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. The PSF-based method computes impulse responses of the operator at scattered points, and interpolates these impulse responses to approximate entries of the integral kernel. To compute impulse responses efficiently, we apply the operator to Dirac combs associated with batches of point sources, which are chosen by solving an ellipsoid packing problem. The ability to rapidly evaluate kernel entries allows us to construct a hierarchical matrix (H-matrix) approximation of the operator. Further matrix computations are then performed with fast H-matrix methods. This end-to-end procedure is illustrated on a blur problem. We demonstrate the PSF-based method's effectiveness by using it to build preconditioners for the Hessian operator arising in two inverse problems governed by partial differential equations (PDEs): inversion for the basal friction coefficient in an ice sheet flow problem and for the initial condition in an advective-diffusive transport problem. While for many ill-posed inverse problems the Hessian of the data misfit term exhibits a low rank structure, and hence a low rank approximation is suitable, for many problems of practical interest the numerical rank of the Hessian is still large. The Hessian impulse responses on the other hand typically become more local as the numerical rank increases, which benefits the PSF-based method. Numerical results reveal that the preconditioner clusters the spectrum of the preconditioned Hessian near one, yielding roughly $5\times$–$10\times$ reductions in the required number of PDE solves, as compared to classical regularization-based preconditioning and no preconditioning. We also present a comprehensive numerical study for the influence of various parameters (that control the shape of the impulse responses and the rank of the Hessian) on the effectiveness of the advection-diffusion Hessian approximation. The results show that the PSF-based method is able to form good approximations of high-rank Hessians using only a small number of operator applications.

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

AMS subject classifications. 35R30, 41A35, 47A52, 47J06, 65D12, 65F08, 65F10, 65K10, 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 $L^2(\Omega)$. By "non-negative integral kernel," we mean that entries of $\mathcal{A}$'s integral kernel are non-negative numbers; this is not the same as positive semi-definiteness of $\mathcal{A}$. Such operators appear, for instance, as Hessians in optimization and inverse problems governed by partial differential equations (PDEs) [14, 20, 47], Schur complements in Schur complement methods for solving partial differential equations and Poincare-Steklov operators in

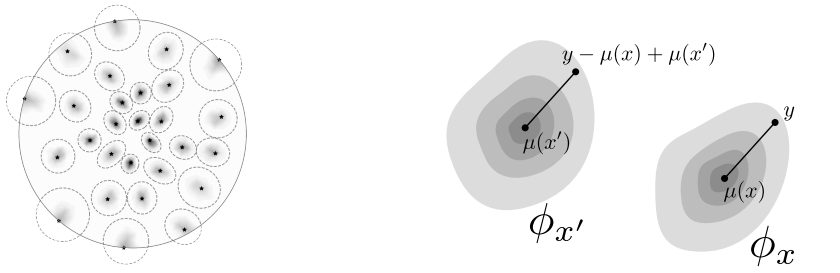
*To appear in SIAM Journal on Scientific Computing (SISC).

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, 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).



(a) One impulse response batch (b) Local mean displacement invariance

Fig. 1: (1a) One batch, η_b, of normalized impulse responses, ϕ_x, that arise from applying $\mathcal{A}$ to a weighted sum of scattered point sources (see Section 5.2). Here, $\mathcal{A}$ is the ice sheet inverse problem data misfit Gauss-Newton Hessian described in Section 7. Black stars are point source locations. Shading shows the magnitude of the normalized impulse responses (darker means larger function values). Dashed gray ellipses are estimated impulse response support ellipsoids based on the moment method in Section 4.1. The large circle is $\partial\Omega$. (1b) Illustration of impulse responses, ϕ_x and $\phi_{x'}$, corresponding to points x and x'. The operator $\mathcal{A}$ is locally mean displacement invariant (Section 4.2) if $\phi_x(y) \approx \phi_{x'}\left(y - \mu(x) + \mu(x')\right)$ when x is close to x'. Here, $\mu(z)$ denotes the mean (center of mass) of ϕ_z.

domain decomposition methods (e.g., Dirichlet-to-Neumann maps) [16, 67, 73], covariance operators in spatial statistics [17, 36, 37, 56], and blurring operators in imaging [22, 60]. Here, "matrix-free" means that we may apply $\mathcal{A}$ and its transpose[1], $\mathcal{A}^T$, to 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 cannot easily access entries of $\mathcal{A}$'s integral kernel. Evaluating the maps in (1.1) may require solving a subproblem that involves PDEs, or performing other costly computations.

The idea of the proposed method, which we refer to throughout the paper as the "PSF-based method," is to use *impulse response interpolation* to form a high rank approximation of $\mathcal{A}$ using a small number of operator applications. 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 1a). Batches of impulse responses may be thought of intuitively as sets of "columns" of the kernel (Figure 2). To choose the batches, we form ellipsoid estimates for the supports of all ϕ_x via a moment method (Figure 3) that involves applying $\mathcal{A}^T$ to a small number of polynomials (see Section 4.1). We then use a greedy

[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 $w \in L^2(\Omega)$.

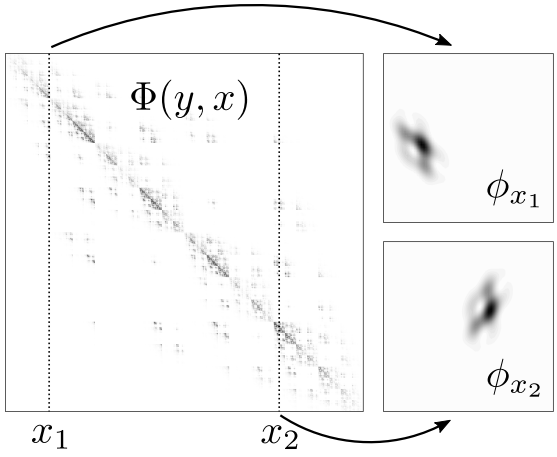


Fig. 2: Left: Matrix created by evaluating the integral kernel Φ for $\mathcal{A}$ (Equation 3.1) at all pairs of mesh vertices. This illustration is for the integral kernel in Equation 7.4. Dark colors indicate large entries and light colors indicate small entries. Rows and columns are ordered according to a kd-tree hierarchical clustering. Right: Impulse responses associated with points $x_1, x_2 \in \Omega$, shown by the two dotted vertical lines. Intuitively, one may think of impulse responses as "columns" of the integral kernel.

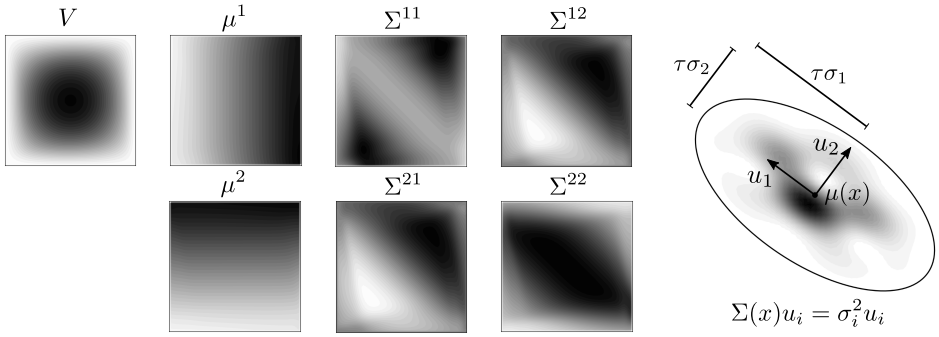


Fig. 3: Left: Impulse response moments. Scaling factor (V), mean (μ), and covariance (Σ). For each point $x \in \Omega$, the quantity $V(x)$ is the integral of ϕ_x over Ω, $\mu(x)$ is the location that ϕ_x is centered at, and $\Sigma(x)$ is a matrix with eigenvectors and eigenvalues that characterize the width of the support of ϕ_x about $\mu(x)$ (see Section 4.1). Right: Ellipsoid support for an impulse response. This ellipsoid is the set of points within τ standard deviations of the mean of the Gaussian distribution with mean $\mu(x)$ and covariance $\Sigma(x)$. The scaling factor $V(x)$ characterizes the magnitude of ϕ_x.

ellipsoid packing algorithm (Figure 4) to maximize the number of impulse responses per batch. Then we interpolate translated and scaled versions of these impulse responses to approximate entries of the operator's integral kernel (Figure 5). Adding more batches yields impulse responses at more points, increasing the approximation accuracy at the cost of one operator application per batch (Figure 6).

The PSF-based method we propose is loosely based on "product convolution"

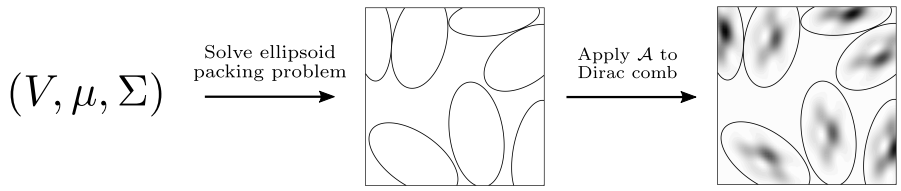


Fig. 4: Illustration of the process to compute one impulse response batch. Impulse response moments are first used to form ellipsoid shaped estimates of the supports of impulse responses (Equation 4.6). Then, an ellipsoid packing problem is solved to choose batches of non-overlapping support ellipsoids (Section 5.1). Finally, $\mathcal{A}$ is applied to a Dirac comb associated with the points x_i, which correspond to the ellipsoids (Section 5.2). The process is repeated to form more batches.

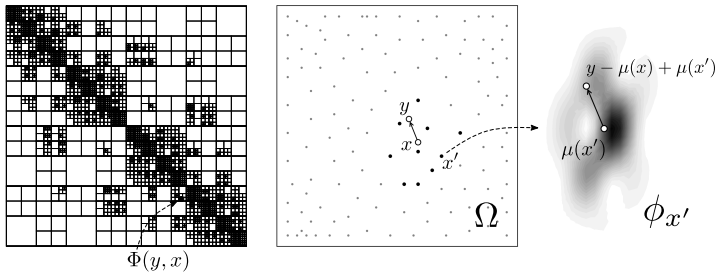


Fig. 5: Left: H-matrix structure for Φ. Computing an entry of this matrix requires evaluating the integral kernel, $\Phi(y, x)$, at a pair of points $(y, x) \in \Omega \times \Omega$. Center: Kernel evaluation points x and y (black circles), sample points for the approximation (light gray and black dots), and the k_n sample points, x', that are nearest to x (black dots). Right: Known impulse response at x'. Using radial basis function interpolation, the desired kernel entry is approximated as a weighted linear combination of translated and scaled versions of impulse responses at the points x' (Section 5.3).

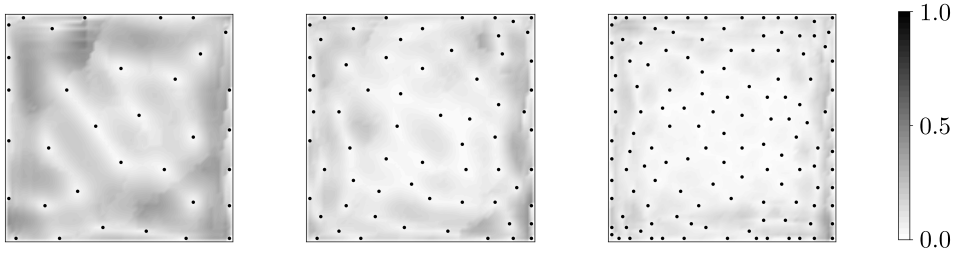


Fig. 6: Relative error, $||\Phi(\cdot, x) - \widetilde{\Phi}(\cdot, x)||/||\Phi(\cdot, x)||$, in the approximation of the "column" of the integral kernel associated with x, using 5 (left), 10 (center) and 20 (right) impulse response batches. Sample points are indicated by black dots. The error associated with the point x is the shade of the image at location x, with white indicating zero error and black indicating 100% error. At the sample points, the error is zero. The further the point x is from the sample points, the larger the error. Adding more batches yields a more accurate approximation.

(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) in which the convolution kernels are constructed from sampling impulse responses of the operator to scattered point sources: [1, 5, 12, 27, 29, 30, 32, 60, 78]. For background on PC and PSF methods, we recommend the following papers: [23, 28, 35]. The proposed PSF-based method improves upon existing PC and PSF methods in the following ways: (1) While PC and PSF approximations are typically based on an assumption of local translation invariance, the method we propose is based on a more general assumption we call "local mean displacement invariance" (Section 4.2 and Figure 1b), which improves the interpolation of the impulse responses. (2) In our previous work [5], we chose point sources in an adaptive grid via a sequential procedure; the refinements to the adaptive grid were chosen to maximally reduce the error at each step. However, in that work each point source required a separate operator application, making the previous method expensive when a large number of impulse responses is desired. In this paper, we use a new moment method (Section 4.1) which permits computation of many impulse responses (e.g., 50) per operator application. We are inspired by resolution analysis in seismic imaging, in which $\mathcal{A}^T$ is applied to a random noise function, and the width of the support of ϕ_x is estimated to be the autocorrelation length of the resultant function near x [31, 74]. The moment method that we use estimates the support of ϕ_x more accurately than random noise probing in resolution analysis, at the cost of the additional constraint that $\mathcal{A}$ has a non-negative integral kernel. (3) The PSF-based method we propose never evaluates computed impulse responses outside of their domain of definition. This eliminates "boundary-artifact" errors (see [5, Section 1.1]) that plague conventional PC and PSF methods.

The ability to rapidly approximate entries of $\mathcal{A}$'s integral kernel allows one to approximate discretized versions of $\mathcal{A}$ using the full arsenal of tools for matrix approximation that rely on fast access to matrix entries. In this work, we form a hierarchical matrix [13, 42] approximation of a discretized version of $\mathcal{A}$. H-matrices are a compressed 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 with rank k_h blocks scales as $O\left(k_h^a N \log(N)^b\right)$ where $a, b \in \{0, 1, 2, 3\}$ are constants which depend on the type of H-matrix used and the operation being performed [40][52, Section 2.1].

2. Why we need more efficient approximations of high rank Hessians. While the PSF-based method proposed in this paper may be used to approximate any operator that has a locally supported non-negative integral kernel, we are primarily motivated by approximation of high-rank Hessians in distributed parameter inverse problems governed by PDEs. In this section, we provide a brief background on this topic, and explain why existing Hessian approximation methods are not satisfactory.

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 [9, 76], or a Bayesian probabilistic framework [49, 70, 72]. In the deterministic framework, one solves an optimization problem to find the parameter that best fits the observations, subject to appropriate regularization [25, 76]. 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 with respect to the parameter in the determinstic optimization problem and the Hessian of the negative log posterior in the Bayesian setting are equal or approximately equal 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 a 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, 38, 75].

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 the conjugate gradient method [66]. In the Bayesian setting, the inverse of the Hessian is the covariance of a local Gaussian approximation of the posterior. This Gaussian distribution can be used directly as an approximation of the posterior, or it can be used as a proposal for Markov chain Monte-Carlo methods for drawing samples from the posterior. For instance, see [50, 62] 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 [38, 63]. 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., [15, 19, 33, 62, 68]. Krylov methods such as Lanczos or randomized methods [18, 44] 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 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, 15, 48].

Examples of inverse problems with high rank data misfit Hessians include large-scale ice sheet inverse problems [45, 48], advection dominated advection-diffusion inverse problems [2][34, Chapter 5], high frequency wave propagation inverse problems [15], inverse problems governed by high Reynolds number flows, and more generally, all inverse problems in which the observations highly inform the parameter. The eigenvalues of the data misfit Hessian characterize how informative the data are about components of the parameter in the corresponding eigenvector directions, hence

more informative data leads to larger eigenvalues and 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 the observations would instead be reconstructed based on the regularization [6, Section 4][76, 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 applications are required to form an accurate approximation of the Hessian using low rank approximation methods. High rank Hessian approximation methods are thus 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 [54, 55, 57, 58, 59]. These methods (which we do not use here) have been used to approximate Hessians in PDE constrained inverse problems [7, 45]. Although these methods are promising, the required number of operator applications is still large (e.g., hundreds to thousands). For example, using the method in [55], the required number of operator applies to construct an H^1 matrix with hierarchical rank r for problems in a 2D domain discretized with a regular grid is $\#\text{levels} \cdot 64 \cdot (r + c)$, where #levels is the depth of the hierarchical partitioning, r is the rank of the blocks (hierarchical rank), and c is an oversampling parameter (see [55, Section 2.4]). On a 64×64 grid with depth 4, hierarchical rank 10, and oversampling parameter $c = 5$, this works out to $4 \cdot 64 \cdot (10+5) = 3840$ operator applies. In Section 7.3, we see numerically that the randomized hierarchical off-diagonal low rank (HODLR) method in [58] requires hundreds to thousands of matrix-vector products to construct approximations of the integral kernel for a blur problem example with modest (e.g., 10%) relative error. Matrix-free H-matrix construction is currently an active area of research, hence these costs may decrease as new algorithms are developed. In this paper, we also form an H-matrix approximation. However, to reduce the required number of operator applications, 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 techniques. Using this two stage approach, we reduce the number of operator applications to a few dozen at most.

Not all data misfit Hessians satisfy the local non-negative integral kernel properties. We note, in particular, that the wave inverse problem data misfit Hessian and Gauss-Newton Hessian have a substantial proportion of negative entries in their integral kernels. In this case more specialized techniques have been developed using, eg., pseudodifferential operator theory [21, 71], and sparsity in the wavelet domain [46]. However, many data misfit Hessians of practical interest do satisfy the local non-negative integral kernel properties (either exactly or approximately), and the PSF-based method we propose 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

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

The linear functional $\mathcal{A}u \in L^2(\Omega)'$ is the result of applying $\mathcal{A}$ to $u \in L^2(\Omega)$, and 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 but is not easily accessible. In this section we describe how to extend the domain of $\mathcal{A}$ to distributions, which allows us to define impulse responses (Section 3.1), we then state the conditions on $\mathcal{A}$ that the PSF-based method requires (Section 3.2), and detail finite element discretization (Section 3.3).

3.1. Distributions and impulse responses. The operator $\mathcal{A}$ may be applied to distributions[3] if Φ is sufficiently regular. Given $\rho \in L^2(\Omega)'$, let $\rho^* \in L^2(\Omega)$ denote the Riesz representative of ρ with respect to the $L^2(\Omega)$ inner product. We have

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

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

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, 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 application of $\mathcal{A}$ to the distribution ρ to be the right hand side of (3.2b). We denote this operator application by "$\mathcal{A}\rho^*$," even if ρ^* does not exist.

Let δ_x denote the delta distribution[4] (i.e., point source, impulse) centered at the point $x \in \Omega$. The *impulse response* of $\mathcal{A}$ associated with x is the function $\phi_x : \Omega \to \mathbb{R}$,

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

that is formed by applying $\mathcal{A}$ to δ_x (per the generalized notion of operator "application" defined above), 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[5] in the sense that

$$\Phi(y,x) \geq 0 \quad \text{for all} \quad (y,x) \in \Omega \times \Omega.$$

The PSF-based method may still perform well if these conditions are relaxed slightly.

It is acceptable if the support of ϕ_x is not perfectly contained in a localized region (violating Assumption 2), so long as the bulk of the "mass" of ϕ_x is contained in a localized region. In principle, the PSF-based method can be applied even if the impulse responses are widely dispersed. However, in this case only a small number of impulse responses can be computed per batch, which means more batches, and hence more operator applies, are needed to form an accurate approximation.

[3] I.e., generalized functions such as the Dirac delta distribution. See, for example, [8, Chapter 5].

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

[5] 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 the PSF-based method, and positive semi-definite operators need not have a non-negative integral kernel.

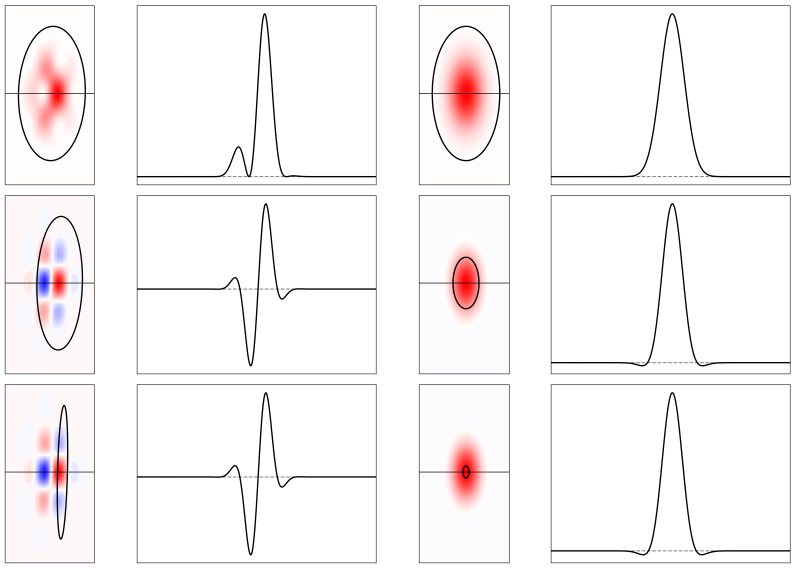


Fig. 7: Illustration of the influence of negative numbers in the integral kernel on the robustness of the ellipsoid estimates for the supports of impulse responses. Left two columns: Blur kernel given in Equation 7.4. Right two columns: Ricker wavelet-type kernel given by $\Phi(y,x) = (1 - a\gamma)\exp(-\gamma/2)$, where $\gamma = (y-x)^T\Sigma^{-1}(y-x)$, and $\Sigma = \mathrm{diag}\,(0.0025, 0.01)$. Ordered from top to bottom, the results are obtained with $a \in \{1.0, 20.0, 27.0\}$ for the left two columns, and $a \in \{0.0, 0.23, 0.249\}$ for the right two columns. Columns 1 and 3: impulse responses with estimated support ellipsoids indicated by the black ellipses. Red and blue represent positive and negative numbers in the integral kernel, respectively. Columns 2 and 4: one-dimensional slice along the horizontal line indicated in the two-dimensional plots. The dashed gray line is at zero.

If there are negative numbers in the integral kernel (violating Assumption 3), the ellipsoid estimation procedure may incur errors or fail, leading to poor performance or failure of the PSF-based method. In Figure 7 we investigate the robustness of the ellipsoid support estimation procedure to violations of Assumption 3. We study two integral kernel examples, both of which are parameterized by a quantity that controls how negative the kernels are. We make the following observations:

- The larger and more numerous the negative numbers are, the more inaccurate the ellipsoid support estimate is.
- The further away from the center of the ellipsoid the negative numbers are, the more influence they have on the quality of the ellipsoid support estimate. This is because moment formulas (Equations 4.2 and 4.3) assign more weight to entries in the kernel that are further from the center.
- Negative numbers affect the ellipsoid estimation method more if they are isolated, and less if they are balanced by nearby positive numbers.
- As kernels become more negative, the ellipsoid estimation performs well up to a certain threshold that depends on the spatial distribution of negative and positive entries. After that threshold is crossed, the estimation rapidly transitions to performing poorly and ultimately failing.

For the kernel in the left two columns of Figure 7, negative numbers are interspersed with positive numbers, allowing us to include a large amount of negative numbers before the ellipsoid estimation fails. For the kernel in the right two columns, the ellipsoid estimate fails with tiny amounts of negative numbers because the negative numbers are far away from the mean and not balanced by positive numbers at similar distances and angles. In the bottom two rows, we see the aforementioned threshold effect, in which the ellipsoid estimation method rapidly transitions from performing reasonably well to performing poorly with only a small change to the integral kernel.

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 the functions that $\mathcal{A}$ and $\mathcal{A}^T$ are applied to using continuous finite elements satisfying the Kronecker property (defined below). With minor modifications, the PSF-based method could be used with more general finite element methods, or other discretization schemes such as finite differences or finite volumes. These restrictions on discretization only apply to functions u that $\mathcal{A}$ and $\mathcal{A}^T$ are applied to. Other functions that arise internally during the process of computing actions of $\mathcal{A}$, such as state variables in a PDE that is solved in a subproblem, may be discretized with any method.

Let $\psi_1, \psi_2, \ldots, \psi_N$ be a set of continuous finite element basis functions used to discretize the problem on a mesh with mesh size parameter h, let $V_h := \operatorname{span}(\psi_1, \psi_2, \ldots, \psi_N)$ be the corresponding finite element space under the L^2 inner product, and let $p_i \in \mathbb{R}^d$, $i = 1, \ldots, N$ be the Lagrange nodes associated with the functions ψ_i. We assume that the finite element basis satisfies the Kronecker property, i.e., $\psi_i(p_i) = 1$ and $\psi_i(p_j) = 0$ if $i \neq j$. For $u_h \in V_h$ we write $\mathbf{u} \in \mathbb{R}^m_{\mathbf{M}}$ to denote the coefficient vector for u_h with respect to the finite element basis, i.e., $u_h(x) = \sum_{i=1}^N \mathbf{u}_i \psi_i(x)$. 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 $\mathbf{M}_{ij} = \int_\Omega \psi_i(x)\psi_j(x)\mathrm{d}x$ 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 $\mathbf{A} : \mathbb{R}^N_{\mathbf{M}} \to \mathbb{R}^N_{\mathbf{M}^{-1}}$ under the isomorphism discussed above. The PSF-based 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. The PSF-based method mitigates this cost by performing as few matrix-vector products as possible. 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:

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

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

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

Broadly, we will construct an H-matrix approximation of $\mathbf{A}$ by forming an H-matrix approximation of $\boldsymbol{\Phi}$, then multiplying $\boldsymbol{\Phi}$ by $\mathbf{M}$ (or a lumped mass version of $\mathbf{M}$) on the left and right 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 the PSF-based 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 $\boldsymbol{\Phi}$ involve inverse mass matrices, which typically contain negative numbers. We therefore recommend replacing the mass matrix, $\mathbf{M}$, with a positive diagonal *lumped mass* approximation. Here, we use the lumped mass matrix in which the ith diagonal entry of the lumped mass matrix is the sum of all entries in the ith row of the mass matrix. Other mass lumping techniques may be used.

4. Key innovations. In this section we present two key innovations that the PSF-based 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 to approximate impulse responses from 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 $V : \Omega \to \mathbb{R}$,

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

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:

$$\mu^i(x) := \frac{1}{V(x)} \int_\Omega \phi_x(y) y^i \ \mathrm{d}y \tag{4.2}$$

$$\Sigma^{ij}(x) := \frac{1}{V(x)} \int_\Omega \phi_x(y) \left(y^i - \mu^i(x)\right)\left(y^j - \mu^j(x)\right) \ \mathrm{d}y, \tag{4.3}$$

where $\mu^i(x)$ and y^i denote the i^{th} components of the vectors $\mu(x)$ and y, respectively, and $\Sigma^{ij}(x)$ denotes the (i,j) entry of the matrix $\Sigma(x)$. The quantities $\mu(x) \in \mathbb{R}^d$ and $\Sigma(x) \in \mathbb{R}^{d\times d}$ 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, 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 our algorithm for picking sample points (Section 5.1 and Figure 4) needs to know $V(x)$, $\mu(x)$, and $\Sigma(x)$ before we compute ϕ_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 the PSF-based method, 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 applications in two spatial dimensions and 10 in three spatial dimensions). This may be motivated by analogy to matrices. If $\mathbf{A} \in \mathbb{R}^{N\times N}$ is a matrix with i^{th} column $\mathbf{a}_i$ and $\mathbf{w} \in \mathbb{R}^N$, then

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

By computing one matrix-vector product of $\mathbf{A}^T$ with $\mathbf{w}$, we compute the inner product of each column of $\mathbf{A}$ 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 $i,j = 1,\dots,d$. 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 $i,j = 1,\dots,d$. 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

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

where τ is a fixed constant (see Figure 3). The ellipsoid E_x is the set of points within τ standard deviations of 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 the PSF-based method 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 $\tau = 3.0$ yields a reasonable balance between these competing interests, and use $\tau = 3.0$ in all numerical results, except for Figure 14, where we study the effects of varying τ. 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:

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

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

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

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

These are four different ways to approximate an impulse response by a nearby impulse response, with each successive approximation building upon the previous ones. The PSF-based method uses (4.10), which is the most sophisticated. 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. 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 translated and rotated frame of reference, while (4.8) will not. 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 1b.

We use approximation (4.10), which is the same as (4.9), except for the factor $V(x)/V(x')$. This factor makes the approximation more accurate 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. Before presenting the technical details of the algorithm in Sections 5.1–5.5, we first provide an overview.

We use (4.5) to compute V, μ, and Σ by applying $\mathcal{A}^T$ to polynomial functions. Then we use (4.6) to form ellipsoid shaped estimates for the support of the ϕ_x's, *without* computing them (see Figure 3). This allows us to compute large numbers of ϕ_{x_i} in "batches," η_b (see Figures 1a and 4). 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 and Figure 5). 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 (RBF) 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 convert $\mathbf{\Phi}_H$ into an H-matrix approximation $\mathbf{A}_H \approx \mathbf{A}$.

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 symmetrize $\mathbf{A}_H$, then modify it via low rank updates to remove erroneous negative eigenvalues (Section 5.5). The complete algorithm for constructing $\mathbf{A}_H$ is shown in Algorithm 1. The computational cost is discussed in Section 6.

Algorithm 1: Construct PSF 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 impulse response batch 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 (Sections 5.3 and 5.4)
Form H-matrix approximation $\mathbf{A}_H$ of $\mathcal{A}$ (Section 5.4)
(optional) Modify $\mathbf{A}_H$ to make it symmetric and remove negative eigenvalues (Section 5.5)

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 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 $V(x_i) > \epsilon_V V_{\max}$. Here, $V_{\max}$ is the largest value of $V(x_j)$ over all points q in the initial set of candidate points, and ϵ_V is a small threshold parameter (we use $\epsilon_V = 10^{-5}$).
3. We make sure that all eigenvalues of $\Sigma(x_i)$ are positive, and the aspect ratio of E_{x_i} (square root of the ratio of the largest eigenvalue of $\Sigma(x_i)$ to the smallest) is bounded by a constant $1/\epsilon_\Sigma$ (we use $1/\epsilon_\Sigma = 20$). Negative integral kernel entries due to discretization error can cause $\Sigma(x_i)$ to be indefinite or highly ill-conditioned.

If x_i passes these checks 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 repeat the point selection process to construct several batches of points $S_1, S_2, \ldots, S_{n_b}$. 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 previous batches.

We check whether $E_{x_i} \cap E_{x_j} = \{\}$ in a two stage process. First, we check whether the axis aligned bounding boxes for the ellipsoids intersect. This quickly rules out intersections of ellipsoids that are far apart. Second, if the bounding boxes intersect, we check if the ellipsoids intersect using the ellipsoid intersection test in [39].

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 response 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 applying the operator once, $\xi_k \mapsto (\mathcal{A}\xi_k^*)^*$, 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 the 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. Here, we describe how to rapidly evaluate arbitrary entries of an approximation to the integral kernel by performing radial basis function interpolation of translated and scaled versions of nearby known impulse responses. In Section 5.4 we use this procedure for rapidly evaluating kernel entries to construct the H-matrix approximation of $\mathbf{A}$.

Given $(y, x) \in \Omega \times \Omega$, let $z_i := y - \mu(x) + \mu(x_i)$ and define

$$f_i := \frac{V(x)}{V(x_i)} \phi_{x_i}(z_i) \tag{5.3}$$

for $i = 1, \ldots, k_n$, 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 [11] of all sample points. We check whether $z_i \in \Omega$ by querying a precomputed axis aligned bounding box tree (AABB tree) [26] of the mesh cells used to discretize the problem. 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 the sample point choosing 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 [77] 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 $\varphi(r) := \exp\left(-\frac{1}{2}\left(C_{\text{RBF}} \frac{r}{r_0}\right)^2\right)$ is a Gaussian kernel radial basis function. Here, $r_0 := \operatorname{diam}\left(\{x_i\}_{i=1}^{k_n}\right)$ is the diameter of the set of sample points used in the interpolation, and C_{RBF} is a user-defined shape parameter that controls the width of the kernel function. 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 $B \in \mathbb{R}^{k_n \times k_n}$, $B_{ij} := \varphi(\|x_i - x_j\|)$, and $f \in \mathbb{R}^{k_n}$ has entries f_i from (5.3).

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 $z_i \in \Omega$.

The shape parameter, C_{RBF}, mediates a tradeoff between accuracy and stability. Small C_{RBF} is required for RBF interpolation with Gaussian kernels to achieve high accuracy, but small C_{RBF} also makes RBF interpolation less robust to errors or non-smoothness in the function being interpolated. For our numerical results involving Hessians in inverse problems governed by PDEs (Sections 7.1 and 7.2), high accuracy is not required because the PSF-based method is used to construct a preconditioner. Hence, for these Hessian approximations we use a conservative choice of $C_{\text{RBF}} = 3.0$ to ensure robustness. For our numerical results for the blur problem example (Section 7.3), we use a smaller value of $C_{\text{RBF}} = 0.5$ so that the RBF interpolation accuracy is not a limiting factor as we study convergence of the PSF-based method.

5.4. Hierarchical matrix construction. We form an H-matrix approximation $\mathbf{A}_H \approx \mathbf{A}$ by forming an H-matrix representation $\mathbf{\Phi}_H$ of $\mathbf{\Phi}$ then multiplying $\mathbf{\Phi}$ with mass matrices $\mathbf{M}$ per (3.5) to form $\mathbf{A}_H = \mathbf{M}\mathbf{\Phi}_H\mathbf{M}$. Here, we use a diagonal lumped mass matrix, so these matrix-matrix multiplications are trivial. If a non-diagonal mass matrix is used, one may form an H-matrix representation of the mass matrix, then perform the matrix-matrix multiplications in (3.5) using H-matrix methods. We use H1 matrices in the numerical results, but any other H-matrix format could be used instead. For more details on H-matrices, see [43].

We form $\mathbf{\Phi}_H$ using the standard geometrical clustering/adaptive cross method implemented within the HLIBpro software package [51]. For details about the algorithms used for geometrical clustering, H-matrix construction, and H-matrix operations in HLIBpro, we refer the reader to [13, 41, 52]. Although $\mathbf{\Phi}$ is a dense $N \times N$ matrix, constructing $\mathbf{\Phi}_H$ only requires evaluation of $O(k_h N \log N)$ kernel entries $\mathbf{\Phi}_{ij} = \widetilde{\Phi}(p_i, p_j)$ (see [10]), 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. We emphasize that the dense matrix $\mathbf{\Phi}$ is never formed.

5.5. Symmetrizing and flipping negative eigenvalues (optional). In many applications, one seeks to approximate an operator $\mathcal{H} = \mathcal{A} + \mathcal{R}$, where $\mathcal{A}$ is a symmetric positive semi-definite operator that we approximate with the PSF-based method to form an H-matrix $\mathbf{A}_H$, and $\mathcal{R}$ is a symmetric positive definite operator that may be easily converted to an H-matrix $\mathbf{R}_H$ without using the PSF-based method. For example, in inverse problems $\mathcal{H}$ is the Hessian, $\mathcal{A}$ is the data misfit term in the Hessian which is dense and available only matrix-free, and $\mathcal{R}$ is the regularization term, which is typically an elliptic differential operator that becomes a sparse matrix after discretization.

The PSF-based approximation $\mathbf{A}_H$, and therefore $\mathbf{A}_H + \mathbf{R}_H$, may be non-symmetric and indefinite because of approximation error. This is undesirable because 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 to perform further operations involving $\mathbf{A}_H + \mathbf{R}_H$, such as using $\mathbf{A}_H + \mathbf{R}_H$ as a preconditioner in the conjugate gradient method.

We modify $\mathbf{A}_H$ to make it symmetric and remove negative eigenvalues via the following procedure. First, we symmetrize $\mathbf{A}_H$ via $\mathbf{A}_H^{\text{sym}} := \frac{1}{2}\left(\mathbf{A}_H + \mathbf{A}_H^T\right)$. Next, we find negative eigenvalues and their corresponding eigenvectors for the generalized eigenvalue problem $\mathbf{A}_H^{\text{sym}}\mathbf{u} = \lambda\mathbf{R}_H$ using a Cayley shift-and-invert Krylov scheme [53]. We flip the signs of these eigenvalues to be positive instead of negative (i.e., $\lambda \to |\lambda|$) by performing a low rank update to $\mathbf{A}_H^{\text{sym}}$. We observe that the eigenvectors associated with large erroneous negative eigenvalues tend to be directions that are nevertheless "important" to $\mathcal{A}$, so flipping the eigenvalues instead of setting them to zero tends to

Symbol	Typical size	Variable name
N	10^3–10^9	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)
$\lvert S_i\rvert$	1–500	Number of sample points in the ith batch

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

yield better approximations. The primary computational task in the Cayley shift-and-invert scheme is the solution of shifted linear systems of the form $(\mathbf{A}_H^{\text{sym}} + \mu_i \mathbf{R}_H)\,\mathbf{x} = \mathbf{b}$, for a small number of positive shifts μ_i. We solve these linear systems by factorizing the matrices $\mathbf{A}_H^{\text{sym}} + \mu_i \mathbf{R}_H$ using fast H-matrix methods. We compute and flip all eigenvalues $\lambda < \epsilon_{\text{flip}}$ which are less than some threshold $\epsilon_{\text{flip}} \in (-1, 0]$. By choosing $\epsilon_{\text{flip}} > -1$, we ensure that the modified version of $\mathbf{A}_H^{\text{sym}} + \mathbf{R}_H$ is positive definite. Choosing $\epsilon_{\text{flip}} = 0$ would remove all erroneous negative eigenvalues. However, this is computationally infeasible if $\mathcal{A}$ has a large or infinite cluster of eigenvalues near zero, a common situation for Hessians in ill-posed inverse problems. We therefore recommend choosing $\epsilon_{\text{flip}} < 0$. In our numerical results, we use $\epsilon_{\text{flip}} = -0.1$.

6. Computational cost. The computational cost of the PSF-based method may be divided into the costs to perform the following tasks: (1) Computing impulse response moments and batches (Lines 1 and 4 in Algorithm 1); (2) Building the H-matrix (Lines 5 and 6 in Algorithm 1); (3) Performing linear algebra operations with the H-matrix. This may optionally include the symmetric positive semi-definite modifications described in Section 5.5. In target applications, (1) is the dominant cost because applying $\mathcal{A}$ to a vector requires an expensive computational procedure such as solving a PDE, and (1) is the only step that requires applying $\mathcal{A}$ to vectors. All operations that do not require applications of $\mathcal{A}$ to vectors are polylog linear (i.e., $O(N \log(N)^b)$ for some b), 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 $\mathcal{A}^T$ to 1, d, and $d(d+1)/2$ vectors, respectively. This works out to 3 applications of $\mathcal{A}^T$ in one spatial dimension, 6 in two dimensions, and 10 in three dimensions. Computing each η_i requires applying $\mathcal{A}$ to one vector, so computing $\{\eta_i\}_{i=1}^{n_b}$ requires n_b operator applications. In total, computing all impulse response moments and batches therefore requires

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

In a typical application one might have $d = 2$ and $n_b = 5$, in which case a modest 11 operator applications 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 intersec-

tion 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 possible with more involved computational geometry methods. However, from a practical perspective, the cost of choosing sample points is small compared to other parts of the algorithm, and hence such improvements are not pursued here.

(2) Building the H-matrix. Classical H-matrix construction techniques require evaluating $O(k_h N \log N)$ matrix entries of the approximation [10], 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 in. 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 those cells, which requires $O(k_n)$ elementary operations. Finally, the radial basis function interpolation requires solving a $k_n \times k_n$ linear system, which requires $O(k_n^3)$ elementary operations. Therefore, building the H-matrix requires

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

(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, matrix inversion, and low rank updates require performing $O\left(k_h^a N \log(N)^b\right)$ elementary operations, where $a, b \in \{0, 1, 2, 3\}$ are constants which depend on the type of H-matrix used and the operation being performed [40][52, Section 2.1]. For our numerical results involving Hessians (Sections 7.1 and 7.2), 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. Symmetrizing $\mathbf{A}_H$ requires one matrix-matrix addition. Flipping negative eigenvalues to be positive requires a handful (typically around 5) of matrix-matrix additions and matrix factorizations to factor the required shifted linear systems, and a number of factorized solves that is proportional to the number of erroneous negative eigenvalues.

In summary, computing all the necessary ingredients to evaluate kernel entries of the PSF-based approximation requires a handful of operator applications (e.g., $6 + n_b$ operator applications in two dimensions, or $10 + n_b$ operator applications in three dimensions, with n_b typically in the range 1–25), plus comparatively cheap additional overhead costs, most notably performing ellipsoid intersection tests while choosing sample point batches. Once these ingredients are computed, no more operator applications (and thus PDE solves) are required, and approximate kernel entries can be evaluated rapidly. Constructing the H-matrix from kernel entries requires a number of elementary operations that scales polylog linearly in N. Using the H-matrix to perform linear algebra operations also scales polylog linearly in N, though the details of these costs depend heavily on the type of H-matrix and operation being performed.

7. Numerical results. We use the PSF-based method to approximate the Newton (or Gauss-Newton) Hessians in inverse problems governed by PDEs which model

steady state ice sheet flow [64] (Section 7.1) and advective-diffusive transport of a contaminant [63] (Section 7.2), and to approximate the integral kernel in a blur problem that is not based on PDEs (Section 7.3). These problems are described in detail in their respective sections.

In both PDE-based inverse problems (Sections 7.1 and 7.2), to reconstruct the unknown parameter fields, denoted q, the inverse problems are formulated as nonlinear least squares optimization problems, whose objective functions consist of a data misfit term (between the observations and model output) and a bi-Laplacian regularization term following [75]. The regularization is centered at a constant function $q_0(x)$. To mitigate boundary effects we use a constant coefficient Robin boundary condition as in [65]. The parameters for the bi-Laplacian operator are chosen so that the Green's function of 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 [75, Section 2.2]. In all numerical results we choose the regularization parameter (which controls the overall strength of the regularization) using the Morozov discrepancy principle [76].

We solve the ice sheet inverse problem with an inexact Newton preconditioned conjugate gradient (PCG) scheme and a globalizing Armijo line search [61]. The Newton search directions, $\widehat{\mathbf{q}}$, are obtained by solving

$$\mathbf{H}\widehat{\mathbf{q}} = -\mathbf{g} \qquad \text{or} \qquad \mathbf{H}_{\text{gn}}\widehat{\mathbf{q}} = -\mathbf{g}, \tag{7.1}$$

wherein we choose the initial guess as the discretization of the constant function q_0. Here, $\mathbf{g}$, $\mathbf{H}$ and $\mathbf{H}_{\text{gn}}$ are the discretized gradient, Hessian, and Gauss-Newton Hessian of the inverse problem objective function, respectively, evaluated at the current Newton iterate. To ensure positive definiteness of the Hessian we use $\mathbf{H}_{\text{gn}}$ for the first five iterations, and $\mathbf{H}$ for all subsequent iterations. The Newton iterations are terminated when $\|\mathbf{g}\| < 10^{-6}\|\mathbf{g}_0\|$, where $\mathbf{g}_0$ is the gradient evaluated at the initial guess. Systems (7.1) are solved inexactly using an inner PCG iteration, which is terminated early based on the Eisenstat-Walker [24] and Steihaug [69] conditions. The inverse problem governed by the advection-diffusion PDE is linear, hence Newton's method converges in one iteration. In this case the Newton linear system, (7.1), is solved using PCG, using termination tolerances described in Section 7.2.

We use the framework described in this paper to generate Hessian preconditioners. We build H-matrix approximations, $\mathbf{A}_H$, of the data misfit Gauss-Newton Hessian (the term in $\mathbf{H}_{\text{gn}}$ that arises from the data misfit). The approximations are indicated by "PSF (n_b)", where n_b is the number of impulse response batches used to build the approximation. The Hessian of the regularization term is a combination of stiffness and mass matrices, which are sparse. Therefore, we form H-matrix representations of these matrices and combine them into an H-matrix approximation of the regularization term in the Hessian, $\mathbf{R}_H$, using standard sparse H-matrix techniques. Then, H-matrix approximations of the Gauss-Newton Hessian, $\mathbf{H}_{\text{gn}} \approx \widetilde{\mathbf{H}} := \mathbf{A}_H + \mathbf{R}_H$, are formed by adding $\mathbf{A}_H$ to $\mathbf{R}_H$ using fast H-matrix arithmetic. We modify $\widetilde{\mathbf{H}}$ to be (approximately) symmetric positive semi-definite via the procedure described in Section 5.5. We factor $\widetilde{\mathbf{H}}$ using fast H-matrix methods, then use the factorization as a preconditioner. We approximate $\mathbf{H}_{\text{gn}}$ rather than $\mathbf{H}$ because $\mathbf{H}$ more often has negative values in its integral kernel. The numerical results show that $\widetilde{\mathbf{H}}$ is a good preconditioner for both $\mathbf{H}_{\text{gn}}$ and $\mathbf{H}$.

7.1. Example 1: Inversion for the basal friction coefficient in an ice sheet flow problem. For this example, we consider a sheet of ice flowing down a mountain (see Figure 8a). 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 friction Robin coefficient field, which governs the resistance to sliding along the base of the ice sheet. The setup, which we briefly summarize, follows [48, 64]. The region of ice is denoted by $\mathcal{D} \subset \mathbb{R}^3$. The basal, lateral and top parts of the boundary $\partial\mathcal{D}$ are denoted by Γ_b, Γ_l, and Γ_t, respectively. The governing equations are the linear incompressible Stokes equations,

$$-\nabla \cdot \sigma(v,p) = f \text{ and } \nabla \cdot v = 0 \quad \text{in } \mathcal{D}, \tag{7.2a}$$

$$\sigma(v,p)\nu = 0 \quad \text{on } \Gamma_t, \tag{7.2b}$$

$$v \cdot \nu = 0 \text{ and } T\left(\sigma(v,p)\nu + \exp(q)\, v\right) = 0 \quad \text{on } \Gamma_b, \tag{7.2c}$$

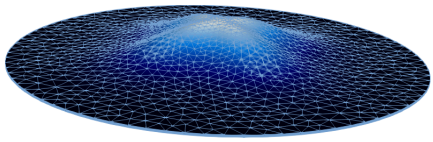
$$\sigma(v,p)\nu + sv = 0 \quad \text{on } \Gamma_l. \tag{7.2d}$$

The solution to these equations is the pair (v,p), where v is the ice flow velocity field[6] and p is the pressure field. Here, q is the unknown logarithmic basal friction field (large q corresponds to large resistance to sliding) defined on the 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 unit normal and T is the tangential projection operator that restricts a vector field to its tangential component along the boundary. We employ a Newtonian constitutive law, $\sigma(v,p) = 2\eta\dot{\varepsilon}(v) - Ip$, where σ is the stress tensor and $\dot{\varepsilon}(v) = \frac{1}{2}\left(\nabla v + \nabla v^\top\right)$ is the strain rate tensor [48]. Here, η is the viscosity and I is the identity operator. Note that while the PDE is linear, the parameter-to-solution map, $q \mapsto (v,p)$, is nonlinear.

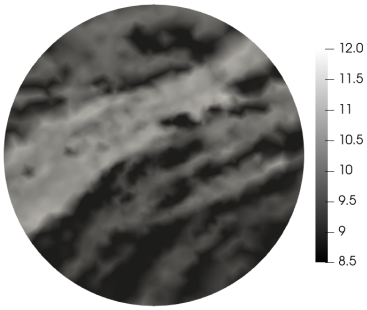
The pressure, p, is discretized with first order scalar continuous Galerkin finite elements defined on a mesh of tetrahedra. The velocity, v, is discretized with second order continuous Galerkin finite elements on the same mesh. The parameter q is discretized with first order scalar continuous Galerkin 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 two-dimensional surface embedded in three dimensions due to the mountain topography. The PSF-based method involves translating impulse responses. Hence it requires either a flat domain, or a notion of local parallel transport. We therefore generate a flattened version of Γ_b, denoted by $\Omega \subset \mathbb{R}^2$, by ignoring the height coordinate. The parameter q is 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 Hessian approximations and defining the regularization. The observations are generated by adding multiplicative Gaussian noise to the tangential component of the velocity field restricted to the top surface of the geometry. We use 5% noise in all cases, except for Figure 9 and Table 3 where the noise is varied from 1% to 25% and the regularization is determined by the Morozov discrepancy principle for each noise level. The true basal friction coefficient and resulting velocity fields, which are obtained by solving (7.2), are shown in Figure 8.

Table 2 shows the performance of the preconditioner for accelerating the solution of the optimization problem to reconstruct $\mathbf{q}$ from observations with 5% noise. We build the PSF (5) preconditioner in the third Gauss-Newton iteration, and reuse it for all subsequent Gauss-Newton and Newton iterations. No preconditioning is used in the iterations before the PSF (5) preconditioner is built. We compare the PSF-based method with the most commonly used existing preconditioners: no preconditioning (NONE), and preconditioning by the regularization term in the Hessian (REG). The

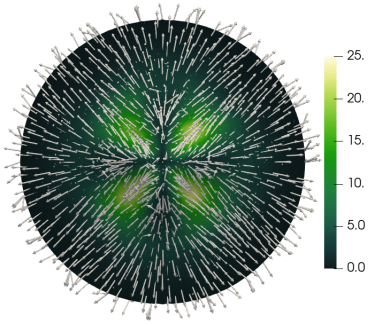
[6] We do not use bold to denote vector or tensor fields to avoid confusion with vectors that arise from finite element discretizations, which are already denoted with bold.



(a) Ice sheet model geometry



(b) q_{true}



(c) v_{true}

Fig. 8: (Ice sheet) (8a) Bird's eye view of 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. (8b) True parameter, q_{true}. (8c) True velocity, v_{true}. Arrows indicate the direction of v_{true} and color indicates the magnitude of v_{true}.

Fig. 9: (Ice sheet) The log basal friction parameter, with color scale as in Figure 8b, computed from the PDE constrained optimization problem with noise levels: 25% (left), 5.0% (middle), and 1.0% (right).

results show that using PSF (5) reduces the total number of Stokes PDE solves to 70, as compared to 908 for regularization preconditioning and 308 for no preconditioning, a reduction in cost of roughly 5×–10×. For problems with a larger physical domain and correspondingly more observations, such as continental scale ice sheet inversion, the speedup will be even greater. This is because the rank of the data misfit Hessian will increase, while the locality of the impulse responses will remain the same. In Figure 9 we show reconstructions for 1%, 5%, and 25% noise.

	PSF (5)			REG			NONE		
Iter	#CG	#Stokes	$\|\mathbf{g}\|$	#CG	#Stokes	$\|\mathbf{g}\|$	#CG	#Stokes	$\|\mathbf{g}\|$
0	1	4	1.9e+7	3	8	1.9e+7	1	4	1.9e+7
1	2	6	6.1e+6	8	18	8.4e+6	2	6	6.1e+6
2	4	10	2.6e+6	16	34	4.1e+6	4	10	2.6e+6
3	2	6+22	6.9e+5	34	70	1.8e+6	14	30	6.9e+5
4	3	8	4.4e+4	52	106	5.6e+5	29	60	1.3e+5
5	5	12	2.2e+3	79	160	9.4e+4	38	78	1.0e+4
6	0	2	1.1e+1	102	206	6.5e+3	58	118	1.8e+2
7	—	—	—	151	304	1.2e+2	0	2	5.5e-1
8	—	—	—	0	2	2.9e-1	—	—	—
Total	17	70	—	445	908	—	146	308	—

Table 2: (Ice sheet) Convergence history for solving the Stokes inverse problem using inexact Newton PCG to tolerance 10^{-6}. Preconditioners shown are the PSF-based method with five batches (PSF (5)) constructed at the third iteration, regularization preconditioning (REG), and no preconditioning (NONE). Columns #CG show the number of PCG iterations used to solve the Newton system for $\widehat{\mathbf{q}}$. Columns $\|\mathbf{g}\|$ show the l^2 norm of the gradient at $\mathbf{q}$. Columns #Stokes show the total number of Stokes PDE solves performed in each Newton iteration. Under PSF (5) and in row Iter 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.

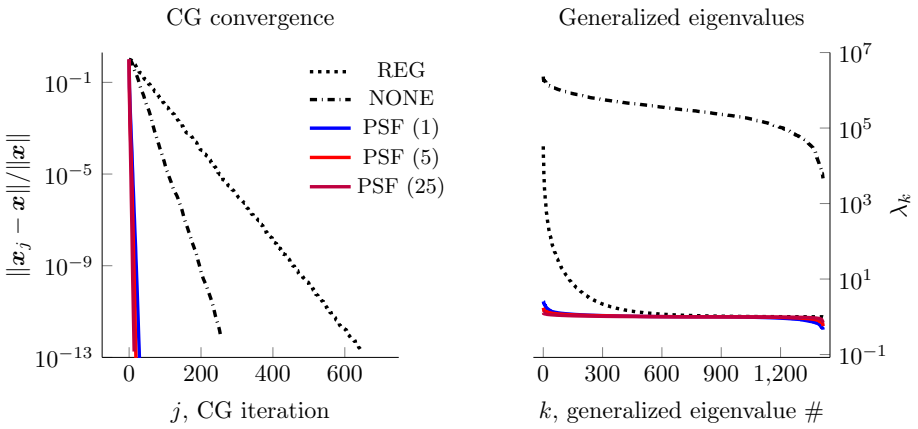


Fig. 10: (Ice sheet) Left: Convergence history for solving $\mathbf{Hx} = \mathbf{b}$ using PCG, where $\mathbf{b}$ has i.i.d. random entries drawn from the standard Gaussian distribution and $\mathbf{H}$ is evaluated at the solution of the inverse problem. Results in these figures are shown for the PSF-based preconditioners 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 $\mathbf{H}_{\text{gn}}$. Right: Generalized eigenvalues for generalized eigenvalue problem $\mathbf{Hu}_k = \lambda_k \widetilde{\mathbf{H}}\mathbf{u}_k$. Here, $\mathbf{H}$ is the Hessian and the matrices $\widetilde{\mathbf{H}}$ are the same Hessian approximations used in the left sub-figure, with NONE corresponding to the identity matrix.

noise	COND($\widetilde{\mathbf{H}}^{-1}\mathbf{H}$)				
level	REG	NONE	PSF (1)	PSF (5)	PSF (25)
25%	1.01e+3	2.96e+3	1.34e+0	1.30e+0	1.18e+0
11%	7.40e+3	1.05e+3	2.27e+0	1.55e+0	1.31e+0
5.0%	3.29e+4	4.96e+2	5.61e+0	3.06e+0	1.92e+0
2.2%	1.66e+5	8.89e+2	1.58e+1	8.07e+0	4.03e+0
1.0%	5.36e+5	1.61e+3	7.17e+1	1.93e+1	9.19e+0

Table 3: (Ice sheet) Condition number for $\widetilde{\mathbf{H}}^{-1}\mathbf{H}$ for the PSF-based preconditioners with 1, 5, and 25 batches (PSF (1), PSF (5), and PSF (25), respectively), no preconditioner (NONE) and regularization preconditioning (REG). All operators are evaluated at the soutions of the inverse problems for their respective noise levels.

Next, we build PSF (1), PSF (5), and PSF (25) preconditioners based on the Gauss-Newton Hessian evaluated at the converged solution $\mathbf{q}$ (note: k in PSF (k) refers to the number of batches; this is not to be confused with the noise levels which range over the same numerical values). We use PCG to solve a linear system with the Hessian as the coefficient operator and a right hand side vector with random independent and identically distributed (i.i.d.) entries drawn from the standard Gaussian distribution. In Figure 10 (left) 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-based preconditioners, with PSF (25) converging fastest, followed by PSF (5), followed by PSF (1), as expected. In Figure 10 (right) we show the generalized eigenvalues for the generalized eigenvalue problem $\mathbf{H}\mathbf{u} = \lambda\widetilde{\mathbf{H}}\mathbf{u}$. The matrix $\widetilde{\mathbf{H}}$ is one of the PSF (1), PSF (5), or PSF (25) Gauss-Newton Hessian approximations, the regularization Hessian (REG), or the identity matrix (NONE). With the PSF-based preconditioners, the generalized eigenvalues cluster near one, with more batches yielding better clustering.

In Table 3, we show the condition number of the preconditioned Hessian for noise levels ranging from 1% to 25%. Note that the condition number using PSF-based preconditioners is extremely small (ranging between 1 and 10) and relatively stable over this range of noise levels. As expected, PSF (25) outperforms PSF (5), which outperforms PSF (1). All PSF-based preconditioners outperform regularization and no preconditioning by several orders of magnitude for all noise levels.

7.2. Example 2: Inversion for the initial condition in an advective-diffusive transport problem. Here, we consider a time-dependent advection-diffusion equation in which we seek to infer the unknown spatially varying initial condition, q, from noisy observation of the full state at a final time, T. This PDE models advective-diffusive transport in a domain $\Omega \subset \mathbb{R}^d$, which is depicted in Figure 11. In this case, the state, $c(x,t)$, could be interpreted as the concentration of a contaminant. The problem description below closely follows [63, 75]. The domain boundaries $\partial\Omega$ include the outer boundaries as well as the internal boundaries of the rectangles, which represent buildings. The parameter-to-observable map $\mathcal{F}$ in this case maps an initial condition $q \in L^2(\Omega)$ to the concentration field at a final time, $c(x,T)$, through

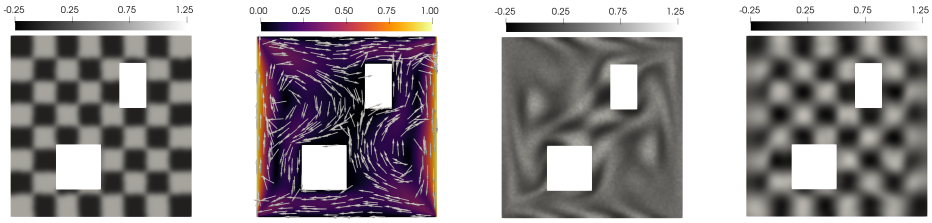
solution of the advection-diffusion equation given by

$$\begin{aligned} c_t - \kappa \Delta c + v \cdot \nabla c &= 0 && \text{in } \Omega \times (0,T), \\ c(\cdot, 0) &= q && \text{in } \Omega, \\ \kappa \nabla c \cdot \nu &= 0 && \text{on } \partial\Omega \times (0,T). \end{aligned} \tag{7.3}$$

Here, $\kappa > 0$ is a diffusivity coefficient, ν is the boundary unit normal vector, and $T > 0$ is the final time. The velocity field, $v : \Omega \to \mathbb{R}^d$, is computed by solving the steady-state Navier-Stokes equations for a two dimensional flow with Reynolds number 50, with boundary conditions $v(x) = (0, 1)$ on the left boundary, $v(x) = (0, -1)$ on the right boundary, and $v(x) = (0, 0)$ on the top and bottom boundaries, as in [63, Section 3]. We use a checkerboard image for the initial condition (Figure 11a) and add 5% multiplicative noise to generate a synthetic observation at the final time, T. The initial condition, velocity field, noisy observations, and reconstructed initial condition are shown in Figure 11. We use $\kappa = 3.2\text{e}{-1}$ and $T = 1.0$ for all results, except for Table 4 and Figure 12 where we vary κ and T.

In Table 4 we show the number of PCG iterations, j, required to solve the Newton linear system to a relative error tolerance of $\|\widehat{\mathbf{q}} - \widehat{\mathbf{q}}_j\| < 10^{-6}\|\widehat{\mathbf{q}}\|$. The solution of the Newton system to which we compare, $\widehat{\mathbf{q}}$, is found via another PCG iteration with a relative residual tolerance of 10^{-11}. We show results for T ranging from 0.5 to 2.0 and κ ranging from 10^{-4} to 10^{-3} using the PSF-based preconditioners with 1, 5, and 25 batches, regularization preconditioning, and no preconditioning. The results show that PSF-based preconditioning outperforms regularization preconditioning and no preconditioning in all cases except one. The exception is $T = 2.0$ and $\kappa = 1.0\text{e}{-3}$, in which PSF (1) performs slightly worse than regularization preconditioning but better than no preconditioning. Adding more batches yields better results, and the impact of adding more batches is more pronounced here than in the ice sheet example. For example, in the mid-range values $T = 1.0$ and $\kappa = 3.2\text{e}{-4}$, PSF (1), PSF (5), and PSF (25) require 1.3×, 2.3×, and 5.1× fewer PCG iterations, respectively, as compared to no preconditioning, and exhibit greater improvements as compared to regularization preconditioning. The PSF preconditioners perform best in the high rank regime where T is small, which makes sense given that short simulation times yield more localized impulse responses (see Figure 4). For example, for $\kappa = 3.2\text{e}{-4}$ using the PSF (5) preconditioner yields 140, 202, and 379 iterations for $T = 0.5$, 1.0, and 2.0, respectively. The performance of the PSF preconditioners as a function of κ does not have as clear of a trend. Reducing κ makes the impulse responses thinner and hence easier to fit in batches, but also increases the complexity of the impulse response shapes, which may reduce the accuracy of the RBF interpolation. The greatest improvements are seen for $T = 0.5$ and $\kappa = 1\text{e}{-3}$, for which PSF (25) requires roughly 10× and 20× fewer PCG iterations than no preconditioning and regularization preconditioning, respectively.

In Figure 13 (left) we show the convergence of PCG for solving $\mathbf{H}\mathbf{x} = \mathbf{b}$, where $\mathbf{b}$ has i.i.d. random entries drawn from a standard Gaussian distribution. The preconditioners used, $\widetilde{\mathbf{H}}$, are the PSF-based preconditioners with 1, 5, or 25 batches, the regularization Hessian, and the identity matrix (i.e., no preconditioning). The results show that PCG converges fastest with the PSF-based preconditioners, with more batches yielding faster convergence. In Figure 13 (right), we show the eigenvalues for the generalized eigenvalue problem $\mathbf{H}\mathbf{u} = \lambda\widetilde{\mathbf{H}}\mathbf{u}$, where the $\widetilde{\mathbf{H}}$ are the preconditioners stated above. With the PSF-based preconditioners the eigenvalues cluster near one, and more batches yields better clustering. With the regularization preconditioner the



(a) True q (b) v (c) Noisy observations (d) Reconstructed q

Fig. 11: (Advective-diffusive transport) (11a) True initial condition. (11b) Velocity field. Color indicates magnitude of velocity vector. (11c) Noisy observations of concentration at the final time. (11d) Reconstructed initial condition.

	κ	REG	NONE	PSF (1)	PSF (5)	PSF (25)
$T = 0.5$	1.0e-4	584	317	311	151	56
	3.2e-4	685	311	233	140	44
	1.0e-3	702	324	122	71	33
$T = 1.0$	1.0e-4	634	449	539	288	100
	3.2e-4	681	459	350	202	90
	1.0e-3	574	520	266	260	208
$T = 2.0$	1.0e-4	609	591	548	520	165
	3.2e-4	524	645	318	379	170
	1.0e-3	349	786	381	262	158

Table 4: (Advective-diffusive transport) Number of PCG iterations required to solve the Newton linear system to tolerance $||\widehat{\mathbf{q}}_j - \widehat{\mathbf{q}}|| < 10^{-6}||\widehat{\mathbf{q}}||$, where $\widehat{\mathbf{q}}_j$ is the jth iterate, and $\widehat{\mathbf{q}}$ is the solution of the Newton linear system. Iteration counts are shown for a variety of different diffusion parameters κ, simulation times T, and preconditioners.

trailing eigenvalues cluster near one, while the leading eigenvalues are amplified.

7.3. Example 3: Spatially varying blurring problem. Here, we define a PDE-free spatially varying blur problem, in which the impulse response, ϕ_x, is a bumpy blob that is centered near x, and is rotated and scaled in a manner that depends on x (see Figure 2). This blur problem is used in Sections 1 and 3 to visually illustrate various stages and aspects of the PSF-based method, and robustness (or lack thereof) of the PSF-based method to violations of the non-negative kernel assumption (Section 3.2). In this section, the blur problem is used to compare the PSF-based method to hierarchical off-diagonal low rank (HODLR) and global low rank (GLR) methods, to study convergence of the PSF-based method, and to investigate the effect of the ellipsoid size parameter τ. The closed form expression for the integral kernel is given by

$$\Phi(y,x) = (1 - af(y,x))g(x) \exp\left(-\frac{1}{2}(h(y,x)^T C^{-1} h(y,x)\right), \tag{7.4}$$

where $f(y,x) = \cos\left(h^1(y,x)/\sqrt{c_1/2}\right) \sin\left(h^2(y,x)/\sqrt{c_2/2}\right)$, with $h^i(y,x)$ the i^{th} component of $R(\theta(x))(y-x)$, with $R(\theta(x))$ a two-dimensional rotation matrix by

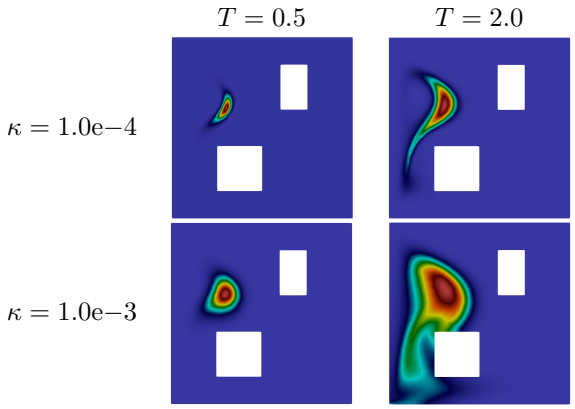


Fig. 12: (Advective-diffusive transport) Impulse responses for small and large diffusion parameters κ and simulation times T.

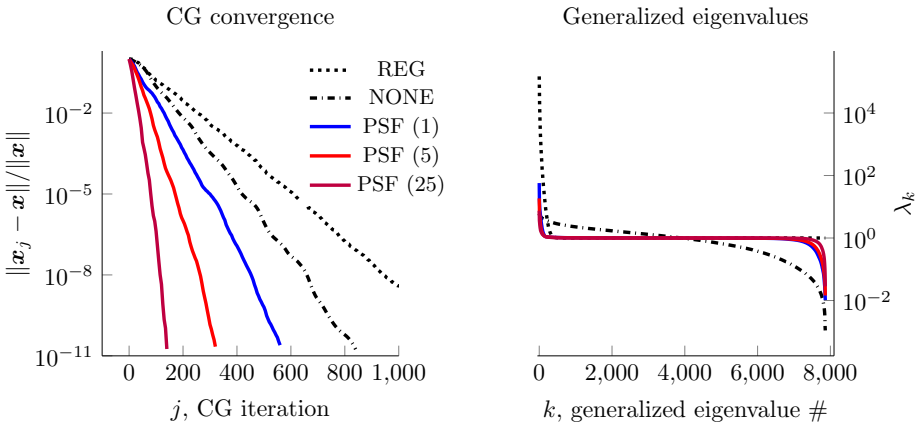


Fig. 13: (Advective-diffusive transport) Left: Convergence history for solving $\mathbf{Hx} = \mathbf{b}$ using PCG, where $\mathbf{b}$ has i.i.d. random entries drawn from the standard Gaussian distribution. Right: Generalized eigenvalues for generalized eigenvalue problem $\mathbf{Hu}_k = \lambda_k \widetilde{\mathbf{H}}\mathbf{u}_k$. Here, $\mathbf{H}$ is the Hessian and the preconditioner, $\widetilde{\mathbf{H}}$, is the PSF-based approximation for 1, 5, or 25 batches (PSF (1), PSF (5), and PSF (25), respectively), the regularization Hessian (REG), or the identity matrix (NONE).

angle $\theta(x) = (x^1 + x^2)\pi/2$, $g(x) = x^1(1-x^1)x^2(1-x^2)$, and with $C = L^2\,\mathrm{diag}(c_1, c_2)$. The constant L controls the width of the blob, and c_1/c_2 controls its aspect ratio. The constant a represents deviation from a Gaussian. When $a = 0$, ϕ_x is a Gaussian, and as a increases, ϕ_x becomes non-Gaussian. When $a > 1$, the integral kernel contains negative values, which allows us to study the robustness of the PSF-based method to violations of Assumption 3 (Section 3.2).

In Table 5 we compare the cost to approximate the blur kernel from Equation (7.4) using the PSF-based method, the randomized HODLR (hierarchical off diagonal low rank) method [58, 45] with 8 levels, and GLR (global low rank) approximation using

	Error tol.	#applies PSF	#applies HODLR	#applies RSVD
$L = 1$	20%	11	592	354
	10%	16	772	520
	5%	22	924	674
$L = 1/2$	20%	8	852	1316
	10%	9	1144	1916
	5%	12	1404	2456
$L = 1/3$	20%	7	932	2624
	10%	8	1264	3734
	5%	8	1520	4660

Table 5: (Blur) Comparison of cost to approximate the blur kernel from Equation (7.4) using the PSF-based method, the randomized HODLR (hierarchical off diagonal low rank) method, and GLR (global low rank) approximation using randomized SVD. The quantity L scales the width of the impulse responses, hence it influences the rank of the operator. Large L means low rank, and small L means high rank. The second column ("Error tol") is the relative error in the approximation of the kernel measured in the Frobenius norm, $||\boldsymbol{\Phi} - \widetilde{\boldsymbol{\Phi}}||_{\text{Fro}}/||\boldsymbol{\Phi}||_{\text{Fro}}$. The remaining three columns show the number of operator applies required to achieve the given error tolerances, using the PSF, HODLR, and GLR methods.

double-pass randomized SVD [44]. For these results we vary the quantity L to scale the width of the impulse responses and hence the rank of the operator. For each case, we calculate the relative error in the approximation of the kernel measured in the Frobenius norm, $||\boldsymbol{\Phi} - \widetilde{\boldsymbol{\Phi}}||_{\text{Fro}}/||\boldsymbol{\Phi}||_{\text{Fro}}$ and show the number of operator applications required to achieve 20%, 10%, and 5% relative error by each method. The results reveal superior performance of the PSF method as compared to the HODLR and GLR methods for all cases. We note that as we increase the rank and decrease the error tolerance, the performance of the HODLR and GLR methods deteriorates.

In Figure 14 we show the convergence of the PSF-based method on the blur kernel as a function of the total number of impulse responses (left), and the number of batches (right). We show convergence for several ellipsoid size parameters τ, ranging from 2.0–4.0. The results in Figure 14 (left) show that the relative error decreases as constant $\times$ $(\#\text{impulse responses})^{-1}$ suggesting linear convergence. The linear convergence stalls at a limit that depends on τ. Increasing τ lowers this limit, allowing the PSF-based method to achieve higher accuracy. In Figure 14 (right), the results show that before this limit is reached, the convergence is faster for smaller τ in terms of the number of batches. This is expected because smaller τ results in more impulse responses per batch. Larger τ causes the PSF-based method to converge more slowly than smaller τ, but with larger τ the PSF-based method stalls at a lower level of error than it does with smaller τ (see, e.g., $\tau = 4.0$ vs. $\tau = 2.5$).

8. Conclusions. We presented an efficient matrix-free PSF-based method for approximating operators with locally supported non-negative integral kernels. The PSF-based method requires access to the operator only via application of the operator to a small number of vectors. The idea of the PSF-based 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

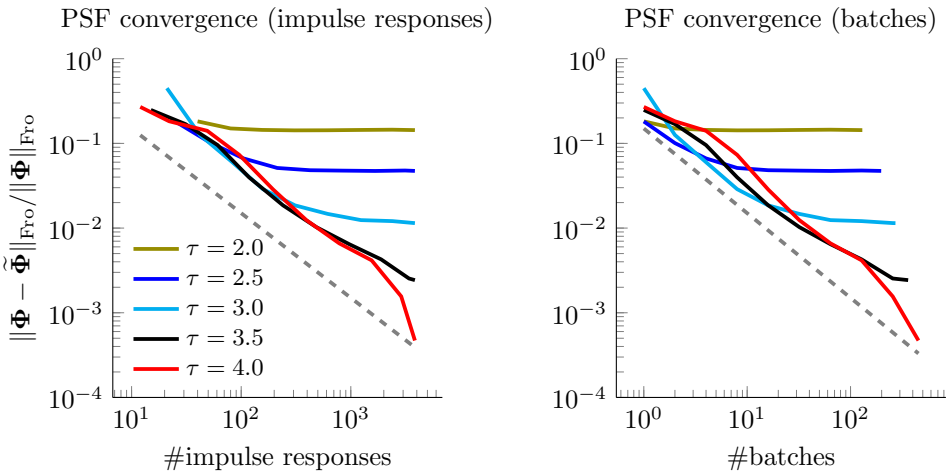


Fig. 14: (Blur) Relative error for different ellipsoid size parameters, τ, vs. the total number of impulse responses (left), and the number of batches (right). The dashed gray lines show linear convergence rates, i.e., constant $\times$ $(\#\text{impulse responses})^{-1}$ on the left, and constant $\times$ $\#\text{batches}^{-1}$ on the right.

mean displacement invariance," which generalizes 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 moment method 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 PSF-based method to approximate the Gauss-Newton Hessians in an ice sheet flow inverse problem governed by a linear Stokes PDE, and an advective-diffusive transport inverse problem governed by an advection-diffusion PDE. We saw that preconditioners based on the PSF-based approximation cluster the eigenvalues of the preconditioned Hessian near one, and allow us to solve the inverse problems using roughly 5×–10× fewer PDE solves. For larger domains with more observations, the rank of the data misfit Hessian will increase, while the locality of impulse responses will remain the same. Hence, we expect the speedup will be even greater for such problems. Although the PSF-based method is not applicable to all Hessians, it is applicable to many Hessians of practical interest. For these Hessians, the PSF-based method offers a *data scalable* alternative to conventional low rank approximation, due to the ability to form high rank approximations of an operator using a small number of operator applications, and thus PDE solves.

Acknowledgments. We thank J.J. Alger, Longfei Gao, Mathew Hu, and Rami Nammour for helpful discussions. We thank Nicole Aretz, Trevor Heise, and the anonymous reviewers for editing suggestions. We thank Georg Stadler for help with the domain setup for the ice sheet problem.

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] T. Arbogast and J. L. Bona, *Methods of Applied Mathematics*, University of Texas at Austin, 2008. Lecture notes in applied mathematics.

[9] H. T. Banks and K. Kunisch, *Estimation techniques for distributed parameter systems*, Springer, 1989.

[10] M. Bebendorf and S. Rjasanow, *Adaptive low-rank approximation of collocation matrices*, Computing, 70 (2003), pp. 1–24.

[11] J. L. Bentley, *Multidimensional binary search trees used for associative searching*, Communications of the ACM, 18 (1975), pp. 509–517.

[12] 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.

[13] S. Börm, L. Grasedyck, and W. Hackbusch, *Introduction to hierarchical matrices with applications*, Engineering analysis with boundary elements, 27 (2003), pp. 405–422.

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

[15] 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.

[16] T. F. Chan and T. P. Mathew, *Domain decomposition algorithms*, Acta Numerica, 3 (1994), pp. 61–143.

[17] J. Chen and M. L. Stein, *Linear-cost covariance functions for Gaussian random fields*, Journal of the American Statistical Association, (2021), pp. 1–18.

[18] 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.

[19] 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.

[20] J. C. De los Reyes, *Numerical PDE-constrained optimization*, Springer, 2015.

[21] L. Demanet, P.-D. Létourneau, N. Boumal, H. Calandra, J. Chiu, and S. Snelson, *Matrix probing: a randomized preconditioner for the wave-equation Hessian*, Applied and Computational Harmonic Analysis, 32 (2012), pp. 155–168.

[22] 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.

[23] 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.

[24] S. C. Eisenstat and H. F. Walker, *Choosing the forcing terms in an inexact newton method*, SIAM Journal on Scientific Computing, 17 (1996), pp. 16–32.

[25] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of Inverse Problems*, vol. 375 of Mathematics and Its Applications, Springer Netherlands, 1996.

[26] C. Ericson, *Real-time collision detection*, Crc Press, 2004.

[27] P. Escande and P. Weiss, *Sparse wavelet representations of spatially varying blurring oper-*

ators, SIAM Journal on Imaging Sciences, 8 (2015), pp. 2976–3014.
[28] P. Escande and P. Weiss, *Approximation of integral operators using product-convolution expansions*, Journal of Mathematical Imaging and Vision, 58 (2017), pp. 333–348.
[29] 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.
[30] P. Escande, P. Weiss, and F. Malgouyres, *Spatially varying blur recovery. Diagonal approximations in the wavelet domain*, in Proceedings of ICPRAM, 2013.
[31] A. Fichtner and T. v. Leeuwen, *Resolution analysis by random probing*, Journal of Geophysical Research: Solid Earth, 120 (2015), pp. 5549–5573.
[32] 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.
[33] 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.
[34] 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.
[35] M. Gentile, F. Courbin, and G. Meylan, *Interpolating point spread function anisotropy*, Astronomy & Astrophysics, 549 (2013).
[36] M. G. Genton, D. E. Keyes, and G. Turkiyyah, *Hierarchical decompositions for the computation of high-dimensional multivariate normal probabilities*, Journal of Computational and Graphical Statistics, 27 (2018), pp. 268–277.
[37] 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.
[38] 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.
[39] 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 International Conference on Information Fusion, IEEE, 2012, pp. 396–401.
[40] L. Grasedyck and W. Hackbusch, *Construction and arithmetics of H-matrices*, Computing, 70 (2003), pp. 295–334.
[41] 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.
[42] W. Hackbusch, *A sparse matrix arithmetic based on H-matrices. Part I: Introduction to H-matrices*, Computing, 62 (1999), pp. 89–108.
[43] W. Hackbusch, *Hierarchical matrices: algorithms and analysis*, vol. 49, Springer, 2015.
[44] 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.
[45] 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 initialization*, Inverse Problems, 39 (2023), p. 085006.
[46] F. J. Herrmann, P. Moghaddam, and C. C. Stolk, *Sparsity-and continuity-promoting seismic image recovery with curvelet frames*, Applied and Computational Harmonic Analysis, 24 (2008), pp. 150–173.
[47] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, *Optimization with PDE constraints*, vol. 23, Springer Science & Business Media, 2008.
[48] 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.
[49] J. Kaipio and E. Somersalo, *Statistical and Computational Inverse Problems*, vol. 160 of Applied Mathematical Sciences, Springer-Verlag New York, 2005.
[50] 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*, ACM Transactions on Mathematical Software, (2023).
[51] R. Kriemann, *HLIBpro user manual*, Max-Planck-Institute for Mathematics in the Sciences, Leipzig, 48 (2008).
[52] R. Kriemann, *H-LU factorization on many-core systems*, Computing and Visualization in

Science, 16 (2013), pp. 105–117.
[53] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, vol. 6, SIAM, 1998.
[54] J. Levitt and P.-G. Martinsson, *Linear-complexity black-box randomized compression of hierarchically block separable matrices*, arXiv preprint arXiv:2205.02990, (2022).
[55] 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.
[56] 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.
[57] 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.
[58] P.-G. Martinsson, *Compressing rank-structured matrices via randomized sampling*, SIAM Journal on Scientific Computing, 38 (2016), pp. A1959–A1986.
[59] P.-G. Martinsson and J. A. Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numerica, 29 (2020), pp. 403–572. Section 20.
[60] 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.
[61] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, 1999.
[62] 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.
[63] N. Petra and G. Stadler, *Model variational inverse problems governed by partial differential equations*, Tech. Report 11-05, The Institute for Computational Engineering and Sciences, The University of Texas at Austin, 2011.
[64] 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.
[65] 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.
[66] Y. Saad, *Iterative methods for sparse linear systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, second ed., 2003.
[67] B. Smith, P. Bjorstad, and W. Gropp, *Domain Decomposition Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, 1996.
[68] 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.
[69] T. Steihaug, *Local and superlinear convergence for truncated iterated projections methods*, Mathematical Programming, 27 (1983), pp. 176–190.
[70] A. Stuart, *Inverse problems: a Bayesian perspective*, Acta Numerica, 19 (2010), pp. 451–559.
[71] W. W. Symes, *Approximate linearized inversion by optimal scaling of prestack depth migration*, Geophysics, 73 (2008), pp. R23–R35.
[72] A. Tarantola, *Inverse Problem Theory and Methods for Model Parameter Estimation*, SIAM, Philadelphia, PA, 2005.
[73] A. Toselli and O. Widlund, *Domain decomposition methods-algorithms and theory*, vol. 34, Springer Science & Business Media, 2004.
[74] J. Trampert, A. Fichtner, and J. Ritsema, *Resolution tests revisited: the power of random numbers*, Geophysical Journal International, 192 (2013), pp. 676–680.
[75] 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, 47 (2021), pp. 1–34.
[76] C. R. Vogel, *Computational Methods for Inverse Problems*, Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2002.
[77] H. Wendland, *Scattered data approximation*, vol. 17, Cambridge University Press, 2004.
[78] 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.