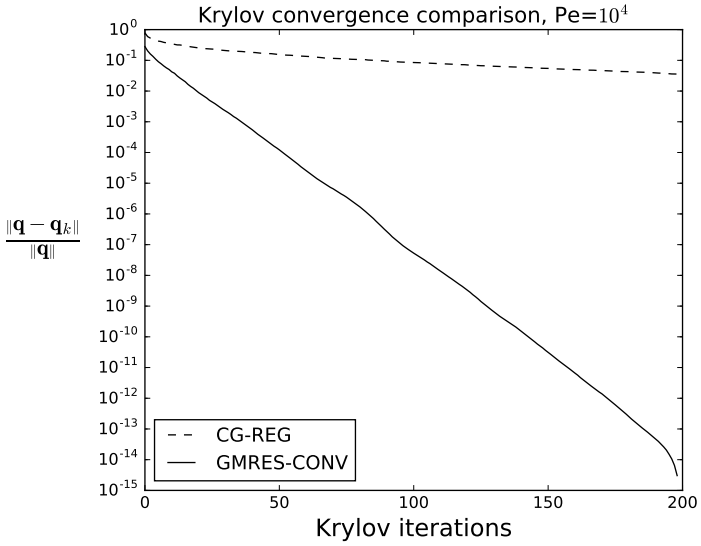


Figure 8.11: **Advection-diffusion inverse problem Hessian:** Convergence of conjugate gradient with regularization preconditioning ('CG-REG'), compared to GMRES with our product-convolution preconditioner, (8.33) ('GMRES-CONV'), for solving the Hessian linear system. Here Pe = 10^4, and the product-convolution approximation is accurate to 5% relative error.

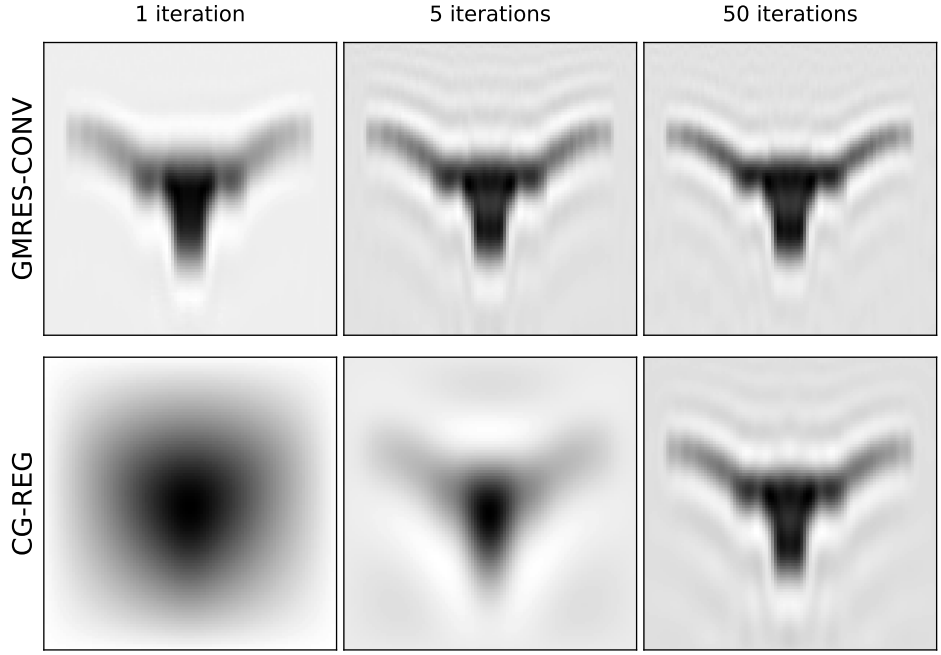


Figure 8.12: **Advection-diffusion inverse problem Hessian:** Comparison of parameter reconstructions associated with terminating the Krylov solver after 1, 5, and 50 iterations, for both GMRES with our preconditioner ('GMRES-CONV'), and conjugate gradient with regularization preconditioning ('CG-REG'). Here Pe = 10^4, and the product-convolution approximation is accurate to 5% relative error.

Chapter 9

Domain Decomposition Wave Preconditioner

In this chapter we construct a domain decomposition solver for the operator

$$\mathbf{B}^H\mathbf{Y}\mathbf{B} + \rho\mathbf{A}^H\mathbf{W}_L^{-1}\mathbf{A}, \tag{9.1}$$

for the single-frequency wave inverse problem. The solver for (9.1) will be robust to large numbers of observations, high wave frequencies, and small ρ. Here bold letters denote the concrete sparse matrix representations of the original (non-bold) operators, and $\mathbf{W}_L$ is a diagonal lumped mass approximation of the mass matrix. Also recall that $\mathbf{A}$ is the coefficient matrix for the state equation, $\mathbf{B}$ is the observation operator, and $\mathbf{Y}$ is a sparse noise coraviance operator. The resulting domain decomposition preconditioner for (9.1), combined with a multigrid preconditioner for $\alpha\mathbf{R}_0 + \rho\mathbf{T}^H\mathbf{W}^{-1}\mathbf{T}$, [1] could be used in the augmented Lagrangian KKT framework from Chapter 7 to precondition the KKT matrix for single frequency single source wave inverse problem. While the preconditioner we present for (9.1) is effective, challenges remain to make the overall KKT preconditioner effective. Nevertheless the preconditioner we present for (9.1) represents a substantial step towards this goal.

The strategy for solving (9.1) will be to break the domain into pieces: one small part, Ω_t, near the surface containing the support of the observation operator, another large part, Ω_b, below the surface, and an interface Γ_i separating Ω_t from Ω_b (see Figure 9.1). Properly addressing the coupling between the subdomains is complicated, and the solver we present will be much more involved than standard domain decomposition solvers for forward problems. The main difficulty will be construction of a preconditioner for a Schur complement associated with degrees of freedom in $\Omega_t \cup \Gamma_i$

[1] Since R_0 is a Laplacian-like operator, and since T^*T is a positive multiplication operator, $\alpha\mathbf{R}_0 + \rho\mathbf{T}^H\mathbf{W}_L^{-1}\mathbf{T}$ can be effectively preconditioned with multigrid. We cannot use multigrid for (9.1) since A is a high frequency wave operator, and multigrid performs poorly for high-frequency wave problems.

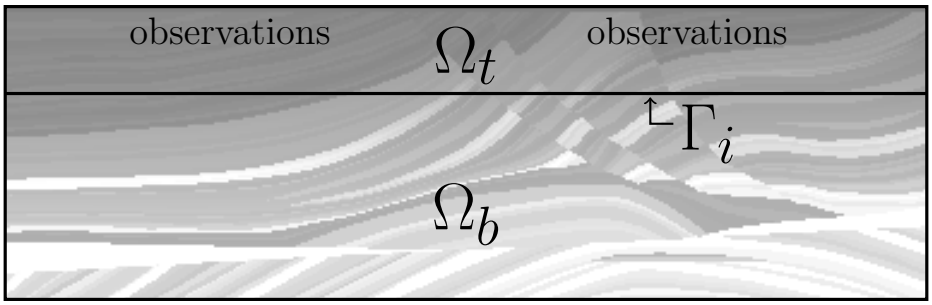


Figure 9.1: The top subdomain containing observations, Ω_t, the bottom subdomain, Ω_b, and the interface separating the subdomains, Γ_i. Not drawn to scale; the top subdomain will be much thinner relative to the bottom subdomain than shown here.

for an expanded matrix that is algebraically equivalent to (9.1) but has twice as many variables. The Schur complement will consist of two terms. The first term will be approximated with a PML truncation, and the second term will be approximated with the product-convolution scheme from Chapter 8. To improve the local translation invariance of the second term, we will use algebraically exact impedance-to-impedance interface conditions to couple the two subdomains.

9.1 Algebraic impedance-to-impedance interface conditions

In forward (non-squared) wave problems, impedance-to-impedance maps have better numerical properties than nonlocal components of Schur complements based on naive algebraic splittings [105]. To illustrate the idea, let $T := (t, i)$ denote the degrees of freedom in the top subdomain (including interface), and consider decomposing the wave operator $\mathbf{A}$ into blocks based on subdomains. Algebraic truncation causes $\mathbf{A}_{TT}$ to have an artificial "hard" boundary at Γ_i that reflects waves, whereas the Schur complement $\mathbf{A}_{TT} - \mathbf{A}_{Tb}\mathbf{A}_{bb}^{-1}\mathbf{A}_{bT}$ cannot have a hard boundary because $\mathbf{\Gamma}_i$ is an internal inferface that allows waves to pass through it. The non-local term,

$$-\mathbf{A}_{Tb}\mathbf{A}_{bb}^{-1}\mathbf{A}_{bT}, \tag{9.2}$$

must contain strong complex long range interactions to correct for this discrepancy. In contrast, impedance boundary conditions are "soft" and allow waves to exit one subdomain and enter another with less reflection. As a result, splittings based on impedance boundary conditions—splittings where the outgoing impedance in one

subdomain is the incoming impedance boundary condition in the other subdomain, and vice versa—yield correction terms of the form (9.2) that contain weaker and less complex long-range interactions.

Later in this chapter we will need to approximate a matrix of the form

$$\mathbf{A}_{Tb}\mathbf{A}_{bb}^{-1}\mathbf{W}_L\mathbf{A}_{bb}^{-H}\mathbf{A}_{bT},$$

which is like (9.2), but squared. We apply an impedance-to-impedance splitting to our problem pre-emptively, so that when this matrix arises, it will be more translation-invariant and therefore easier to approximate with our product-convolution scheme from Chapter 8. Unfortunately, modifying the interface conditions at the continuum (PDE) level results in discrepancies along Γ_i upon discretization. Waves propagate these discrepancies throughout the whole domain, polluting the solution everywhere and causing considerable error. To avoid these errors, we change interface conditions algebraically, ensuring that the modified system has the same solution as the original system at the discrete level.

Let

$$\begin{bmatrix} \mathbf{A}_{tt} & \mathbf{A}_{ti} \\ \mathbf{A}_{it} & \widetilde{\mathbf{A}}_{ii}^{\text{top}} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \widetilde{\mathbf{A}}_{ii}^{\text{bot}} & \mathbf{A}_{ib} \\ \mathbf{A}_{bi} & \mathbf{A}_{bb} \end{bmatrix} \tag{9.3}$$

be the coefficient matrices that result from discretizing the Helmholtz equation on $\Omega_t \cup \Gamma_i$ and $\Gamma_i \cup \Omega_b$, respectively, with the same mesh and finite elements used for $\mathbf{A}$, but with incoming impedance boundary conditions on Γ_i in both cases. That is, boundary conditions of the form

$$\frac{du}{dn} - iq^{1/2}u = z \quad \text{on } \Gamma_i,$$

where n is the normal vector to the subdomain and z is an arbitrary forcing impedance. Here the submatrices in the (i,i) blocks, $\widetilde{\mathbf{A}}_{ii}^{\text{top}}$ and $\widetilde{\mathbf{A}}_{ii}^{\text{bot}}$, differ from $\mathbf{A}_{ii}$ due to the inclusion of impedance boundary conditions. By duplicating interface variables within the linear system

$$\begin{bmatrix} \mathbf{A}_{tt} & \mathbf{A}_{ti} & 0 \\ \mathbf{A}_{it} & \mathbf{A}_{ii} & \mathbf{A}_{ib} \\ 0 & \mathbf{A}_{bi} & \mathbf{A}_{bb} \end{bmatrix} \begin{bmatrix} \mathbf{u}_t \\ \mathbf{u}_i \\ \mathbf{u}_b \end{bmatrix} = \begin{bmatrix} \mathbf{f}_t \\ \mathbf{f}_i \\ \mathbf{f}_b \end{bmatrix}, \tag{9.4}$$

we create the following system defined on $\Omega_t \cup \Gamma_i \cup \Gamma_i \cup \Omega_b$:

$$\underbrace{\begin{bmatrix} \mathbf{A}_{tt} & \mathbf{A}_{ti} & 0 & 0 \\ \mathbf{A}_{it} & \widetilde{\mathbf{A}}_{ii}^{\text{top}} & \mathbf{A}_{ii} - \widetilde{\mathbf{A}}_{ii}^{\text{top}} & \mathbf{A}_{ib} \\ \mathbf{A}_{it} & \mathbf{A}_{ii} - \widetilde{\mathbf{A}}_{ii}^{\text{bot}} & \widetilde{\mathbf{A}}_{ii}^{\text{bot}} & \mathbf{A}_{ib} \\ 0 & 0 & \mathbf{A}_{bi} & \mathbf{A}_{bb} \end{bmatrix}}_{\widetilde{\mathbf{A}}} \begin{bmatrix} \mathbf{u}_t \\ \mathbf{u}_i \\ \mathbf{u}_j \\ \mathbf{u}_b \end{bmatrix} = \begin{bmatrix} \mathbf{f}_t \\ \mathbf{f}_i \\ \mathbf{f}_i \\ \mathbf{f}_b \end{bmatrix}. \tag{9.5}$$

We denote the overall coefficient matrix in (9.5) by $\widetilde{\mathbf{A}}$, and define for later convenience the matrices

$$\mathbf{Z}_{\uparrow} := \mathbf{A}_{ii} - \widetilde{\mathbf{A}}_{ii}^{\text{top}}, \quad \text{and} \quad \mathbf{Z}_{\downarrow} := \mathbf{A}_{ii} - \widetilde{\mathbf{A}}_{ii}^{\text{bot}}.$$

The matrices $\mathbf{A}$ and $\widetilde{\mathbf{A}}$ are algebraically equivalent in the sense that if $(\mathbf{u}_t, \mathbf{u}_i, \mathbf{u}_j, \mathbf{u}_b)$ solves (9.5), then $(\mathbf{u}_t, \mathbf{u}_i, \mathbf{u}_b)$ solves (9.4). But in $\widetilde{\mathbf{A}}$, the top (T) and bottom ($B := (i, b)$) subdomains are coupled with impedance-to-impedance interface conditions on Γ_i.

9.2 Replacing A with $\widetilde{\mathbf{A}}$ in $\mathbf{B}^H\mathbf{Y}\mathbf{B} + \rho\mathbf{A}^H\mathbf{W}_L^{-1}\mathbf{A}$

By performing algebraic manipulations, we may replace $\mathbf{A}$ with $\widetilde{\mathbf{A}}$ in any linear system with (9.1) as the coefficient matrix. Consider the system

$$\mathbf{B}^H\mathbf{Y}\mathbf{B}\mathbf{u} + \rho\mathbf{A}^H\mathbf{W}_L^{-1}\mathbf{A}\mathbf{u} = \mathbf{f}. \tag{9.6}$$

Applying $\mathbf{A}^{-H}$ to (9.6) on the left, applying $\mathbf{A}^{-1}$ on the right, and defining the auxiliary variables $\mathbf{w} := \mathbf{A}\mathbf{u}$ and $\mathbf{p} := \mathbf{A}^{-H}\mathbf{f}$, yields the following equivalent system

$$\mathbf{A}^{-H}\mathbf{B}^H\mathbf{Y}\mathbf{B}\mathbf{A}^{-1}\mathbf{w} + \rho\mathbf{W}_L^{-1}\mathbf{w} = \mathbf{p}. \tag{9.7}$$

Let $\mathbf{E}_1$ be the matrix that takes the vector $\left[\mathbf{u}_t, \mathbf{u}_i, \mathbf{u}_b\right]^T$ and duplicates interface variables, mapping it to the vector $\left[\mathbf{u}_t, \mathbf{u}_i, \mathbf{u}_i, \mathbf{u}_b\right]^T$. Let $\mathbf{E}_2$ be the matrix that takes the vector $\left[\mathbf{u}_t, \mathbf{u}_i, \mathbf{u}_j, \mathbf{u}_b\right]^T$ and deletes the second interface variable, mapping it to the vector $\left[\mathbf{u}_t, \mathbf{u}_i, \mathbf{u}_b\right]^T$. The algebraic equivalence of $\mathbf{A}$ and $\widetilde{\mathbf{A}}$ may be restated as the equality

$$\mathbf{A}^{-1} = \mathbf{E}_2\widetilde{\mathbf{A}}^{-1}\mathbf{E}_1. \tag{9.8}$$

Substituting (9.8) into (9.7) yields the system

$$\mathbf{E}_1^T \widetilde{\mathbf{A}}^{-H} \mathbf{E_2}^T \mathbf{B}^H \mathbf{Y} \mathbf{B} \mathbf{E}_2 \widetilde{\mathbf{A}}^{-1} \mathbf{E_1} \mathbf{w} + \rho \mathbf{W}_L^{-1} \mathbf{w} = \mathbf{p}, \tag{9.9}$$

which contains $\widetilde{\mathbf{A}}$ instead of $\mathbf{A}$. To solve (9.6), we may instead compute $\mathbf{p} = \mathbf{A}^{-H}\mathbf{f}$, then solve (9.9) for $\mathbf{w}$, then compute $\mathbf{u} = \mathbf{A}^{-1}\mathbf{w}$.

9.3 Expanded system

Matrix entries of the coefficient matrix in (9.9) are not available, owing to the presence of the dense matrices $\widetilde{\mathbf{A}}^{-1}$ and $\widetilde{\mathbf{A}}^{-H}$ within it. However, just as the dense Hessian is equivalent to the algebraically equivalent sparse KKT matrix (see Chapter 5), the dense linear system (9.9) is equivalent to the following sparse linear system:

$$\begin{bmatrix} \mathbf{W}_L^{-1} & 0 & \mathbf{E}_1^T \\ 0 & \frac{1}{\rho}\mathbf{E}_2^T \mathbf{B}^T \mathbf{Y} \mathbf{B} \mathbf{E}_2 & \widetilde{\mathbf{A}}^H \\ \mathbf{E}_1 & \widetilde{\mathbf{A}} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \boldsymbol{\xi} \\ \boldsymbol{\eta} \end{bmatrix} = \begin{bmatrix} \mathbf{p} \\ 0 \\ 0 \end{bmatrix}, \tag{9.10}$$

where $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ are auxiliary variables.

Performing a block LU factorization of the coefficient matrix in (9.10), then performing algebraic manipulations, shows that $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ solve the following linear system:

$$\begin{bmatrix} \frac{1}{\rho}\widetilde{\mathbf{B}}^T \mathbf{Y} \widetilde{\mathbf{B}} & \widetilde{\mathbf{A}}^H \\ \widetilde{\mathbf{A}} & -\widetilde{\mathbf{W}}_L \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi} \\ \boldsymbol{\eta} \end{bmatrix} = \begin{bmatrix} 0 \\ -\mathbf{E}_1 \mathbf{W}_L \mathbf{p} \end{bmatrix}, \tag{9.11}$$

where $\widetilde{\mathbf{W}}_L := \mathbf{E}_1 \mathbf{W}_L \mathbf{E}_1^T$ and $\widetilde{\mathbf{B}} := \mathbf{B}\mathbf{E}_2$. To solve (9.9) (and therefore solve (9.6), as per the discussion in Section 9.2), we may instead solve (9.11) for $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$, then compute $\mathbf{w} = \mathbf{W}_L(\mathbf{p} - \mathbf{E}_1^T \boldsymbol{\eta})$.

9.4 Reordering and block-factoring the expanded system

Reordering system (9.11) so that all degrees of freedom are grouped by their subdomain (top, T, or bottom, B), rather than by the variable (ξ or η) yields the

equivalent linear system

$$\underbrace{\left[\begin{array}{cc:cc} \frac{1}{\rho}\left(\widetilde{\mathbf{B}}^H\mathbf{Y}\widetilde{\mathbf{B}}\right)_{TT} & \widetilde{\mathbf{A}}_{TT}^H & 0 & \widetilde{\mathbf{A}}_{TB}^H \\ \widetilde{\mathbf{A}}_{TT} & -\left(\widetilde{\mathbf{W}}_L\right)_{TT} & \widetilde{\mathbf{A}}_{TB} & -\left(\widetilde{\mathbf{W}}_L\right)_{TB} \\ \hdashline 0 & \widetilde{\mathbf{A}}_{BT}^H & 0 & \widetilde{\mathbf{A}}_{BB}^H \\ \widetilde{\mathbf{A}}_{BT} & -\left(\widetilde{\mathbf{W}}_L\right)_{BT} & \widetilde{\mathbf{A}}_{BB} & -\left(\widetilde{\mathbf{W}}_L\right)_{BB} \end{array}\right]}_{\mathbf{M}} \underbrace{\left[\begin{array}{c} \boldsymbol{\xi}_T \\ \boldsymbol{\eta}_T \\ \hdashline \boldsymbol{\xi}_B \\ \boldsymbol{\eta}_B \end{array}\right]}_{\mathbf{x}} = \underbrace{\left[\begin{array}{c} 0 \\ -\left(\mathbf{W}_L\mathbf{p}\right)_T \\ \hdashline 0 \\ -\left(\mathbf{W}_L\mathbf{p}\right)_B \end{array}\right]}_{\mathbf{g}}, \tag{9.12}$$

or $\mathbf{M}\mathbf{x} = \mathbf{g}$, where $\mathbf{M}$, $\mathbf{x}$, and $\mathbf{g}$ are the coefficient matrix, unknown, and right hand side of (9.12), respectively. Let $\mathbf{M}_{TT}$, $\mathbf{M}_{TB}$, $\mathbf{M}_{BT}$, and $\mathbf{M}_{BB}$ denote the coarse-level blocks of $\mathbf{M}$ (the four 2×2 blocks separated by dashed lines in (9.12)). With respect to this partitioning, $\mathbf{M}$ has the following block triangular factorization:

$$\begin{bmatrix} \mathbf{M}_{TT} & \mathbf{M}_{TB} \\ \mathbf{M}_{BT} & \mathbf{M}_{BB} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{M}_{TB}\mathbf{M}_{BB}^{-1} \\ 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{S} & 0 \\ \mathbf{M}_{BT} & \mathbf{M}_{BB} \end{bmatrix}, \tag{9.13}$$

where

$$\mathbf{S} := \mathbf{M}_{TT} - \mathbf{M}_{TB}\mathbf{M}_{BB}^{-1}\mathbf{M}_{BT} \tag{9.14}$$

is the Schur complement for the top degrees of freedom. Replacing $\mathbf{S}$ with an approximation $\widehat{\mathbf{S}}$ (to be discussed in Section 9.5) in (9.13) yields the following overall preconditioner $\widehat{\mathbf{M}}$ for $\mathbf{M}$:

$$\widehat{\mathbf{M}} := \begin{bmatrix} \mathbf{I} & \mathbf{M}_{TB}\mathbf{M}_{BB}^{-1} \\ 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \widehat{\mathbf{S}} & 0 \\ \mathbf{M}_{BT} & \mathbf{M}_{BB} \end{bmatrix}.$$

If we can apply $\widehat{\mathbf{M}}^{-1}$ to vectors, then we can use $\widehat{\mathbf{M}}$ as a preconditioner within Krylov methods to efficiently solve $\mathbf{M}\mathbf{x} = \mathbf{g}$. As per the discussion in previous sections, this allows us to solve linear systems with $\mathbf{B}^H\mathbf{Y}\mathbf{B} + \rho\mathbf{A}^H\mathbf{W}_L^{-1}\mathbf{A}$ as the coefficient matrix, our desired goal.

Applying $\widehat{\mathbf{M}}^{-1}$ to vectors requires us to performing a sequence of two block triangular solves. The nontrivial operations required by these block triangular solves are the application of $\mathbf{M}_{BB}^{-1}$ to two vectors, and the application of $\widehat{\mathbf{S}}^{-1}$ to one vector (see Algorithm 5). Since $\widetilde{\mathbf{M}}_{BB}$ is permutation equivalent to a block-triangular matrix with $\widetilde{\mathbf{A}}_{BB}$ and $\widetilde{\mathbf{A}}_{BB}^H$ on the diagonal blocks, we can apply $\mathbf{M}_{BB}^{-1}$ to vectors by solving

a linear system with $\widetilde{\mathbf{A}}_{BB}^H$ as the coefficient matrix, then solving a linear system with $\widetilde{\mathbf{A}}_{BB}$ as the coefficient matrix, in sequence (see Algorithm 6). These Helmholtz systems are the same as the forward and adjoint problems, except they are defined on a slightly smaller domain ($\Gamma_i \cup \Omega_b$ instead of $\Omega_t \cup \Gamma_i \cup \Omega_b$) and have impedance boundary conditions instead of Dirichlet boundary conditions on their top surface. Thus the same methods one uses for solving the forward and adjoint equations can be used to solve these systems, at essentially the same cost. Constructing a good preconditioner $\widehat{\mathbf{S}} \approx \mathbf{S}$ is more difficult; this is the subject of the next section.

9.5 Interface Schur complement preconditioner

Direct calculation shows that the non-sparse component of $\mathbf{S}$ can be separated into two terms as follows:

$$\mathbf{M}_{TB}\mathbf{M}_{BB}^{-1}\mathbf{M}_{BT} = \mathbf{M}_{TB}\mathbf{D}^{-1}\mathbf{M}_{BT} + \mathbf{E}_3\mathbf{F}\mathbf{E}_3^T \tag{9.15}$$

where we define

$$\mathbf{D} := \begin{bmatrix} 0 & \widetilde{\mathbf{A}}_{BB}^H \\ \widetilde{\mathbf{A}}_{BB} & 0 \end{bmatrix}, \quad \text{and} \quad \mathbf{F} := \widetilde{\mathbf{A}}_{iB}\widetilde{\mathbf{A}}_{BB}^{-1}\left(\widetilde{\mathbf{W}}_L\right)_{BB}\widetilde{\mathbf{A}}_{BB}^{-H}\widetilde{\mathbf{A}}_{iB}^H,$$

and $\mathbf{E}_3^T$ as the matrix that takes a vector $(\boldsymbol{\xi}_T, \boldsymbol{\eta}_T)$, and returns $\boldsymbol{\eta}_i$. In other words, $\mathbf{E}_3\mathbf{F}\mathbf{E}_3^T$ is a matrix containing zeros in all locations, except for the diagonal block corresponding to interface degrees of freedom for $\boldsymbol{\eta}$, which contains $\mathbf{F}$.

We will carry out the following plan to create a preconditioner $\widehat{\mathbf{S}} \approx \mathbf{S}$:

1. We approximate $\mathbf{M}_{TB}\mathbf{D}^{-1}\mathbf{M}_{BT}$ with a PML truncation (Section 9.5.1)
2. We approximate $\mathbf{F}$ with our product-convolution approximation (Section 9.5.2).
3. After replacing these terms with their approximations within $\mathbf{S}$, we create an approximate expanded system, $\mathbf{M}^{\text{trunc}}$, by undoing the process that formed $\mathbf{S}$ from $\mathbf{M}$. But because of the PML truncation, $\mathbf{M}^{\text{trunc}}$ will be much smaller than $\mathbf{M}$, and because of the product-convolution approximation, the matrix entries of $\mathbf{M}^{\text{trunc}}$ will still be available (Section 9.5.3).

4. We form a hierarchical matrix representation of $\mathbf{M}^{\text{trunc}}$, and factorize it using H-matrix arithmetic.

To approximately solve a linear system with $\mathbf{S}$ as the coefficient operator, we instead solve the approximate expanded system with $\mathbf{M}^{\text{trunc}}$ as the coefficient operator, and extract out the necessary components of the result. We define $\widehat{\mathbf{S}}^{-1}$ to be the operator that carries out this process.

9.5.1 PML truncation

The first term in (9.15),

$$\mathbf{M}_{TB}\mathbf{D}^{-1}\mathbf{M}_{BT},$$

has the following pattern of action:

1. Wavefields in Ω_T are used to generate two sources of waves on Γ_i.
2. Two uncoupled wave equations are solved in the bottom subdomain Ω_B, using the sources on Γ_i from step 1 as the right hand sides.
3. Properties of the resulting wave fields are observed along Γ_i, then transferred to Ω_T.

We can approximate this 3-step process at greatly reduced cost by using partially matched layer (PML) truncations. In Step 2, instead of solving the wave equations in all of Ω_B, we truncate the domain just below Γ_i, using a PML layer to simulate the effect of waves generated on Γ_i leaving the truncated domain without reflection (see Figure 9.2). Then the wave equations are solved in a smaller truncated domain, $\Gamma_i \cup \Omega_k$, where Ω_k is a thin subdomain just below Γ_i containing the PML layer. By performing this PML truncation, we are neglecting the effect of wave reflections off of features below the PML layer, as these reflections are typically less significant than direct, unreflected, waves. PML truncation techniques like this have already been used to great effect for preconditioning *forward* wave problems [84, 179, 197].

Let $\widetilde{\mathbf{A}}^{\text{trunc}}$ and $\widetilde{\mathbf{W}}_L^{\text{trunc}}$ denote the PML-truncated versions of $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{W}}_L$, respectively, which are defined on $\Omega_t \cup \Gamma_i \cup \Gamma_i \cup \Omega_k$ and include the effects of the PML layer in

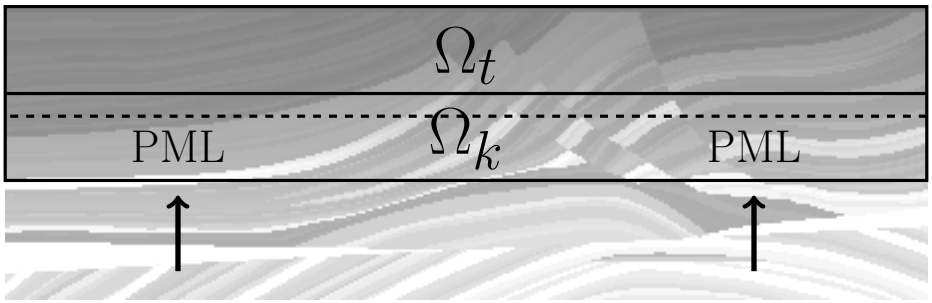


Figure 9.2: Truncated domain. Partially matched layer (PML) below the dashed line in Ω_k. Not drawn to scale; the top and truncated domains will be only 2 finite element cells thick (much thinner than shown here).

Ω_k. Also let $\mathbf{M}^{\text{trunc}}$ and $\mathbf{D}^{\text{trunc}}$ be the same as $\mathbf{M}$ and $\mathbf{D}$, respectively, except defined on $\Omega_t \cup \Gamma_i \cup \Gamma_i \cup \Omega_k$, and with $\widetilde{\mathbf{A}}^{\text{trunc}}$ and $\widetilde{\mathbf{W}}_L^{\text{trunc}}$ replacing $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{W}}_L$, respectively. We approximate the first term in (9.15) as follows:

$$\mathbf{M}_{TB}\mathbf{D}^{-1}\mathbf{M}_{BT} \approx \mathbf{M}_{TK}^{\text{trunc}} \left(\mathbf{D}^{\text{trunc}}\right)^{-1} \mathbf{M}_{KT}^{\text{trunc}},$$

where the subscript $K := (i, k)$ denotes the degrees of freedom that reside in $\Gamma_i \cup \Omega_k$.

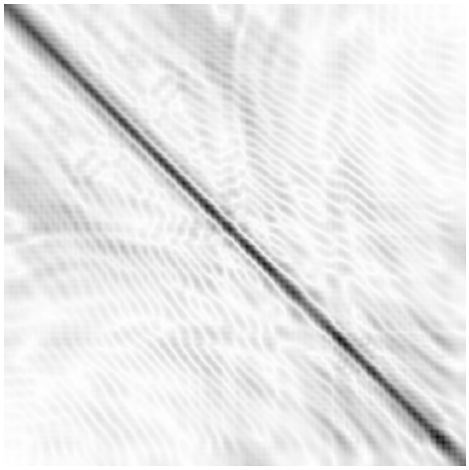
9.5.2 Product-convolution approximation

The operator

$$\mathbf{F} = \widetilde{\mathbf{A}}_{iB}\widetilde{\mathbf{A}}_{BB}^{-1}\left(\widetilde{\mathbf{W}}_L\right)_{BB}\widetilde{\mathbf{A}}_{BB}^{-H}\widetilde{\mathbf{A}}_{iB}^{H}$$

is not amenable to PML truncation approximation since the action of $\mathbf{F}$ involves a sequence of *two* wave solves in Ω_b instead of one. Waves propagate from the interface into the domain in the first wave solve, then the resulting wave field is used as a distributed source for propaging waves back to the interface. Truncating the domain would neglect back propagating waves emanating from below the truncated domain in the second wave solve, which are significant. Instead we approximate $\mathbf{F}$ using product-convolution approximation.

If the soundspeed were horizontally uniform (but possibly vertically variable), and if the domain were semi-infinite in the horizontal direction, and if we ignore discretization concerns and consider the operator $\mathbf{F}$ at the continuum level, then by symmetry $\mathbf{F}$ would be perfectly translation-invariant, so our product-convolution scheme would approximate $\mathbf{F}$ perfectly with one sample point. With a soundspeed



(a) $|\mathbf{F}_{\text{grid}}|$ without impedance-to-impedance interface conditions

(b) $|\mathbf{F}_{\text{grid}}|$ with impedance-to-impedance interface conditions

Figure 9.3: Comparison of $|\mathbf{F}_{\text{grid}}|$ without (9.3a) and with (9.3b) the algebraic change to impedance-to-impedance interface conditions discussed in Section 9.1. Changing to impedance-to-impedance interface conditions reduces complex long-range interactions in $\mathbf{F}_{\text{grid}}$, thus increasing translation-invariance.

that varies horizontally, short-range interactions within $\mathbf{F}$ will be locally translation-invariant, and the length scale for local translation invariance of these short-range interactions will be the length over which the soundspeed varies substantially in the horizontal direction. This length scale is a property of the medium and does not depend on the wave frequency. However, translation invariance fails for long range interactions within $\mathbf{F}$, since these interactions depend on complicated long-distance pathing of high-frequency waves in a non-uniform medium. This is where the change to impedance interface conditions from Section 9.1 pays off. With impedance interface conditions, long range interactions within $\mathbf{F}$ are damped, as compared to long range interactions within the analogous operator that would arise if we did not modify the interface conditions (see Figure 9.3). This damping of long range interactions within $\mathbf{F}$ greatly reduces the convolution-rank of $\mathbf{F}$, so that fewer sample points are required to approximate it.

Two discretization issues prevent us from applying the product-convolution approximation to the matrix $\mathbf{F}$ directly. First, translation-invariance only makes sense at the continuum level for an operator if that operator maps from a dual space of

distributions to a space of functions, so that the impulse response of the operator to a delta function source makes sense. At the discrete level this means the matrix must take dual vectors as input and yield vectors as output. But $\mathbf{F}$ maps vectors to dual vectors instead. Therefore, we sandwich $\mathbf{F}$ between inverse mass matrices $\mathbf{W}_{ii}^{-1}$ so that the resulting operator, $\mathbf{W}_{ii}^{-1}\mathbf{F}\mathbf{W}_{ii}^{-1}$, maps dual vectors to vectors, as desired. Second, the degree of freedom locations for the finite element Lagrange nodes may not reside on a uniform grid, as is required for our discrete product-convolution scheme. We overcome this problem by interpolating functions onto a regular grid before performing the product-convolution approximation, then mapping the resulting functions back after the product-convolution approximation.

Let $\mathcal{V}_\Gamma$ be the restriction to Γ_i of the finite element space used for $\mathbf{u}$, let $\{\phi_k\}_{k=1}^N$ be the finite element basis for $\mathcal{V}_\Gamma$, and let $\{x\}_{k=1}^M$ be the Lagrange nodes for the basis $(\phi_k(x_j) = \delta_{ij})$. Likewise, let $\mathcal{G}$ be the space of piecewise linear functions defined on a regular grid covering Γ_i (not including the PML layer), let $\{\phi'_k\}_{k=1}^{M'}$ be a finite basis for $\mathcal{G}$, and let $\{x'\}_{k=1}^{M'}$ the associated Lagrange nodes. We define $\mathbf{P}_1 : \mathcal{V}_\Gamma \to \mathcal{G}$ and $\mathbf{P}_2 : \mathcal{G} \to \mathcal{V}_\Gamma$ to be the following interpolation matrices mapping between these spaces:

$$(\mathbf{P}_1)_{ij} := \phi_j(x'_i) \quad \text{and} \quad (\mathbf{P}_2)_{ij} := \phi'_j(x_i).$$

By choosing the regular grid sufficiently fine, we have $\mathbf{P}_2\mathbf{P}_1 \approx I$, since this combined operator interpolates functions in the $\mathcal{V}_\Gamma$ to $\mathcal{G}$, then back to $\mathcal{V}_\Gamma$. Thus we have

$$\begin{aligned}\mathbf{F} &\approx \mathbf{W}_{ii}\mathbf{P}_2\mathbf{P}_1\mathbf{W}_{ii}^{-1}\mathbf{F}\mathbf{W}_{ii}^{-1}\mathbf{P}_1^T\mathbf{P}_2^T\mathbf{W}_{ii} \\ &= \mathbf{W}_{ii}\mathbf{P}_2\mathbf{F}_{\text{grid}}\mathbf{P}_2^T\mathbf{W}_{ii}\end{aligned}$$

where we define

$$\mathbf{F}_{\text{grid}} := \mathbf{P}_1\mathbf{W}_{\text{ii}}^{-1}\mathbf{F}\mathbf{W}_{\text{ii}}^{-1}\mathbf{P}_1^T.$$

The operator $\mathbf{F}_{\text{grid}}$ operates on functions defined on a regular grid, and maps dual vectors to vectors. Thus $\mathbf{F}_{\text{grid}}$ satisfies the necessary conditions for us to approximate it with a product-convolution approximation. We illustrate $\mathbf{F}_{\text{grid}}$'s local translation invariance in Figure 9.3. We form the approximation

$$\widehat{\mathbf{F}}_{\text{grid}} \approx \mathbf{F}_{\text{grid}}$$

by applying our product-convolution scheme from Chapter 8 to $\mathbf{F}_{\text{grid}}$. Then we symmetrize the result to get the approximation

$$\widehat{\mathbf{F}}_{\text{grid}}^{\text{sym}} := \left(\widehat{\mathbf{F}}_{\text{grid}} + \widehat{\mathbf{F}}_{\text{grid}}^{H}\right) \Big/ 2.$$

Each application of $\mathbf{F}_{\text{grid}}$ to a vector costs one wave solve with $\widetilde{\mathbf{A}}_{BB}$ as the coefficient matrix and one wave solve with $\widetilde{\mathbf{A}}_{BB}^{H}$ as the coefficient matrix, so constructing $\widehat{\mathbf{F}}_{\text{grid}}$ requires $2r + 2q$ wave solves. Recall that r is the convolution-rank of the approximation, and q is the number of adjoint matrix-vector products used to form the error estimators. Once $\mathbf{F}_{\text{grid}}$ is constructed, we may use it to form the following approximation of $\mathbf{F}$:

$$\widehat{\mathbf{F}} := \mathbf{W}_{ii}\mathbf{P}_2\widehat{\mathbf{F}}_{\text{grid}}^{\text{sym}}\mathbf{P}_2^T\mathbf{W}_{ii}.$$

Since $\widehat{\mathbf{F}}$ is a product of sparse matrices with a product-convolution operators, we may efficiently compute matrix entries of $\widehat{\mathbf{F}}$ and apply blocks of $\widehat{\mathbf{F}}$ to vectors.

9.5.3 Expanded truncated system

Substituting our approximations from Section 9.5.1 and Section 9.5.2 into the the appropriate terms in $\mathbf{S}$ yields the following approximation:

$$\mathbf{S} \approx \widehat{\mathbf{S}} := \mathbf{M}_{TT} - \mathbf{M}_{TK}^{\text{trunc}} \left(\mathbf{D}^{\text{trunc}}\right)^{-1} \mathbf{M}_{KT}^{\text{trunc}} - \mathbf{E}_3\widehat{\mathbf{F}}\mathbf{E}_3^T. \tag{9.16}$$

Just as $\mathbf{S}$ is the Schur complement for the top variables for $\mathbf{M}$, $\widehat{\mathbf{S}}$ is the Schur complement for the top variables for the matrix

$$\widehat{\mathbf{M}}^{\text{trunc}} := \begin{bmatrix} \mathbf{M}_{TT} - \mathbf{E}_3\widehat{\mathbf{F}}\mathbf{E}_3 & \mathbf{M}_{TK}^{\text{trunc}} \\ \mathbf{M}_{KT}^{\text{trunc}} & \mathbf{D}^{\text{trunc}} \end{bmatrix} \tag{9.17}$$

Because of the PML truncation, the matrix $\widehat{\mathbf{M}}^{\text{trunc}}$ is much smaller than $\mathbf{M}$. All matrices within $\widehat{\mathbf{M}}^{\text{trunc}}$ are sparse, except for $\widehat{\mathbf{F}}$, which is a product of sparse matrices with a product-convolution operator. We can solve any linear system with $\widehat{\mathbf{S}}$ as the coefficient operator by instead extending the right hand side by zero into the truncated domain, solving the expanded linear system with $\widehat{\mathbf{M}}^{\text{trunc}}$ as the coefficient operator, then extracting the top components of the solution (see Algorithm 2).

9.5.4 Hierarchical matrix approximation of $\widehat{\mathbf{M}}^{\text{trunc}}$ and definition of $\widehat{\mathbf{S}}$

To solve linear systems with $\widehat{\mathbf{M}}^{\text{trunc}}$ as the coefficient matrix, we construct a hierarchical matrix representation of $\widehat{\mathbf{M}}^{\text{trunc}}$, then factor $\widehat{\mathbf{M}}^{\text{trunc}}$ using H-matrix arithmetic. Since $\widehat{\mathbf{M}}$ is sparse, except for the presence of $\widehat{\mathbf{F}}$, and since $\widehat{\mathbf{F}}$ is the product of sparse matrices with a product-convolution operator, we convert $\widehat{\mathbf{M}}^{\text{trunc}}$ to H-matrix format by forming low-rank approximations of its admissible blocks via randomized SVD (as described in Section 8.3.4). Using CUR approximations to form low rank approximations of admissible blocks is also possible.

To approximately solve the linear system $\mathbf{S}\mathbf{x}_T = \mathbf{y}_T$, we extend $\mathbf{x}_T$ by zero so that it is defined in the truncated domain, permute variables to make the ordering consistent with $\widetilde{\mathbf{M}}^{\text{trunc}}$, apply $\left(\widetilde{\mathbf{M}}^{\text{trunc}}\right)^{-1}$ using the H-matrix factorization of $\widetilde{\mathbf{M}}^{\text{trunc}}$, then extract the top and bottom components of the result and unpermute. We define $\widehat{\mathbf{S}}^{-1}$ to be the result of this process. This process is detailed in Algorithm 2.

Algorithm 2 Application of $\widehat{\mathbf{S}}^{-1}$ to a vector

Computes: $\widehat{\mathbf{x}}_T = \widehat{\mathbf{S}}^{-1}\mathbf{y}_T$
Requires: solver for $\widehat{\mathbf{M}}^{\text{trunc}}$

1: **procedure** SOLVE_S_HAT($\mathbf{y}_T$)
2: Solve $\widehat{\mathbf{M}}^{\text{trunc}} \begin{bmatrix} \widehat{\mathbf{x}}_T \\ \widehat{\mathbf{x}}_K \end{bmatrix} = \begin{bmatrix} \mathbf{y}_T \\ 0 \end{bmatrix}$ using H-matrix factorization of $\widehat{\mathbf{M}}^{\text{trunc}}$.
3: **return** $\widehat{\mathbf{x}}_T$

We could have formed and factorized an H-matrix approximation of $\mathbf{M}$, avoiding the PML truncation of Section 9.5.1 and product-convolution approximation of Section 9.5.2. Why did we not do this? Why is H-matrix approximation of $\widehat{\mathbf{M}}^{\text{trunc}}$ prefferable H-matrix approximation of $\mathbf{M}$? The reasons are:

- $\widehat{\mathbf{M}}^{\text{trunc}}$ is much smaller than $\mathbf{M}$ since it operates on the truncated domain which is quasi lower-dimensional. Thus, even if the H-rank of $\mathbf{M}$ and $\widetilde{\mathbf{M}}^{\text{trunc}}$ were the same, constructing and factorizing $\widetilde{\mathbf{M}}^{\text{trunc}}$ would be much cheaper than constructing and factorizing $\mathbf{M}$.

- The observation operator block, $\frac{1}{\rho}\widetilde{\mathbf{B}}^H\mathbf{Y}\widetilde{\mathbf{B}}$, has a local regularizing effect near the observations since it causes waves to scatter off of the observation locations and decohere. This regularizing effect is minimal within $\mathbf{M}$ since most of the domain is far from the observations. But the regularizing effect is large in $\widehat{\mathbf{M}}^{\text{trunc}}$ since the degrees of freedom for $\widehat{\mathbf{M}}^{\text{trunc}}$ are all in the truncated domain near the observations. As a result, the H-rank of $\widehat{\mathbf{M}}^{\text{trunc}}$ is much less than the H-rank of $\mathbf{M}$.

These two reasons make $\widehat{\mathbf{M}}^{\text{trunc}}$ substantially more amenable to H-matrix approximation than $\mathbf{M}$.

9.6 Setup and solve algorithms

The overall process for solving (9.6) consists of a setup phase (Algorithm 3) and a solve phase (Algorithm 4). The solve phase requires applying $\widehat{\mathbf{M}}^{-1}$ to vectors (Algorithm 5), which in turn requires applying $\mathbf{M}_{BB}^{-1}$ to vectors (Algorithm 6) and applying $\widehat{\mathbf{S}}$ to vectors (Algorithm 2). We describe Algorithm 3 and Algorithm 4 here, and Algorithm 5 and Algorithm 6 in in Appendix C. We already detailed Algorithm 2 in Section 9.5.4.

Algorithm 3 Setup for $\widehat{\mathbf{M}}$ (preconditioner for $\mathbf{M}$)

1: Assemble Helmholtz impedance matrices $\widetilde{\mathbf{A}}_{ii}^{\text{top}}$ and $\widetilde{\mathbf{A}}_{ii}^{\text{bot}}$.
2: Form $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{W}}_L$.
3: Form $\mathbf{M}$.
4: Form PML truncation matrices $\widetilde{\mathbf{A}}^{\text{trunc}}$ and $\widetilde{\mathbf{W}}_L^{\text{trunc}}$.
5: Construct solvers for $\widetilde{\mathbf{A}}_{BB}$ and $\widetilde{\mathbf{A}}_{BB}^H$.
6: Construct solver for $\mathbf{W}_{ii}$.
7: Construct regular grid covering Γ_i.
8: Form $\mathbf{P}_1$ and $\mathbf{P}_2$.
9: Construct product-convolution aproximation $\widehat{\mathbf{F}}_{\text{grid}}$. ▷ Cost: $2r + 2q$ wave solves.
10: Construct H-matrix representation of $\widehat{\mathbf{M}}^{\text{trunc}}$.
11: Factorize H-matrix representation of $\widehat{\mathbf{M}}^{\text{trunc}}$.

Algorithm 4 Solver for $\mathbf{B}^H\mathbf{Y}\mathbf{B} + \rho\mathbf{A}^H\mathbf{W}_L^{-1}\mathbf{A}$

Solves: $\mathbf{B}^H\mathbf{Y}\mathbf{B}\mathbf{u} + \rho\mathbf{A}^H\mathbf{W}_L^{-1}\mathbf{A}\mathbf{u} = \mathbf{f}$
Requires: $\mathbf{M}$, $\mathbf{W}_L$, solver for $\mathbf{A}$, solver for $\mathbf{A}^H$, solver for $\widehat{\mathbf{M}}$

procedure SOLVE_AABB($\mathbf{f}$)
  $\mathbf{p} \leftarrow \mathbf{A}^{-H}\mathbf{f}$
  $\mathbf{g} \leftarrow \begin{bmatrix}\mathbf{0}_{n_T} & -(\mathbf{W}_L\mathbf{p})_T & \mathbf{0}_{n_B} & -(\mathbf{W}_L\mathbf{p})_B\end{bmatrix}^T$
  Solve $\mathbf{M}\mathbf{x} = \mathbf{g}$ for $\mathbf{x}$ with GMRES, using $\widehat{\mathbf{M}}$ as a preconditioner
  $\begin{bmatrix}\boldsymbol{\xi}_T & \boldsymbol{\eta}_T & \boldsymbol{\xi}_B & \boldsymbol{\eta}_B\end{bmatrix}^T \leftarrow \mathbf{x}$
  $\begin{bmatrix}\boldsymbol{\eta}_t & \boldsymbol{\eta}_i\end{bmatrix}^T \leftarrow \boldsymbol{\eta}_T$
  $\begin{bmatrix}\boldsymbol{\eta}_j & \boldsymbol{\eta}_b\end{bmatrix}^T \leftarrow \boldsymbol{\eta}_B$
  $\mathbf{q} \leftarrow \begin{bmatrix}\boldsymbol{\eta}_t & \boldsymbol{\eta}_i + \boldsymbol{\eta}_j & \boldsymbol{\eta}_b\end{bmatrix}^T$
  $\mathbf{w} \leftarrow \mathbf{W}_L(\mathbf{p} - \mathbf{q})$
  $\mathbf{u} \leftarrow \mathbf{A}^{-1}\mathbf{w}$.
  return $\mathbf{u}$

9.7 Numerical Results

We use our method, as described in Algorithm 4, to solve the linear system (9.6). We also use conjugate gradient to solve the system, using $\mathbf{A}^H\mathbf{W}_L^{-1}\mathbf{A}$ as the preconditioner. We denote our method by 'DDWAVE,' and we denote conjugate gradient preconditioned by $\mathbf{A}^H\mathbf{W}_L^{-1}\mathbf{A}$ by 'AA.' We compare the number of Krylov iterations to solve the linear system for DDWAVE to the number of Krylov iterations to solve the linear system for AA. We report results for error tolerances ranging from 10^{-1} to 10^{-6}. We use a fixed error tolerance of 20% for the product-convolution approximation used when constructing the DDWAVE preconditioner. For observations, we use Neumann observations along the top surface of the domain. We discretize the problem with P^2 finite elements on a regular mesh of tetrahedra, with approximately 5 mesh vertices per wavelength. For DDWAVE, GMRES iterations are used to solve the auxiliary linear system $\mathbf{M}\mathbf{x} = \mathbf{g}$, and $\widehat{\mathbf{M}}$ is used as a preconditioner. In either case (DDWAVE or AA), each Krylov iteration requires one forward wave solve and one adjoint wave solve.

In Table 9.1 we compare DDWAVE to AA over a range of penalty parameters, ρ, ranging from 10^2 to 10^{-8}. We use a fixed frequency $\omega = 50$. For all ρ considered,

Table 9.1: **(Iterations vs. ρ)** Number of iterations required to solve $(\rho \mathbf{A}^H \mathbf{W}_L^{-1} \mathbf{A} + \mathbf{B}^H \mathbf{Y} \mathbf{B})\mathbf{u} = \mathbf{f}$, for a variety of penalty parameters $\rho = 10^2, 10^1, \ldots, 10^{-8}$ (Column 1) and relative error tolerances, ranging from 10^{-1} to 10^{-6}. That is the number of Krylov iterations used to construct an approximtion solution $\mathbf{u}$ such that $||\mathbf{u} - \mathbf{u}_{\text{true}}||/||\mathbf{u}_{\text{true}}||$ is less than the desired tolerance, where $\mathbf{u}_{\text{true}}$ is the exact solution to the linear system. Here $\mathbf{f}$ is a i.i.d Gaussian random vector, and $\omega = 50$. We indicate our wave domain decomposition method with 'DDWAVE', and indicate using $\rho \mathbf{A}^H \mathbf{W}_L^{-1} \mathbf{A}$ as a preconditioner by 'AA'. Column 2 shows the number of sample points used to form the product-convolution approximation of F_{grid}, so that the relative error in the product-convolution approximation is less than 20%. Columns 3-8 show the number of GMRES iterations required for solving $\mathbf{M}\mathbf{x} = \mathbf{g}$, so that when $\mathbf{u}$ is built from the $\mathbf{x}$ using the methods discussed in this chapter, $\mathbf{u}$ satisfies the desired error tolerance. Columns 9-14 show the number of conjugate gradient iterations for solving the linear system using $\rho \mathbf{A}^H \mathbf{W}_L^{-1} \mathbf{A}$ as the preconditioner. Dashed entries ('—'), indicate that the method did not converge to the desired tolerance.

ρ		DDWAVE						AA					
	r	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
10^2	16	3	6	8	11	13	15	57	90	138	268	454	661
10^1	18	3	5	8	10	13	15	88	137	227	562	—	—
10^0	18	4	6	8	10	12	15	135	262	553	986	—	—
10^{-1}	16	4	6	9	11	13	15	269	532	—	—	—	—
10^{-2}	18	3	6	8	10	13	15	417	761	—	—	—	—
10^{-3}	16	3	6	8	10	13	15	—	—	—	—	—	—
10^{-4}	15	4	6	8	10	13	15	—	—	—	—	—	—
10^{-5}	18	3	6	8	10	13	15	—	—	—	—	—	—
10^{-6}	15	3	6	8	10	12	15	—	—	—	—	—	—
10^{-7}	14	4	6	8	11	13	—	—	—	—	—	—	—
10^{-8}	16	3	6	9	11	13	—	—	—	—	—	—	—

DDWAVE requires far fewer iterations to converge to the same tolerance. Furthermore, while the required number of Krylov iterations with AA grows as ρ decreases, the number of Krylov iterations required by DDWAVE, and the number of sample points required to construct the DDWAVE preconditioner, remain constant. Overall, DDWAVE is more efficient than AA, and DDWAVE is scalable with respect to ρ (penalty parameter scalable), while AA is not.

In Table 9.2, we compare DDWAVE to AA over a range of frequencies, ω, ranging from 10 to 100. As ω increases, the mesh used to discretize the problem is refined, causing the number of observations to increase. We use a fixed penalty parameter, $\rho = 10^{-1}$. For all frequencies considered, DDWAVE substantially outperforms AA. As a general trend, the number of Krylov iterations for AA grows as ω increases,

Table 9.2: **(Iterations vs. ω)** Number of iterations required to solve $(\rho\mathbf{A}^H\mathbf{W}_L^{-1}\mathbf{A} + \mathbf{B}^H\mathbf{Y}\mathbf{B})\mathbf{u} = \mathbf{f}$, for a variety of angular frequencies $\omega = 10, 20, \ldots, 100$ (Column 1) and relative error tolerances rangin from 10^{-1} to 10^{-6}. That is the number of Krylov iterations used to construct an approximtion solution $\mathbf{u}$ such that $||\mathbf{u} - \mathbf{u}_{\text{true}}||/||\mathbf{u}_{\text{true}}||$ is less than the desired tolerance, where $\mathbf{u}_{\text{true}}$ is the exact solution to the linear system. Here $\mathbf{f}$ is an i.i.d Gaussian random vector, and $\rho = 10^{-1}$. We indicate our wave domain decomposition method with 'DDWAVE', and indicate using $\rho\mathbf{A}^H\mathbf{W}_L^{-1}\mathbf{A}$ as a preconditioner by 'AA'. Column 2 shows the number of sample points used to form the product-convolution approximation of F_{grid}, so that the relative error in the product-convolution approximation is less than 20%. Columns 3-8 show the number of GMRES iterations required for solving $\mathbf{M}\mathbf{x} = \mathbf{g}$, so that when $\mathbf{u}$ is built from the $\mathbf{x}$ using the methods discussed in this chapter, $\mathbf{u}$ satisfies the desired error tolerance. Columns 9-14 show the number of conjugate gradient iterations for solving the linear system using $\rho\mathbf{A}^H\mathbf{W}_L^{-1}\mathbf{A}$ as the preconditioner. Dashed entries ('—'), indicate that the method did not converge to the desired tolerance.

ω		DDWAVE						AA					
	r	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
10	5	3	6	7	9	10	12	118	119	119	119	119	120
20	8	3	5	7	9	11	13	360	429	435	438	439	439
30	12	3	5	8	10	13	15	284	581	759	956	1000	—
40	13	4	6	8	10	12	15	266	435	765	—	—	—
50	16	3	6	8	10	12	14	269	503	—	—	—	—
60	18	3	6	8	11	14	15	235	454	882	—	—	—
70	20	3	6	8	11	14	16	211	414	987	—	—	—
80	21	4	6	9	12	14	16	232	453	768	—	—	—
90	23	4	6	9	13	15	17	209	377	705	—	—	—
100	25	3	6	9	12	15	17	200	384	702	—	—	—

while the number of Krylov iterations for DDWAVE remains constant as ω increases. The required number of sample points used to construct DDWAVE does grow as ω increases, but the growth rate is slow and the number of sample points remains small for all frequencies considered. Overall, DDWAVE performs substantially better than AA. DDWAVE is scalable with respect to ω (frequency scalable), while AA is not.

In Table 9.3, we report the number of sample points required to achieve a desired relative error tolerance for the product-convolution approximation used within DDWAVE. We also report the number of Krylov iterations required for the overall relative error in the solution, $\mathbf{u}$, to drop below a fixed tolerance of 10^{-5}. Here $\omega = 50$ and $\rho = 10^{-2}$. For loose tolerances in the product-convolution approximation, the number of sample points remains small. As the tolerance tightens, the number of

Table 9.3: Impact of the error tolerance for approximating F_{grid} with our product-convolution scheme on computational cost. Here $\omega = 50$ and $\rho = 10^{-2}$. Row 1 shows the tolerance used for $\|F_{\text{grid}} - \widehat{F}_{\text{grid}}\|/\|F_{\text{grid}}\|$. Row 2 shows the number of sample points, r, required to achieve the tolerance. Row 3 shows the number of GMRES iterations required for solving $\mathbf{Mx} = \mathbf{g}$ so that when $\mathbf{u}$ is built from the $\mathbf{x}$ using the methods discussed in this chapter, $||\mathbf{u} - \mathbf{u}_{\text{true}}||/||\mathbf{u}_{\text{true}}|| < 10^{-5}$, where $\mathbf{u}_{\text{true}}$ solves $(\rho\mathbf{A}^H\mathbf{W}_L^{-1}\mathbf{A} + \mathbf{B}^H\mathbf{YB})\mathbf{u}_{\text{true}} = \mathbf{f}$.

product-convolution tolerance	0.50	0.44	0.37	0.31	0.24	0.18	0.11	0.05	0.02
r	5	5	5	9	11	20	35	78	253
Krylov iterations	21	21	21	15	14	12	11	11	11

sample points increases. At the same time, the required number of Krylov iterations decreases as the product-convolution tolerance tightens and the approximation becomes more accurate. Because of error due to the PML truncation (Section 9.5.1), the number of Krylov iterations asymptotes to 11 as the error in the product-convolution approximation becomes small. Overall, as the product-convolution approximation becomes more accurate, the required number of sample points increases while the required number of Krylov iterations decreases. It appears that the 'sweet spot" to minimize both the number of sample points and the number of Krylov iterations occurs for a tolerance of 20% for the product-convolution approximation.

Chapter 10

Conclusion

Existing methods for solving large-scale inverse problems perform poorly if the data are highly informative about the unknown parameter. As certain quantities in the inverse problem increase—e.g., the number of observations, or the Peclet number, or the wave frequency—the informativeness of the data about the parameter increases. This makes small eigenvalues of the data misfit Hessian become large, and large eigenvalues become larger. Then, as these eigenvalues become larger, existing methods for solving the inverse problem perform worse. But if data-scalable Hessian preconditioners are available, they can be used to solve the inverse problem efficiently regardless of how informative the data are about the parameter.

Building data-scalable Hessian preconditioners is difficult; at present, few data-scalable Hessian preconditioners exist. To address this, we developed a novel KKT preconditioner for a diffusion inverse problem, a novel Hessian preconditioner for an advection inverse problem, and a novel preconditioner for an operator that arises in connection with preconditioning the KKT operator for a wave inverse problem. We showed that these preconditioners are data-scalable, and outperform existing preconditioners.

To precondition the diffusion KKT operator, we created an augmented KKT preconditioner based on a block diagonal approximation to an augmented Lagrangian version of the KKT operator. We proved bounds on the condition number of the preconditioned system in an abstract setting, specialized the analysis to the case of source inversion problems with spectral filtering regularization, and tested the preconditioner numerically on the diffusion source inversion problem with highly informative data and small regularization parameter. Our analysis and numerical results showed that the augmented KKT preconditioner is mesh and data scalable when the regularization does not over-penalize highly informed parameter modes and does not

under-penalize uninformed modes. The development and analysis of the augmented KKT preconditioner were problem-independent, so it could also be applied to other inverse problems that satisfy certain requirements. The primary limitation of the augmented KKT preconditioner is that it applies only to inverse problems with invertible T operators. Many problems of practical interest have invertible T operators, including source inversion problems in which the parameter and state variables are discretized with the same mesh and finite elements. Extending the preconditioner to non-invertible T operators is a direction for future research.

To precondition the advection Hessian, we created a matrix-free adaptive product-convolution operator approximation scheme. The efficiency of the scheme depends on the degree to which the operator being approximated is locally translation-invariant. The scheme improves on existing product-convolution schemes by providing an automated method for performing adaptivity, and by addressing issues related to boundaries. We used the scheme to build a preconditioner for the advection Hessian that far outperforms low-rank approximation and regularization preconditioning. As the Peclet number increased, the cost to solve the Hessian system remained roughly constant using our new preconditioner.

Our product-convolution scheme is also well-suited for approximating or preconditioning operators that arise in Schur complement methods for solving partial differential equations (PDEs), operators that arise in image deblurring, integral operators, covariance operators with spatially varying kernels, and Dirichlet-to-Neumann maps or other Poincaré–Steklov operators in multiphysics problems. These operators are often dense, implicitly defined, and high-rank, making them difficult to approximate with standard techniques. In Appendix E we provide numerical results of applying the scheme to other, non-Hessian, operators. The product-convolution scheme is best suited to moderate accuracy approximations (say, 80% to 99% accuracy), which are well-suited for the purpose of building preconditioners. A target for future research is to apply our product-convolution scheme to the wave Hessian directly.

As a step towards preconditioning the wave KKT operator, we created a domain decomposition preconditioner for the operator $B^*B+\rho A^*A$. This operator arises when

one applies the augmented Lagrangia KKT framework to the wave KKT operator for inversion using a single frequency and a single source. Within the preconditioner, we used the adaptive product-convolution scheme to approximate a second order interface Schur complement. We saw that our preconditioner for $B^*B + \rho A^*A$ is scalable with respect to the wave frequency and the penalty parameter: the preconditioner performs well when ω is large, and when ρ is small. Using the preconditioner for $B^*B+\rho A^*A$ to build a preconditioner for the complete inverse problem KKT operator is a direction for future research.

Appendices

Appendix A

Bayesian Solution Methods

Whereas the computational task in the deterministic framework is obvious—solve an optimization problem to get q—the task in the Bayesian framework is less clear. The "solution" to the Bayesian inverse problem, $\pi(q|y)$, is a posterior probability distribution defined on a high-dimensional space, and is therefore far too large to be discretized, computed, and stored on a computer.[1]

In order to extract useful statistical information from $\pi(q|y)$, it is sufficient to generate samples from it. With samples $\{q_k\}_{i=1}^n$ drawn from $\pi(q|y)$, expectations of arbitrary functions $\psi(q)$ can be computed with Monte-Carlo approximations, e.g.,

$$\mathbb{E}_{\pi(q|y)}\psi \approx \frac{1}{n}\sum_{i=1}^{n}\psi(q_i).$$

This allows one to, for example, compute the mean, variance, and higher moments of quantities of interest.

Bayesian sampling methods that do not account for the directional scalings of the probability distribution being sampled from cannot efficiently sample from high-dimensional probability distributions. The eigenstructure of the Hessian locally characterizes the directional scalings of the posterior probability distribution in an inverse problem. Hence, in order to be effective for large-scale problems, Bayesian sampling methods must either use the Hessian, or perform operations that are effectively equivalent to using the Hessian. Data-scalable use of the Hessian in these methods can be performed efficiently if a good Hessian preconditioner is available.

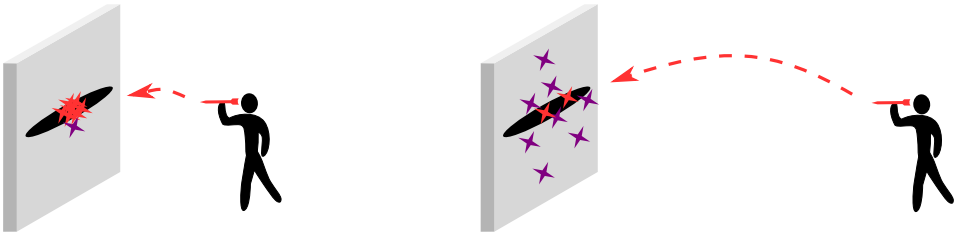
[1] For low- and moderate-dimensional parameters, one can directly characterize $\pi(q|y)$ using polynomial chaos or Karhunen–Loève expansions [101], measure-theoretic approaches towards the inverse problem [57], sparse grids [63], or other non-sampling based methods. However, in large-scale big data problems (where the dimension of the parameter subspace informed by the data is large) these approaches are infeasible.

A.1 Throwing darts at an ellipsoid

Imagine throwing darts at an ellipsoidal dartboard (See Figure A.1). This serves as an analogy for the proposal step in Markov Chain Monte-Carlo (MCMC), importance sampling, and other procedures that sample from the posterior using a proposal distribution. The dart-throwing distribution corresponds to the proposal distribution, and the ellipsoid corresponds to the region of non-negligible probability for the posterior.

If you stand close to the dartboard, your throws will hit the ellipsoid with high probability, but will cluster near the center and rarely hit regions to either side. If you stand far away from the dartboard you will rarely hit the ellipsoid, but your hits will cover the ellipsoid more evenly. More generally, if the directional scalings of your dart throwing distribution do not match those of the dartboard, you cannot simultaneously hit the dartboard with high probability and achieve high coverage of the dartboard.

Likewise, sampling procedures that use a proposal are efficient when samples from the proposal provide good coverage of the region of high probability of the posterior, and land in that region often. Thus the directional scalings of the proposal distribution should match the directional scalings of the posterior as much as possible.



(a) High success, low coverage.

(b) Low success, high coverage.

Figure A.1: Using a proposal distribution with covariance that does not match that of the true distribution is analogous to trying to hit a ellipsoidal dartboard by throwing darts at it. A.1a: if you stand close to the dartboard, your throws will usually hit it, but the hits will provide poor coverage. A.1b: if you stand far from the dartboard, your throws will usually miss, but the hits will provide good coverage.

A.2 Sampling methods

Since the Hessian is the inverse covariance of a local Gaussian approximation to the posterior (see Section 2.6), sampling methods that do not use the Hessian (either explicitly or implicitly) struggle to capture the directional scalings of the posterior, and therefore perform poorly. In Metropolis-Hastings Markov Chain Monte-Carlo, poorly scaled proposal distributions result in either low acceptance rates or small step sizes. In the dart throwing analogy from Section A.1, low acceptance rates correspond to standing far away from the dartboard, and small step sizes correspond to standing close to the dartboard. Both of these scenarios lead to high autocorrelation of the chain, so that long sample chains are required to generate a small number of statistically independent samples.

In resampling methods such as importance sampling, failure to account for the directional scalings of the posterior leads to weight collapse, where almost all of the weight is assigned to a single sample. In the dart throwing analogy, this corresponds to the case where only one dart hits the dartboard. In particle methods, the analog of weight collapse is particle collapse, where the particles all converge to the same point. To prevent particle collapse, a mutation step is typically performed. The mutation step uses other techniques, which require properly scaled proposal distributions.

Delayed rejection adaptive metropolis (DRAM) [110] and ensemble Kalman filtering methods [9, 89] account for the directional scalings of the posterior by iteratively approximating the posterior covariance during the sampling process. However, building the covariance approximation from scratch is slow, so these methods also perform poorly on large-scale big data distributed parameter inverse problems. Rather than build the proposal covariance iteratively, the Laplace approximation proposal uses the local Gaussian approximation from Section 2.6, evaluated at the MAP point. In this case, the inverse of the Hessian is the proposal covariance. Extensions of this idea use information from the Hessian at many points to build an approximation of the global covariance [70, 71, 178]. Stochastic Newton MCMC [140, 164] uses Hessian information to construct a Gaussian approximation of the posterior at each step within an MCMC chain. Proposals at a given point in the chain are drawn from the

local Gaussian approximation at that point.

More advanced methods such as randomize-then-optimize and Riemannian manifold MCMC account for the bending of the posterior in addition to its directional scalings. Randomize-then-optimize [25, 158] generates proposed samples by solving nonlinear deterministic optimization problems of the form (2.8), but with randomly perturbed data. In Riemannian manifold MCMC [56], one views the Gauss-Newton Hessian as a Riemannian metric on the parameter space. High-quality proposals are generated by tracing geodesics on the associated Riemannian manifold.

Stein variational gradient methods [133] move particles so that they approximate the posterior as well as possible, as measured by KL-divergence. Unlike other particle methods that require mutation, Stein variational methods avoid particle collapse by including a constraint that pushes particles apart. Standard versions of Stein variational gradient methods use a modified version of the gradient of the negative log posterior to move the particles at each stage, yielding a method analogous to gradient descent. Like gradient-based methods for the deterministic problem, Stein variational gradient methods require large numbers of iterations to converge for ill-conditioned problems. However, a Newton version of the Stein variational gradient method, requiring Hessian solves, was recently proposed [78]. On examples presented in [78], the standard version of the method required thousands of iterations for convergence while the Newton version required approximately ten iterations.

A.3 Required Hessian operations

The Bayesian sampling methods from Section A.2 that use the Hessian must perform subsets of the following operations:

(a) Applying the Hessian to vectors:

$$z \mapsto Hz.$$

(b) Performing Hessian solves:

$$z \mapsto H^{-1}z.$$

(c) Applying the Hessian inverse square root:

$$z \mapsto H^{-1/2} z.$$

(d) Computing the ratio of the determinant of the Hessian at one point (H_1) to that of another point (H_2):

$$\frac{\det(H_1)}{\det(H_2)}. \tag{A.1}$$

The Laplace approximation and stochastic Newton require (a) and (d) to evaluate the proposal density, and (c) to generate samples. Randomize-then-optimize requires solving deterministic optimization problems. This can be done efficiently with Newton's method, which requires (b). Evaluating the density of the randomize-then-optimize proposal (in order to metropolize the method) also requires (a) and (d). Riemannian manifold MCMC requires (a), (c), and (d). Riemannian manifold MCMC also requires computing the action of higher-order derivative tensors, but performing Hessian operations remains the primary computational bottleneck. The Newton variant of the Stein variational gradient method requires (a) and (b).

Applying the Hessian to vectors, (a), can be performed efficiently using adjoint methods. We will describe how to do this in Section 5.2. Hessian solves, (b), can be performed efficiently using Krylov methods if a good Hessian preconditioner is available (see Section 3.3).

Current implementations of sampling methods that use the Hessian inverse square root, (c), and the Hessian determinant ratio, (d), compute low-rank approximiations of the prior preconditioned data misfit Hessian, then use these low-rank approximations to perform these operations. However, in Chapter 4 we will see that the rank of H_d grows with the informativeness of the data, making computation of these low-rank approximations more expensive as more informative data are included in the inversion. Thus these methods are not data-scalable in current implementations.

However, if a data-scalable Hessian preconditioner is available, we can use rational approximations to perform operations (c) and (d) in an efficient, data-scalable manner. We briefly summarize rational matrix function approximations in Section A.3.1, and describe how to use these approximations to perform (c) in Section A.3.2 and (d) in Section A.3.3. For (c) and (d) we will use rational approximations to the inverse square root and logarithm.

A.3.1 Rational matrix function approximation

Rational approximation of a function f of a matrix M takes the form

$$f(M) \approx \frac{p(M)}{q(M)},$$

where p and q are polynomials. For concreteness, assume that the degree of q is m, that the degree of p is less than or equal to m, and that the roots r_i of q are negative. This is the case for the inverse square root and the logarithm. In this case, using partial fractions yields the following expansion of the rational function:

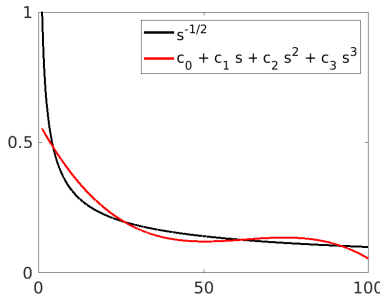
$$\frac{p(M)}{q(M)} = c_0 + c_1 \left(M + \mu_1 I\right)^{-1} + c_2 \left(M + \mu_2 I\right)^{-1} + \cdots + c_m \left(M + \mu_m I\right)^{-1},$$

where $\mu_i := -r_i$ are positive. Applying this rational approximation to a vector,

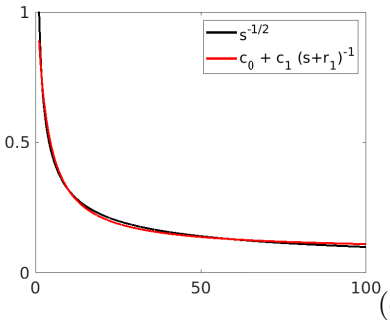
$$z \mapsto \frac{p(M)}{q(M)} z,$$

therefore requires solving m linear systems with positively shifted versions of M as the coefficient operator. These linear systems can be solved efficiently with Krylov methods if a good preconditioner is available (see Section 3.3). Rather than nest a Krylov method within a rational approximation, one may instead use preconditioned *rational Krylov methods* [108, 109] to perform the whole process at once.

Rational approximations converge rapidly. Specifically, suppose that f is analytic on $\mathbb{C}$ except for singularities or a branch cut on $(-\infty, 0]$, and suppose that M is a real matrix with positive eigenvalues. In this case several methods exist for choosing c_i and μ_i such that the rational approximations converge geometrically in m, and the convergence constant scales logarithmically in the condition number of M. See [117] and the references therein for a discussion of these issues.



(a) Third order polynomial approximation of $s^{-1/2}$.



(b) First order rational approximation of $s^{-1/2}$.

order	polynomial error	rational error
1	$3.6 \cdot 10^{-1}$	$4.5 \cdot 10^{-2}$
2	$2.6 \cdot 10^{-1}$	$3.1 \cdot 10^{-3}$
3	$1.9 \cdot 10^{-1}$	$2.1 \cdot 10^{-4}$
4	$1.4 \cdot 10^{-1}$	
5	$1.1 \cdot 10^{-1}$	

(c) L^2 relative error for polynomial and rational approximations of $s^{-1/2}$ on $[1, 100]$.

Figure A.2: Least-squares optimal approximation of $s^{-1/2}$ on $[1, 100]$ using polynomial approximation compared to rational approximation. Rational approximation performs much better. Intuitively, this makes sense. The graph of $s^{-1/2}$ looks like a shifted version of the graph of s^{-1}, but does not look like shifted versions of the graphs of s, s^2, s^3, etc.

It follows that highly accurate rational approximations can be achieved with

$$m = O\left(\log \kappa\right)$$

terms, where κ is the condition number of M. In contrast, oscillations cause polynomial approximation of $f(M)$ to perform much worse, leading to algebraic dependence on κ. Figure A.2 illustrates the difference between polynomial approximation and rational approximation for the inverse square root.

A.3.2 Inverse square root

To apply the Hessian inverse square root to a vector ((c) in Section A.3), one can use the rational approximation method of Section A.3.1, with $f(s) = s^{-1/2}$ and $M = H$. This requires solving m linear systems of the form

$$(H + \mu I)z = b,$$

which can be performed efficiently if good Hessian preconditioners are available. The number of solves, m, will grow with the logarithm of the condition number of H, which grows as increasingly informative data are included in the inverse problem. But since the dependence on the condition number is only logarithmic, we still consider this method to be data-scalable.

A.3.3 Determinant ratio

Computing the determinant ratio from (d) in Section A.3 is equivalent to computing

$$\det(H_2^{-1}H_1). \tag{A.2}$$

Recalling the identity

$$\log\det(M) = \operatorname{trace}(\log(M)),$$

we see that(A.2) can also be computed by computing

$$\operatorname{trace}(\underbrace{\log(H_2^{-1}H_1)}_{X}), \tag{A.3}$$

and exponentiating the result. The idea here is to compute this trace with a matrix-free randomized trace estimator [18], using rational approximations to the matrix logarithm to apply $\log(H_2^{-1}H_1)$ to vectors as needed. Specifically, given a matrix X, we may estimate its trace with the Monte-Carlo sum:

$$\operatorname{trace}(X) = \mathbb{E}\left(z^*Xz\right) \approx \frac{1}{n}\sum_{i=1}^{n} z_i^*Xz_i,$$

where z and $\{z_i\}_{i=1}^n$ are random vectors with Gaussian independent and identically distributed entries. Computing this Monte-Carlo sum requires applying X to z_i for $i = 1, \ldots, n$. For us, $X = \log(H_2^{-1}H_1)$. Hence we may use this randomized trace estimator to compute (A.3) if we can perform n matrix-vector products of the form

$$z \mapsto \log(H_2^{-1}H_1)z.$$

Each of these matrix-vector products can be performed using the rational approximation method described in Section A.3.1, with $M = H_2^{-1}H_1$ and $f(s) = \log(s)$. This requires solving linear systems of the form $(H_2^{-1}H_1 + \mu I)z = b$, or equivalently,

$$\left(H_1 + \mu H_2\right)z = H_2 b.$$

These systems can be solved efficiently if good Hessian preconditioners are available.

The numerical properties of $H_2^{-1}H_1$ work in our favor. Likely, the spectrum of $H_2^{-1}H_1$ will be clustered around 1, and rational approximations to the matrix logarithm perform best when the spectrum is clustered in this way.

Appendix B

Dimension dependence of sparse-direct methods

The dominant cost within sparse-direct methods for problems with d spacetime dimensions is the construction and factorization of dense Schur complement matrices associated with $(d-1)$-dimensional groups of degrees of freedom called "separator fronts." For a problem discretized on cube with n gridpoints per dimension (e.g., $n \times n \times n$ for $d = 3$), constructing and factoring these Schur complements requires $O\left(\left(n^{(d-1)}\right)^3\right)$ operations, and $O\left(\left(n^{(d-1)}\right)^2\right)$ memory. If $d = 2$, it turns out that this is the same amount of memory that is required to store the right hand side vector for the problem (up to a constant), so sparse-direct methods are mesh-scalable, in terms of memory, in 2 spacetime dimensions. In 3 or more dimensions, significantly more memory is required to factorize the Schur complement as compared to storing the right hand side, so sparse-direct methods are not mesh-scalable in terms of memory in 3 dimensions. Sparse-direct methods are mesh-scalable in terms of operation count in 1 dimension, but not in 2 or more dimensions.

We illustrate the dimension-dependence of sparse-direct methods by describing LU factorization of a matrix with nested-dissection ordering. All other memory-efficient large-scale sparse-direct methods, e.g., multi-frontal methods, have similar properties. In nested-dissection ordering, the degrees of freedom are partitioned into three groups: two groups which do not interact with each other, and a third group called the *separator front* which interacts with itself and the other two groups (see Figure B.1). Ideally, we want the separator front to be as small as possible. Each of the two non-interacting groups are partitioned into three subgroups, separated by new separator fronts. The partitioning process continues recursively until the number of degrees of freedom in each non-interacting group is less than a predetermined threshold. Abstractly, the partitions form a binary tree in which the internal nodes of the tree are the separator fronts, and the leaves are small non-interacting groups

of degrees of freedom that are not partitioned any further.

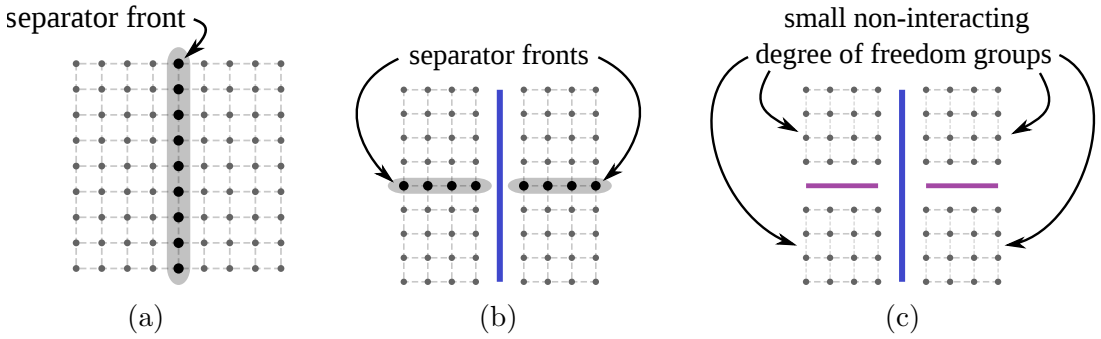


Figure B.1: Separator fronts partition degrees of freedom into three parts. Two parts (gray dots) that do not interact with each other, and one (black dots) that interact with both other parts. The process continues recursively until only small non-interacting degree of freedom groups remain. Illustrations depict degree-of-freedom locations in physical space. Dashed lines indicate which degrees of freedom interact with which other degrees of freedom; here degrees of freedom interact with only their neighbors.

The matrix is now ordered so that rows and columns corresponding to degrees of freedom in lower levels of the tree precede rows and columns corresponding to degrees of freedom in higher levels of the tree. When Gaussian elimination eliminates a group of degrees of freedom in this ordering, Schur complement updates fill in portions of the L and U factor matrices associated with all separator fronts "up the tree" which touch these degrees of freedom (see Figure B.2). Whereas the subatrices associated with separator fronts are sparse in the original matrix, these Schur complement updates make them dense in the factor matrices. We thus must store and perform factorizations on dense submatrices associated with internal interactions within each separator front.

In d dimensions, the separator fronts are $(d-1)$-dimensional (see Figure B.3). Thus sparse-direct methods for d-dimensional problems require storing and factorizing dense matrices associated with internal interactions within $(d-1)$-dimensional separator fronts. Standard dense factorization of these matrices requires $O\left((d-1)^2\right)$ memory and $O\left((d-1)^3\right)$ operations.

Suppose we have a 2D $n \times n$ mesh of gridpoints. The largest separator front is a collection of n gridpoints residing on a 1D line, so the Schur complement for the separator front is a dense $n \times n$ matrix, which requires $O(n^2)$ memory and $O(n^3)$

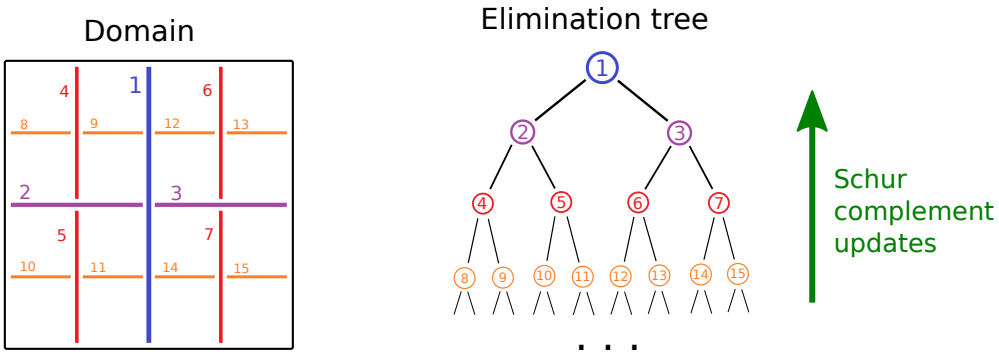


Figure B.2: B.2a: partitioning of the degrees of freedom in a 2D domain by nested-dissection ordering. Lines indicate seperator fronts. B.2b: tree of degree of freedom groups. The root is the biggest separator front. Internal nodes are separator fronts. Leaves are the noninteracting boxes at the lowest level.

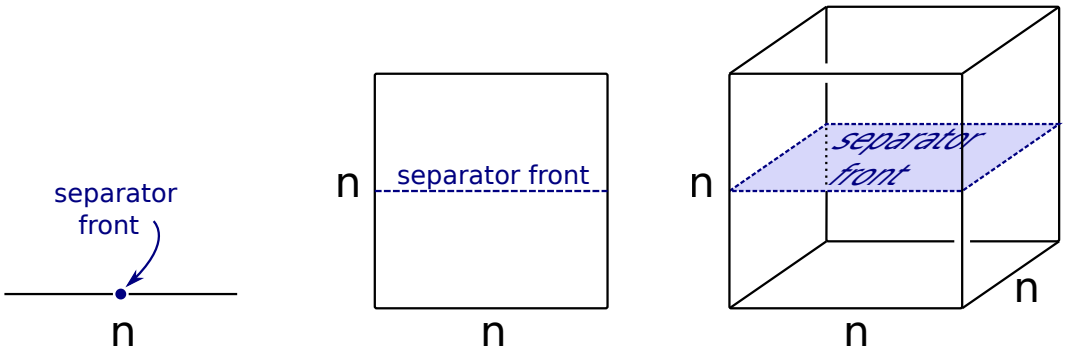


Figure B.3: Separator fronts in a d-dimensional problem are $(d-1)$-dimensional. In a 1D problem with n degrees of freedom, the separator front contains $O(1)$ points. In a 2D $n \times n$ problem, the separator front contains $O(n)$ points. In a 3D $n \times n \times n$ problem, the separator front contains $O(n^2)$ points.

operations to build and factorize. Since the total number of degrees of freedom is n^2, building and factorizing this Schur complement requires the same amount of memory (up to a constant) as simply storing the solution to the problem (n^2 vs. $O(n^2)$). This makes sparse-direct factorization mesh-scalable in terms of memory. However, it takes n times more operations to build and factorize this Schur complement than it does to look at the solution (n^2 vs. $O(n^3)$), so sparse-direct factorization is not mesh-scalable in terms of operations.

If we have a 3D $n \times n \times n$ mesh, the largest separator front is a collection of n^2 gridpoints on a 2D plane, so building and factorizing the Schur complement associated with that separator front requires $O\left((n^2)^2\right) = O(n^4)$ memory and $O\left((n^2)^3\right) = O(n^6)$ operations. Since the number of degrees of freedom in the mesh is only n^3, sparse direct factorization is therefore not mesh-scalable in 3D.

Appendix C

Additional Algorithms

Algorithm 5 Application of $\widehat{\mathbf{M}}^{-1}$ to a vector

Solves: $\underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{M}_{TB}\mathbf{M}_{BB}^{-1} \\ 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \widehat{\mathbf{S}} & 0 \\ \mathbf{M}_{BT} & \mathbf{M}_{BB} \end{bmatrix}}_{\widehat{\mathbf{M}}} \underbrace{\begin{bmatrix} \mathbf{x}_T \\ \mathbf{x}_B \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \mathbf{g}_T \\ \mathbf{g}_B \end{bmatrix}}_{\mathbf{g}}$

Requires: blocks of $\mathbf{M}$, solver for $\mathbf{M}_{BB}$, solver for $\widehat{\mathbf{S}}$

1: **procedure** SOLVE_M_HAT($\mathbf{g}$)
2: $\begin{bmatrix} \mathbf{g}_T & \mathbf{g}_B \end{bmatrix}^T \leftarrow \mathbf{g}$
3: $\mathbf{z} \leftarrow \mathbf{g}_T - \mathbf{M}_{TB}\mathbf{M}_{BB}^{-1}\mathbf{g}_B$ ▷ Cost: 2 wave solves.
4: $\mathbf{x}_T \leftarrow \widehat{\mathbf{S}}^{-1}\mathbf{z}$.
5: $\mathbf{x}_B \leftarrow \mathbf{M}_{BB}^{-1}\left(\mathbf{g}_B - \mathbf{M}_{BT}\mathbf{x}_T\right)$ ▷ Cost: 2 wave solves.
6: $\mathbf{x} \leftarrow \begin{bmatrix} \mathbf{x}_T & \mathbf{x}_B \end{bmatrix}^T$
7: **return** $\mathbf{x}$

Algorithm 6 Application of $\mathbf{M}_{BB}^{-1}$ to a vector

Solves: $\underbrace{\begin{bmatrix} 0 & \widetilde{\mathbf{A}}_{BB}^H \\ \widetilde{\mathbf{A}}_{BB} & -\left(\widetilde{\mathbf{W}}_L\right)_{BB} \end{bmatrix}}_{\mathbf{M}_{BB}} \underbrace{\begin{bmatrix} \boldsymbol{\xi}_B \\ \boldsymbol{\eta}_B \end{bmatrix}}_{\mathbf{x}_B} = \underbrace{\begin{bmatrix} \mathbf{b}_B \\ \mathbf{c}_B \end{bmatrix}}_{\mathbf{g}_B}.$

Requires: $\widetilde{\mathbf{W}}_L$, solver for $\widetilde{\mathbf{A}}_{BB}$, solver for $\widetilde{\mathbf{A}}_{BB}^H$

1: **procedure** SOLVE_MBB($\mathbf{g}_B$)
2: $\begin{bmatrix} \mathbf{b}_B & \mathbf{c}_B \end{bmatrix}^T \leftarrow \mathbf{g}_B$
3: $\boldsymbol{\eta}_B \leftarrow \widetilde{\mathbf{A}}_{BB}^{-H}\mathbf{b}_B$ ▷ Cost: 1 wave solve.
4: $\boldsymbol{\xi}_B \leftarrow \widetilde{\mathbf{A}}_{BB}^{-1}\left(\mathbf{c}_B + \left(\widetilde{\mathbf{W}}_L\right)_{BB}\boldsymbol{\eta}_B\right)$ ▷ Cost: 1 wave solve.
5: $\mathbf{x}_B \leftarrow \begin{bmatrix} \boldsymbol{\xi}_B & \boldsymbol{\eta}_B \end{bmatrix}^T$
6: **return** $\mathbf{x}_B$

Appendix D

Additional Proofs

Proof of Theorem 2. Elements of the Krylov subspace $\mathcal{K}_j$ are in bijective correspondence with the set of all $(j-1)^{\text{th}}$ order polynomials, $\mathcal{P}_{j-1}$, via the map $P \leftrightarrow P(M)b$ (including polynomials P such that $P(0) \neq 1$). This follows from the definition of $\mathcal{K}_i$ and invertibility of M. Using this bijection, along with the definition of x_i and algebraic manipulations, yields

$$\begin{aligned} \|b - Mx_j\|^2 &= \min_{z \in \mathcal{K}_j} \|b - Mz\|^2 \\ &= \min_{P \in \mathcal{P}_{j-1}} \|b - MP(M)b\|^2 \\ &= \min_{Q \in \mathcal{Q}_j} \|Q(M)b\|^2. \end{aligned} \tag{D.1}$$

The theorem follows from expanding $Q(M)$ and b in the eigenvector basis of M in (D.1). □

Proof of Theorem 5. We proceed one derivative at a time, using ordinary techniques from multivariable calculus.

Zero$^{\text{th}}$ derivative: The procedure for computing $\mathcal{J}$ is a restatement of the definitions of $\mathcal{J}$ and u.

First derivative: By the chain rule,

$$g^* := \frac{d\mathcal{J}}{dq} = \frac{\partial \mathcal{J}_d}{\partial u}\frac{du}{dq} + \frac{d\mathcal{J}_R}{dq}. \tag{D.2}$$

Direct calculation shows that

$$\frac{\partial \mathcal{J}_d}{\partial u} = (Bu - y)^* YB \tag{D.3}$$

and

$$\frac{d\mathcal{J}_R}{dq} = (q - \overline{q}_0)^* R. \tag{D.4}$$

Differentiating the state equation yields:

$$\begin{aligned} 0 = \frac{d}{dq}\left(Au - f\right) &= A\frac{du}{dq} + \frac{\partial A}{\partial q}u - \frac{\partial f}{\partial q} \\ &= A\frac{du}{dq} + T, \end{aligned}$$

which implies that

$$\frac{du}{dq} = -A^{-1}T. \tag{D.5}$$

Substituting (D.3), (D.4), and (D.5) into (D.2) yields:

$$g^* = \underbrace{-\left(Bu - y\right)^* Y B A^{-1}}_{=:\lambda^*} T + \left(q - \overline{q}_0\right)^* R.$$

Defining a new variable $\lambda := -A^{-*}B^*Y\left(Bu - y\right)$, which is equivalent to the adjoint equation in Table 5.1 after rearrangement, we see that the gradient takes the form shown in Table 5.1.

Second derivative: To compute Hp, we compute the directional derivative of g in direction p, reducing expressions as far as possible using the chain rule and the product rule, stopping when we reach variables we already know, operators we already know, and the new variables $\eta = \frac{du}{dq}p$ and $\xi = \frac{d\lambda}{dq}p$. This yields:

$$\begin{aligned} Hp = \frac{dg}{dq}p &= Rp + \left(\frac{dT^*}{dq}p\right)\lambda + T^*\left(\frac{d\lambda}{dq}p\right) \\ &= Rp + \left(\frac{dT^*}{dq}p\right)\lambda + T^*\xi \\ &= Rp + \left(\frac{\partial T^*}{\partial q}p\right)\lambda + \left(\frac{\partial T^*}{\partial u}\frac{\partial u}{\partial q}p\right)\lambda + T^*\xi \\ &= Rp + \Xi p + \Theta^*\eta + T^*\xi. \end{aligned}$$

Differentiating the state equation in direction p and performing algebraic manipulations yields the incremental forward equation for η shown in Table 5.1.

To derive the incremental adjoint equation for ξ, we differentiate the adjoint equation in Table 5.1 in direction p. This yields:

$$\frac{d}{dq}\left(A^*\lambda\right)p = \frac{d}{dq}\left(-B^*Y\left(Bu - y\right)\right)p,$$

which expands to

$$A^* \frac{d\lambda}{dq} p + \left(\frac{dA^*}{dq} \lambda \right) p = -B^* Y B \frac{du}{dq} p,$$

and this simplifies to the incremental adjoint equation shown in Table 5.1.

□

Proof of Theorem 6. Differentiating $\mathcal{G} = Bu(q)$ with respect to q then using (D.5) yields (5.1). Formula (5.2) follows from substituting (5.1) into the formula $H_d^{\text{gn}} = GY^*G$. Formula (5.3) follows directly from the definitions. Formulas (5.4), (5.5), and (5.6) follow from computation of the partial derivatives of $\mathcal{L}$ and $\mathcal{L}^{\text{gn}}$, which are defined in (3.9) and (3.12), respectively. □

Proof of Theorem 7. Substituting the formulas for blocks of K in (5.5) into the right hand side of (5.7) and applying the result to a vector p yields:

$$Rp + \Xi p - \begin{bmatrix} \Theta^* & T^* \end{bmatrix} \begin{bmatrix} B^* Y B & A^* \\ A & \end{bmatrix}^{-1} \begin{bmatrix} \Theta p \\ Tp \end{bmatrix}. \tag{D.6}$$

Let η be the solution to the incremental forward equation and let ξ be the solution to the incremental adjoint equation. Direct calculation shows that (D.6) simplifies to

$$Rp + \Xi p - \begin{bmatrix} \Theta^* & T^* \end{bmatrix} \begin{bmatrix} -\eta \\ -\xi \end{bmatrix},$$

which further simplifies to

$$Rp + \Xi p + \Theta^* \eta + T^* \xi.$$

Since this is the formula for applying H to p from Table 5.1, and since p was arbitrary, the left and right sides of (5.7) must be equal.

The fact that H^{gn} is the Schur complement of K^{gn} for q follows from the formulas for these operators in Theorem 6 and block linear algebra. □

Proof of Corollary 2. Let the subscript x denote degrees of freedom associated with both u and λ in K. Since H is the Schur complement for q in K, we have the following block-LU factorization of K:

$$K = \begin{bmatrix} K_{qq} & K_{qx} \\ K_{xq} & K_{xx} \end{bmatrix} = \begin{bmatrix} I & K_{qx} K_{xx}^{-1} \\ & I \end{bmatrix} \begin{bmatrix} H & \\ K_{xq} & K_{xx} \end{bmatrix}. \tag{D.7}$$

In these LU factors, we encounter the operator

$$K_{xx} = \begin{bmatrix} B^*YB & A^* \\ A & \end{bmatrix}.$$

The block structure of K_{xx} lets us solve a linear system with K_{xx} as the coefficient operator by performing a sequence of solves with A and A^* as coefficient operators:

$$\begin{bmatrix} B^*YB & A^* \\ A & \end{bmatrix} \begin{bmatrix} w \\ \chi \end{bmatrix} = \begin{bmatrix} e \\ f \end{bmatrix} \quad \Leftrightarrow \quad \begin{cases} Aw = f \\ A^*\chi = e - B^*Bw. \end{cases} \tag{D.8}$$

The implication for K and H follows from solving the equation on the right side of the implication in (5.8) by performing two triangular solves using the LU factorization of K in (D.7), and using (D.8) both times K_{xx}^{-1} must be applied in these triangular solves.

The same reasoning yields the implication for K^{gn} and H^{gn}. □

Proof of Theorem 8. To prove the result for H and H^{gn}, recall that $H_d \simeq \frac{d^2\mathcal{J}_d}{dq^2}$ and notice that $\mathcal{J}_d = \frac{1}{2}\rho^*Y\rho$, where $\rho := \mathcal{G}(q) - y$ denotes the data residual. Differentiating $\mathcal{J}_d$ twice, we have

$$\frac{d^2\mathcal{J}_d}{dq^2} = \frac{d^2}{dq^2}\left(\frac{1}{2}\rho^*Y\rho\right) = \left(\frac{d\rho}{dq}\right)^* Y \left(\frac{d\rho}{dq}\right) + \left(\frac{d^2\rho}{dq^2}\right)^* Y\rho.$$

Because $\frac{d\rho}{dq} = G$, the linear operator associated with the first term is $G^*YG = H_d^{\text{gn}}$. The second term is linear in $Y^{1/2}\rho$. Thus

$$H_d = H_d^{\text{gn}} + O(\|\rho\|_Y). \tag{D.9}$$

Adding R to each side of (D.9) yields (5.9).

To prove (5.10), recall that solutions to the full-space problem must satisfy $0 = \frac{\partial\mathcal{L}}{\partial z}$. In particular,

$$0 = \left(\frac{\partial\mathcal{L}}{\partial u}\right)^* = B^*Y\rho + A^*\lambda.$$

Solving this equation for λ yields $\lambda = -A^{-*}B^*Y\rho$, which implies $\lambda = O(\|\rho\|_Y)$, which implies $\Theta = O(\|\rho\|_Y)$ and $\Xi = O(\|\rho\|_Y)$. From the formula for K in (5.5) and the formula for K^{gn} in (5.6), we see that K may be written as K^{gn} plus a block matrix containing only the operators Θ and Ξ, which implies (5.10). □

Appendix E

Additional Numerical Results

Although we developed our adaptive product-convolution scheme (Chapter 8) to approximate operators that arise in PDE-constrained inverse problems, the scheme is effective for approximating operators that arise in many other applications, including image deblurring and domain decomposition methods for numerically solving PDEs.[1] In this appendix we use our product-convolution scheme to approximate a spatially varying blur operator (Section E.1), and the non-local component of the Schur complement associated with restricting the Poisson operator to an internal interface (Section E.2). For the spatially varying blur operator, our scheme refines towards the boundary between blur kernels and refines almost nowhere else, therefore outperforming the standard non-adaptive scheme which refines everywhere uniformly. For the Poisson interface Schur complement, our scheme is mesh scalable: it requires roughly the same convolution rank (number of terms in (8.12)) to achieve a desired error tolerance regardless of how fine the mesh is. For the Poisson Schur complement, our scheme, in combination with H-matrix methods, can be used to build an excellent preconditioner.

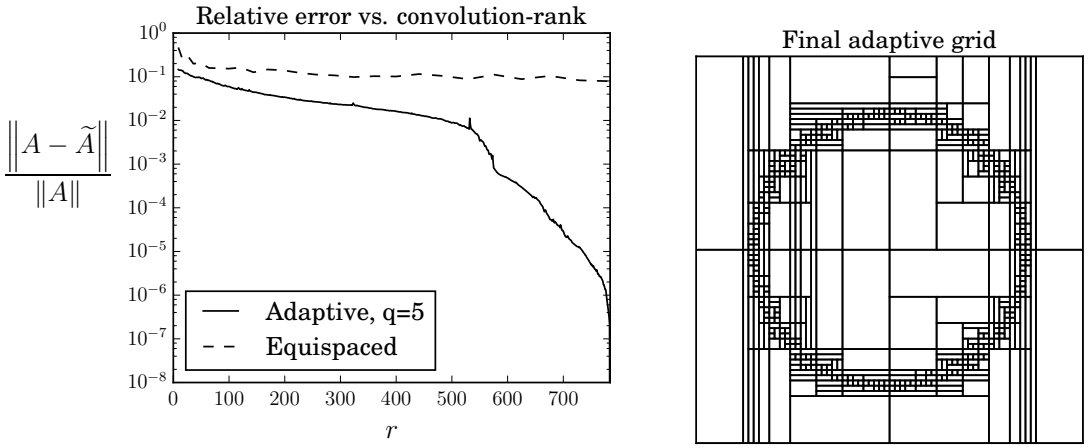
In this appendix, as in Chapter 8, 'A' denotes a generic locally translation-invariant operator to be approximated (this notation differs from the rest of the dissertation, in which A denotes the state operator).

E.1 Spatially varying blur

Problem setup Let a be the following spatially varying blurring kernel,

$$a(s,t) := \exp\left(-\frac{s^2+t^2}{2\sigma^2(s,t)}\right), \quad \text{where} \quad \sigma(s,t) = \begin{cases} 0.1, & s^2+t^2 < 0.5, \\ 0.2, & s^2+t^2 \geq 0.5. \end{cases}$$

[1] This appendix contains content from [7] (Nick Alger, Vishwas Rao, Aaron Myers, Tan Bui-Thanh, and Omar Ghattas. Scalable matrix-free adaptive product-convolution approximation for locally translation-invariant operators. arXiv preprint arXiv:1805.06018, 2018. Submitted.).



(a) (b)

Figure E.1: **Spatially varying blur:** Product-convolution approximation of the spatially varying blur operator defined in Section E.1. (a) Convergence of our adaptive scheme, compared to convergence of standard product-convolution approximation with an equispaced regular grid of sample points and local bilinear interpolant weighting functions. (b) Final grid generated by our adaptive scheme.

Here A is the matrix generated by sampling a on $[-1,1]^2$ with a 75×75 equally spaced regular grid.

Results Figure E.1a compares product-convolution approximation of A using our adaptive scheme, versus standard product-convolution approximation of A using an equally spaced regular grid of sample points, with bilinear interpolation of impulse response functions, no adaptivity and no boundary extension procedure. Our adaptive scheme converges much faster than the regular grid scheme.

Figure E.1b shows the final grid generated by our adaptive scheme, in which the boundary of the circle $s^2 + t^2 = 1$ is fully resolved with 2×2 cells. Error in the adaptive procedure is zero (within machine epsilon) for this final grid.

E.2 Poisson interface Schur complement

Problem setup Here we consider the discretized (negative) Laplace operator $K \approx -\Delta$ on the interior of the cube, $(-1,1)^3$. To build K, we discretize the Laplace operator on the whole cube, $[-1,1]^3$ with piecewise linear finite elements on a regular

$n \times n \times n$ mesh of tetrahedra, so that there are $(n+1)^3$ mesh gridpoints. Then we exclude rows and columns from the resulting matrix that correspond to boundary degrees of freedom. The resulting $(n-1)^3 \times (n-1)^3$ matrix, K, is the coefficient matrix for the linear system that would need to be solved to determine the solution on the interior degrees of freedom for the Poisson problem in the cube with Dirichlet boundary conditions.

Let 'i' denote the degrees of freedom on the interface hyperplane at $z = 0$ that separates[2] the degrees of freedom in the top half of the cube from the bottom half of the cube. Let 't' denote the degrees of freedom in the top half of the cube ($z > 0$), and let 'b' denote degrees of freedom in the bottom half of the cube ($z < 0$), not including the interface in both cases. Denote the associated blocks of K by K_{it}, K_{tt}, K_{ti}, K_{ib} and so forth. We use our adaptive product-convolution scheme to approximate the operator

$$A \coloneqq K_{it} K_{tt}^{-1} K_{ti} + K_{ib} K_{bb}^{-1} K_{bi}.$$

The matrix $-A$ is the non-local component of the Schur complement for degrees of freedom on the interface hyperplane, i.e., the matrix

$$S \coloneqq K_{ii} - K_{it} K_{tt}^{-1} K_{ti} - K_{ib} K_{bb}^{-1} K_{bi}.$$

Matrix entries of A are not directly available; we apply A to vectors by performing matrix-vector products with K_{bi}, K_{ib}, K_{ti}, and K_{it}, and solving linear systems with K_{tt} and K_{bb} as the coefficient matrices. After approximating A with $\widetilde{A}$ using our product-convolution scheme, we also construct the Schur complement approximation

$$\widetilde{S} \coloneqq K_{ii} - \widetilde{A}.$$

Such Schur complement approximations could be constructed recursively. One would subdivide the top and bottom subdomains, then subdivide the subdivisions, and so on. Approximations of Schur complements at deeper levels of the recursion would be used when constructing approximations at shallower levels. Here we only present results for one subdivision.

[2] We choose n even so that the interface is at $z = 0$, rather than being slightly offset.

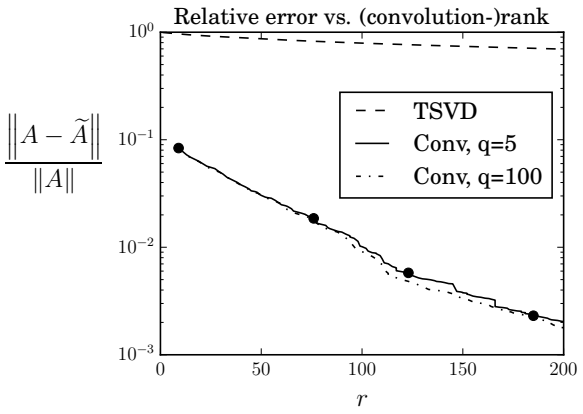


Figure E.2: **Poisson Schur complement:** Relative error in truncated SVD low-rank approximation ('TSVD') compared to our product-convolution approximation ('Conv') as the (convolution) rank, r, changes. We show convergence curves for our scheme using both $q = 5$ and $q = 100$ random samples for the a-posteriori error estimator. Black dots correspond to the adaptive grids visualized in Figure E.3.

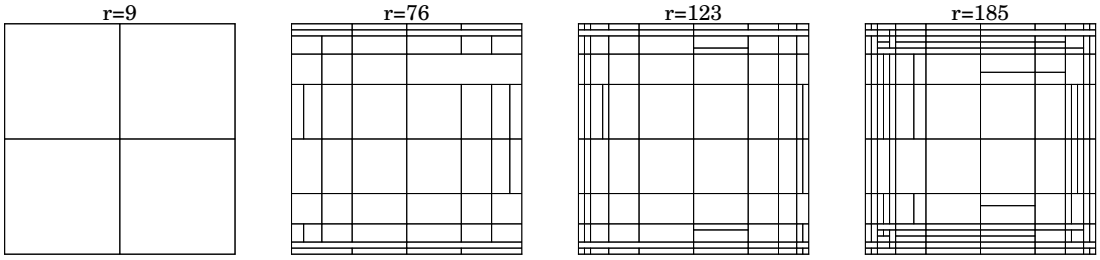


Figure E.3: **Poisson Schur complement:** Intermediate stages of adaptive grid refinement corresponding to black dots in Figure E.2.

Results Figure E.2 compares the convergence of our scheme to truncated SVD ('TSVD') approximation for $n = 40$ ($N = (n-1)^2 = 1521$). Since the Poisson Schur complement is high rank, TSVD performs poorly. In contrast, our scheme performs well: at $r = 200$ our scheme has less than 0.03% error, whereas TSVD has approximately 69% error. Figure E.2 also shows that our scheme performs well even when we use a small number of random samples for the a-posteriori error estimator: the convergence curve for $q = 5$ is almost identical to the convergence curve for $q = 100$. Figure E.3 displays the adaptive meshes from four different stages of the adaptive refinement process from Figure E.2. Our scheme adaptively refines towards the boundary, then the corners. This is expected since boundary effects are the only source of translation-invariance failure.

Figure E.4 compares our scheme to TSVD on a sequence of progressively finer meshes, from $h \approx 0.1$ to $h \approx 0.01$, where h is the distance between adjacent gridpoints in the mesh. The curves show the (convolution) rank, r, required to achieve a relative

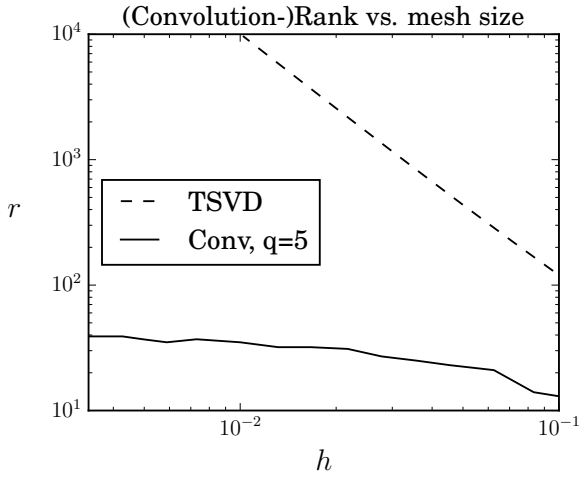


Figure E.4: **Poisson Schur complement:** The (convolution) rank, r, required to achieve a relative approximation error of 5%, for a variety of mesh sizes, h. 'TSVD' indicates truncated SVD low rank approximation, and 'Conv' indicates our product-convolution scheme.

error tolerance of 5%. The rank for TSVD grows with the number of degrees of freedom on the top surface ($r \sim O(1/h^2)$), offering little improvement over directly building a dense matrix representation of A column-by-column. In contrast, the convolution rank for our scheme remains small for all h considered.

Figure E.5 compares the time required to apply A to a vector, versus the time required to apply $\widetilde{A}$ to a vector. When applying A to vectors, we solve the necessary linear systems with K_{tt} and K_{bb} as coefficient operators using PyAMG's [159] rootnode algebraic multigrid. When applying $\widetilde{A}$ to vectors, we use the FFT, as discussed in Section 8.3.2. For large n, applying $\widetilde{A}$ to a vector is much cheaper than applying A to a vector.

In Table E.1 we compare the condition number of the Schur complement, S, with the condition numbers of the preconditioned Schur complement, $\widetilde{S}^{-1}S$, for $n \times n \times n$ meshes ranging from $n = 10$ to $n = 100$. Here $\widetilde{S}^{-1}$ is constructed by converting $\widetilde{S}$ to H-matrix format, then inverting it using H-matrix arithmetic. For H-matrices, we use the standard coordinate splitting nested-bisection binary cluster tree,[3] and the standard diameter-less-than-distance admissibilty condition[4]. Here, we use a tolerance of

[3]Degrees of freedom are split into two equally-sized clusters by a hyperplane normal to widest coordinate direction for that cluster. Then each cluster is split into two smaller clusters in the same way, and so on, recursively. The splitting continues until the number of degrees of freedom in a cluster is less than 32.

[4]We mark a block of the matrix as low rank (admissible) if the distance between the degree of freedom cluster associated with the rows of the block and the diameter of the degree of freedom cluster associated with the columns of the block is less than or equal to the diameter of the smaller of the two degree of freedom clusters.

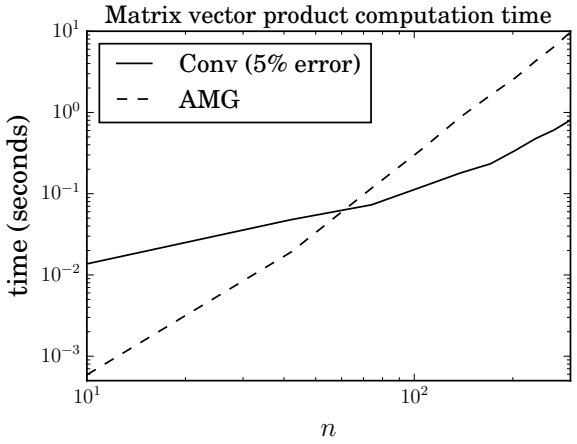


Figure E.5: **Poisson Schur complement:** The time required to apply $\widetilde{A}$ to a vector using the FFT to compute the convolutions ('Conv'), compared to the time required to apply A to a vector, using multigrid to apply the matrices K_{tt}^{-1} and K_{bb}^{-1} to vectors ('AMG'). For our product-convolution scheme, the average slope between $n = 171$ and $n = 300$ (from the final upturn to the end, containing 5 equally spaced n) is 2.2, suggesting an asymptotic cost of $O(n^{2.2})$ (theory predicts $O(n^2 \log n)$). For algebraic multigrid, the average slope between $n = 171$ and $n = 300$ is 3.2, suggesting an asymptotic cost of $O(n^{3.2})$ (theory predicts $O(n^3)$).

10^{-6} for the low-rank approximations performed during H-matrix construction and arithmetic. The condition number of the (unpreconditioned) Schur complement grows as $O(1/h)$, where $h \approx 1/n$ is the mesh size. In contrast, the preconditioned Schur complement remains extremely well conditioned: the largest value of cond $\left(\widetilde{S}^{-1}S\right)$ is 1.9 for all meshes considered.

n	$\mathrm{cond}\,(S)$	$\mathrm{cond}\left(\widetilde{S}^{-1}S\right)$	r
10	10.3	1.1	9
20	21.3	1.2	20
30	32.2	1.3	27
40	43.0	1.4	28
50	53.8	1.5	31
60	64.5	1.5	33
70	75.3	1.8	32
80	86.1	1.8	35
90	96.9	1.8	35
100	107.7	1.9	35

Table E.1: **Poisson Schur complement:** Comparison of condition numbers for the Poisson interface Schur complement for a range of $n \times n \times n$ meshes. S is the unpreconditioned Schur complement. $\widetilde{S}$ is the approximate Schur complement generated by replacing the nonlocal terms, A, within the Schur complement, with our convolution aproximation, $\widetilde{A}$, with a 5% relative error tolerance. The last column shows r, the convolution-rank of $\widetilde{A}$.

Bibliography

[1] Santi S. Adavani and George Biros. Multigrid algorithms for inverse problems with linear parabolic PDE constraints. *SIAM Journal on Scientific Computing*, 31:369–397, 2008.

[2] Santi S. Adavani and George Biros. Fast algorithms for source identification problems with elliptic PDE constraints. *SIAM Journal on Imaging Sciences*, 3:791–808, 2010.

[3] Hans-Martin Adorf. Towards HST restoration with a space-variant PSF, cosmic rays and other missing data. In *The Restoration of HST Images and Spectra-II*, page 72, 1994.

[4] Volkan Akçelik, George Biros, Andrei Draganescu, Omar Ghattas, Judith Hill, and Bart van Bloeman Waanders. Dynamic data-driven inversion for terascale simulations: Real-time identification of airborne contaminants. In *Proceedings of SC2005*, Seattle, 2005.

[5] Volkan Akçelik, George Biros, Omar Ghattas, Judith Hill, David Keyes, and Bart van Bloeman Waanders. Parallel PDE-constrained optimization. In M. Heroux, P. Raghaven, and H. Simon, editors, *Parallel Processing for Scientific Computing*. SIAM, 2006.

[6] Nick Alger. Relating condition number of Hessian to the rate of convergence. Mathematics Stack Exchange, 2018. https://math.stackexchange.com/q/2847190.

[7] Nick Alger, Vishwas Rao, Aaron Myers, Tan Bui-Thanh, and Omar Ghattas. Scalable matrix-free adaptive product-convolution approximation for locally translation-invariant operators. *arXiv preprint arXiv:1805.06018*, 2018. Submitted.

[8] Nick Alger, Umberto Villa, Tan Bui-Thanh, and Omar Ghattas. A data scalable augmented Lagrangian KKT preconditioner for large-scale inverse problems. *SIAM Journal on Scientific Computing*, 39(5):A2365–A2393, 2017.

[9] Jeffrey L. Anderson. An adaptive covariance inflation error correction algorithm for ensemble filters. *Tellus A: Dynamic Meteorology and Oceanography*, 59(2):210–224, 2007.

[10] Eyal Arian. *Multigrid methods for optimal shape design governed by elliptic systems*. PhD thesis, The Weizmann Institute of Science, 1994.

[11] Eyal Arian. Analysis of the Hessian for aeroelastic optimization. Technical report, ICASE, 1995.

[12] Eyal Arian and Angelo Iollo. Analytic Hessian derivation for the quasi-one-dimensional Euler equations. *Journal of Computational Physics*, 2009.

[13] Eyal Arian and Shlomo Ta'asan. Smoothers for optimization problems. In *Seventh Copper Mountain Conference on Multigrid Methods*, pages 15–30. Citeseer, 1996.

[14] Eyal Arian and Shlomo Ta'asan. Analysis of the Hessian for aerodynamic optimization: inviscid flow. *Computers and Fluids*, 28(7):853, 1999.

[15] Eyal Arian and Veer N. Vatsa. A preconditioning method for shape optimization governed by the Euler equations. *International Journal of Computational Fluid Dynamics*, 12(1):17–27, 1999.

[16] Uri M. Ascher and Eldad Haber. A multigrid method for distributed parameter estimation problems. *Electronic Transactions on Numerical Analysis*, 15:1–17 (electronic), 2003. Tenth Copper Mountain Conference on Multigrid Methods (Copper Mountain, CO, 2001).

[17] Richard C. Aster, Brian Borchers, and Clifford H. Thurber. *Parameter estimation and inverse problems*. Academic Press, 2013.

[18] Haim Avron and Sivan Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM (JACM)*, 58(2):8, 2011.

[19] Owe Axelsson and Janos Karatson. Mesh independent superlinear PCG rates via compact-equivalent operators. *SIAM Journal on Numerical Analysis*, 45(4):1495–1516, 2007.

[20] Ivo Babuska and Jens M. Melenk. The partition of unity method. In *International Journal of Numerical Methods in Engineering*. Citeseer, 1996.

[21] Randolph E. Bank and Todd Dupont. An optimal order process for solving finite element equations. *Mathematics of Computation*, 36(153):35–51, 1981.

[22] Randolph E. Bank, Bruno D. Welfert, and Harry Yserentant. A class of iterative methods for solving saddle point problems. *Numerische Mathematik*, 56:645–666, 1990.

[23] Gang Bao and William W. Symes. Computation of pseudo-differential operators. *SIAM Journal on Scientific Computing*, 17(2):416–429, 1996.

[24] Johnathan M. Bardsley, Stuart Jefferies, James Nagy, and Robert Plemmons. A computational method for the restoration of images with an unknown, spatially-varying blur. *Optics Express*, 14(5):1767–1782, 2006.

[25] Jonathan M. Bardsley, Antti Solonen, Heikki Haario, and Marko Laine. Randomize-then-optimize: a method for sampling from posterior distributions in nonlinear inverse problems.

[26] Andrew T. Barker, Tyrone Rees, and Martin Stoll. A fast solver for an H_1 regularized PDE-constrained optimization problem. *Communications in Computational Physics*, 19:143–167, 2016.

[27] Frank Bauer and Mark A. Lukas. Comparing parameter choice methods for regularization of ill-posed problems. *Mathematics and Computers in Simulation*, 81(9):1795–1841, 2011.

[28] Frank Bauer and Peter Mathé. Parameter choice methods using minimization schemes. *Journal of Complexity*, 27(1):68–85, 2011.

[29] Mario Bebendorf. Approximation of boundary element matrices. *Numerische Mathematik*, 86(4):565–589, 2000.

[30] Nathan Bell, Luke N. Olson, and Jacob B. Schroder. PyAMG: Algebraic multigrid solvers in Python, 2013. Version 2.1.

[31] Stefania Bellavia, Jacek Gondzio, and Benedetta Morini. A matrix-free preconditioner for sparse symmetric positive definite systems and least-squares problems. *SIAM Journal on Scientific Computing*, 35(1):A192–A211, 2013.

[32] Michele Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, 182(2):418–477, 2002.

[33] Michele Benzi, Jane K. Cullum, and Miroslav Tuma. Robust approximate inverse preconditioning for the conjugate gradient method. *SIAM Journal on Scientific Computing*, 22(4):1318–1332, 2000.

[34] Michele Benzi, Gene H. Golub, and Jörg Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[35] Michele Benzi, Eldad Haber, and Lauren Taralli. A preconditioning technique for a class of PDE-constrained optimization problems. *Advances in Computational Mathematics*, 35(2):149–173, 2011.

[36] Michele Benzi and Miroslav Tuma. A comparative study of sparse approximate inverse preconditioners. *Applied Numerical Mathematics*, 30(2):305–340, 1999.

[37] Jean-Pierre Berenger. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics*, 114(2):185–200, 1994.

[38] Jérémie Bigot, Paul Escande, and Pierre Weiss. Estimation of linear operators from scattered impulse responses. *arXiv preprint arXiv:1610.04056*, 2016.

[39] George Biros and Omar Ghattas. Parallel Lagrange–Newton–Krylov–Schur methods for PDE–constrained optimization. Part I: The Krylov–Schur solver. *SIAM Journal on Scientific Computing*, 27(2):687–713, 2005.

[40] George Biros and Omar Ghattas. Parallel Lagrange–Newton–Krylov–Schur methods for PDE–constrained optimization. Part II: The Lagrange–Newton solver and its application to optimal control of steady viscous flows. *SIAM Journal on Scientific Computing*, 27(2):714–739, 2005.

[41] Joseph E. Bishop. A displacement-based finite element formulation for general polyhedra using harmonic shape functions. *International Journal for Numerical Methods in Engineering*, 97(1):1–31, 2014.

[42] Steffen Börm. *Efficient numerical methods for non-local operators: H2-matrix compression, algorithms and analysis*, volume 14. European Mathematical Society, 2010.

[43] Steffen Börm and Lars Grasedyck. Hybrid cross approximation of integral operators. *Numerische Mathematik*, 101(2):221–249, 2005.

[44] Alfio Borzi. Multigrid methods for parabolic distributed optimal control problems. *Journal Of Computational And Applied Mathematics*, 157(2):365–382, 2003.

[45] Alfio Borzi and Roland Griesse. Experiences with a space-time multigrid method for the optimal control of a chemical turbulence model. *International Journal For Numerical Methods In Fluids*, 47(8-9):879–885, 2005.

[46] Alfio Borzı and Volker Schulz. Multigrid methods for PDE optimization. *SIAM Review*, 51(2):361–395, 2009.

[47] Wajih Boukaram, George Turkiyyah, and David Keyes. Randomized GPU algorithms for the construction of hierarchical matrices from mat-vec operations. Personal communication, July 2018.

[48] Dietrich Braess and Wolfgang Hackbusch. A new convergence proof for the multigrid method including the V-cycle. *SIAM Journal on Numerical Analysis*, 20(5):967–975, 1983.

[49] Franco Brezzi and Michel Fortin. *Mixed and hybrid finite element methods*, volume 15 of *Springer Series in Computational Mathematics*. Springer-Verlag, New York, 1991.

[50] Alexander N. Brooks and Thomas J. R. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 32(1-3):199–259, 1982.

[51] Tan Bui-Thanh, Carsten Burstedde, Omar Ghattas, James Martin, Georg Stadler, and Lucas C. Wilcox. Extreme-scale UQ for Bayesian inverse problems governed by PDEs. In *SC12: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2012.

[52] Tan Bui-Thanh and Omar Ghattas. Analysis of the Hessian for inverse scattering problems. Part I: Inverse shape scattering of acoustic waves. *Inverse Problems*, 28(5):055001, 2012.

[53] Tan Bui-Thanh and Omar Ghattas. Analysis of the Hessian for inverse scattering problems. Part II: Inverse medium scattering of acoustic waves. *Inverse Problems*, 28(5):055002, 2012.

[54] Tan Bui-Thanh and Omar Ghattas. Analysis of the Hessian for inverse scattering problems. Part III: Inverse medium scattering of electromagnetic waves. *Inverse Problems and Imaging*, 7(4):1139–1155, 2013.

[55] Tan Bui-Thanh, Omar Ghattas, James Martin, and Georg 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.

[56] Tan Bui-Thanh and Mark Girolami. Solving large-scale PDE-constrained Bayesian inverse problems with Riemann manifold Hamiltonian Monte Carlo. *Inverse Problems*, 30(11):114014, 2014.

[57] Troy Butler, Donald Estep, and Jeff Sandelin. A computational measure theoretic approach to inverse sensitivity problems II: A posteriori error analysis. *SIAM Journal on Numerical Analysis*, 50(1):22–45, 2012.

[58] Daniela Calvetti, Bryan Lewis, and Lothar Reichel. Restoration of images with spatially variant blur by the GMRES method. In *Advanced Signal Processing Algorithms, Architectures, and Implementations X, ed. FT Luk, Proceedings of the Society of Photo-Optical Instrumentation Engineers (SPIE)*, volume 4116, pages 364–374, 2000.

[59] Stéphanie Chaillat and George Biros. FaIMS: A fast algorithm for the inverse medium problem with multiple frequencies and multiple sources for the scalar Helmholtz equation. *Journal of Computational Physics*, 231(12):4403–4421, 2012.

[60] Raymond H. Chan and Michael K. Ng. Conjugate gradient methods for Toeplitz systems. *SIAM Review*, 38(3):427–482, 1996.

[61] Shivkumar Chandrasekaran, Ming Gu, and William Lyons. A fast adaptive solver for hierarchically semiseparable representations. *Calcolo*, 42(3):171–185, 2005.

[62] Pei Chen. Hessian matrix vs. Gauss–Newton Hessian matrix. *SIAM Journal on Numerical Analysis*, 49(4):1417–1435, 2011.

[63] Peng Chen and Christoph Schwab. Sparse-grid, reduced-basis Bayesian inversion. *Computer Methods in Applied Mechanics and Engineering*, 297:84–115, 2015.

[64] Peng Chen, Umberto Villa, and Omar Ghattas. Taylor approximation and variance reduction for PDE-constrained optimal control under uncertainty. *Preprint*, 2018.

[65] Hongwei Cheng, Zydrunas Gimbutas, Per-Gunnar Martinsson, and Vladimir Rokhlin. On the compression of low rank matrices. *SIAM Journal on Scientific Computing*, 26(4):1389–1404, 2005.

[66] Jiawei Chiu and Laurent Demanet. Matrix probing and its conditioning. *SIAM Journal on Numerical Analysis*, 50(1):171–193, 2012.

[67] Y. Choi. *Simultaneous analysis and design in PDE-constrained optimization*. PhD thesis, Stanford, 2012.

[68] Steven C. Constable, Robert L. Parker, and Catherine G. Constable. Occam's inversion: a practical algorithm for generating smooth models from electromagnetic sounding data. *Geophysics*, 52(3):289–300, 1987.

[69] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012. Chapter 8.

[70] Tiangang Cui, Kody Law, and Youssef M. Marzouk. Dimension-independent likelihood-informed MCMC. *Journal of Computational Physics*, 304:109–137, 2016.

[71] Tiangang Cui, James Martin, Youssef M. Marzouk, Antti Solonen, and Alessio Spantini. Likelihood-informed dimension reduction for nonlinear inverse problems. *Inverse Problems*, 30(11):114015, 2014.

[72] Timothy A. Davis. *Direct methods for sparse linear systems*, volume 2. SIAM, 2006.

[73] Valentia De Simone and Daniela di Serafino. A matrix-free approach to build band preconditioners for large-scale bound-constrained optimization. *Journal of Computational and Applied Mathematics*, 268:82–92, 2014.

[74] Laurent Demanet, Pierre-David Létourneau, Nicolas Boumal, Henri Calandra, Jiawei Chiu, and Stanley Snelson. Matrix probing: a randomized preconditioner for the wave-equation Hessian. *Applied and Computational Harmonic Analysis*, 32(2):155–168, 2012.

[75] Ron S. Dembo, Stanley C. Eisenstat, and Trond Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400–408, 1982.

[76] Leszek Demkowicz. "Babuška ⇔ Brezzi?". Technical Report 06-08, Institute for Computational Engineering and Sciences, the University of Texas at Austin, April 2006.

[77] Loïc Denis, Eric Thiébaut, Ferréol Soulez, Jean-Marie Becker, and Rahul Mourya. Fast approximations of shift-variant blur. *International Journal of Computer Vision*, 115(3):253–278, 2015.

[78] Gianluca Detommaso, Tiangang Cui, Youssef Marzouk, Robert Scheichl, and Alessio Spantini. A Stein variational Newton method. *arXiv preprint arXiv:1806.03085*, 2018.

[79] Andrei Druaguanescu and Todd Dupont. Optimal order multilevel preconditioners for regularized ill-posed problems. *Mathematics of Computation*, 2008.

[80] Andrei Druaguanescu and Ana Maria Soane. Multigrid solution of a distributed optimal control problem constrained by the Stokes equations. *Applied Mathematics and Computation*, 2013.

[81] Jurjen Duintjer Tebbens and Miroslav Tuma. Preconditioner updates for solving sequences of linear systems in matrix-free environment. *Numerical Linear Algebra with Applications*, 17(6):997–1019, 2010.

[82] Howard C. Elman, David J. Silvester, and Andrew J. Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics.* Oxford University Press, 2014.

[83] Heinz W. Engl, Martin Hanke, and Andreas Neubauer. *Regularization of Inverse Problems*. Springer Netherlands, 1996.

[84] Björn Engquist and Lexing Ying. Sweeping preconditioner for the helmholtz equation: moving perfectly matched layers. *Multiscale Modeling & Simulation*, 9(2):686–710, 2011.

[85] Ioannis Epanomeritakis, Volkan Akçelik, Omar Ghattas, and Jacobo Bielak. A Newton-CG method for large-scale three-dimensional elastic full-waveform seismic inversion. *Inverse Problems*, 24(3):034015, 2008.

[86] Paul Escande and Pierre Weiss. Sparse wavelet representations of spatially varying blurring operators. *SIAM Journal on Imaging Sciences*, 8(4):2976–3014, 2015.

[87] Paul Escande and Pierre Weiss. Approximation of integral operators using convolution-product expansions. *Journal of Mathematical Imaging and Vision*, 58(3):333–348, 2017.

[88] Paul Escande, Pierre Weiss, and François Malgouyres. Spatially varying blur recovery. Diagonal approximations in the wavelet domain. 2012.

[89] Geir Evensen. *Data assimilation: the ensemble Kalman filter*. Springer Science & Business Media, 2009.

[90] Giovanni Fasano and Massimo Roma. On the iterative computation of a l2-norm scaling based preconditioner.

[91] Giovanni Fasano and Massimo Roma. Preconditioning large indefinite linear systems, 2012.

[92] Giovanni Fasano and Massimo Roma. Preconditioning Newton–Krylov methods in nonconvex large scale optimization. *Computational Optimization and Applications*, 56(2):253–290, 2013.

[93] Massimiliano Ferronato. Preconditioning for sparse linear systems at the dawn of the 21st century: History, current developments, and future perspectives. *IRSN Applied Mathematics*, 2012.

[94] D. A. Fish, Jan Grochmalicki, and E. R. Pike. Scanning singular-value-decomposition method for restoration of images with space-variant blur. *JOSA A*, 13(3):464–469, 1996.

[95] Pearl H. Flath. *Hessian-based response surface approximations for uncertainty quantification in large-scale statistical inverse problems, with applications to groundwater flow*. PhD thesis, The University of Texas at Austin, 2013.

[96] Pearl H. Flath, Lucas C. Wilcox, Volkan Akçelik, Judy Hill, Bart van Bloemen Waanders, and Omar 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.

[97] Ralf C. Flicker and François J. Rigaut. Anisoplanatic deconvolution of adaptive optics images. *JOSA A*, 22(3):504–513, 2005.

[98] Zenon Fortuna. Some convergence properties of the conjugate gradient method in Hilbert space. *SIAM Journal on Numerical Analysis*, 16:380–394, 1979.

[99] Haohuan Fu, Junfeng Liao, Jinzhe Yang, Lanning Wang, Zhenya Song, Xiaomeng Huang, Chao Yang, Wei Xue, Fangfang Liu, Fangli Qiao, et al. The Sunway TaihuLight supercomputer: system and applications. *Science China Information Sciences*, 59(7):072001, 2016.

[100] Marc Gentile, Frederic Courbin, and Georges Meylan. Interpolating point spread function anisotropy. *Astronomy & Astrophysics*, 549, 2013.

[101] Roger G. Ghanem and Pol D. Spanos. *Stochastic Finite Elements: A Spectral Approach*. Dover publications, 2003.

[102] Amir Gholami, Andreas Mang, and George Biros. An inverse problem formulation for parameter estimation of a reaction–diffusion model of low grade gliomas. *Journal of Mathematical Biology*, 2016.

[103] Erez Gilad and Jost Von Hardenberg. A fast algorithm for convolution integrals with space and time variant kernels. *Journal of Computational Physics*, 216(1):326–336, 2006.

[104] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, 1981.

[105] Adrianna Gillman, Alex H. Barnett, and Per-Gunnar Martinsson. A spectrally accurate direct solution technique for frequency-domain scattering problems with variable media. *BIT Numerical Mathematics*, 55(1):141–170, 2015.

[106] Gene H. Golub, Chen Greif, and James M. Varah. An algebraic analysis of a block diagonal preconditioner for saddle point systems. *SIAM Journal on Matrix Analysis and Applications*, 27(3):779–792, 2006.

[107] Max D. Gunzburger. *Perspectives in Flow Control and Optimization*. SIAM, Philadelphia, 2003.

[108] Stefan Güttel. *Rational Krylov methods for operator functions*. PhD thesis, 2010.

[109] Stefan Güttel. Rational Krylov approximation of matrix functions: Numerical methods and optimal pole selection. *GAMM-Mitteilungen*, 36(1):8–31, 2013.

[110] Heikki Haario, Marko Laine, Antonietta Mira, and Eero Saksman. Dram: efficient adaptive mcmc. *Statistics and computing*, 16(4):339–354, 2006.

[111] Eldad Haber and Uri M. Ascher. Preconditioned all-at-once methods for large, sparse parameter estimation problems. *Inverse Problems*, 17(6):1847–1864, 2001.

[112] Wolfgang Hackbusch. A sparse matrix arithmetic based on H-matrices. Part I: Introduction to H-matrices. *Computing*, 62(2):89–108, 1999.

[113] Wolfgang Hackbusch. *Hierarchical Matrices: Algorithms and Analysis*, volume 49. Springer, 2015.

[114] Wolfgang Hackbusch, Boris Khoromskij, and Stefan A. Sauter. On H^2-matrices. In *Lectures on Applied Mathematics: Proceedings of the Symposium Organized by the Sonderforschungsbereich 438 on the Occasion of Karl-Heinz Hoffmann's 60th Birthday, Munich, June 30–July 1, 1999*, page 9. Springer Science & Business Media, 2000.

[115] Saima Ben Hadj, Laure Blanc-Féraud, Gilles Aubert, and Gilbert Engler. Blind restoration of confocal microscopy images in presence of a depth-variant blur and Poisson noise. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 915–919. IEEE, 2013.

[116] William W. Hager and Hongchao Zhang. A survey of nonlinear conjugate gradient methods. *Pacific Journal of Optimization*, 2(1):35–58, 2006.

[117] Nicholas Hale, Nicholas J. Higham, and Lloyd N. Trefethen. Computing a^{α},$log(a)$, and related matrix functions by contour integrals. *SIAM Journal on Numerical Analysis*, 46(5):2505–2523, 2008.

[118] Nathan Halko, Per Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

[119] Per Christian Hansen. The L-curve and its use in the numerical treatment of inverse problems. In *Computational Inverse Problems in Electrocardiology, ed. P. Johnston, Advances in Computational Bioengineering*, pages 119–142. WIT Press, 2000.

[120] Felix J. Herrmann, Peyman Moghaddam, and Christiaan C. Stolk. Sparsity- and continuity-promoting seismic image recovery with curvelet frames. *Applied and computational harmonic analysis*, 2008.

[121] Roland Herzog and Ekkehard Sachs. Preconditioned conjugate gradient method for optimal control problems with control and state constraints. *SIAM Journal on Matrix Analysis and Applications*, 31:2291–2317, 2010.

[122] Roland Herzog and Ekkehard Sachs. Superlinear convergence of Krylov subspace methods for self-adjoint problems in Hilbert space. *SIAM Journal on Numerical Analysis*, 53:1304–1324, 2015.

[123] Michael Hirsch, Suvrit Sra, Bernhard Schölkopf, and Stefan Harmeling. Efficient filter flow for space-variant multiframe blind deconvolution. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 607–614. IEEE, 2010.

[124] Kenneth L. Ho and Lexing Ying. Hierarchical interpolative factorization for elliptic operators: differential equations. *Communications in Pure and Applied Mathematics*, 69(8):1415–1451, 2015.

[125] Tomasz Hrycak, Saptarshi Das, Gerald Matz, and Hans G. Feichtinger. Low complexity equalization for doubly selective channels modeled by a basis expansion. *IEEE Transactions on Signal Processing*, 58(11):5706–5719, 2010.

[126] Tobin Isaac, Noemi Petra, Georg Stadler, and Omar Ghattas. Scalable and efficient algorithms for the propagation of uncertainty from data through inference to prediction for large-scale problems, with application to flow of the Antarctic ice sheet. *Journal of Computational Physics*, 296:348–368, September 2015.

[127] Jim E. Jones and Panayot S. Vassilevski. AMGe based on element agglomeration. *SIAM Journal on Scientific Computing*, 23(1):109–133, 2001.

[128] Wolfgang Krendl, Valeria Simoncini, and Walter Zulehner. Stability estimates and structural spectral properties of saddle point problems. *Numerische Mathematik*, 124(1):183–213, 2013.

[129] Patrick Le Tallec and Abani Patra. Non-overlapping domain decomposition methods for adaptive hp approximations of the stokes problem with discontinuous pressure fields. *Computer Methods in Applied Mechanics and Engineering*, 145(3-4):361–379, 1997.

[130] Lin Lin, Jianfeng Lu, and Lexing Ying. Fast construction of hierarchical matrix representation from matrix–vector multiplication. *Journal of Computational Physics*, 230(10):4071–4087, 2011.

[131] Finn Lindgren, Håvard Rue, and Johan Lindström. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(4):423–498, 2011.

[132] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS methods for large scale optimization. *Math. Prog.*, 45:503–528, 1989.

[133] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. In *Advances In Neural Information Processing Systems*, pages 2378–2386, 2016.

[134] Anders Logg, Kent-Andre Mardal, and Garth Wells. *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS book*, volume 84. Springer Science & Business Media, 2012.

[135] Ladislav Lukšan, Ctirad Matonoha, and Jan Vlcek. Band preconditioners for the matrix free truncated Newton method. Technical report, Institute of Computer Science, Academy of Sciences of the Czech Republic, 2010.

[136] Ladislav Lukšan and Jan Vlček. Efficient tridiagonal preconditioner for the matrix-free truncated Newton method. *Applied Mathematics and Computation*, 235:394–407, 2014.

[137] Michael W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

[138] Kent-Andre Mardal, Bjørn Fredrik Nielsen, and Magne Nordaas. Robust preconditioners for PDE-constrained optimization with limited observations. *BIT Numerical Mathematics*, 57(2):405–431, 2017.

[139] Kent-Andre Mardal and Ragnar Winther. Preconditioning discretizations of systems of partial differential equations. *Numerical Linear Algebra with Applications*, 18(1):1–40, 2010.

[140] James Martin, Lucas C. Wilcox, Carsten Burstedde, and Omar Ghattas. A stochastic Newton MCMC method for large-scale statistical inverse problems with application to seismic inversion. *SIAM Journal on Scientific Computing*, 34(3):A1460–A1487, 2012.

[141] Per-Gunnar Martinsson. A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1251–1274, 2011.

[142] Per-Gunnar Martinsson. Compressing rank-structured matrices via randomized sampling. *SIAM Journal on Scientific Computing*, 38(4):A1959–A1986, 2016.

[143] Per-Gunnar Martinsson and Vladimir Rokhlin. A fast direct solver for boundary integral equations in two dimensions. *Journal of Computational Physics*, 205(1):1–23, 2005.

[144] James A. Mercer, Nick Alger, Bruce M. Howe, Rex K. Andrew, and John A. Colosi. Moving ship thermometry. *The Journal of the Acoustical Society of America*, 120(5):3061–3061, 2006.

[145] José Luis Morales and Jorge Nocedal. Automatic preconditioning by limited memory quasi-Newton updating. *SIAM Journal on Optimization*, 10(4):1079–1096, 2000.

[146] José Luis Morales and Jorge Nocedal. Enriched methods for large-scale unconstrained optimization. *Computational Optimization and Applications*, 21(2):143–154, 2002.

[147] Vladimir Alekseevich Morozov. *Methods for solving incorrectly posed problems*. Springer, 1984.

[148] Malcolm F. Murphy, Gene H. Golub, and Andrew J. Wathen. A note on preconditioning for indefinite linear systems. *SIAM Journal on Scientific Computing*, 22:1969–1972, 2000.

[149] James G. Nagy and Dianne P. O'Leary. Restoring images degraded by spatially variant blur. *SIAM Journal on Scientific Computing*, 19(4):1063–1082, 1998.

[150] James G. Nagy, Katrina Palmer, and Lisa Perrone. Iterative methods for image deblurring: a Matlab object-oriented approach. *Numerical Algorithms*, 36(1):73–93, 2004.

[151] Larry Nazareth. A relationship between the BFGS and conjugate gradient algorithms and its implications for new algorithms. *SIAM Journal on Numerical Analysis*, 16(5):794–800, 1979.

[152] Yurii Nesterov. A method of solving a convex programming problem with convergence rate o(1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[153] James Ng, Richard Prager, Nick Kingsbury, Graham Treece, and Andrew Gee. Wavelet restoration of medical pulse-echo ultrasound images in an EM framework. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 54(3), 2007.

[154] Bjørn Fredrik Nielsen and Kent-Andre Mardal. Efficient preconditioners for optimality systems arising in connection with inverse problems. *SIAM Journal on Control and Optimization*, 48(8):5143–5177, 2010.

[155] Bjørn Fredrik Nielsen and Kent-Andre Mardal. Analysis of the minimum residual method applied to ill posed optimality systems. *SIAM Journal on Scientific Computing*, 35(2):A785–A814, 2012.

[156] J.A. Nitsche. Über ein variationsprinzip zur lösung Dirichlet-problem bei verwendung von teilräumen, die kienen randbedingungen unteworfen sind,. *Abh. Math. Sem. Univ. Hamburg*, 36:9–15, 1971.

[157] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Verlag, Berlin, Heidelberg, New York, second edition, 2006.

[158] Dean S. Oliver, Albert C. Reynolds, and Ning Liu. *Inverse Theory for Petroleum Reservoir Characterization and History Matching*. Cambidge University Press, 2008.

[159] L. N. Olson and J. B. Schroder. PyAMG: Algebraic multigrid solvers in Python v4.0, 2018. Release 4.0.

[160] Luke Olson, Jacob Schroder, and Ray. TuminaTuminaro. A general interpolation strategy for algebraic multigrid using energy minimization. *SIAM Journal on Scientific Computing*, 33(2):966–991, 2011.

[161] Chris C. Paige and Michael A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

[162] John W. Pearson, Martin Stoll, and Andrew J. Wathen. Regularization-robust preconditioners for time-dependent PDE-constrained optimization problems. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1126–1152, 2012.

[163] John W. Pearson and Andrew J. Wathen. A new approximation of the Schur complement in preconditioners for PDE-constrained optimization. *Numerical Linear Algebra with Applications*, 19(5):816–829, 2012.

[164] Noemi Petra, James Martin, Georg Stadler, and Omar Ghattas. A computational framework for infinite-dimensional Bayesian inverse problems, Part II: Stochastic Newton MCMC with application to ice sheet flow inverse problems. *SIAM Journal on Scientific Computing*, 36(4):A1525–A1555, 2014.

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

[166] Chrysanthe Preza and José-Angel Conchello. Depth-variant maximum-likelihood restoration for three-dimensional fluorescence microscopy. *JOSA A*, 21(9):1593–1601, 2004.

[167] Tyrone Rees, Sue H. Dollar, and Andrew J. Wathen. Optimal solvers for PDE-constrained optimization. *SIAM Journal on Scientific Computing*, 32(1):271–298, 2010.

[168] Tyrone Rees, Martin Stoll, and Andrew J. Wathen. All-at-once preconditioning in PDE-constrained optimization. *Kybernetika*, 46(2):341–360, 2010.

[169] Tyrone Rees and Andrew J. Wathen. Preconditioning iterative methods for the optimal control of the Stokes equations. *SIAM Journal on Scientific Computing*, 33(5):2903–2926, 2011.

[170] Adam Rogers and Jason D. Fiege. Strong gravitational lens modeling with spatially variant point-spread functions. *The Astrophysical Journal*, 743(1):68, 2011.

[171] Massimo Roma. Dynamic scaling based preconditioning for truncated Newton methods in large scale unconstrained optimization. *Optimization Methods and Software*, 20(6):693–713, 2005.

[172] Youcef Saad and Martin H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[173] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, first edition, 2000.

[174] Yousef Saad and Maria Sosonkina. Distributed schur complement techniques for general sparse linear systems. *SIAM Journal on Scientific Computing*, 21(4):1337–1356, 1999.

[175] Anton Schiela and Stefan Ulbrich. Operator preconditioning for a class of inequality constrained optimal control problems. *SIAM Journal on Optimization*, 24(1):435–466, 2014.

[176] Joachim Schöberl and Walter Zulehner. Symmetric indefinite preconditioners for saddle point problems with applications to PDE-constrained optimization problems. *SIAM Journal on Matrix Analysis and Applications*, 29(3):752–773, 2007.

[177] Valeria Simoncini. Reduced order solution of structured linear systems arising in certain PDE-constrained optimization problems. *Computational Optimization and Applications*, 53(2):591–617, 2012.

[178] Alessio Spantini, Antti Solonen, Tiangang Cui, James Martin, Luis Tenorio, and Youssef Marzouk. Optimal low-rank approximations of Bayesian linear inverse problems. *SIAM Journal on Scientific Computing*, 37(6):A2451–A2487, 2015.

[179] Christiaan C Stolk. An improved sweeping domain decomposition preconditioner for the Helmholtz equation. *Advances in Computational Mathematics*, 43(1):45–76, 2017.

[180] Martin Stoll and Andrew J. Wathen. All-at-once solution of time-dependent PDE-constrained optimization problems. Technical report, 2010.

[181] Andrew M. Stuart. Inverse problems: a Bayesian perspective. *Acta Numerica*, 19(1):451–559, 2010.

[182] Stefan Takacs and Walter Zulehner. Convergence analysis of multigrid methods with collective point smoothers for optimal control problems. *Computing and Visualization in Science*, 2011.

[183] Albert Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. SIAM, 2005.

[184] Henry J. Trussell and Sergei Fogel. Identification and restoration of spatially variant motion blurs in sequential images. *IEEE Transactions on Image Processing*, 1(1):123–126, 1992.

[185] Henry J. Trussell and B. Hunt. Sectioned methods for image restoration. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(2):157–164, 1978.

[186] Eugene Tyrtyshnikov. Incomplete cross approximation in the mosaic-skeleton method. *Computing*, 64(4):367–380, 2000.

[187] Henk A. Van der Vorst. *Iterative Krylov methods for Large Linear Systems*, volume 13. Cambridge University Press, 2003.

[188] Charles F. Van Loan. Generalizing the singular value decomposition. *SIAM Journal on Numerical Analysis*, 13(1):76–83, 1976.

[189] Petr Vaněk, Jan Mandel, and Marian Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56(3):179–196, 1996.

[190] Nimisha Vedanti, Ravi P. Srivastava, John Sagode, and Vijay P. Dimri. An efficient 1D Occam's inversion algorithm using analytically computed first-and second-order derivatives for DC resistivity soundings. *Computers & Geosciences*, 31(3):319–328, 2005.

[191] Curt R. Vogel. *Computational Methods for Inverse Problems*. Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2002.

[192] Andrew J. Wathen, Bernd Fischer, and David J. Silvester. The convergence rate of the minimal residual method for the Stokes system. *Numerische Mathematik*, 71:121–134, 1995.

[193] Andrew J. Wathen and David J. Silvester. Fast iterative solution of stabilised stokes systems. Part I: using simple diagonal preconditioners. *SIAM Journal on Numerical Analysis*, 30(3):630–649, 1993.

[194] Jianlin Xia, Shivkumar Chandrasekaran, Ming Gu, and Xiaoye S. Li. Superfast multifrontal method for large structured linear systems of equations. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1382–1411, 2009.

[195] Jinchao Xu and Ludmil Zikatanov. Some observations on Babuška and Brezzi theories. Technical Report AM222, Penn State University, September 2000. http://www.math.psu.edu/ccma/reports.html.

[196] Zhiguang Xue, Nick Alger, and Sergey Fomel. Full-waveform inversion using smoothing kernels. In *SEG Technical Program Expanded Abstracts 2016*, pages 1358–1363. Society of Exploration Geophysicists, 2016.

[197] Leonardo Zepeda-Núñez and Laurent Demanet. The method of polarized traces for the 2D Helmholtz equation. *Journal of Computational Physics*, 308:347–388, 2016.

[198] Hejun Zhu, Siwei Li, Sergey Fomel, Georg Stadler, and Omar Ghattas. A Bayesian approach to estimate uncertainty for full-waveform inversion using a priori information from depth migration. *Geophysics*, 2016.

[199] Walter Zulehner. Nonstandard norms and robust estimates for saddle point problems. *SIAM Journal on Matrix Analysis and Applications*, 32:536–560, 2011.