POINT SPREAD FUNCTION APPROXIMATION OF HIGH RANK HESSIANS WITH LOCALLY SUPPORTED NON-NEGATIVE INTEGRAL KERNELS*

NICK ALGER†, TUCKER HARTLAND‡, NOEMI PETRA§, AND OMAR GHATTAS¶

Abstract. We present an efficient matrix-free point spread function (PSF) method for approximating operators that have locally supported non-negative integral kernels, with particular focus on approximating high rank Hessians in distributed parameter inverse problems governed by partial differential equations (PDEs). The proposed method uses impulse responses of the underlying operator (computed at a collection of scattered points) and interpolates translated and scaled versions of these impulse responses to approximate entries of the integral kernel. The interpolation uses a "local mean displacement invariance" approximation, which improves upon classical local translation invariance. Impulse responses are computed by applying the operator to a small number of Dirac combs associated with "batches" of point sources. By solving an ellipsoid packing problem, we choose as many point sources as possible per batch, while ensuring that the supports of the impulse responses within each batch do not overlap. Support ellipsoids are estimated a-priori based on impulse response moments, and these moments are computed for impulse responses at all points simultaneously via a procedure that involves applying the operator to a small number of polynomial functions. The ability to rapidly evaluate kernel entries allows us to construct a hierarchical matrix (H-matrix) approximation of the operator. Fast H-matrix methods are used to perform further matrix computations in nearly linear complexity, including matrix-matrix addition and multiplication, matrix inversion and factorization, and matrix-vector products. We illustrate our approach by applying the method to approximate the Hessians in an ice sheet flow PDE constrained inverse problem with highly informative data, and an advection dominated advection-diffusion PDE constrained inverse problem. Numerical results demonstrate that the proposed method substantially outperforms existing methods. We show that our method can approximate high rank operators using only a small number of operator applies.

Key words. data scalability, Hessian, hierarchical matrix, matrix-free, operator approximation, PDE constrained inverse problems, point spread function, product convolution, local translation invariance, impulse response, moment methods

AMS subject classifications. 35R30, 41A05, 41A35, 41A36, 42A85, 47A52, 47A58, 49K20, 65D12, 65F08, 65F10, 65N20, 65N21, 86A22, 86A40

1. Introduction.

We present an efficient *matrix-free* point spread function (PSF) method for approximating operators, $\mathcal{A} : L^2(\Omega) \to L^2(\Omega)'$, that have locally supported non-negative integral kernels. Here $\Omega \subset \mathbb{R}^d$ is a bounded domain, and $L^2(\Omega)'$ is the space of real-valued continuous linear functionals on $L^2(\Omega)$. We focus on approximating Hessians in optimization and inverse problems governed by partial differential equations (PDEs) [5, 44], though such operators also arise as certain Schur complements in Schur complement methods for solving partial differential equations (PDEs) [50, 64], Poincare-Steklov operators in domain decomposition methods (e.g.,

*Submitted to the editors DATE.

Funding: This research was supported by the National Science Foundation under Grant No. DMS-1840265 and CAREER-1654311, DOD grant FA9550-21-1-0084, and DOE grants DE-SC0021239 and DE-SC0019303.

†Oden Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX (nalger@oden.utexas.edu).

‡Department of Applied Mathematics, University of California, Merced, Merced, CA. (thartland@ucmerced.edu).

§Department of Applied Mathematics, University of California, Merced, Merced, CA. (npetra@ucmerced.edu).

¶Oden Institute for Computational Engineering and Sciences and Walker Department of Mechanical Engineering, The University of Texas at Austin, Austin, TX (omar@oden.utexas.edu).

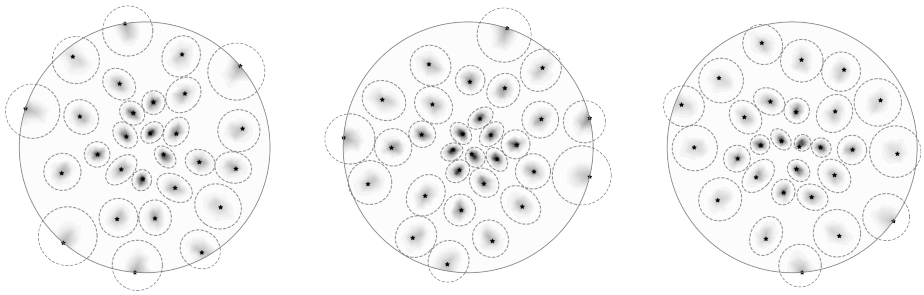


Fig. 1: Batches, η_b, of normalized impulses, ϕ_x, for the Stokes inverse problem data misfit Gauss-Newton Hessian (Section 7). Black stars are point source locations. Dashed gray ellipses are estimated impulse response support ellipsoids. The large circle is $\partial\Omega$.

Dirichlet-to-Neumann maps), covariance operators in spatial statistics [14, 31, 53], and blurring operators in imaging [17, 58]. Here "matrix-free" means that we may apply $\mathcal{A}$ and its transpose[1], $\mathcal{A}^T$, to arbitrary functions,

$$u \mapsto \mathcal{A}u \qquad \text{and} \qquad w \mapsto \mathcal{A}^T w, \tag{1.1}$$

via a black box computational procedure, but we cannot easily access entries of $\mathcal{A}$'s integral kernel. Evaluating the maps in (1.1) may require, as a sub procedure, performing a costly computation, such as solving a large linear system or time stepping.

We use *impulse response interpolation* to form a high rank operator approximation using a small number of operator applies. The impulse response, ϕ_x, associated with a point x is the Riesz representation[2] of the linear functional that results from applying $\mathcal{A}$ to a delta distribution (i.e., point source, impulse) centered at x. We compute batches of impulse responses by applying $\mathcal{A}$ to weighted sums of delta distributions associated with batches of points scattered throughout the domain (see Figure 1). Then we interpolate translated and scaled versions of these impulse responses to approximate entries of the operator's integral kernel. Picking the batches of points requires us to estimate the supports of the impulse responses ϕ_x *before* we compute them. The idea of estimating the supports of the functions ϕ_x a-priori was inspired by techniques from resolution analysis in seismic imaging. There, $\mathcal{A}^T$ is applied to a random noise function, and the width of ϕ_x is estimated to be the autocorrelation length of the resultant function near x [26, 68]. We use polynomial functions instead of random noise functions (see Section 4.1). Our method estimates the support of ϕ_x more accurately and reliably than random noise probing at the cost of the additional constraint that $\mathcal{A}$ has a non-negative integral kernel. Our method may be categorized as a PSF method that is loosely based on "product convolution" (PC) approximations, which are approximations of an operator by weighted sums of convolution operators with spatially varying weights. PC and PSF methods have a long history dating back several decades. We note the following papers (among many others), [1, 5, 10, 22,

[1]Recall that $\mathcal{A}^T : L^2(\Omega) \to L^2(\Omega)'$ is the unique operator satisfying $(\mathcal{A}u)(w) = (\mathcal{A}^T w)(u)$ for all $u, w \in L^2(\Omega)$, where $\mathcal{A}u \in L^2(\Omega)'$ is the result of applying $\mathcal{A}$ to $u \in L^2(\Omega)$, and $(\mathcal{A}u)(w)$ is the result of applying that linear functional to $w \in L^2(\Omega)$, and similar for operations with $\mathcal{A}^T$.

[2]Recall that the Riesz representative of a functional $\rho \in L^2(\Omega)'$ with respect to the L^2 inner product is the unique function $\rho^* \in L^2(\Omega)$ such that $\rho(w) = (\rho^*, w)_{L^2(\Omega)}$ for all $w \in L^2(\Omega)$.

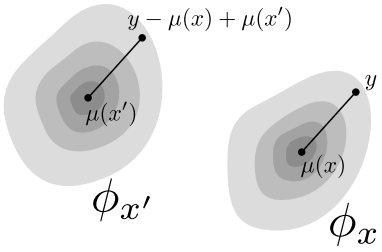


Fig. 2: $\mathcal{A}$ is locally mean displacement invariant if $\phi_x(y) \approx \phi_{x'}(y - \mu(x) + \mu(x'))$ when x is close to x'. Here $\mu(z)$ denotes the mean (center of mass) of ϕ_z.

24, 25, 27, 58, 72], in which the convolution kernels are constructed from sampling impulse responses of the operator to scattered point sources. For background on PC and PSF methods, we recommend the following papers: [18, 23, 30]. While PC approximations are based on an assumption of local translation invariance, the method we propose is based on an assumption we call "local mean displacement invariance," which is more general than local translation invariance, and includes both CUR low rank approximation and local translation invariance as special cases (Section 4.2 and Figure 2).

The ability to rapidly approximate entries of $\mathcal{A}$'s integral kernel allows us to form a hierarchical matrix (H-matrix) [11, 38] approximation of a discretized version of $\mathcal{A}$. H-matrices are a matrix format in which the rows and columns of the matrix are re-ordered, then the matrix is recursively subdivided into blocks, in such a way that many off-diagonal blocks are low rank, even though the matrix as a whole may be high rank. H-matrix methods permit us to perform matrix-vector products cheaply, and perform other useful linear algebra operations that cannot be done easily using the original operator. These operations include matrix-matrix addition, matrix-matrix multiplication, matrix factorization, and matrix inversion. The work and memory required to perform these operations for an $N \times N$ H-matrix scales *nearly linearly* in N (i.e., $o(N^{1+\epsilon})$ for any $\epsilon > 0$). The exact cost depends on the type of H-matrix used, the operation being performed, and the rank of the off-diagonal blocks [35]. The ability to perform these operations permits, for example, fast solution of Newton linear systems in PDE constrained optimization, fast sampling of ill-conditioned posterior distribution in Bayesian inverse problems, and construction of high rank Gaussian surrogate models that can be used for uncertainty quantification.

2. Why we need high rank Hessian approximations. In distributed parameter inverse problems governed by PDEs, one seeks to infer an unknown spatially varying parameter field from limited observations of a state variable that depends on the parameter implicitly through the solution of a PDE. Conventionally, the inverse problem is formulated using either a deterministic framework [8, 70], or a Bayesian probabilistic framework [46, 66, 67]. In the deterministic framework, one solves an optimization problem to find the parameter that best fits the observations, subject to appropriate regularization [20]. In the probabilistic framework, Bayes' theorem combines the observations with prior information to form a posterior distribution over the space of all possible parameter fields, and computations are performed to extract statistical information about the parameter from this posterior. The Hessian of the objective function in the determinstic optimization problem and the Hessian of the

negative log posterior in the Bayesian setting are equal or approximately equal to each other under typical noise, regularization, and prior models, so we refer to both of these Hessians as "the Hessian." The Hessian consists of a data misfit term (the "data misfit Hessian"), which depends on discrepancy between the observations and the associated model predictions, and a regularization or prior term (the "regularization Hessian") which does not depend on the observations. For more details on the Hessian, see [4, 32, 69].

Hessian approximations and preconditioners are highly desirable because the Hessian is central to efficient solution of inverse problems in both the deterministic and Bayesian settings. When solving the deterministic optimization problem with Newton-type methods, the Hessian is the coefficient operator for the linear system that must be solved or approximately solved at every Newton iteration. Good Hessian preconditioners reduce the number of iterations required to solve these Newton linear systems with Krylov methods such as conjugate gradient [63]. In the Bayesian setting, the inverse of the Hessian is the covariance of a local Gaussian approximation of the posterior. Such a Gaussian distribution can be used as a direct approximations of the posterior, or as a proposal for Markov chain Monte-Carlo methods for drawing samples from the posterior. For instance, see [47, 60] and the references therein.

Owing to the implicit dependence of predicted observations on the parameter, entries of the Hessian are not easily accessible. Rather, the Hessian may be applied to a vector via a computational process that involves solving a pair of forward and adjoint PDEs which are linearizations of the original PDE [32, 37]. The most popular matrix-free Hessian approximation methods are based on low rank approximation of either the data misfit Hessian, or the data misfit Hessian preconditioned by the regularization Hessian, e.g., [13, 16, 28, 60, 65]. Krylov methods such as Lanczos or randomized methods [15, 41] are typically used to construct these low rank approximations by applying the Hessian to vectors. Using these methods, the required number of Hessian applications (and hence the required number of PDE solves) is typically proportional to the rank of the low rank approximation. Low rank approximation methods are justified by arguing that the numerical rank of the data misfit Hessian is insensitive to the dimension of the discretized parameter. This means that the required number of PDE solves remains the same as the mesh used to discretize the parameter is refined. However, in many inverse problems of practical interest the numerical rank of the data misfit Hessian, while mesh independent, is still large, which makes it costly to approximate the Hessian using low rank approximation methods [7, 13, 44].

Examples of inverse problems with high rank data misfit Hessians include advection dominated advection-diffusion inverse problems [2][29, Chapter 5], high frequency wave inverse problems [13], and more generally, all inverse problems in which the observations highly inform the parameter. The eigenvalues of the data misfit Hessian characterize the sensitivity of the predicted observations to perturbations of the parameter in the corresponding eigenvector directions, so more informative data leads to larger eigenvalues and therefore a larger numerical rank [3][4, Section 1.4 and Chapter 4]. Roughly speaking, the numerical rank of the data misfit Hessian is the dimension of the subspace of parameter space that is informed by the data. The numerical rank of the regularization preconditioned data misfit Hessian may be reduced by increasing the strength of the regularization, but this throws away useful information—components of the parameter that could be learned from components of the observations that have a large signal to noise ratio are instead reconstructed based on the regularization [6, Section 4][70, Chapters 1 and 7]. Hence, low rank approximation methods suffer from a predicament—if the data highly inform the parameter

and the regularization is chosen appropriately, then a large number of operator applies are required to form an accurate approximation of the Hessian using low rank approximation methods. High rank Hessian approximation methods are needed.

Recently there have been improvements in matrix-free H-matrix construction methods in which an operator is applied to structured random vectors, and the response of the operator to those random vectors is processed to construct an H-matrix approximation [51, 52, 55, 56, 57]. These methods (which we do not use here) have been used to approximate Hessians in PDE constrained inverse problems [7, 42]. Although the required number of matrix-vector products grows nearly linearly in the discretized parameter dimension and linearly in the rank of the off-diagonal blocks, the required number of matrix-vector products is still large (e.g., hundreds to thousands). In this paper, we also form an H-matrix Hessian approximation. However, to reduce the required number of operator actions, we first form a PSF approximation of the data misfit Hessian by exploiting locality and non-negative integral kernel properties, then form the H-matrix using classical H-matrix techniques. Not all data misfit Hessians satisfy the local non-negative integral kernel properties (for example, the wave inverse problem data misfit Hessian has a substantial amount of negative entries in its integral kernel). However, many data misfit Hessians of practical interest do satisfy these properties (either exactly or approximately), and our method is targeted at approximating these Hessians.

3. Preliminaries.

Let $\Omega \subset \mathbb{R}^d$ be a bounded domain (typically $d = 1$, 2, or 3). We seek to approximate integral operators $\mathcal{A} : L^2(\Omega) \to L^2(\Omega)'$ of the form

$$(\mathcal{A}u)(w) := \int_\Omega \int_\Omega w(y)\Phi(y,x)u(x)dxdy. \tag{3.1}$$

Here the linear functional $\mathcal{A}u \in L^2(\Omega)'$ is the result of applying $\mathcal{A}$ to $u \in L^2(\Omega)$, the scalar $(\mathcal{A}u)(w)$ is the result of applying that linear functional to $w \in L^2(\Omega)$. The integral kernel, $\Phi : \Omega \times \Omega \to \mathbb{R}$, exists in principle but is not easily accessible. In this section we describe how to extend the action of $\mathcal{A}$ to distributions, which allows us to define impulse responses (Section 3.1), we state the conditions on $\mathcal{A}$ that our method requires (Section 3.2), and we detail finite element discretization (Section 3.3).

3.1. Distributions and impulse responses.

The action of $\mathcal{A}$ may be extended to distributions if Φ is sufficiently regular. Let $\rho \in L^2(\Omega)'$, and let $\rho^* \in L^2(\Omega)$ denote the Riesz representative of ρ with respect to the $L^2(\Omega)$ inner product. We have

$$(\mathcal{A}\rho^*)(w) = \int_\Omega \int_\Omega w(y)\Phi(y,x)\rho^*(x)dx\ dy \tag{3.2a}$$

$$= \int_\Omega w(y) \int_\Omega \Phi(y,x)\rho^*(x)dx\ dy = \int_\Omega w(y)\rho\left(\Phi(y,\cdot)\right)dy, \tag{3.2b}$$

where $\Phi(y,\cdot)$ denotes the function $x \mapsto \Phi(y,x)$. Now let $\mathcal{D}(\Omega) \subset L^2(\Omega)$ be a suitable space of test functions and let $\rho : \mathcal{D}(\Omega) \to \mathbb{R}$ be a distribution. In this case, the Riesz representative ρ^* may not exist, so the derivation in (3.2) is not valid. However, if Φ is sufficiently regular such that the function $y \mapsto \rho\left(\Phi(y,\ \cdot\ )\right)$ is well defined for almost all $y \in \Omega$, and if this function is in $L^2(\Omega)$, then the right hand side of (3.2b) is well defined. Hence, we *define* the action of $\mathcal{A}$ on the distribution ρ to be the right hand side of (3.2b). We denote this action by "$\mathcal{A}\rho^*$," even if ρ^* does not exist.

Let δ_x denote the delta distribution[3] (i.e., point source, impulse) centered at the

[3]Recall that the delta distribution $\delta_x : \mathcal{D}(\Omega) \to \mathbb{R}$ is defined by $\delta_x(w) = w(x)$ for all $w \in \mathcal{D}(\Omega)$.

point $x \in \Omega$. The *impulse response* of $\mathcal{A}$ associated with x is the function $\phi_x : \Omega \to \mathbb{R}$,

$$\phi_x := (\mathcal{A}\delta_x^*)^*, \tag{3.3}$$

that is formed by applying $\mathcal{A}$ to δ_x, then taking the Riesz representation of the resulting linear functional. Using (3.2b) and the definition of the delta distribution, we see that ϕ_x may also be written as the function $\phi_x(y) = \Phi(y, x)$.

3.2. Required conditions. We focus on approximating operators that satisfy the following conditions:

1. The kernel Φ is sufficiently regular so that ϕ_x is well defined for all $x \in \Omega$.
2. The supports of the impulse responses ϕ_x are contained in localized regions.
3. The integral kernel is non-negative in the sense that

$$\Phi(y, x) \geq 0$$

for all $(y, x) \in \Omega \times \Omega$.[4]

Our method may still perform well if these conditions are relaxed slightly. It is fine if the support of ϕ_x is not perfectly contained in a localized region, as long as the bulk of the "mass" of ϕ_x is contained in a localized region. The integral kernel does not need to be non-negative for all pairs of points $(y, x) \in \Omega \times \Omega$, as long as it is non-negative for the vast majority of pairs of points (y, x), and as long as the negative numbers are comparatively small. If these conditions are violated, our method will incur additional error. If these conditions are violated too much, our method may fail.

3.3. Finite element discretization. In computations, functions are discretized and replaced by finite dimensional vectors, and operators mapping between infinite dimensional spaces are replaced by operators mapping between finite dimensional spaces. In this paper we discretize using continuous Galerkin finite elements satisfying the Kronecker property (defined below). With minor modifications, our method could be used with more general finite element methods, or other discretization schemes such as finite differences or finite volumes.

Let $\psi_1, \psi_2, \ldots, \psi_N$ be a set of continuous Galerkin finite element basis functions used to discretize the problem on a mesh with mesh size parameter h, let $V_h := \text{span}\,(\psi_1, \psi_2, \ldots, \psi_N)$ be the corresponding finite element space under the L^2 inner product, and let $\zeta_i \in \mathbb{R}^d$, $i = 1, \ldots, N$ be the Lagrange nodes associated with the functions ψ_i. We assume that the finite element basis satisfies the Kronecker property, i.e., $\psi_i(\zeta_i) = 1$ and $\psi_i(\zeta_j) = 0$ if $i \neq j$. For $u_h \in V_h$ we write $\boldsymbol{u} \in \mathbb{R}^m_{\mathbf{M}}$ to denote the coefficient vector for u_h with respect to the finite element basis, i.e., $u_h(x) = \sum_{i=1}^N \boldsymbol{u}_i \psi_i(x)$. Linear functionals $\rho_h \in V_h'$ have coefficient dual vectors $\boldsymbol{\rho} \in \mathbb{R}^m_{\mathbf{M}^{-1}}$, with entries $\boldsymbol{\rho}_i = \rho_h(\psi_i)$ for $i = 1, \ldots, m$. Here $\mathbf{M} \in \mathbb{R}^{N \times N}$ denotes the sparse finite element mass matrix which has entries $\mathbf{M}_{ij} = \int_\Omega \psi_i(x)\psi_j(x)dx$ for $i, j = 1, \ldots, N$. The space $\mathbb{R}^N_{\mathbf{M}}$ is $\mathbb{R}^N$ with the inner product $(\mathbf{u}, \mathbf{w})_{\mathbf{M}} := \mathbf{u}^T\mathbf{M}\mathbf{w}$, and $\mathbb{R}^N_{\mathbf{M}^{-1}}$ is the analogous space with $\mathbf{M}^{-1}$ replacing $\mathbf{M}$. Direct calculation shows that $\mathbb{R}^N_{\mathbf{M}}$ and $\mathbb{R}^N_{\mathbf{M}^{-1}}$ are isomorphic to V_h and V_h' as Hilbert spaces, respectively.

After discretization, the operator $\mathcal{A} : L^2(\Omega) \to L^2(\Omega)'$ is replaced by an operator $A_h : V_h \to V_h'$, which becomes an operator

$$\mathbf{A} : \mathbb{R}^N_{\mathbf{M}} \to \mathbb{R}^N_{\mathbf{M}^{-1}}$$

[4]Note that having a non-negative integral kernel is different from positive semi-definiteness. The operator $\mathcal{A}$ need not be positive semi-definite to use our method, and positive semi-definite operators need not have a non-negative integral kernel.

under the isomorphism discussed above. Our method is agnostic to the computational procedure for approximating $\mathcal{A}$ with $\mathbf{A}$. What is important is that we do not have direct access to matrix entries $\mathbf{A}_{ij}$. Rather, we have a computational procedure that allows us to compute matrix-vector products $\mathbf{u} \mapsto \mathbf{A}\mathbf{u}$ and $\mathbf{w} \mapsto \mathbf{A}^T\mathbf{w}$, and computing these matrix-vector products is costly. Of course, matrix entries can be computed via matrix-vector products as $\mathbf{A}_{ij} = (\mathbf{A}\mathbf{e}_j)_i$, where $\mathbf{e}_j = (0, \ldots, 0, 1, 0, \ldots, 0)^T$ is the length N unit vector with one in the jth coordinate and zeros elsewhere. But computing the matrix-vector product $\mathbf{e}_j \mapsto \mathbf{A}\mathbf{e}_j$ is costly, and therefore wasteful if we do not use other matrix entries in the jth column of $\mathbf{A}$. Hence, methods for approximating $\mathbf{A}$ are computationally intractable if they require accessing scattered matrix entries from many different rows and columns of $\mathbf{A}$.

The operator $A_h : V_h \to V_h'$ can be written in integral kernel form, (3.1), but with Φ replaced by a slightly different integral kernel, Φ_h, which we do not know, and which differs from Φ due to discretization error. Since the functions in V_h are continuous at x, the delta distribution δ_x is a continuous linear functional on V_h, which has a discrete dual vector $\boldsymbol{\delta}_x \in \mathbb{R}^N_{\mathbf{M}^{-1}}$ with entries $(\boldsymbol{\delta}_x)_i = \psi_i(x)$ for $i = 1, \ldots, N$. Additionally, it is straightforward to verify that the Riesz representation, $\rho_h^* \in V_h$, of a functional $\rho \in V_h'$ has coefficient vector $\boldsymbol{\rho}^* = \mathbf{M}^{-1}\boldsymbol{\rho}$. Therefore, the formula for the impulse response from (3.3) becomes $\boldsymbol{\phi}_x = (A_h\delta_x^*)^* = \mathbf{M}^{-1}\mathbf{A}\mathbf{M}^{-1}\boldsymbol{\delta}_x$, and the (y, x) kernel entry of Φ_h may be written as $\Phi_h(y, x) = \boldsymbol{\delta}_y^T\boldsymbol{\phi}_x = \boldsymbol{\delta}_y^T\mathbf{M}^{-1}\mathbf{A}\mathbf{M}^{-1}\boldsymbol{\delta}_x$. Now define $\boldsymbol{\Phi} \in \mathbb{R}^{N\times N}$ to be the following dense matrix of kernel entries evaluated at all pairs of Lagrange nodes:

$$\boldsymbol{\Phi}_{ij} := \Phi_h(\zeta_i, \zeta_j). \tag{3.4}$$

Because of the Kronecker property of the finite element basis, we have $\boldsymbol{\delta}_{\zeta_i} = \mathbf{e}_i$. Thus, we have $\Phi_h(\zeta_i, \zeta_j) = \left(\mathbf{M}^{-1}\mathbf{A}\mathbf{M}^{-1}\right)_{ij}$, which implies

$$\mathbf{A} = \mathbf{M}\boldsymbol{\Phi}\mathbf{M}. \tag{3.5}$$

Broadly, our method will construct an H-matrix approximation of $\mathbf{A}$ by forming H-matrix approximations of $\boldsymbol{\Phi}$ and $\mathbf{M}$ (or a lumped mass version of $\mathbf{M}$), then multiplying these matrices per (3.5) using H-matrix methods. Classical H-matrix construction methods require access to arbitrary matrix entries $\boldsymbol{\Phi}_{ij}$, but these matrix entries are not easily accessible. The bulk of our method is therefore dedicated to forming approximations of these matrix entries that can be evaluated rapidly.

Lumped mass matrix. At the continuum level, Φ is assumed to be non-negative. However, entries of Φ_h, involve inverse mass matrices, which typically contain negative numbers. If there are too many negative numbers, or if the negative numbers are too large, our algorithm will be less robust. We therefore recommend replacing the mass matrix, $\mathbf{M}$, with a positive diagonal *lumped mass* approximation. In our numerical results, we use the lumped mass matrix constructed by replacing off-diagonal entries of the mass matrix by zero. Other mass lumping techniques may be used [43].

4. Key innovations. In this section we present two key innovations that our method is based on. First, we define moments of the impulse responses, ϕ_x, show how these moments can be computed efficiently, and use these moments to form ellipsoid shaped a-priori estimates for the supports of the impulse responses (Section 4.1). Second, we describe an improved method of approximating impulse responses by other nearby impulse responses, which we call "normalized local mean displacement invariance" (Section 4.2).

4.1. Impulse response moments and ellipsoid support estimate. The impulse response ϕ_x may be interpreted as a scaled probability distribution because of the non-negative integral kernel property. Let $V : \Omega \to \mathbb{R}$,

$$V(x) := \int_\Omega \phi_x(y) dy \tag{4.1}$$

denote the spatially varying scaling factor, and for $i, j = 1, \ldots, d$ define $\mu : \Omega \to \mathbb{R}^d$ and $\Sigma : \Omega \to \mathbb{R}^{d\times d}$ as follows:

$$\mu^i(x) := \int_\Omega (\phi_x(y)/V(x)) y^i \, dy \tag{4.2}$$

$$\Sigma^{ij}(x) := \int_\Omega (\phi_x(y)/V(x)) \left(y^i - \mu^i(x)\right) \left(y^j - \mu^j(x)\right) \, dy, \tag{4.3}$$

where $\mu^i(x)$ denotes the i^{th} component of the vector $\mu(x)$, and $\Sigma^{ij}(x)$ denotes the (i, j) entry of the matrix $\Sigma(x)$. The vector $\mu(x) \in \mathbb{R}^d$ and the matrix $\Sigma(x) \in \mathbb{R}^{d\times d}$ are the mean and covariance of the normalized version of ϕ_x, respectively.

The direct approach to compute $V(x)$, $\mu(x)$, and $\Sigma(x)$ is to apply $\mathcal{A}$ to a point source centered at x to obtain ϕ_x, as per (3.3). Then one can post process ϕ_x to determine $V(x)$, $\mu(x)$, and $\Sigma(x)$. However, this direct approach is not feasible because we need to know $V(x)$, $\mu(x)$, and $\Sigma(x)$ before we compute ϕ_x, in order choose the point x. Computing ϕ_x in order to determine $V(x)$, $\mu(x)$, and $\Sigma(x)$ would be extremely computationally expensive, and defeat the purpose of our algorithm, which is to reduce the computational cost by computing impulse responses in batches. Fortunately, it is possible to compute $V(x)$, $\mu(x)$, and $\Sigma(x)$ indirectly, *for all points* $x \in \Omega$ *simultaneously*, by applying $\mathcal{A}^T$ to one constant function, d linear functions, and $d(d + 1)/2$ quadratic functions (e.g., 6 total operator applies in two spatial dimensions and 10 in three spatial dimensions). This may be motivated by analogy to matrices. If $\mathbf{B} \in \mathbb{R}^{N\times N}$ is a matrix with i^{th} column $\mathbf{b}_i$ and $\mathbf{w} \in \mathbb{R}^N$, then

$$\mathbf{B}^T\mathbf{w} = \begin{bmatrix} — & \mathbf{b}_1^T & — \\ & \vdots & \\ — & \mathbf{b}_N^T & — \end{bmatrix} \mathbf{w} = \begin{bmatrix} \mathbf{b}_1^T\mathbf{w} \\ \vdots \\ \mathbf{b}_N^T\mathbf{w} \end{bmatrix}.$$

By computing one matrix-vector product of $\mathbf{B}^T$ with $\mathbf{w}$, we compute the inner product of each column of $\mathbf{B}$ with $\mathbf{w}$ simultaneously. The operator case is analogous, with ϕ_x taking the place of a matrix column. We have

$$\left(\mathcal{A}^T w\right)^* (x) = \int_\Omega \Phi(y, x) w(y) dy = (\phi_x, w)_{L^2(\Omega)}. \tag{4.4}$$

By computing one operator application of $\mathcal{A}^T$ to w, we compute the inner product of each ϕ_x with w, for all points x simultaneously.

Let C, L^i, and Q^{ij} be the following constant, linear, and quadratic functions:

$$C(x) := 1, \qquad L^i(x) := x^i, \qquad Q^{ij}(x) := x^i x^j$$

for $i, j = 1, \ldots, d$. Using the definition of V in (4.1) and using (4.4), we have

$$V(x) = \int_\Omega \phi_x(y) C(y) \, dy = (\phi_x, C)_{L^2(\Omega)} = \left(\mathcal{A}^T C\right)^* (x).$$

Hence, we compute $V(x)$ for all x simultaneously by applying $\mathcal{A}^T$ to C. Analogous manipulations show that $\mu(x)$ and $\Sigma(x)$ may be computed for all points x simultaneously by applying $\mathcal{A}^T$ to the functions L^i and Q^{ij}, respectively. We have

$$V = \left(\mathcal{A}^T C\right)^* \tag{4.5a}$$

$$\mu^i = \left(\mathcal{A}^T L^i\right)^* / V \tag{4.5b}$$

$$\Sigma^{ij} = \left(\mathcal{A}^T Q^{ij}\right)^* / V - \mu^i \cdot \mu^j \tag{4.5c}$$

for $i, j = 1, \ldots, d$. Here u/w denotes pointwise division, $(u/w)(x) = u(x)/w(x)$, and $u \cdot w$ denotes pointwise multiplication, $(u \cdot w)(x) = u(x)w(x)$.

We approximate the support of ϕ_x with the ellipsoid

$$E_x := \{x' \in \Omega : (x' - \mu(x))^T \Sigma(x)^{-1} (x' - \mu(x)) \leq \tau^2\}, \tag{4.6}$$

where τ is a fixed constant. The ellipsoid E_x is the set of points within τ standard deviations from the mean of the Gaussian distribution with mean $\mu(x)$ and covariance $\Sigma(x)$, i.e., the Gaussian distribution which has the same mean and covariance as the normalized version of ϕ_x. The quantity τ is a parameter that must be chosen appropriately. The larger τ is, the larger the ellipsoid E_x is, and the more conservative the estimate is for the support of ϕ_x. However, in Section 5.1 we will see that the cost of our algorithm depends on how many non-overlapping ellipsoids E_x we can "pack" in the domain Ω (more ellipsoids is better), and choosing a larger value of τ means that fewer ellipsoids will fit in Ω. In practice, we find that $\tau = 3$ yields a reasonable balance between these competing interests, and use $\tau = 3$ in our numerical results. The fraction of the "mass" of ϕ_x residing outside of E_x is less than $1/\tau^2$ by Chebyshev's inequality, though this bound is typically conservative.

4.2. Local mean displacement invariance. Let x and x' be points in Ω that are "close" to each other, and consider the following approximations:

$$\phi_x(y) \approx \phi_{x'}(y) \tag{4.7}$$

$$\phi_x(y) \approx \phi_{x'}(y - x + x') \tag{4.8}$$

$$\phi_x(y) \approx \phi_{x'}\left(y - \mu(x) + \mu(x')\right) \tag{4.9}$$

$$\phi_x(y)/V(x) \approx \phi_{x'}\left(y - \mu(x) + \mu(x')\right)/V(x'). \tag{4.10}$$

These are four different ways to approximate an impulse response by a nearby impulse response, with each successive approximation building upon the previous ones. Our method uses (4.10), which is the most sophisticated of these approximations. Approximation (4.7) says that ϕ_x can be approximated by $\phi_{x'}$ when x and x' are close. Operators satisfying (4.7) can be well approximated via low rank CUR approximation [54]. However, the required rank in the low rank approximation can be large, which makes algorithms based on (4.7) expensive. Operators that satisfy (4.8) are called "locally translation invariant" because integral kernel entries $\Phi(y, x)$ for such operators are approximately invariant under translation of x and y by the same displacement, i.e., $x \to x + h$ and $y \to y + h$. It is straightforward to show that if equality holds in (4.8), then $\mathcal{A}$ is a convolution operator. Locally translation invariant operators act like convolutions locally, and can therefore be well approximated by PC approximations.

Approximation (4.9) improves upon (4.7) and (4.8), and generalizes both. On one hand, if (4.7) holds, then $\mu(x) \approx \mu(x')$, and so (4.9) holds. On the other hand,

translating a distribution translates its mean, so if (4.8) holds, then $\mu(x')-\mu(x) \approx x'-x$, so again (4.9) holds. But approximation (4.9) can hold in situations where neither (4.7) nor (4.8) holds. For example, because the expected value commutes with affine transformations, (4.9) will hold when $\mathcal{A}$ is locally translation invariant with respect to a rotated frame of reference, while (4.8) will not hold in this case. Additionally, (4.9) generalizes to operators $\mathcal{A} : L^2(\Omega_1) \to L^2(\Omega_2)'$ that map between function spaces on different domains Ω_1 and Ω_2, and even operators that map between domains with different spatial dimensions. In contrast, (4.8) does not naturally generalize to operators that map between function spaces on different domains, because the formula $y - x + x'$ requires vectors in Ω_2 and Ω_1 to be added together. We call (4.9) "local mean displacement invariance," and illustrate (4.9) in Figure 2.

We use approximation (4.10), which is the same as (4.9), except for the extra factors of $1/V(x)$ on the left hand side and $1/V(x')$ on the right hand side. These factors make the approximation more robust if $V(x)$ varies widely. Approximation (4.10) is equivalent to (4.9), but with ϕ_x replaced by its normalized version, $\phi_x/V(x)$. We call (4.10) "normalized local mean displacement invariance."

5. Operator approximation algorithm. We use (4.5) in Section 4.1 to compute V, μ, and Σ by applying $\mathcal{A}^T$ to a small number of polynomial functions. Then we use (4.6) to form ellipsoid shaped estimates for the support of the ϕ_x, *without* computing them. This allows us to compute large numbers of ϕ_{x_i} in "batches," η_b (see Figure 1). We compute one batch, denoted η_b, by applying $\mathcal{A}$ to a weighted sum of point sources (Dirac comb) associated with a batch, S_b, of points x_i scattered throughout Ω (Section 5.2). The batch of points S_b is chosen via a greedy ellipsoid packing algorithm so that, for $x_i, x_j \in S_b$, the support ellipsoid for ϕ_{x_i} and the support ellipsoid for ϕ_{x_j} do not overlap if $i \neq j$ (Section 5.1). Because these supports do not overlap (or do not overlap much), we can post process η_b to recover the functions ϕ_{x_i} associated with all points $x_i \in S_b$—with one application of $\mathcal{A}$, we recover many ϕ_{x_i} (Section 5.2). The process is repeated until a desired number of batches is reached.

Once the batches η_b are computed, we approximate the integral kernel $\Phi(y, x)$ at arbitrary points (y, x) by interpolation of translated and scaled versions of the computed ϕ_{x_i} (Section 5.3). The key idea behind the interpolation is the normalized local mean displacement invariance assumption discussed in Section 4.2. Specifically, we approximate $\Phi(y, x) = \phi_x(y)$ by a weighted linear combination of the values $\frac{V(x)}{V(x_i)}\phi_{x_i}(y-\mu(x)+\mu(x_i))$ for a small number of sample points x_i near x. The weights are determined by radial basis function interpolation.

The ability to rapidly evaluate approximate kernel entries $\Phi(y, x)$ allows us to construct an H-matrix approximation, $\mathbf{\Phi}_H \approx \mathbf{\Phi}$, using the conventional adaptive cross H-matrix construction method (Section 5.4). In this method, one forms low rank approximations of off-diagonal blocks of the matrix by sampling rows and columns of those blocks. We then use H-matrix methods to convert $\mathbf{\Phi}_H$ into an H-matrix approximation $\mathbf{A}_H \approx \mathbf{A}$. The complete algorithm for constructing $\mathbf{A}_H$ is shown in Algorithm 1. The cost is discussed in Section 6.

When $\mathcal{A}$ is symmetric positive semi-definite, $\mathbf{A}_H$ may be non-symmetric and indefinite due to errors in the approximation. In this case, one may (optionally) modify the H-matrix representation of $\mathbf{A}_H$ to make it symmetric positive semi-definite using a rational function method that we describe in Appendix B.

Algorithm 1: Construct H-matrix approximation

Input : Linear operator $\mathcal{A}$, parameter n_b
Output: H-matrix $\mathbf{A}_H$
Compute V, μ, and Σ (Equations (4.5) in Section 4.1)
for $k = 1, 2, \ldots, n_b$ **do**
    Choose a batch of sample points, S_k (Section 5.1)
    Compute η_k by applying $\mathcal{A}$ to the Dirac comb for S_k (Section 5.2)
Form H-matrix approximation $\mathbf{\Phi}_H$ of integral kernel (Section 5.4)
Form H-matrix approximation $\mathbf{A}_H$ of $\mathcal{A}$ (Section 5.4)
(optional) Modify $\mathbf{A}_H$ to make it symmetric positive definite (Appendix B)

5.1. Sample point selection via greedy ellipsoid packing. We choose sample points, x_i, in batches S_k. We use a greedy ellipsoid packing algorithm to choose as many points as possible per batch, while ensuring that there is no overlap between the support ellipsoids, E_{x_i}, associated with the sample points within a batch.

We start with a finite set of candidate points X and build S_k incrementally with points selected from X. For simplicity of explanation, here S_k and X are mutable sets that we add points to and remove points from. First we initialize S_k as an empty set. Then we select the candidate point $x_i \in X$ that is the farthest away from all points in previous sample point batches $S_1 \cup S_2 \cup \cdots \cup S_{k-1}$. Candidate points for the first batch S_1 are chosen randomly from X. Once x_i is selected, we remove x_i from X. Then we perform the following two checks:

1. We check whether x_i is sufficiently far from all of the previously chosen points in the current batch, in the sense that $E_{x_i} \cap E_{x_j} = \{\}$ for all $x_j \in S_k$.
2. We make sure that $V(x_i)$ is not too small, by checking whether $V(x_i) > \epsilon V_{\max}$. Here $V_{\max}$ is the largest value of $V(x_j)$ over all points q in the initial set of candidate points, and ϵ is a small threshold parameter (we use $\epsilon = 10^{-5}$).

If x_i passes both these checks (i.e., if x_i is sufficiently far from other points in the batch, and $V(x_i)$ is not too small) then we add x_i to S_k. Otherwise we discard x_i. This process repeats until there are no more points in X. We check whether $E_{x_i} \cap E_{x_j} = \{\}$ using the ellipsoid intersection test described in Appendix A. We repeat the process to construct several batches of points $S_1, S_2, \ldots, S_{n_b}$. In our implementation, for each batch, X is initialized as the set of all Lagrange nodes for the finite element basis functions used to discretize the problem, except for points in previously chosen batches.

5.2. Impulse response batches. We compute impulse responses ϕ_{x_i} in batches by applying $\mathcal{A}$ to Dirac combs. The Dirac comb, ξ_k, associated with a batch of sample points, S_k, is the following weighted sum of Dirac distributions (point sources) centered at the points $x_i \in S_k$:

$$\xi_k := \sum_{x_i \in S_k} \delta_{x_i} / V(x_i).$$

We compute the *impulse batch*, η_k, by applying $\mathcal{A}$ to the Dirac comb:

$$\eta_k := (\mathcal{A}\xi_k^*)^* = \sum_{x_i \in S_k} \phi_{x_i} / V(x_i). \tag{5.1}$$

The last equality in (5.1) follows from linearity and the definition of ϕ_{x_i} in (3.3). Since the points x_i are chosen so that the ellipsoid E_{x_i} that (approximately) supports ϕ_i, and the ellipsoid E_{x_j} that (approximately) supports ϕ_j do not overlap when $i \neq j$, we have (approximately)

$$\phi_{x_i}(z) = \begin{cases} \eta_k(z)V(x_i), & z \in E_{x_i} \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

for all $x_i \in S_k$. By performing one matrix-vector product, $\xi_k \mapsto (\mathcal{A}\xi_k^*)^*$, we recover ϕ_{x_i} for every point $x_i \in S_k$.

Each point source δ_{x_i} is scaled by $1/V(x_i)$ so that the resulting scaled impulse responses within η_k are comparable in magnitude. Without this scaling, the portion of ϕ_{x_i} outside of E_{x_i}, which we neglect, may overwhelm ϕ_{x_j} for a nearby point x_j if $V(x_i)$ is much larger than $V(x_j)$. Note that we are not in danger of dividing by zero, because our ellipsoid packing procedure from Section 5.1 excludes x_i from consideration as a sample point if $V(x_i)$ is smaller than a predetermined threshold.

5.3. Approximate integral kernel entries. Given $(y, x) \in \Omega \times \Omega$, let

$$z_i := y - \mu(x) + \mu(x_i), \qquad i = 1, \ldots, k_n,$$

where $\{x_i\}_{i=1}^{k_n}$ are the k_n nearest sample points to x, excluding points x_i for which $z_i \notin \Omega$. Here k_n is a small user defined parameter; e.g., $k_n = 10$. We find the k_n nearest sample points to x by querying a precomputed kd-tree [9] of all sample points. We check whether $z_i \in \Omega$ by querying a precomputed axis aligned bounding box tree (AABB tree) [21] of the mesh cells used to discretize the problem. Further, let

$$f_i := \frac{V(x)}{V(x_i)} \phi_{x_i}(z_i), \qquad i = 1, \ldots, k_n. \tag{5.3}$$

Note that $\phi_{x_i}(z_i)$ is well defined because $z_i \in \Omega$, and $\frac{V(x)}{V(x_i)}$ is well defined because our sample point picking procedure in Section 5.1 ensures that $V(x_i) > 0$. Per the discussion in Section 4.2, we expect $\Phi(y, x) \approx f_i$ for $i = 1, \ldots, k_n$; the closer x_i is to x, the better we expect the approximation to be. We therefore approximate $\Phi(y, x)$ by interpolating the (point,value) pairs $\{(x_i, f_i)\}_{i=1}^{k_n}$ at the point x. Interpolation is performed using the following radial basis function [71] scheme:

$$\Phi(y, x) \approx \widetilde{\Phi}(y, x) := \sum_{i=1}^{k_n} c_i \, \varphi(\|x - x_i\|), \tag{5.4}$$

where c_i are weights, and φ is a polyharmonic spline radial basis function, which is defined as $\varphi(r) := r^d$ if d is odd, and $\varphi(r) := r^d \log r$ for $r > 0$ and $\varphi(0) := 0$ if d is even. The vector of weights, $c = (c_1, c_2, \ldots, c_{k_n})^T$, is found as the solution to the $k_n \times k_n$ linear system

$$Bc = f, \tag{5.5}$$

where $B \in \mathbb{R}^{k_n \times k_n}$ has entries $B_{ij} := \varphi(\|x_i - x_j\|)$, and $f \in \mathbb{R}^{k_n}$ has entries f_i from (5.3). Polyharmonic spline radial basis function interpolation yields the smoothest interpolant of the given points and values, in a certain least squares sense [19].

Symbol	Typical size	Variable name
N	10^2–10^9	Number of finite element degrees of freedom
n_b	1–25	Number of batches
k_h	5–50	H-matrix rank
k_n	5–15	Number of nearest neighbors for RBF interpolation
d	1–3	Spatial dimension
m	10^1–10^4	Total number of sample points (all batches)
$\|S_i\|$	1–500	Number of sample points in the ith batch
l	1–2	Rational function parameter for SPSD modification

Table 1: Symbols used for variables in computational cost estimates, and approximate ranges for their sizes in practice.

To evaluate f_i, we check whether $z_i \in E_{x_i}$ using (4.6). If $z_i \notin E_{x_i}$, then z_i is outside the estimated support of ϕ_{x_i}, so we set $f_i = 0$. If $z_i \in E_{x_i}$, we look up the batch index b such that $x_i \in S_b$, and evaluate f_i via the formula $f_i = V(x)\eta_b(z_i)$, per (5.2). Note that z_i is typically not a gridpoint of the mesh used to discretize the problem, even if y, x, and x_i are gridpoints. Hence, evaluating $\eta_b(z_i)$ requires determining which mesh cell contains z_i, then evaluating finite element basis functions on that mesh cell. Fortunately, the mesh cell containing z_i was determined as a side effect of querying the AABB tree of mesh cells when we checked whether $z_i \in \Omega$.

5.4. Hierarchical matrix construction. We form our H-matrix approximation $\mathbf{A}_H \approx \mathbf{A}$ by forming H-matrix representations $\mathbf{\Phi}_H$ and $\mathbf{M}_H$ of $\mathbf{\Phi}$ and $\mathbf{M}$, respectively, then using fast H-matrix methods to multiply these matrices per (3.5) to form $\mathbf{A}_H = \mathbf{M}_H\mathbf{\Phi}_H\mathbf{M}_H$. We use H1 matrices in our numerical results, but any other H-matrix format could be used instead. For more details on H-matrices, see [39].

We form $\mathbf{\Phi}_H$ using the standard geometrical clustering/adaptive cross method implemented within the HLIBpro software package [48]. For details about the algorithms used for geometrical clustering, H-matrix construction, and H-matrix operations in HLIBpro, we recommend the the following papers [11, 36, 49]. Although $\mathbf{\Phi}$ is a dense $N \times N$ matrix, constructing $\mathbf{\Phi}_H$ only requires evaluation of $O(k_h^2 N \log N)$ kernel entries $\mathbf{\Phi}_{ij} = \widetilde{\Phi}(\zeta_i, \zeta_j)$, and these entries are computed via the radial basis function interpolation method described in Section 5.3. Here k_h is the rank of the highest rank block in the H-matrix. A dense representation of $\mathbf{\Phi}$ is never formed. We form $\mathbf{M}_H$ using standard H-matrix methods for sparse matrices implemented within HLIBpro, using the same geometrical clustering block cluster tree as was used for $\mathbf{\Phi}$.

6. Computational cost. The computational cost of our method may be divided into the costs to perform the following tasks: (1) Computing impulse moments and batches; (2) Building the H-matrix; (3) Performing linear algebra operations with the H-matrix. This may optionally include modifying the H-matrix to make it symmetric positive semi-definite. In practice, (1) is the dominant cost, because (1) is the only task that requires applying $\mathcal{A}$ to vectors, and our method is targeted at applications in which applying $\mathcal{A}$ to a vector requires an expensive computational procedure such as solving a large scale PDE. All operations that do not require applications of $\mathcal{A}$ to vectors are nearly linear, and therefore scalable, in the size of the problem, N. We now describe these costs in detail. For convenience, Table 1 lists variable symbols and their approximate sizes.

(1) Computing impulse response moments and batches. Computing V, μ, and Σ requires applying $\mathcal{A}$ to 1, d, and $d(d+1)/2$ vectors, respectively. Computing each η_i requires applying $\mathcal{A}$ to one vector, so computing $\{\eta_i\}_{i=1}^{n_b}$ requires n_b operator applies. In total, computing all impulse response moments and batches therefore requires

$$1 + d + d(d+1)/2 + n_b \qquad \text{operator applies.}$$

In a typical application one might have $d = 2$ and $n_b = 5$, in which case a modest 11 operator applies are required.

Computing the impulse response batches also requires choosing sample point batches via the greedy ellipsoid packing algorithm described in Section 5.1. Choosing the ith batch of sample points may require performing up to $N|S_i|$ ellipsoid intersection tests, where $|S_i|$ is the number of sample points in the ith batch. Choosing all of the sample points therefore requires performing at most

$$Nm \qquad \text{ellipsoid intersection tests,}$$

where m is the total number of sample points in all batches. The multiplicative dependence of N with m is undesirable since m may be large, and reducing this cost is a target for future work. However, from a practical perspective, the cost for picking sample points is small compared to other parts of the approximation algorithm, so we do not pursue these improvements in this paper.

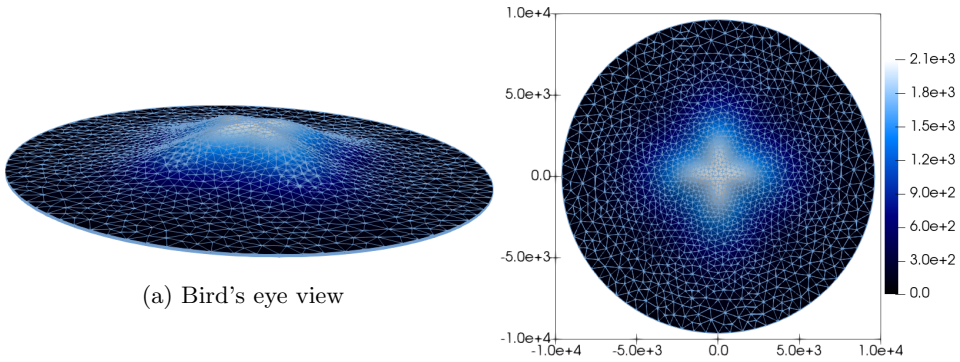
(2) Building the H-matrix. The classical H-matrix construction technique requires evaluating $O(k_h^2 N \log N)$ matrix entries of the approximation, where k_h is the H-matrix rank, i.e, the maximum rank among the blocks of the H-matrix. To evaluate one matrix entry, first one must find the k_n nearest sample points to a given point, where k_n is the number of impulse responses used in the RBF interpolation. This is done using a precomputed kd-tree of sample points, and requires $O(k_n \log m)$ floating point and logical elementary operations. Second, one must find the mesh cells that the points $\{z_i\}_{i=1}^{k_n}$ reside. This is done using an AABB tree of mesh cells, and requires $O(k_n \log N)$ elementary operations. Third, one must evaluate finite element basis functions on each of those cells, which requires $O(k_n)$ elementary operations. Finally, one must solve a $k_n \times k_n$ linear system to perform the radial basis function interpolation, which requires $O(k_n^3)$ elementary operations, which typically is negligible. Therefore, building the H-matrix requires

$$O\left(\left(k_h^2 N \log N\right)\left(k_n \log N + k_n^3\right)\right) \qquad \text{elementary operations.}$$

(3) Performing linear algebra operations with the H-matrix. It is well known that H-matrix methods for matrix-vector products, matrix-matrix addition, matrix-matrix multiplication, matrix factorization, and matrix inversion, and low rank updates require

$$O(k_h^2 N \log(N)^a) \qquad \text{elementary operations,} \tag{6.1}$$

where $a \in \{0, 1, 2, 3\}$ [35]. The precise cost depends on the type of H-matrix used and the operation being performed. In the numerical results (Section 7), we use one matrix-matrix addition to add the H-matrix approximation of the data misfit term in the Hessian to the regularization term in the Hessian. Constructing an H-matrix representation of the regularization term in the Hessian requires one matrix inversion and two matrix-matrix multiplications. Then, we use one matrix factorization to form



(a) Bird's eye view

(b) Top down view

Fig. 3: The ice sheet, discretized by a mesh of tetrahedra. Color indicates the height of the base of the ice sheet (i.e., the mountain topography). The radius of the domain is 10^4 meters, the maximum height of the mountain is 2.1×10^3 meters, and the average thickness of the ice sheet is 250 meters.

a preconditioner for the Newton linear system. The rational method for modifying $\mathbf{A}_H$ to be positive semi-definite, described in Appendix B, is performed using two matrix-matrix additions, $l+1$ matrix-matrix multiplications, and one matrix inversion, where 2^l is the rational function order.

7. Numerical results.

7.1. Example 1: Inversion for the basal sliding coefficient in an ice sheet flow problem. We use our method to approximate the data misfit Gauss-Newton Hessian in an inverse problem governed by a PDE which models steady state ice sheet flow [61]. Briefly, a sheet of ice is flowing down a mountain (see Figure 3). Given observations of the tangential component of the ice velocity on the top surface of the ice, we invert for the logarithm of the unknown spatially varying basal sliding parameter field, which governs the resistance to sliding along the base of the ice sheet. The inverse problem considered here is synthetic, but in practice surface velocity observations are available from satellites.

The domain that is filled with ice is denoted by $U \subset \mathbb{R}^3$. The basal, lateral and top parts of ∂U are denoted by Γ_b, Γ_l, and Γ_t, respectively. The governing equations are the linear incompressible Stokes equations,

$$-\nabla \cdot \sigma = f \quad \text{in } U, \tag{7.1a}$$

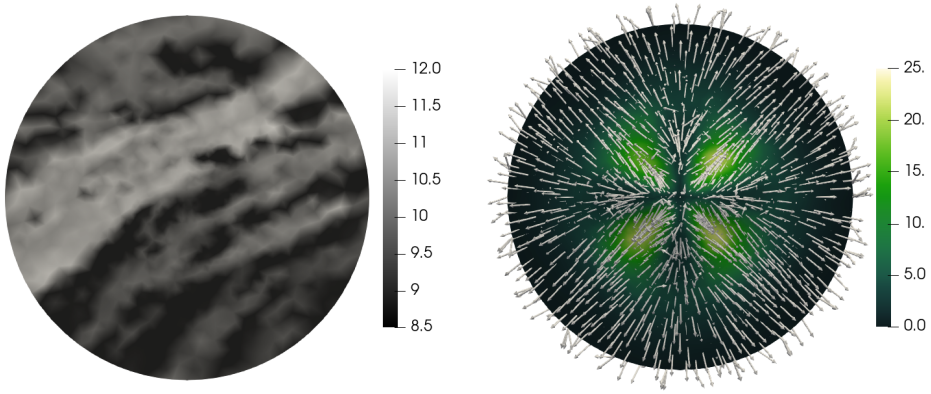
$$\nabla \cdot v = 0 \quad \text{in } U, \tag{7.1b}$$

$$\sigma\nu = 0 \quad \text{on } \Gamma_t, \tag{7.1c}$$

$$v \cdot \nu = 0 \quad \text{and} \quad T\left(\sigma\nu + \exp(q)\, v\right) = 0 \quad \text{on } \Gamma_b, \tag{7.1d}$$

$$\sigma\nu + sv = 0 \quad \text{on } \Gamma_l. \tag{7.1e}$$

The solution to these equations is the pair (v, p), where v is the ice flow velocity field, and p is the pressure field. Here, q is the unknown logarithmic basal sliding field



(a) q_{true}. (b) v_{true}.

Fig. 4: True parameter, q_{true}, (left) and true velocity v_{true} (right). In the right plot, arrows indicate the direction of v_{true} and color indicates the magnitude of v_{true}.

(note that large q corresponds to high resistance to sliding) which is defined on the 2D surface Γ_b. The quantity f is the body force density due to gravity, $s = 10^6$ is a Robin boundary condition constant, ν is the outward normal and T is the tangential projection operator that restricts a vector field to its tangential component along the boundary. We employ Glen's flow law [34], $\sigma = 2\eta\dot{\varepsilon} - Ip$, which is a constitutive law for ice that relates the stress tensor, σ, to the strain rate tensor, $\dot{\varepsilon} = \frac{1}{2}\left(\nabla v + \nabla v^\top\right)$. Here η is the effective viscosity and I is the unit matrix. Glen's exponent has been chosen to be one, which provides for a linear Stokes model. Note that while the PDE is linear, the parameter-to-solution map, $q \mapsto (v, p)$, is nonlinear.

The pressure, p, is discretized with first order scalar continuous Galerkin finite elements defined on a mesh of tetrahedra. The velocity, v, is discretized with second order continuous Galerkin finite elements on the same mesh. The parameter q is discretized with first order scalar continuous Galerkin finite elements on the mesh of triangles that results from restricting the tetrahedral mesh to the basal boundary, Γ_b. Note that Γ_b is a 2D surface embedded in 3D due to the mountain topography. We also generate a flattened version of Γ_b, denoted by $\Omega \subset \mathbb{R}^2$, by ignoring the height coordinate. The parameter q is dually viewed as a function on Γ_b for the purpose of solving the Stokes equations, and as a function on Ω for the purpose of building our Hessian approximations and defining the regularization. The observations are generated by adding 1% multiplicative Gaussian noise to the tangential component of the velocity field restricted to the top surface of the geometry. The true basal sliding coefficient field and the true velocity field, which is obtained by solving (7.1), are shown in Figure 4.

To reconstruct the basal sliding parameter field q, we formulate an inverse problem governed by the Stokes problem (7.1). In particular this problem is formulated as a nonlinear least squares optimization problem, whose cost function consists of a misfit term (between the observations and model output) and a bi-Laplacian regularization term following [45]. The regularization is centered at the constant function $q_0(x) = 10.5$. To mitigate boundary effects we use a constant coefficient Robin boundary condition as in [62]. The parameters are chosen so that the Green's function of

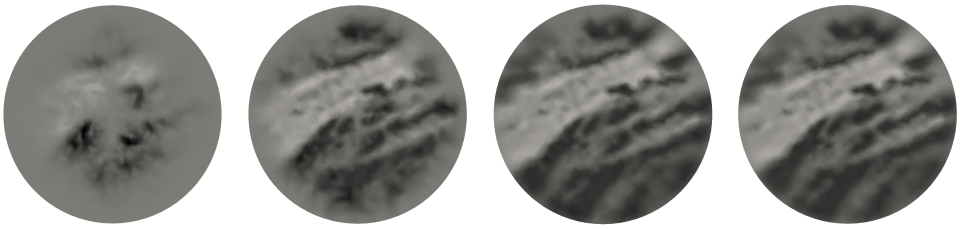


Fig. 5: The log basal sliding parameter after 1 (left), 3 (second from left), and 4 (third from left) Newton iterations, and the final Newton iterate (right). The PSF (5) preconditioner is constructed between Newton iterations 3 and 4.

the Hessian of the regularization has a characteristic length of 0.25 of the domain radius. For the specific setup, we refer the reader to [69, Section 2.2]. We choose the regularization parameter, γ, that controls the overall strength of the regularization using Morozov's discrepancy principle. The optimal value of γ is 7.3×10^3, which is used for all the numerical results except for Figure 6b, where we vary γ to study how changing γ impacts the effectiveness of the proposed preconditioner.

We solve the inverse problem with an inexact Newton preconditioned conjugate gradient (PCG) scheme and a globalizing Armijo line search [59]. The (discrete) Newton updates, $\widehat{\boldsymbol{q}}_k$, are obtained by solving

$$\boldsymbol{H}\widehat{\boldsymbol{q}}_k = -\boldsymbol{g}_k \qquad \text{or} \qquad \boldsymbol{H}_{\text{gn}}\widehat{\boldsymbol{q}}_k = -\boldsymbol{g}_k, \tag{7.2}$$

wherein we choose the initial guess as the discretization of the constant function q_0. Here $\boldsymbol{g}_k$, $\boldsymbol{H}$ and $\boldsymbol{H}_{\text{gn}}$ are the discretized gradient, Hessian, and Gauss-Newton Hessian of the inverse problem cost function, respectively. To ensure positive definiteness of the Hessian we use $\boldsymbol{H}_{\text{gn}}$ for the first five iterations, and $\boldsymbol{H}$ for all subsequent iterations. The Newton iterations are terminated when $||\boldsymbol{g}_k||/||\boldsymbol{g}_0|| < 10^{-8}$. Systems (7.2) are solved inexactly using an inner PCG iteration, which terminates when the norm of the linear system residual is less than $\min\left(0.5, \sqrt{||\boldsymbol{g}_k||}\right)||\boldsymbol{g}_k||$.

We use the framework described in this paper to generate Hessian preconditioners. We build H-matrix approximations of the data misfit Gauss-Newton Hessian (the term in $\boldsymbol{H}_{\text{gn}}$ that arises from the data misfit) using 1, 5, and 25 batches. These approximations are denoted by PSF (1), PSF (5), and PSF (25), respectively. The resulting H-matrices, which may be indefinite due to approximation error, are modified to be symmetric positive definite via the procedure described in Appendix B, with $l = 2$. We approximate $\boldsymbol{H}_{\text{gn}}$ rather than $\boldsymbol{H}$ because $\boldsymbol{H}$ more often has negative values in its integral kernel, especially when one is away from the optimal basal sliding parameter. Our results show that our preconditioners are also good preconditioners for $\boldsymbol{H}$. The H-matrix approximation of the data misfit Gauss-Newton Hessian is added to an H-matrix approximation of the Hessian of the regularization term to form an H-matrix approximation of $\boldsymbol{H}_{\text{gn}}$. We note that the Hessian of the regularization is a combination of stiffness and mass matrices, which are sparse. Therefore, we form H-matrix representations of these matrices and combine them using standard sparse H-matrix techniques. We factor the overall Gauss-Newton Hessian approximation, denoted $\widetilde{\boldsymbol{H}}$, using fast H-matrix methods, then use the factorization as a preconditioner.

Table 2 shows the performance of our preconditioner for accelerating the solution of optimization problem (**??**). We build the PSF (5) preconditioner in the third

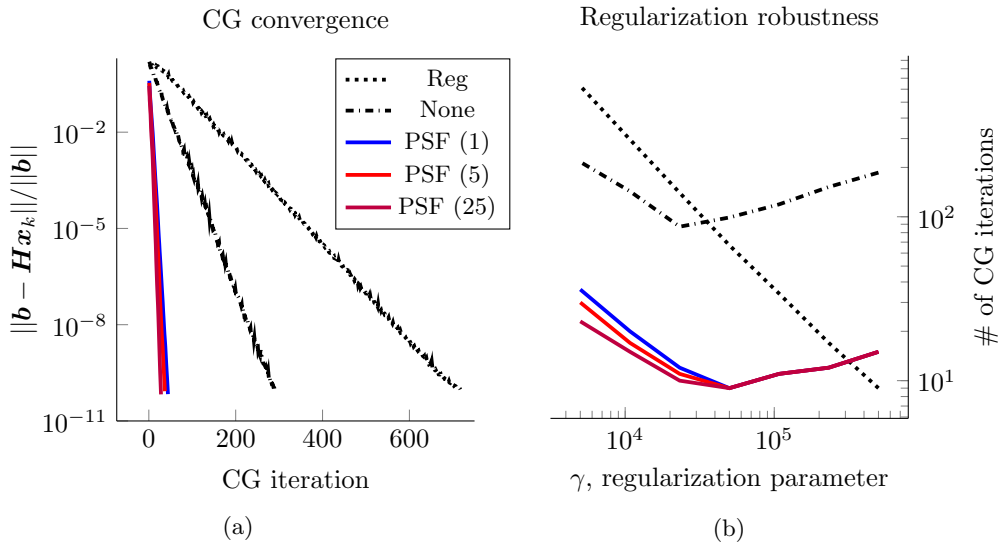
	PSF (5)			REG			NONE		
Iter	#CG	#Stokes	$\|\|\boldsymbol{g}\|\|$	#CG	#Stokes	$\|\|\boldsymbol{g}\|\|$	#CG	#Stokes	$\|\|\boldsymbol{g}\|\|$
0	1	4	2.0e+7	1	4	2.0e+7	1	4	2.0e+7
1	2	6	5.8e+6	2	6	7.9e+6	2	6	5.8e+6
2	4	10	2.4e+6	5	12	3.9e+6	4	10	2.4e+6
3	2	6+22	5.8e+5	13	28	1.6e+6	13	28	5.8e+5
4	5	12	5.6e+4	42	86	4.8e+5	33	68	9.1e+4
5	10	22	3.5e+3	84	170	7.7e+4	57	116	6.2e+3
6	14	30	2.7e+1	125	252	5.0e+3	76	154	1.1e+2
7	0	2	4.1e-2	194	390	7.9e+1	117	236	2.3e-1
8	—	—	—	0	2	1.6e-1	0	2	4.1e-2
Total	38	114	—	466	950	—	303	624	—

Table 2: Convergence history for solving the Stokes inverse problem using preconditioned inexact Newton PCG. Preconditioners shown are our method with five batches (PSF (5)) constructed at the third iteration, regularization preconditioning (REG), and no preconditioning (NONE). Columns titled #CG show the number of PCG iterations used to solve the Newton system for $\widehat{\boldsymbol{q}}_k$. Columns titled $||\boldsymbol{g}||$ show the l^2 norm of the gradient at $\boldsymbol{q}_k$. Columns titled #Stokes show the total number of Stokes PDE solves performed in each Newton iteration. This consists of Stokes solves for u performed during the linesearch going from $\boldsymbol{q}_{k-1}$ to $\boldsymbol{q}_k$, plus one adjoint Stokes solve to compute the gradient at $\boldsymbol{q}_k$, plus one incremental forward and one incremental adjoint Stokes solve per PCG iteration for solving the Newton system. In the PSF (5) portion of row 3, we write $6+22$ to indicate that 6 Stokes solves were used during the standard course of the iteration, and 22 Stokes solves were used to build the PSF (5) preconditioner.

Gauss-Newton iteration, and reuse it for all subsequent Gauss-Newton and Newton iterations. No preconditioning is used in the Gauss-Newton iterations before the PSF (5) preconditioner is built. We compare our method with the most commonly used existing preconditioners: no preconditioning (NONE), and preconditioning by the regularization term in the Hessian (REG). Using PSF (5) reduces the total number of Stokes PDE solves by a factor of six compared to no preconditioning, and by a factor of nine compared to regularization preconditioning[5]. In Figure 5 we show select Newton iterates using our PSF (5) preconditioner.

Next, we build PSF (1), PSF (5), and PSF (25) preconditioners based on the Gauss-Newton Hessian evaluated at the converged solution $\boldsymbol{q}$. We use PCG to solve a linear system with the Hessian as the coefficient operator and a right hand vector with random i.i.d. entries drawn from the standard normal distribution. In Figure 6a we compare the convergence of PCG for solving this linear system using the PSF (1), PSF (5), PSF (25), REG, and NONE preconditioners. PCG converges fastest with the PSF preconditioners, with PSF (25) converging fastest, followed by PSF (5), followed by PSF (1), as expected. PCG converges much slower with no preconditioning and regularization preconditioning than it does with PSF preconditioning, with no preconditioning outperforming regularization preconditioning. In Figure 6b,

[5]Interestingly, we observe that no preconditioning outperforms regularization preconditioning here because the noise level (and hence γ) is small.

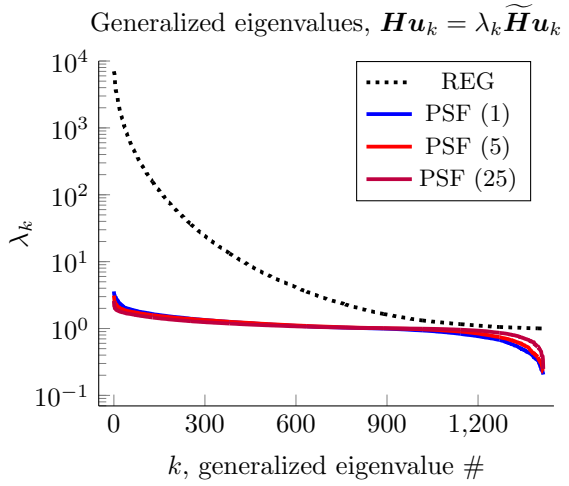


(a) (b)

Fig. 6: Left (6a) shows the convergence history for solving $\boldsymbol{H}\boldsymbol{x} = \boldsymbol{b}$ using PCG, where $\boldsymbol{b}$ has i.i.d. random entries drawn from the standard normal distribution. Here we use $\gamma = 7.3 \times 10^3$, which is determined by the Morozov discrepancy principle. Right (6b) shows the number of PCG iterations required to achieve $||\boldsymbol{b} - \boldsymbol{H}\boldsymbol{x}_k||/||\boldsymbol{b}|| < 10^{-6}$, where $\boldsymbol{x}_k$ is the kth PCG iterate, for a range of different γ. Results in these figures are shown for our preconditioner with 1, 5, and 25 batches (PSF (1), PSF (5), and PSF(25), respectively), regularization preconditioning (REG), and no preconditioning (NONE). The preconditioner is constructed using $\boldsymbol{H}_{\text{gn}}$. For a fair comparison, even as γ changes, $\boldsymbol{H}$ and $\boldsymbol{H}_{\text{gn}}$ are always evaluated at the same $\boldsymbol{q}$, which is the optimal point for the inverse problem with $\gamma = 7.3 \times 10^3$.

we perform PCG solves on linear systems of the same form, except now we vary γ. The performance of our PSF preconditioners is good and relatively stable over a wide range of γ's. All PSF preconditioners perform the same for medium and large values of γ. For small values of γ PSF (25) performs slightly better than PSF (5), which performs slightly better than PSF (1). As expected, regularization preconditioning performs well for large γ and poorly for small γ. Our PSF preconditioners outperform no preconditioning and regularization preconditioning for medium and small γ, and perform similarly to regularization preconditioning for large γ.

In Figure 7a we show the generalized eigenvalues for the generalized eigenvalue problem $\boldsymbol{H}\boldsymbol{u} = \lambda\widetilde{\boldsymbol{H}}\boldsymbol{u}$. Here $\boldsymbol{H}$ is the Hessian evaluated at the reconstructed parameter $\boldsymbol{q}$ for γ chosen to satisfy the Morozov discrepancy principle. The matrix $\widetilde{\boldsymbol{H}}$ is one of the PSF (1), PSF (5), or PSF (25) Gauss-Newton Hessian approximations constructed at that $\boldsymbol{q}$, or the regularization Hessian. The PSF preconditioners cluster the eigenvalues of the Hessian near one, with more batches yielding better clustering. The regularization preconditioner clusters the trailing eigenvalues but amplifies leading eigenvalues. In Figure 7b we show the condition numbers of $\widetilde{\boldsymbol{H}}^{-1}\boldsymbol{H}$ and $\widetilde{\boldsymbol{H}}^{-1}\boldsymbol{H}_{\text{gn}}$. The PSF preconditioners yield the the smallest condition numbers, with more batches yielding smaller condition numbers. Regularization preconditioning and no precondi-



(a)

	$\kappa(\widetilde{\boldsymbol{H}}^{-1}\boldsymbol{H})$	$\kappa(\widetilde{\boldsymbol{H}}^{-1}\boldsymbol{H}_{\text{gn}})$
REG	7.02e+3	7.02e+3
NONE	8.71e+2	9.22e+2
PSF (1)	1.75e+1	1.76e+1
PSF (5)	1.39e+1	1.39e+1
PSF (25)	1.03e+1	1.06e+1

(b)

Fig. 7: Left (7a): eigenvalues for generalized eigenvalue problem $\boldsymbol{H}\boldsymbol{u}_k = \lambda_k \widetilde{\boldsymbol{H}}\boldsymbol{u}_k$. Here $\boldsymbol{H}$ is the Hessian and $\widetilde{\boldsymbol{H}}$ is the PSF approximation constructed using the Gauss-Newton Hessian, $\boldsymbol{H}_{\text{gn}}$, for 1, 5, and 25 batches (PSF (1), PSF (5), and PSF (25), respectively), or the regularization Hessian (REG). Right (7b): condition number for $\widetilde{\boldsymbol{H}}^{-1}\boldsymbol{H}$ and $\widetilde{\boldsymbol{H}}^{-1}\boldsymbol{H}_{\text{gn}}$ for these different preconditioners, and no preconditioner (NONE). All operators are evaluated at the $\boldsymbol{q}$ that solves the inverse problem for $\gamma = 7.3 \times 10^3$, as determined by the Morozov discrepancy principle. COND(H) LABELS!

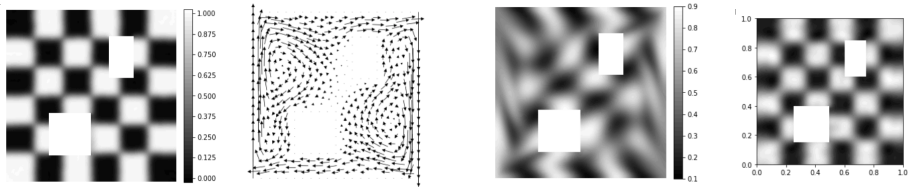


Fig. 8: (a) true initial condition. (b) velocity field. (c) observed concentration at final time. (d) reconstruction of initial condition. black=0, white=1

tioning yield larger condition numbers. The condition numbers are similar for both $\widetilde{\boldsymbol{H}}^{-1}\boldsymbol{H}$ and $\widetilde{\boldsymbol{H}}^{-1}\boldsymbol{H}_{\text{gn}}$, demonstrating that the preconditioner built based on $\boldsymbol{H}_{\text{gn}}$ is a good preconditioner for $\boldsymbol{H}$.

7.2. Example 2: Inversion for the initial condition in an advection-diffusion problem.

- Picture of velocity field, initial condition, final condition, reconstruction (1) (OKOK)
- Impulse responses: vary time and Peclet number (3) (OKOK)
- CG convergence (1) CG iters vs Peclet number (Morozov) (2) (OKOK)

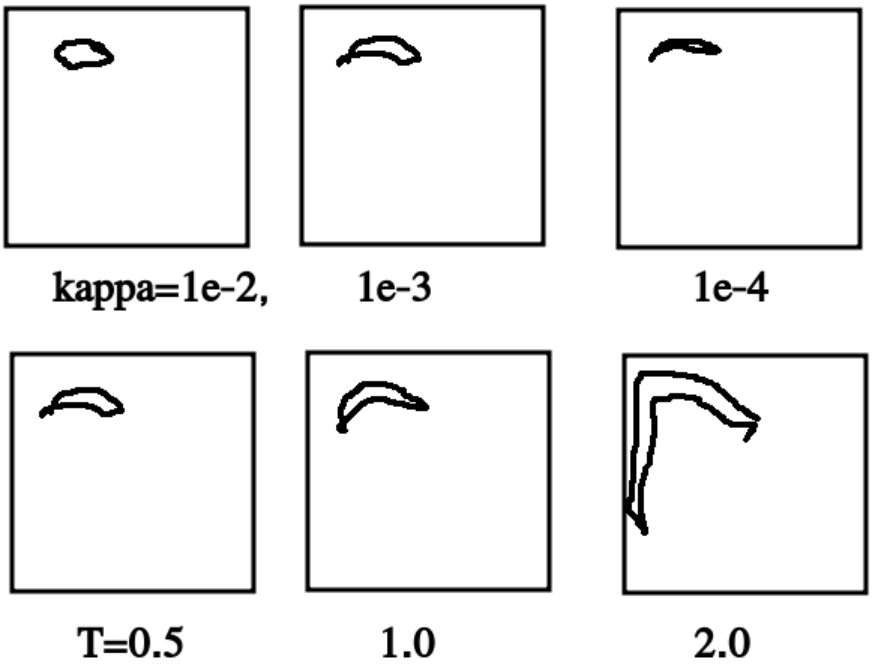


Fig. 9: Impulse responses for various diffusivities kappa and fixed simulation time Tf=0.5 (top) and simulation times with fixed kappa=1e-3 (bot). Colorbar is normalized independently for each subfigure

- Preconditioned spectrum, condition numbers? (1)

8. Conclusions. We presented an efficient matrix-free PSF method for approximating operators with locally supported non-negative integral kernels. The method only requires access to the operator via application of the operator to a small number of vectors. The idea of the method is to compute batches of impulse responses by applying the operator to Dirac combs of scattered point sources, then interpolate these impulse responses to approximate entries of the operator's integral kernel. The interpolation is based on a new principle we call "local mean displacement invariance," which generalizes and improves upon classical local translation invariance. The ability to quickly approximate arbitrary integral kernel entries permits us to form an H-matrix approximation of the operator. Fast H-matrix arithmetic is then used to perform further linear algebra operations that cannot be performed easily with the original operator, such as matrix factorization and inversion. The supports of the impulse responses are estimated to be contained in ellipsoids, which are determined a-priori via a new procedure that involves applying the operator to a small number of polynomial functions. Point source locations for the impulse response batches are chosen using a greedy ellipsoid packing procedure, in which we choose as many impulse responses per batch as possible, while ensuring that the corresponding ellipsoids do not overlap. We applied the method to approximate the Gauss-Newton Hessian in a Stokes ice sheet inverse problem, and saw that the approximation substantially outper-

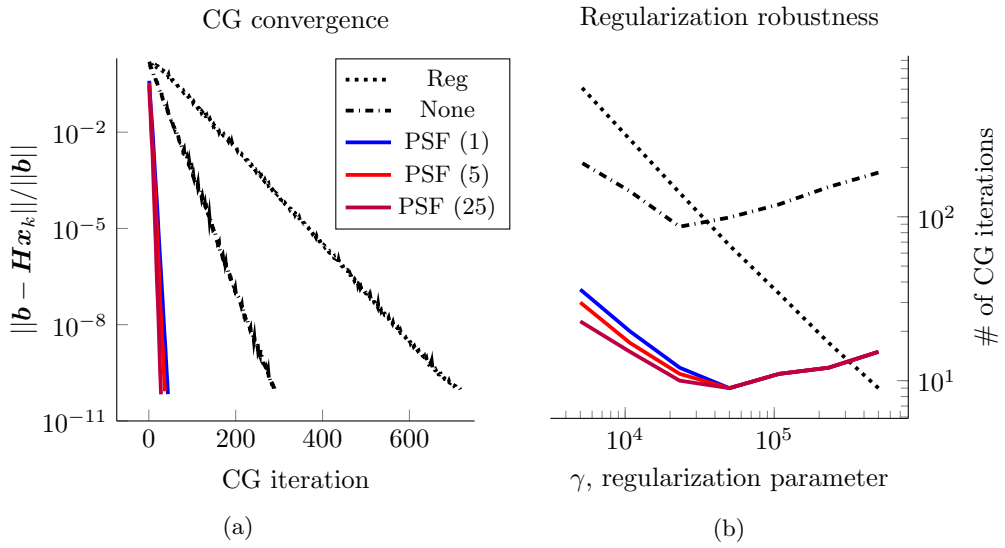


Fig. 10: Left (10a) shows the convergence history for solving $\boldsymbol{H}\boldsymbol{x} = \boldsymbol{b}$ using PCG, where $\boldsymbol{b}$ has i.i.d. random entries drawn from the standard normal distribution. BLAH BLAH BLAH Here we use $\gamma = 9999 \times 10^9$, which is determined by the Morozov discrepancy principle. Right (10b) shows the number of PCG iterations required to achieve $||\boldsymbol{b} - \boldsymbol{H}\boldsymbol{x}_k||/||\boldsymbol{b}|| < 10^{-6}$, where $\boldsymbol{x}_k$ is the kth PCG iterate, for a range of different κ. Results in these figures are shown for our preconditioner with 5, 15, and 30 batches (PSF (5), PSF (15), and PSF(30), respectively), regularization preconditioning (REG), and no preconditioning (NONE). The preconditioner is constructed using $\boldsymbol{H}_{\text{gn}}$.

forms existing Hessian approximation methods. Although the method we presented is not applicable to all Hessians, it is applicable to many Hessians of practical interest. For these Hessians, our method offers a *data scalable* alternative to conventional low rank approximation, because our method can form high rank approximations of an operator using a small number of operator applies, and thus forward and adjoint PDE solves.

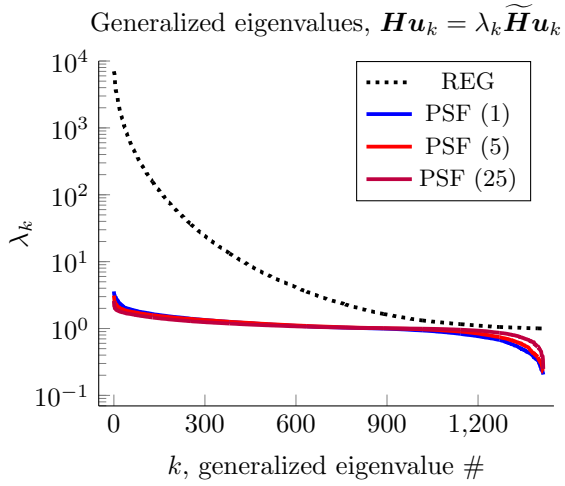
Appendix A. Ellipsoid intersection test. The procedure for choosing sample points relies on quickly determining whether two ellipsoids intersect. Let E_1 and E_2 be the ellipsoids defined as

$$E_i := \{x : (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) \leq \tau^2\}$$

for $i \in \{1, 2\}$, where $\mu_1, \mu_2 \in \mathbb{R}^d$, and $\Sigma_1, \Sigma_2 \in \mathbb{R}^{d \times d}$ are positive definite. Let K be the following one dimensional convex function:

$$K(s) := 1 - \frac{1}{\tau^2}(\mu_1 - \mu_2)^T \left(\frac{1}{1-s}\Sigma_1 + \frac{1}{s}\Sigma_2 \right)^{-1} (\mu_1 - \mu_2).$$

In [33] it is shown that $E_1 \cap E_2 = \{\}$ if and only if $K(s) < 0$ for some $s \in (0, 1)$.



(a)

	$\kappa(\widetilde{\boldsymbol{H}}^{-1}\boldsymbol{H})$
REG	7.02e+3
NONE	8.71e+2
PSF (5)	1.75e+1
PSF (15)	1.39e+1
PSF (30)	1.03e+1

(b)

Fig. 11: Left (11a): eigenvalues for generalized eigenvalue problem $\boldsymbol{H}\boldsymbol{u}_k = \lambda_k \widetilde{\boldsymbol{H}}\boldsymbol{u}_k$. Here $\boldsymbol{H}$ is the Hessian and $\widetilde{\boldsymbol{H}}$ is the PSF approximation constructed using the Gauss-Newton Hessian, $\boldsymbol{H}_{\text{gn}}$, for 5, 15, and 30 batches (PSF (5), PSF (15), and PSF (30), respectively), or the regularization Hessian (REG). Right (11b): condition number for $\widetilde{\boldsymbol{H}}^{-1}\boldsymbol{H}$ and $\widetilde{\boldsymbol{H}}^{-1}\boldsymbol{H}_{\text{gn}}$ for these different preconditioners, and no preconditioner (NONE). COND(H) LABELS!

The function $K(s)$ may be evaluated quickly for many s by computing the solution to the generalized eigenvalue problem $\Sigma_1 P = \Sigma_2 P \Lambda$, where $P \in \mathbb{R}^{d\times d}$ is the matrix of generalized eigenvectors, and $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d)$ is the diagonal matrix of generalized eigenvalues λ_i. After some algebraic manipulations, we have

$$K(s) = 1 - \frac{1}{\tau^2}\sum_{i=1}^{d} \frac{s(1-s)}{1+s(\lambda_i - 1)} v_i^2, \tag{A.1}$$

where $v := P^T(\mu_1 - \mu_2)$. To check whether E_1 and E_2 intersect, we compute the generalized eigenvalue decomposition of Σ_1 and Σ_2, form v, then minimize $K(s)$ in the form (A.1) on $(0,1)$ using Brent's method [12]. If $K(s^*) < 0$ at the minimizer, s^*, then $E_1 \cap E_2 = \{\}$. Otherwise $E_1 \cap E_2 \neq \{\}$.

This ellipsoid intersection test is only performed if the coordinate axis aligned bounding boxes for the ellipsoids intersect. The smallest coordinate axis aligned box that contains E_i is the box $\prod_{k=1}^{d}\left[\mu_i^k - \tau\sqrt{\Sigma_i^{kk}}, \mu_i^k + \tau\sqrt{\Sigma_i^{kk}}\right]$, where μ_i^k is the kth entry of μ_i, and Σ_i^{kk} is the kth diagonal entry of Σ_i[6]. The boxes $\prod_{k=1}^{d}\left[a_-^k, a_+^k\right]$ and $\prod_{k=1}^{d}\left[b_-^k, b_+^k\right]$ intersect if and only if $a_-^k \leq b_+^k$ and $b_-^k \leq a_+^k$ for $k = 1, \ldots, d$.

Appendix B. Rational positive semi-definite modification. In many problems $\mathcal{A}$ is symmetric positive semi-definite. However, $\mathbf{A}_H$ is non-symmetric and

[6] See https://math.stackexchange.com/q/3928964

indefinite because of approximation error. This is undesirable. Symmetry and positive semi-definiteness are important properties which should be preserved if possible. Also, lacking these properties may prevent one from using highly effective algorithms, such as the conjugate gradient method, to perform further operations involving $\mathbf{A}_H$.

If $\mathcal{A}$ is symmetric, we symmetrize $\mathbf{A}_H$ as follows: $\mathbf{A}_H^{\text{sym}} := \frac{1}{2}\left(\mathbf{A}_H + \mathbf{A}_H^T\right)$. If $\mathcal{A}$ is positive semi-definite, we adapt the rational method from [40, Section 13.8] to approximate the matrix absolute value of $\mathbf{A}_H^{\text{sym}}$, which we denote by $|\mathbf{A}_H^{\text{sym}}|$. That is, $|\mathbf{A}_H^{\text{sym}}|$ has the same eigenvectors as $\mathbf{A}_H^{\text{sym}}$, but the eigenvalues of $|\mathbf{A}_H^{\text{sym}}|$ are the absolute values of the eigenvalues of $\mathbf{A}_H^{\text{sym}}$. Once constructed, we use the approximation of $|\mathbf{A}_H^{\text{sym}}|$ in place of $\mathbf{A}_H$ in further linear algebra operations.

In detail, let a be any scalar satisfying $-a < \lambda_{\min}$, where $\lambda_{\min}$ is the smallest eigenvalue of $\mathbf{A}_H^{\text{sym}}$, and set

$$\mathbf{T} := \left(2\mathbf{A}_H^{\text{sym}} + a\mathbf{I}\right)/a, \tag{B.1}$$

From the results in [40, Section 13.8] it is straightforward to show that

$$\mathbf{\Pi}_l := \left(\mathbf{I} + \mathbf{T}^{2^l}\right)^{-1} \tag{B.2}$$

approximates the spectral projector onto the eigenspace of $\mathbf{A}_H^{\text{sym}}$ corresponding to negative eigenvalues. Thus, we have

$$|\mathbf{A}_H^{\text{sym}}| \approx \left(\mathbf{I} - 2\mathbf{\Pi}_l\right)\mathbf{A}_H^{\text{sym}}. \tag{B.3}$$

To form an approximation of $|\mathbf{A}_H^{\text{sym}}|$, first, we compute $\lambda_{\min}$ using the Lanczos method, and set $a = \gamma\lambda_{\min}$, where $\gamma \geq 1$ is a parameter that must be chosen. The method is relatively insensitive to γ and numerically we find that the method works well for any $\gamma \in [1, 2]$. We use $\gamma = 1.5$ in our numerical results. Next, we compute $\mathbf{T}$ per (B.1), compute $\mathbf{T}^{2^l}$ per (B.2) via repeated squaring, compute $\mathbf{\Pi}_l$ per (B.2), and form an approximation of $|\mathbf{A}_H^{\text{sym}}|$ per (B.3). Fast H-matrix methods are used to perform the required matrix-matrix additions and multiplications, and matrix inversion. We recommend using $l = 1$ or 2. Larger l is unnecessary and may make the method unstable.

Acknowledgments. We thank J.J. Alger, Longfei Gao, Mathew Hu, and Rami Nammour for helpful discussions. We thank Trevor Heise for editing suggestions. We thank Georg Stadler for domain geometry help.

REFERENCES

[1] H.-M. Adorf, *Towards HST restoration with a space-variant PSF, cosmic rays and other missing data*, in The Restoration of HST Images and Spectra-II, 1994, p. 72.

[2] V. Akçelik, G. Biros, A. Drăgănescu, O. Ghattas, J. Hill, and B. van Bloeman Waanders, *Dynamic data-driven inversion for terascale simulations: Real-time identification of airborne contaminants*, in Proceedings of SC2005, Seattle, 2005.

[3] A. Alexanderian, P. J. Gloor, and O. Ghattas, *On Bayesian A-and D-optimal experimental designs in infinite dimensions*, Bayesian Analysis, 11 (2016), pp. 671–695.

[4] N. Alger, *Data-scalable Hessian preconditioning for distributed parameter PDE-constrained inverse problems*, PhD thesis, The University of Texas at Austin, 2019.

[5] N. Alger, V. Rao, A. Myers, T. Bui-Thanh, and O. Ghattas, *Scalable matrix-free adaptive product-convolution approximation for locally translation-invariant operators*, SIAM Journal on Scientific Computing, 41 (2019), pp. A2296–A2328.

[6] N. Alger, U. Villa, T. Bui-Thanh, and O. Ghattas, *A data scalable augmented Lagrangian KKT preconditioner for large-scale inverse problems*, SIAM Journal on Scientific Computing, 39 (2017), pp. A2365–A2393.

[7] I. Ambartsumyan, W. Boukaram, T. Bui-Thanh, O. Ghattas, D. Keyes, G. Stadler, G. Turkiyyah, and S. Zampini, *Hierarchical matrix approximations of Hessians arising in inverse problems governed by PDEs*, SIAM Journal on Scientific Computing, 42 (2020), pp. A3397–A3426.
[8] R. C. Aster, B. Borchers, and C. H. Thurber, *Parameter estimation and inverse problems*, Elsevier, 2018.
[9] J. L. Bentley, *Multidimensional binary search trees used for associative searching*, Communications of the ACM, 18 (1975), pp. 509–517.
[10] J. Bigot, P. Escande, and P. Weiss, *Estimation of linear operators from scattered impulse responses*, Applied and Computational Harmonic Analysis, 47 (2019), pp. 730–758.
[11] S. Börm, L. Grasedyck, and W. Hackbusch, *Introduction to hierarchical matrices with applications*, Engineering analysis with boundary elements, 27 (2003), pp. 405–422.
[12] R. P. Brent, *An algorithm with guaranteed convergence for finding a zero of a function*, The computer journal, 14 (1971), pp. 422–425.
[13] T. Bui-Thanh, O. Ghattas, J. Martin, and G. Stadler, *A computational framework for infinite-dimensional Bayesian inverse problems Part I: The linearized case, with application to global seismic inversion*, SIAM Journal on Scientific Computing, 35 (2013), pp. A2494–A2523.
[14] J. Chen and M. L. Stein, *Linear-cost covariance functions for Gaussian random fields*, Journal of the American Statistical Association, (2021), pp. 1–18.
[15] H. Cheng, Z. Gimbutas, P.-G. Martinsson, and V. Rokhlin, *On the compression of low rank matrices*, SIAM Journal on Scientific Computing, 26 (2005), pp. 1389–1404.
[16] T. Cui, J. Martin, Y. M. Marzouk, A. Solonen, and A. Spantini, *Likelihood-informed dimension reduction for nonlinear inverse problems*, Inverse Problems, 30 (2014), p. 114015.
[17] L. Denis, E. Thiébaut, and F. Soulez, *Fast model of space-variant blurring and its application to deconvolution in astronomy*, in Image Processing (ICIP), 2011 18th IEEE International Conference on, IEEE, 2011, pp. 2817–2820.
[18] L. Denis, E. Thiébaut, F. Soulez, J.-M. Becker, and R. Mourya, *Fast approximations of shift-variant blur*, International Journal of Computer Vision, 115 (2015), pp. 253–278.
[19] J. Duchon, *Splines minimizing rotation-invariant semi-norms in Sobolev spaces*, in Constructive theory of functions of several variables, Springer, 1977, pp. 85–100.
[20] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of Inverse Problems*, vol. 375 of Mathematics and Its Applications, Springer Netherlands, 1996.
[21] C. Ericson, *Real-time collision detection*, Crc Press, 2004.
[22] P. Escande and P. Weiss, *Sparse wavelet representations of spatially varying blurring operators*, SIAM Journal on Imaging Sciences, 8 (2015), pp. 2976–3014.
[23] P. Escande and P. Weiss, *Approximation of integral operators using product-convolution expansions*, Journal of Mathematical Imaging and Vision, 58 (2017), pp. 333–348.
[24] P. Escande and P. Weiss, *Fast wavelet decomposition of linear operators through product-convolution expansions*, IMA Journal of Numerical Analysis, 42 (2022), pp. 569–596.
[25] P. Escande, P. Weiss, and F. Malgouyres, *Spatially varying blur recovery. Diagonal approximations in the wavelet domain*, (2012).
[26] A. Fichtner and T. v. Leeuwen, *Resolution analysis by random probing*, Journal of Geophysical Research: Solid Earth, 120 (2015), pp. 5549–5573.
[27] D. Fish, J. Grochmalicki, and E. Pike, *Scanning singular-value-decomposition method for restoration of images with space-variant blur*, JOSA A, 13 (1996), pp. 464–469.
[28] H. P. Flath, L. C. Wilcox, V. Akçelik, J. Hill, B. van Bloemen Waanders, and O. Ghattas, *Fast algorithms for Bayesian uncertainty quantification in large-scale linear inverse problems based on low-rank partial Hessian approximations*, SIAM Journal on Scientific Computing, 33 (2011), pp. 407–432.
[29] P. 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.
[30] M. Gentile, F. Courbin, and G. Meylan, *Interpolating point spread function anisotropy*, Astronomy & Astrophysics, 549 (2013).
[31] C. J. Geoga, M. Anitescu, and M. L. Stein, *Scalable gaussian process computations using hierarchical matrices*, Journal of Computational and Graphical Statistics, 29 (2020), pp. 227–237.
[32] O. Ghattas and K. Willcox, *Learning physics-based models from data: perspectives from inverse problems and model reduction*, Acta Numerica, 30 (2021), pp. 445–554.
[33] I. Gilitschenski and U. D. Hanebeck, *A robust computational test for overlap of two arbitrary-dimensional ellipsoids in fault-detection of Kalman filters*, in 2012 15th Inter-

national Conference on Information Fusion, IEEE, 2012, pp. 396–401.
[34] J. W. Glen, *The creep of polycrystalline ice*, Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences, 228 (1955), pp. 519–538.
[35] L. Grasedyck and W. Hackbusch, *Construction and arithmetics of H-matrices*, Computing, 70 (2003), pp. 295–334.
[36] L. Grasedyck, R. Kriemann, and S. Le Borne, *Parallel black box $\mathcal{H}$-LU preconditioning for elliptic boundary value problems*, Computing and visualization in science, 11 (2008), pp. 273–291.
[37] M. D. Gunzburger, *Perspectives in Flow Control and Optimization*, SIAM, Philadelphia, 2003.
[38] W. Hackbusch, *A sparse matrix arithmetic based on H-matrices. Part I: Introduction to H-matrices*, Computing, 62 (1999), pp. 89–108.
[39] W. Hackbusch, *Hierarchical matrices: algorithms and analysis*, vol. 49, Springer, 2015.
[40] W. Hackbusch and W. Kress, *A projection method for the computation of inner eigenvalues using high degree rational operators*, Computing, 81 (2007), pp. 259–268.
[41] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM review, 53 (2011), pp. 217–288.
[42] T. Hartland, G. Stadler, M. Perego, K. Liegeois, and N. Petra, *Hierarchical off-diagonal low-rank approximation of Hessians in inverse problems, with application to ice sheet model initializaiton*, 2023, doi:10.48550/ARXIV.2301.03644, https://arxiv.org/abs/2301.03644.
[43] T. J. Hughes, *The finite element method: linear static and dynamic finite element analysis*, Courier Corporation, 2012.
[44] T. Isaac, N. Petra, G. Stadler, and O. 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 (2015), pp. 348–368.
[45] T. Isaac, N. Petra, G. Stadler, and O. 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 (2015), pp. 348–368, doi:10.1016/j.jcp.2015.04.047.
[46] J. Kaipio and E. Somersalo, *Statistical and Computational Inverse Problems*, vol. 160 of Applied Mathematical Sciences, Springer-Verlag New York, 2005, doi:10.1007/b138659, https://link.springer.com/978-0-387-27132-3.
[47] K.-T. Kim, U. Villa, M. Parno, Y. Marzouk, O. Ghattas, and N. Petra, *hIPPYlib-MUQ: A bayesian inference software framework for integration of data with complex predictive models under uncertainty*, arXiv preprint arXiv:2112.00713, (2021).
[48] R. Kriemann, *HLIBpro user manual*, Max-Planck-Institute for Mathematics in the Sciences, Leipzig, 48 (2008).
[49] R. Kriemann, *H-LU factorization on many-core systems*, Computing and Visualization in Science, 16 (2013), pp. 105–117.
[50] P. Le Tallec and A. 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 (1997), pp. 361–379.
[51] J. Levitt and P.-G. Martinsson, *Linear-complexity black-box randomized compression of hierarchically block separable matrices*, arXiv preprint arXiv:2205.02990, (2022).
[52] L. Lin, J. Lu, and L. Ying, *Fast construction of hierarchical matrix representation from matrix–vector multiplication*, Journal of Computational Physics, 230 (2011), pp. 4071–4087.
[53] F. Lindgren, H. Rue, and J. 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 (2011), pp. 423–498.
[54] M. W. Mahoney and P. Drineas, *CUR matrix decompositions for improved data analysis*, Proceedings of the National Academy of Sciences, 106 (2009), pp. 697–702.
[55] P.-G. Martinsson, *A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix*, SIAM Journal on Matrix Analysis and Applications, 32 (2011), pp. 1251–1274.
[56] P.-G. Martinsson, *Compressing rank-structured matrices via randomized sampling*, SIAM Journal on Scientific Computing, 38 (2016), pp. A1959–A1986.
[57] P.-G. Martinsson and J. A. Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numerica, 29 (2020), pp. 403–572. Section 20.
[58] J. G. Nagy and D. P. O'Leary, *Restoring images degraded by spatially variant blur*, SIAM

Journal on Scientific Computing, 19 (1998), pp. 1063–1082.
[59] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, 1999.
[60] N. Petra, J. Martin, G. Stadler, and O. 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 (2014), pp. A1525–A1555.
[61] N. Petra, H. Zhu, G. Stadler, T. Hughes, and O. Ghattas, *An inexact Gauss-Newton method for inversion of basal sliding and rheology parameters in a nonlinear Stokes ice sheet model*, Journal of Glaciology, 58 (2012), pp. 889–903, doi:10.3189/2012JoG11J182.
[62] L. Roininen, J. M. Huttunen, and S. Lasanen, *Whittle-Matérn priors for Bayesian statistical inversion with applications in electrical impedance tomography*, Inverse Problems & Imaging, 8 (2014), p. 561.
[63] Y. Saad, *Iterative methods for sparse linear systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, second ed., 2003, doi:10.1137/1.9780898718003, https://doi.org/10.1137/1.9780898718003.
[64] Y. Saad and M. Sosonkina, *Distributed Schur complement techniques for general sparse linear systems*, SIAM Journal on Scientific Computing, 21 (1999), pp. 1337–1356.
[65] A. Spantini, A. Solonen, T. Cui, J. Martin, L. Tenorio, and Y. Marzouk, *Optimal low-rank approximations of Bayesian linear inverse problems*, SIAM Journal on Scientific Computing, 37 (2015), pp. A2451–A2487.
[66] A. Stuart, *Inverse problems: a Bayesian perspective*, Acta Numerica, 19 (2010), pp. 451–559.
[67] A. Tarantola, *Inverse Problem Theory and Methods for Model Parameter Estimation*, SIAM, Philadelphia, PA, 2005.
[68] J. Trampert, A. Fichtner, and J. Ritsema, *Resolution tests revisited: the power of random numbers*, Geophysical Journal International, 192 (2013), pp. 676–680.
[69] U. Villa, N. Petra, and O. Ghattas, *hIPPYlib: an extensible software framework for large-scale inverse problems governed by PDEs: part i: deterministic inversion and linearized Bayesian inference*, ACM Transactions on Mathematical Software (TOMS), 47 (2021), pp. 1–34.
[70] C. R. Vogel, *Computational Methods for Inverse Problems*, Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2002.
[71] H. Wendland, *Scattered data approximation*, vol. 17, Cambridge University Press, 2004.
[72] H. Zhu, S. Li, S. Fomel, G. Stadler, and O. Ghattas, *A Bayesian approach to estimate uncertainty for full waveform inversion with a priori information from depth migration*, Geophysics, 81 (2016), pp. R307–R323.