SIAM J. SCI. COMPUT.
Vol. 41, No. 4, pp. A2296–A2328

SCALABLE MATRIX-FREE ADAPTIVE PRODUCT-CONVOLUTION APPROXIMATION FOR LOCALLY TRANSLATION-INVARIANT OPERATORS[*]

NICK ALGER[†], VISHWAS RAO[‡], AARON MYERS[†],
TAN BUI-THANH[†][§], AND OMAR GHATTAS[†][¶]

Abstract. We present an adaptive grid matrix-free operator approximation scheme based on a "product-convolution" interpolation of convolution operators. This scheme is appropriate for operators that are locally translation-invariant, even if these operators are high rank or full rank. Such operators arise in Schur complement methods for solving partial differential equations (PDEs), as Hessians in PDE-constrained optimization and inverse problems, as integral operators, as covariance operators, and as Dirichlet-to-Neumann maps. Constructing the approximation requires computing the impulse responses of the operator to point sources centered on nodes in an adaptively refined grid of sample points. A randomized a posteriori error estimator drives the adaptivity. Once constructed, the approximation can be efficiently applied to vectors using the fast Fourier transform. The approximation can be efficiently converted to hierarchical matrix (H-matrix) format, then inverted or factorized using scalable H-matrix arithmetic. The quality of the approximation degrades gracefully as fewer sample points are used, allowing cheap lower quality approximations to be used as preconditioners. This yields an automated method to construct preconditioners for locally translation-invariant Schur complements. We directly address issues related to boundaries and prove that our scheme eliminates boundary artifacts. We test the scheme on a spatially varying blurring kernel, on the nonlocal component of an interface Schur complement for the Poisson operator, and on the data misfit Hessian for an advection dominated advection-diffusion inverse problem. Numerical results show that the scheme outperforms existing methods.

Key words. convolution, operator approximation, PDE constrained inverse problems, hierarchical matrix, matrix-free, data scalability

AMS subject classifications. 41A05, 41A35, 42A61, 42A85, 47A58, 49K20, 65F08, 65J22, 65N21, 65T50, 94A12

DOI. 10.1137/18M1189324

1. Introduction.

We present an adaptive product-convolution scheme for approximating locally translation-invariant operators. That is, operators $A : l^2(\mathbf{\Omega}) \rightarrow l^2(\mathbf{\Omega})$ satisfying

$$A\,[y, x] \approx A\,[y - x + p, p] \tag{1}$$

[*] Submitted to the journal's Methods and Algorithms for Scientific Computing section May 23, 2018; accepted for publication (in revised form) May 17, 2019; published electronically July 23, 2019.
https://doi.org/10.1137/18M1189324

Funding: This work was partially supported by AFOSR FA9550-17-1-0190, NSF ACI-1550593 and CBET-1508713, and DOE DE-SC0010518, DE-SC0009286, and DE-SC0019393. The work of the third and fourth authors was partially supported by the DOE grant DE-SC0018147. The work of the second author was partially supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research Program under contract DE-AC02-06CH11357.

[†] Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX 78712 (nalger@ices.utexas.edu, aaron@ices.utexas.edu).

[‡] Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL 60439 (vhebbur@anl.gov).

[§] Institute for Computational Engineering and Sciences, Department of Aerospace Engineering and Engineering Mechanics, and Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX 78712 (tanbui@ices.utexas.edu).

[¶] Institute for Computational Engineering and Sciences, and Departments of Geological Sciences and Mechanical Engineering, The University of Texas at Austin, Austin, TX 78712 (omar@ices.utexas.edu).

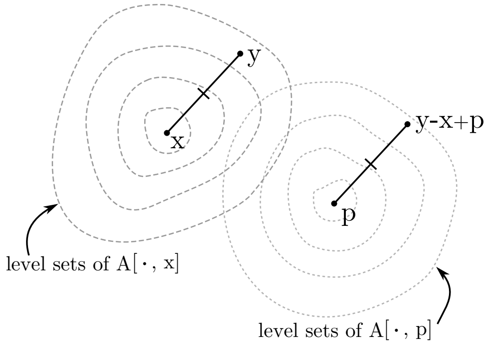


FIG. 1. *Our product-convolution scheme is suitable for operators that are locally translation-invariant. That is, operators for which the impulse response at y to a point source at x is similar to the impulse response at $y - x + p$ given a point source of equal magnitude at p if x is near p.*

whenever x is not too far from p (see Figure 1). Here we consider the case in which $\mathbf{\Omega}$ is a box[1] in $\mathbb{Z}^d$. Our scheme is well suited for approximating or preconditioning operators that arise in Schur complement techniques [43, 53] for solving partial differential equations (PDEs), reduced Hessians in PDE-constrained optimization and inverse problems, integral operators, covariance operators with spatially varying kernels, and Dirichlet-to-Neumann maps or other Poincaré–Steklov operators in multiphysics problems. These operators are typically dense and implicitly defined, and often do not admit a global low-rank approximation, making them difficult to approximate with standard techniques.

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

$$\varphi_p\left[z\right] = \left(A\delta_p\right)\left[z+p\right], \quad z \in \mathbf{\Omega} - p. \tag{2}$$

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

$$Af \approx \widetilde{A}f \coloneqq \sum_{k=1}^{r} \varphi_k^E * (w_k \cdot f), \tag{3}$$

where the w_k are locally supported weighting functions that overlap and form a partition of unity, "$\cdot$" denotes pointwise multiplication of functions, $*$ denotes convolution (see section 1.3 for more details on notation), and the functions φ_k^E are modified[2] versions of the (translated, recentered) impulse responses φ_{p_k} associated with a collection of sample points, p_k. Each point p_k is contained within the support of the associated weighting function w_k.

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

[2] To address issues with boundary artifacts, we construct φ_k^E by extending the function φ_{p_k} outside of $\mathbf{\Omega} - p_k$ using information from neighboring functions, φ_{p_j} (more on this in section 2.5).

The basic form of (3) is known as a product-convolution approximation, and is well established in the literature (see section 1.2.2). Here we improve upon existing schemes by

- adaptively and automatically choosing the sample points p_k;
- addressing issues related to boundaries.

In section 2 we derive our scheme, explain how we choose p_k, and detail the process for constructing w_k and φ_k^E. In section 3 we detail how $\widetilde{A}$ can be used once constructed, including how to efficiently convert it to hierarchical matrix (H-matrix) format. In section 4 we perform an a priori error analysis of our scheme. We demonstrate our scheme numerically in section 5 and give concluding remarks in section 6. In the remainder of this section we summarize our results (section 1.1), review existing work (section 1.2), and define our setting and notation (section 1.3).

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

Often the ultimate goal is to solve linear systems with A as the coefficient operator. Krylov methods can be used to solve these systems [21]. However, the convergence of Krylov methods depends heavily on the spectral structure of the coefficient operator, leading to slow convergence when A is ill-conditioned. To address this, we explain how $\widetilde{A}$ can be efficiently converted to H-matrix format. Once in H-matrix format, $\widetilde{A}$ can be efficiently factorized or inverted using H-matrix arithmetic, then used as a preconditioner. Alternatively, one can build circulant preconditioners from $\widetilde{A}$ [22, 47].

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

Boundaries introduce two difficulties for product-convolution schemes:

1. *Boundary artifacts:* The impulse response associated with p_k is naturally defined on $\mathbf{\Omega} - p_k$, but the product-convolution scheme (3) requires it to be defined on a larger set. The three standard extension techniques—extending the impulse response by zero, reflecting it across the boundary, or replicating it periodically—all create boundary artifacts wherever artificial data are used in place of undefined data.
2. *Boundary effects:* The underlying operator may fail to be translation-invariant near boundaries due to boundary conditions or other physically meaningful effects.

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

In Theorem 5, we prove that the error in our scheme is controlled by the local failure of translation-invariance in A. This, together with adaptivity, implies convergence: our scheme will continue to add new sample points until it achieves the desired error tolerance. The more translation-invariant A is, the fewer sample points will be used. Additionally, Theorem 5 implies that our approximation scheme will not introduce boundary artifacts. Without our impulse response extension procedure, the bound in Theorem 5 would fail near the boundary.

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

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

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

Although our scheme will eventually converge to any desired error tolerance, it is most useful for computing moderately accurate approximations (say, 80% to 99% accurate) of "difficult" operators that are poorly approximated by standard techniques. In our numerical tests, we observe that the convergence slows beyond this accuracy. Moderate accuracy approximation is sufficient for many engineering applications, and is ideal for building preconditioners.

1.2. Existing work.

The most widely used, robust, and general purpose matrix-free operator approximation schemes are based on low-rank approximation (section 1.2.1). However, many important operators in PDEs, PDE-constrained optimization and inverse problems, and integral equations are not low rank. Our scheme fits within a class of operator approximation schemes based on interpolation of convolution operators (section 1.2.2). Hierarchical matrices (section 1.2.3) are another well-established operator approximation format; they are simultaneously a tool we use (section 3.4), and an alternative to our scheme.

1.2.1. Low-rank approximation.

Low-rank approximations—matrix factorizations of the form $A \approx BC$, where B is $N \times r$ (tall), and C is $r \times N$ (wide)—can be efficiently constructed in a matrix-free setting by using Krylov methods (Lanczos or Arnoldi), randomized SVD [38] or CUR decomposition/skeletonization [24, 34, 45, 57]. Although low-rank approximations have been used for Dirichlet-to-Neumann maps [19, 20], full-rank or high-rank operators typically still retain a high rank after being restricted to a boundary as a Schur complement. Likewise, although low-rank approximations have been used to approximate the (prior preconditioned) Hessian of the data misfit term in PDE-constrained inverse problems [18, 26, 30, 50, 54], the numerical rank of this term grows as the informativeness of the data in the inverse problem grows [4], making low-rank approximation inefficient for highly informative data. Even when the operator is low rank in the sense that $r \ll N$, the cost of computing the low-rank approximation may be prohibitive. For example, a low-rank approximation of the Hessian in a PDE constrained optimization or inverse problem requires $O(r)$ linearized forward/adjoint PDE solves, so that for large-scale problems with, e.g., N of order 10^6, even a compression of 0.1% still means that thousands of forward solves are needed, which is often an expensive proposition [17, 23, 41]. Our scheme is motivated by a desire to go beyond low-rank approximation in these applications.

1.2.2. Convolution interpolation.

Since the linear operator that performs a convolution may be numerically full rank (e.g., convolution with δ_0: the identity operator) or high rank (e.g., convolution with a Gaussian with a small width), interpolation of convolution operators can, where applicable, be used to approximate dense operators with far fewer terms than the rank of the operator.

Operator approximation schemes based on weighted sums of convolution operators with spatially varying weights ("convolution interpolations") fall into two categories: product-convolution schemes where one performs elementwise products with weighting functions first and convolutions second, and convolution-product schemes where this order is reversed:

$$Af \approx \underbrace{\sum_{k=1}^{r} \psi_k * (\omega_k \cdot f)}_{\text{product-convolution}} \quad \text{versus} \quad \underbrace{\sum_{k=1}^{r} \omega_k \cdot (\psi_k * f)}_{\text{convolution-product}}. \tag{4}$$

The terms "product-convolution" and "convolution-product" refer to the general format of the approximations in (4), where ψ_k and ω_k could be any functions. For us, ψ_k are (modified) impulse response functions and ω_k form a partition of unity. Since the entries of a convolution operator $M : f \mapsto \psi * f$ are $M[y,x] = \psi[y-x]$, product-convolution and convolution-product approximations have the following (y,x) matrix entries:

$$A[y,x] \approx \underbrace{\sum_{k=1}^{r} \omega_k[x]\,\psi_k[y-x]}_{\text{product-convolution}} \quad \text{versus} \quad \underbrace{\sum_{k=1}^{r} \omega_k[y]\,\psi_k[y-x]}_{\text{convolution-product}}. \tag{5}$$

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

$$\underbrace{\sum_{k=1}^{r} \psi_k * (\omega_k \cdot f)}_{\widetilde{A}f} \quad \xleftrightarrow{\text{adjoint}} \quad \underbrace{\sum_{k=1}^{r} \overline{\omega}_k \cdot \left(\operatorname{flip}\left(\overline{\psi_k}\right) * f\right)}_{\widetilde{A}^* f}, \tag{6}$$

where $\operatorname{flip}(\psi)[x] := \psi[-x]$, and the overline indicates the complex conjugate. Here we use a product-convolution scheme.

Convolution interpolation schemes have been used in image restoration and deblurring [29, 47, 55] in photography [56], astronomy [1, 31, 52], and microscopy [51], as well as in wireless communication signal processing [40], ultrasound imaging [48], systems biology [33], and Hessian approximation in seismic inversion [58].[3] Aside from the application, convolution interpolation schemes differ based on how they construct the functions ω_k and ψ_k. For a comprehensive overview of existing schemes, we refer the reader to the summaries in [27, 28, 32].

Existing schemes can be categorized based on whether the span of the functions ω_k is fixed, or the span of the functions ψ_k is fixed, or both of the spans are fixed, or neither of the spans are fixed. Schemes then attempt to find the remaining (not fixed) functions and the coefficients for linear combinations of the fixed functions so that the error in the resulting operator approximation is small. Established choices for the span of the functions ψ_k include the span of impulse responses of A to point sources at a collection of fixed locations (we do this), subspaces of this span, and the span of functions with known analytic forms (e.g., Gaussians, spherical harmonics). Established choices for the span of the functions ω_k include spans of Fourier modes, piecewise polynomials on a regular grid (e.g., piecewise constants, piecewise linear functions, B-splines), wavelets, radial basis functions [11], and functions based on kriging.

On one hand, existing schemes in which the functions ω_k are not fixed[4] require more access to A than just the ability to apply it to vectors. On the other hand, existing schemes in which the functions ω_k are fixed do not permit spatial adaptivity, with one exception. This includes existing sectioning approaches that partition the domain into pieces on a regular grid, then use different functions ψ_k for each piece [47]. The exception is [8], which, like this paper, proposes partitioning the domain with an adaptively refined grid. However, [8] only proposes the concept; they do not provide practical algorithms to perform the adaptivity.

In [10], matrix probing [25] using basis matrices with (y, x) entries that take the form $\omega_k\,[x+y]\,\psi_k\,[y-x]$ is used to approximate the exterior Dirichlet-to-Neumann map for a forward wave propagation problem. This approximation could be viewed as a middle ground between a product-convolution scheme and a convolution-product scheme, which would correspond to basis matrices of the form $\omega_k\,[x]\,\psi_k\,[y-x]$ and $\omega_k\,[y]\,\psi_k\,[y-x]$, respectively. After constructing the approximation, [10] proposes converting it to H-matrix format for further use. Our approximation is different, but we also propose the same subsequent H-matrix conversion.

1.2.3. Hierarchical matrices.

Hierarchical matrices [35] are matrices that may be full rank, but the blocks of the matrix associated with clusters of degrees of freedom that are far away from each other (or satisfy some other admissibility condition) are low rank. This structure allows for compressed storage and fast (nearly linear) matrix arithmetic, including matrix inversion and factorization. Special subclasses of H-matrices such as H^2-matrices [37] (among others) allow for greater compression and faster matrix arithmetic. For an overview of H- and H^2-matrices, see [13, 36].

Classical H-matrix construction techniques require access to the matrix entries of A, and hence are not applicable here. There exist matrix-free H-matrix construction techniques based on a recursive “peeling-process” [44], but these techniques have several subtle limitations. Although asymptotically scalable in theory, in practice the peeling process must apply the original operator to a large number of vectors.

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

[4] The terminology for this is potentially confusing: in the literature, computed (rather than fixed) functions ω_k are known as “adaptive” weighting functions, but this is unrelated to our “adaptive grid” weighting functions.

Furthermore, attempting to construct a less accurate approximation by applying the original operator to fewer vectors is not advisable (unlike our scheme where this is fine). Errors at any step of the peeling process compound during subsequent steps. Finally, the peeling process is purely algebraic. This makes the peeling process more general, at the cost of potentially being less efficient than specialized schemes (like ours) that take advantage of local translation-invariance or other properties of the operator being approximated.

1.3. Setting and notation.

We work in l^2 spaces on $\mathbb{Z}^d$ or subsets of $\mathbb{Z}^d$; these spaces arise when one discretizes a function on a continuous domain using a regular grid. Norms are denoted by $\|\cdot\|$, or occasionally $\|\cdot\|_{l^2(X)}$ if the domain is not clear from context. For linear operators we always use the Frobenius norm (square root of the sum of squares of all entries of the matrix representation of the linear operator).

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

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

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

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

The evaluation of f at x is denoted $f[x]$, and $f[\mathbf{C}] \in l^2(\mathbf{C} - c_{\min})$, with $(f[\mathbf{C}])[x] := f[x + c_{\min}]$. Likewise, $M[y, x]$ is the (y, x) "matrix entry" of M, and $M[\mathbf{T}, \mathbf{S}] \in l^2((\mathbf{T} - t_{\min}) \times (\mathbf{S} - s_{\min}))$ with $(M[\mathbf{T}, \mathbf{S}])[y, x] := M[y + t_{\min}, x + s_{\min}]$. That is, $M[\mathbf{T}, \mathbf{S}]$ is the $\mathbf{T}, \mathbf{S}$ "block" of M. A dot within indexing brackets, as in $M[\mathbf{C}, \ \cdot\ ]$ or $M[\ \cdot\ , \mathbf{C}]$, indicates the matrix of all columns or rows of M corresponding to points in $\mathbf{C}$, respectively. The action of a linear operator M on a vector f is denoted by Mf. We write M^* to denote the adjoint of M. That is, $M^*[y, x] = \overline{M[x, y]}$, where the overline indicates the complex conjugate.

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

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

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

$$(\psi * f)[y] := \sum_{x \in \mathbb{Z}^d} f[x]\, \psi[y - x]. \tag{7}$$

If the domains of functions f, ψ are only subsets of $\mathbb{Z}^d$, we define their convolution to be the result of extending f, ψ by zero so that they are defined on all of $\mathbb{Z}^d$, then convolving them using formula (7). We use the term "convolution rank" to denote the number of terms in a weighted sum of convolution operators (e.g., r in (3)).

We define the functions

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

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

2. The adaptive product-convolution approximation.

As discussed in section 1, if A were translation-invariant (i.e., if (1) held with equality for all $x, y \in \mathbb{Z}^d$), then A would be the convolution operator defined by the action $Af = \varphi_p * f$, where φ_p is the impulse response of A at p, as defined in (2). For example, the solution operator for a homogeneous PDE on an unbounded domain is translation-invariant, and φ_p is the Green's function for the PDE. Of course, translation-invariant operators are rare in practice. It is more common for A to only be *approximately* translation-invariant (see Figure 1), and for the approximate translation-invariance to be valid only *locally.* That is,

$$A[p + y - x, p] \approx A[y, x] \quad \text{when } x \in U \tag{8}$$

for some neighborhood U consisting of points "near" p. We will provide a rigorous analysis of approximation errors in section 4; for now we leave the exact nature of this approximate equality ($\approx$) intentionally vague. Just as translation-invariance of A implies that A is a convolution operator, local approximate translation-invariance of A implies that A can be locally approximated by a convolution operator. Specifically, (8) implies

$$Ag \approx \varphi_p * g \quad \text{when } \operatorname{supp}(g) \subset U. \tag{9}$$

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

$$Ag \approx \varphi_k * g \quad \text{when } \operatorname{supp}(g) \subset U_k, \quad k = 1, \ldots, r, \tag{10}$$

then A can be globally approximated as follows:

$$Af = A \sum_{k=1}^{r} w_k \cdot f = \sum_{k=1}^{r} A(w_k \cdot f) \approx \sum_{k=1}^{r} \varphi_k * (w_k \cdot f). \tag{11}$$

The first equality follows from the partition unity property of the functions w_k, the second follows from the linearity of A, and the approximate equality follows from the local approximation property (10) and the fact that $\operatorname{supp}(w_k \cdot f) \subset U_k$.

2.1. Overview of the approximation. The previous derivation leads us to approximate A with the following *product-convolution approximation*,

$$\widetilde{A}f := \sum_{k=1}^{r} \varphi_k^E * (w_k \cdot f), \tag{12}$$

where

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

  $$\varphi_k\left[z\right] = \left(A\delta_{p_k}\right)\left[z + p_k\right], \quad z \in \mathbf{\Omega} - p_k, \tag{13}$$

  for a collection of *sample points* $\{p_k\}_{k=1}^r$;
- the sample points $\{p_k\}_{k=1}^r$ reside in a collection of overlapping sets $\{U_k\}_{k=1}^r$ that cover $\mathbf{\Omega}$:

  $$p_k \in U_k \text{ for } k = 1, \dots, r \quad \text{and} \quad \mathbf{\Omega} \subset \bigcup_{k=1}^{r} U_k;$$
- $\{w_k\}_{k=1}^r$ is a partition of unity subordinate to the cover:

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

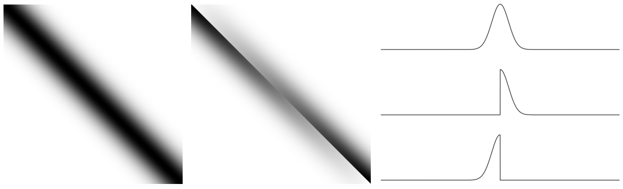
Our scheme is defined by the points p_k, the sets U_k, the partition of unity weighting functions w_k, and the extended impulse response functions φ_k^E.

In general, translation-invariance varies spatially. By this, we mean that the size of the neighborhood U on which the error in (8) is sufficiently small depends on the location of U. To fix ideas, suppose that A is the solution operator for an inhomogeneous elliptic PDE. In this case, the size of U will typically be small if the coefficient in the PDE varies over short length scales within U, and large if the coefficient varies over large length scales within U. In order to capture such spatial variations in translation-invariance while minimizing the number of sample points used, we choose p_k and U_k adaptively (sections 2.2 and 2.3). A randomized adjoint based a posteriori error estimator (section 2.6) drives the adaptivity.

Due to boundary effects, translation-invariance typically fails in directions perpendicular to a boundary, but holds in directions parallel to that boundary. For example, let φ_p be the Green's function at p for a homogeneous PDE on an infinite half-space. Although φ_p changes as p approaches the boundary, by symmetry it does not change as p moves parallel to the boundary. In order to address this direction-dependent translation-invariance, we refine anisotropically, subdividing preferentially in directions that φ_p changes the most as a function of p (section 2.7).

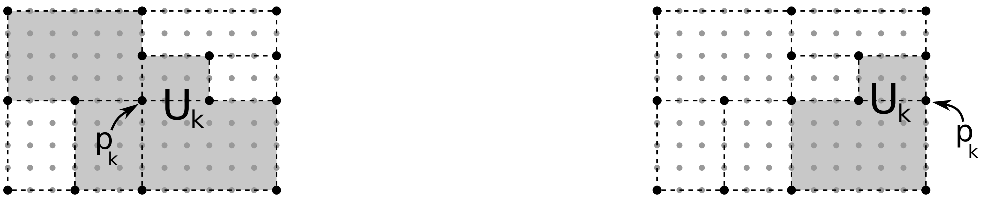
The adaptive refinement procedure creates unusually shaped neighborhoods U_k. We construct harmonic weighting functions, w_k, on these sets by solving local Laplace problems (section 2.4).

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



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

FIG. 2. *Extending impulse responses by zero leads to boundary artifacts even if A is, itself, a convolution operator. Here A, Figure* 2(a)*, takes a function defined on $[1, N]$, extends it by zero to $\mathbb{Z}$, convolves it with a Gaussian φ, Figure* 2(c)*, then restricts the result to $[1, N]$. The approximation, $\widetilde{A}$, Figure* 2(b)*, linearly interpolates between convolution with φ_{left} at* 1 *and φ_{right} at N, where φ_{left} and φ_{right}, Figure* 2(c)*, are the impulse responses of A to point sources centered at* 1 *and N, respectively, with extension by zero used as needed. Black indicates value* 1*, and white indicates value* 0 *in Figures* 2(a) *and* 2(b)*.*



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

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

FIG. 3. *Sample points p_k (black points) form an adaptively refined grid within $\mathbf{\Omega}$ (all gray and black points). The blocky neighborhood U_k associated with sample point p_k (shaded light gray region) is the union of all leaf cells that contain p_k.*

Figure 2). To eliminate such boundary artifacts, we extend the functions φ_k outside of their natural support by using information from neighboring functions φ_j to create extended impulse response functions φ_k^E (section 2.5).

2.2. Adaptive grid structure. We will choose the sample points, p_k, so that they form an adaptively refined rectilinear grid (for example, see Figure 3). This section defines the structure of the adaptive grid; the procedure for constructing it will be explained in section 2.3.

We organize the domain $\mathbf{\Omega}$ into a binary tree, $\mathcal{T}$, of boxes $\mathbf{C} \subset \mathbf{\Omega}$ which we call *cells*. The root of $\mathcal{T}$ is the whole domain $\mathbf{\Omega}$. Cells may be either refined or not refined; refined cells are internal nodes in $\mathcal{T}$ and unrefined cells are leaves of $\mathcal{T}$. We denote the set of all leaves of the tree by leaves($\mathcal{T}$). Refined cells $\mathbf{C}$ are subdivided in a chosen direction into a set of two child cells that share an internal facet (more about how we choose the subdivision direction in section 2.7). We denote the set of children of $\mathbf{C}$ by children($\mathbf{C}$). The corners of all cells form the set of sample points:

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

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

We write

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

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

$$U_k \coloneqq \bigcup_{\mathbf{C}_i \in \operatorname{cells}(p_k)} \mathbf{C}_i. \tag{14}$$

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

2.3. Adaptive refinement algorithm. Starting with $\mathbf{\Omega}$ subdivided once in all directions, we repeatedly estimate the error in all cells in $\operatorname{leaves}(\mathcal{T})$ using an a posteriori error estimator, then refine the leaf cell with the largest error. The refinement process continues until either (a) the desired error in the approximation is achieved, or (b) a predetermined maximum number of sample points p_k is reached. At each step of the refinement process we construct or modify the functions w_k and φ_k^E using methods that will be described in sections 2.4, 2.5, and 2.8. We perform the a posteriori error estimation with a randomized method that will be described in section 2.6. We choose which direction to subdivide cells in using a method that will be described in section 2.7. The complete algorithm is summarized in Algorithm 2.1.

2.4. Harmonic weighting functions. We construct harmonic partition of unity weighting functions, w_k, by solving discrete local Laplace (diffusion) problems recursively on subsets of U_k. This process is equivalent to the construction of harmonic basis functions in finite element methods [12], and also shares conceptual ties with partition of unity finite element methods [6] and the construction of coarse basis functions in agglomerated element algebraic multigrid [42].

The blocky neighborhood U_k is a union of d-dimensional boxes. The boundary of each d-dimensional box is a union of $(d-1)$-dimensional facets, each of which is a box. There are $2d$ facets, corresponding to either the front or the back of the box in each coordinate direction. Facets that contain hanging nodes ("broken facets") are the union of several smaller $(d-1)$-dimensional boxes. Hence the boundary of each d-dimensional box can be expressed as the union of $(d-1)$-dimensional boxes, where we exclude broken facets in favor of their constituent smaller boxes. In the same way, the boundary of each $(d-1)$-dimensional box is a union of $(d-2)$-dimensional boxes, and so forth all the way down until we reach a set of 0-dimensional sample points. We build harmonic weighting functions by solving the Laplace equation ($-\Delta w_k = 0$) on these boxes recursively in dimension, using the values from lower-dimensional boxes as Dirichlet boundary conditions for higher-dimensional boxes. For sample points p_j (the lowest level), we assign $w_k[p_k] = 1$ and $w_k[p_j] = 0$ for $j \neq k$. Figure 4 illustrates this process for $d = 2$. Linearity, the maximum principle, and induction on boxes of increasing dimension show that the functions w_k form a partition of unity on $\mathbf{\Omega}$.

For the discrete Laplace equation we use the (positive definite) discrete graph Laplacian; this is equivalent to discretizing the continuous Laplacian using a standard Kronecker sum finite difference approximation on a regular grid. The local Laplace problems can be solved efficiently (in time proportional to the number of unknowns) with multigrid [7, 15].

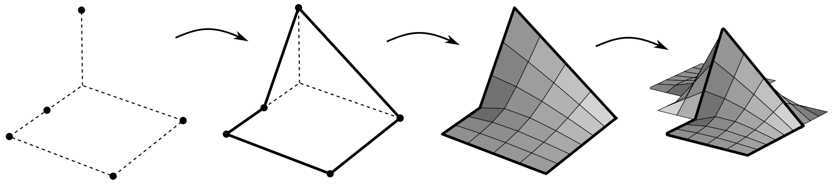
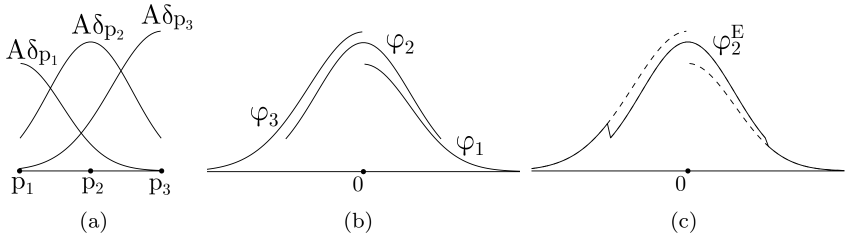


FIG. 4. *Construction of w_k for $d = 2$. For each box in U_k we assign w_k the value 1 at sample point p_k and 0 at all other sample points. For edges between sample points, we compute the values of w_k by solving the discrete 1-dimensional Laplace equation, using the previously assigned values at sample points as Dirichlet boundary conditions. For faces, we compute the values of w_k by solving the discrete 2-dimensional Laplace equation, using the previously computed edge values as Dirichlet boundary conditions. Finally, we form w_k on U_k by combining its constituent pieces on each box.*



(a) (b) (c)

FIG. 5. *Illustration of impulse response extension procedure in 1 dimension. To construct φ_2^E, we extend the support of φ_2 by filling in regions where φ_2 is undefined with values from φ_1 and φ_3.*

2.5. Extended impulse response functions. To construct φ_k^E, we first compute the impulse responses φ_k of A at the points p_k by applying A to point sources, then translating the results (see (13)). To eliminate boundary artifacts, we create φ_k^E by extending the support of φ_k, using data from neighboring functions φ_j, to fill in regions outside of $\operatorname{supp}(\varphi_k)$.

1. For z within $\operatorname{supp}(\varphi_k)$, we set $\varphi_k^E[z] := \varphi_k[z]$.
2. For z outside $\operatorname{supp}(\varphi_k)$ but within $\operatorname{supp}(\varphi_j)$ for at least one neighboring φ_j, we define $\varphi_k^E[z]$ as the average of all neighboring $\varphi_j[z]$ whose support contains z.
3. For z outside $\operatorname{supp}(\varphi_k)$ and outside $\operatorname{supp}(\varphi_j)$ for all neighboring φ_j, we set $\varphi_k^E[z] := 0$.

Figure 5 illustrates this procedure for a 1-dimensional example. Our theory still holds if we use any weighted average of neighboring $\varphi_j[z]$ in step 2, provided the weights are nonnegative and sum to one. We use the average since it simplifies the implementation and the explanation, and since more elaborate schemes are likely to yield only minimal improvements. The fact that we set some entries of $\varphi_k^E[z]$ to zero in step 3 is irrelevant since our scheme never accesses these entries (this will follow from Proposition 3).

In preparation for the theory in section 4, we now describe the process of constructing φ_k^E more precisely. First, we construct the following counting functions:

$$c_k := \mathbb{1}_{\boldsymbol{\Omega}-p_k} + \sum_{\substack{j\in\mathrm{nbrs}(k)\\ j\neq k}} \mathbb{1}_{(\boldsymbol{\Omega}-p_j)\setminus(\boldsymbol{\Omega}-p_k)}.$$

Since $\operatorname{supp}(\varphi_j) = \boldsymbol{\Omega} - p_j$, $c_k[z]$ counts how many φ_j will contribute to $\varphi_k^E[z]$. Next we compute

$$v_k[z] \coloneqq \begin{cases} 1/c_k[z], & z \in \operatorname{supp}(c_k), \\ 0, & \text{otherwise,} \end{cases}$$

and define

$$v_k^{(j)} \coloneqq \begin{cases} v_k \cdot \mathbb{1}_{\boldsymbol{\Omega}-p_k}, & j = k, \\ v_k \cdot \mathbb{1}_{(\boldsymbol{\Omega}-p_j)\setminus(\boldsymbol{\Omega}-p_k)}, & j \in \operatorname{nbrs}(k),\ j \neq k. \end{cases} \tag{15}$$

The function $v_k^{(j)}[z]$ is the weight given to neighboring impulse response φ_j at point z when constructing φ_k^E. Finally, we construct φ_k^E:

$$\varphi_k^E \coloneqq \sum_{j \in \operatorname{nbrs}(k)} v_k^{(j)} \cdot \varphi_j. \tag{16}$$

2.6. Randomized a posteriori error estimator. In order to decide which cells to refine, we wish to compute the error in the approximation

$$e_{\mathbf{C}} \coloneqq \left\| \left(\widetilde{A} - A \right) [\boldsymbol{\Omega}, \mathbf{C}] \right\| \tag{17}$$

for all cells $\mathbf{C} \in \operatorname{leaves}(T)$. Computing these norms is prohibitively expensive, so instead we estimate them. If M is any matrix with N columns, then the following sample average approximation estimates the square of its Frobenius norm:

$$\|M\|^2 = \mathbb{E}\left(\|M\zeta\|^2\right) \approx \frac{1}{q}\sum_{i=1}^{q} \|M\zeta_i\|^2 = \frac{1}{q}\|MZ\|^2, \tag{18}$$

where $\zeta, \zeta_i \sim N(0,1)^N$, are independent and identically distributed (i.i.d.) Gaussian random vectors, $\mathbb{E}$ is the expected value, q is the number of samples used in the sample average approximation, and $Z \sim N(0,1)^{N\times q}$ is an i.i.d. Gaussian random matrix (the matrix with columns ζ_i) [5]. Hence we can form an estimator, $\eta_{\mathbf{C}} \approx e_{\mathbf{C}}$, by forming a random matrix $Z \sim N(0,1)^{N\times q}$, computing

$$Y = A^* Z \quad \text{and} \quad \widetilde{Y} = \widetilde{A}^* Z,$$

then extracting blocks of the results, and taking norms:

$$\eta_{\mathbf{C}} \coloneqq \frac{1}{\sqrt{q}} \|\widetilde{Y}[\mathbf{C}, \ \cdot\ ] - Y[\mathbf{C}, \ \cdot\ ]\|. \tag{19}$$

By performing the randomized sample average approximation with the adjoints A^* and $\widetilde{A}^*$, we apply these operators once per sample, then postprocess the results to get estimators for all cells. Using the original operators A and $\widetilde{A}$ instead would require us to apply these operators to new random vectors for each cell.

It is straightforward to adapt the Chernoff bound in [5] to get an upper bound on the number of samples required. However, this bound is overly pessimistic; in practice we find the estimator is effective with only a handful of samples.

2.7. Anisotropic refinement: Choosing the subdivision direction. We refine anisotropically by estimating the direction that φ_p changes the most as a function of p, then subdividing in that direction. This allows us to capture changes to φ_p in directions where translation-invariance fails, without refining the grid in directions where translation-invariance holds.

Let $\mathbf{C}$ be a cell that we have chosen to subdivide based on the randomized a posteriori error estimator described in section 2.6. For each coordinate direction i in which $\mathbf{C}$ is big enough to be refined ($c^i_{\max} - c^i_{\min} > 2$), we partition the functions φ^E_k associated with the corners of $\mathbf{C}$ into two groups. One group is the set of φ^E_k associated with corners in the "front" of the cell (+) in the ith coordinate direction, and the other group is the set of φ^E_k associated with the "back" of the cell (−) in the ith coordinate direction:

$$\Psi^{i+} := \{\varphi^E_k : p_k \in \text{corners}(\mathbf{C}),\ p^i_k = c^i_{\max}\},$$
$$\Psi^{i-} := \{\varphi^E_k : p_k \in \text{corners}(\mathbf{C}),\ p^i_k = c^i_{\min}\}.$$

Next, we construct "average" φ^E_k functions for the front and back of the cell, respectively:

$$\varphi^{i+} := \frac{1}{2^{d-1}} \sum_{\varphi^E \in \Psi^{i+}} \varphi^E \quad \text{and} \quad \varphi^{i-} := \frac{1}{2^{d-1}} \sum_{\varphi^E \in \Psi^{i-}} \varphi^E.$$

Then we determine how much these average impulse responses change from the front to the back in direction i by computing $\|\varphi^{i+} - \varphi^{i-}\|_{l^2(\mathbf{\Omega} - c_{\text{mid}})}$. Finally, we subdivide $\mathbf{C}$ in the coordinate direction i in which the average impulse response changes the most.

2.8. Construction cost. Algorithm 2.1 shows the complete algorithm for constructing $\widetilde{A}$. Updating $\widetilde{A}$ after refining a cell requires us to apply A to point sources centered at the new sample points created during the refinement. Hence the entire refinement process requires us to apply A to r vectors, where r is the total number of sample points in the final product-convolution approximation.

The dominant cost in the error estimation process is the cost of computing A^*Z and $\widetilde{A}^*Z$ for a random matrix Z with q columns. Since A^*Z is constant throughout the refinement process, we compute it once at the beginning.

Although $\widetilde{A}$ changes every time we refine a cell, after performing a refinement we do not have to recompute $\widetilde{A}^*Z$ from scratch. To see this, recall from (6) that the adjoint of our product-convolution operator is a convolution-product operator with the convolution functions reflected about the origin and complex conjugated. In order to recompute $\widetilde{A}^*Z$ after refining cells, we only need to compute the convolutions $\text{flip}(\overline{\varphi^E_k}) * \zeta_i$ for each column, ζ_i, in Z, and each sample point, p_k, that is *new* or has a *new neighbor*.[5] The convolutions for old sample points without new neighbors have been computed previously and can be reused within (6). Thus the error estimation process requires computing $O(rq)$ convolutions. As we will discuss in section 3.2, each of these convolutions can be done with the FFT in $O(N \log N)$ work. Updating the functions w_k can be done locally. This requires negligible work compared to the other costs already discussed. Putting all these pieces together, constructing $\widetilde{A}$ requires

$$O\left(rC + qC_* + rqN \log N\right)$$

work, where C and C_* are the costs to apply A and A^* to one vector, respectively.

[5] The function φ^E_k depends on neighboring impulse responses due to the extension procedure.

Algorithm 2.1 Construction of $\widetilde{A}$.

```
Input: $v \mapsto Av$, $v \mapsto A^* v$, $\boldsymbol{\Omega}$, $\tau$, $q$
Output: $\left(w_k, \varphi_k^E\right)_{k=1}^r$

Draw random matrix $Z \sim N(0,1)^{N \times q}$
Compute $Y = A^* Z$                                   ▷ Cost: $q$ applications of $A^*$
Initialize $\mathcal{T}$ with $\boldsymbol{\Omega}$ as its root (section 2.2)
Refine $\mathcal{T}$ by subdividing $\boldsymbol{\Omega}$ once in each coordinate direction
Construct blocky neighborhoods $U_k$ (14)
Construct harmonic weighting function $w_k$ (section 2.4)
Compute impulse response functions $\varphi_k = A\delta_{p_k}$          ▷ Cost: $3^d$ applications of $A$
Construct extended impulse response functions $\varphi_k^E$ (section 2.5)
Compute $\widetilde{Y} = \widetilde{A}^* Z$                               ▷ Cost: $q \times 3^d$ convolutions
Form local error estimators $\eta_{\mathbf{C}}$ (19)
Form overall error estimator $\eta_{\boldsymbol{\Omega}}$
while $\eta_{\boldsymbol{\Omega}} > \tau$ do
    Find cell $\mathbf{C} \in$ leaves $\mathcal{T}$ with the largest $\eta_{\mathbf{C}}$
    Determine subdivision direction, $i$, for $\mathbf{C}$ (section 2.7)
    Subdivide $\mathbf{C}$ in direction $i$
    Construct $U_k$ that are new or modified by the refinement
    Construct $w_k$ for new or modified $U_k$
    Compute $\varphi_k = A\delta_{p_k}$ for all new $p_k$      ▷ Cost: 1 application of $A$ per new $p_k$
    Construct new or modified $\varphi_k^E$
    Update $\widetilde{Y}$ (section 2.8)                   ▷ Cost: $O(q)$ convolutions per new $p_k$
    Form new or modified local error estimators $\eta_{\mathbf{C}}$
    Form overall error estimator $\eta_{\boldsymbol{\Omega}}$
```

3. Using the product-convolution approximation.

The product-convolution format allows us to perform useful operations with $\widetilde{A}$ that we cannot perform with A.

3.1. Computing matrix entries of $\widetilde{A}$.

Our approximation $\widetilde{A}$ is a product-convolution scheme and, therefore (as seen in (5)), has the following matrix entries:

$$\widetilde{A}\left[y,x\right] = \sum_{k=1}^{r} w_k\left[x\right]\varphi_k^E\left[y-x\right] = \sum_{k:x\in U_k} w_k\left[x\right]\varphi_k^E\left[y-x\right]. \tag{20}$$

Using (20) we can compute individual matrix entries of $\widetilde{A}$ in $O(1)$ time even though $\widetilde{A}$ is not stored in memory in the conventional sense.

3.2. Applying $\widetilde{A}$ or $\widetilde{A}^*$ to vectors.

Applying $\widetilde{A}$ or $\widetilde{A}^*$ to a vector requires computing r convolutions, r pointwise vector multiplications, and some vector additions (see (12) or (6), respectively). Out of these operations, the r convolutions are the most computationally expensive. Since the convolution theorem allows us to compute each of these convolutions using the FFT (after appropriate zero padding) [39] at $O(N \log N)$ cost, the cost of applying $\widetilde{A}$ or $\widetilde{A}^*$ to a vector is $O(rN\log N)$.

3.3. Applying blocks of $\widetilde{A}$ or $\widetilde{A}^*$ to vectors.

One can implicitly apply a convolution operator to a function that is supported in a source box $\mathbf{S}$ then restrict the results to another target box $\mathbf{T}$, by performing a convolution between a function supported on a box with the same shape as $\mathbf{S}$ and a function supported on a box with the same shape as $\mathbf{T}-\mathbf{S}$, then translating the results. Specifically, a change of variables shows that if f is supported on $\mathbf{S}$, then

$$(\varphi * f)\left[\mathbf{T}\right] = (\varphi_0 * f_0)\left[\mathbf{T}'\right],$$

where

$$f_0\left[x_0\right] \coloneqq \begin{cases} f\left[x_0 + s_{\min}\right], & x_0 \in \mathbf{S}_0, \\ 0, & \text{else}, \end{cases} \qquad \varphi_0\left[z_0\right] \coloneqq \begin{cases} \varphi\left[z_0 + g_{\min}\right], & z_0 \in \mathbf{G}_0, \\ 0, & \text{else}, \end{cases}$$

and $\mathbf{G} \coloneqq \mathbf{T}-\mathbf{S}$, $\mathbf{S}_0 \coloneqq \mathbf{S} - s_{\min}$, $\mathbf{T}' \coloneqq \mathbf{T} - t_{\min} + s_{\max} - s_{\min}$, and $\mathbf{G}_0 \coloneqq \mathbf{G} - g_{\min}$. Thus one can apply a block of a convolution operator to a vector in work that scales nearly linearly with the linear dimensions of the block: $O(\sigma \log \sigma)$, where $\sigma = |\mathbf{S}|+|\mathbf{T}|$. To apply $\widetilde{A}\left[\mathbf{T},\mathbf{S}\right]$ or $\widetilde{A}^*\left[\mathbf{T},\mathbf{S}\right]$ to a vector, we use this method for each convolution in the sums ((12) and (6)) defining $\widetilde{A}$ or $\widetilde{A}^*$, respectively, that could be nonzero. Since the functions w_k are supported on the sets U_k, the terms in these sums that could be nonzero correspond to sets U_k that intersect $\mathbf{S}$ when multiplying $\widetilde{A}\left[\mathbf{T},\mathbf{S}\right]$ with a vector, and $\mathbf{T}$ when multiplying with $\widetilde{A}^*\left[\mathbf{T},\mathbf{S}\right]$ with a vector. As a result, it costs

$$(21) \qquad \underbrace{O(r_{\mathbf{S}}\ \sigma \log \sigma)}_{f \mapsto \widetilde{A}[\mathbf{T},\mathbf{S}]f} \qquad \text{and} \qquad \underbrace{O(r_{\mathbf{T}}\ \sigma \log \sigma)}_{f \mapsto \widetilde{A}^*[\mathbf{T},\mathbf{S}]f}$$

work to apply $\widetilde{A}\left[\mathbf{T},\mathbf{S}\right]$ and $\widetilde{A}^*\left[\mathbf{T},\mathbf{S}\right]$ to vectors, respectively. Here $r_{\mathbf{S}}$ and $r_{\mathbf{T}}$ are the number of sets U_k that intersect $\mathbf{S}$ and $\mathbf{T}$, respectively.

3.4. Conversion to H-matrix format.

Construction of a H-matrix proceeds in the following way:

1. The degrees of freedom are partitioned hierarchically into a cluster tree.
2. The matrix entries are partitioned hierarchically into a block cluster tree.
3. A low-rank approximation is constructed for each block of the matrix that is marked as low rank (i.e., admissible) within the block cluster tree.

The H-matrix construction process is scalable if we can construct low-rank approximations (see section 1.2.1) of the low-rank blocks (step 3) in work that scales nearly linearly with the dimensions of the block. The method for efficiently applying blocks of $\widetilde{A}$ and $\widetilde{A}^*$ to vectors, outlined in section 3.3, allows us to do this using Krylov methods or randomized SVD. Whenever the Krylov method or randomized SVD requires the application of a block or its adjoint to a vector, we perform this computation using the method in section 3.3. Alternatively, formula (20) for the matrix entries of $\widetilde{A}$ allows us to construct a low-rank approximation of a block by forming a CUR approximation, as is done in [9, 14, 57]. Whenever the CUR approximation algorithm requires a row, column, or entry of the block, we access it using (20).

Since applying the block $\widetilde{A}\left[\mathbf{T},\mathbf{S}\right]$ to a vector costs $O(\sigma \log \sigma)$ work, where $\sigma = |\mathbf{S}| + |\mathbf{T}|$, whereas accessing a row or column costs $O(\sigma)$ work, the CUR approach is asymptotically more scalable than the Krylov or randomized SVD approaches by a log factor. However, the CUR approach is less robust, and typically has poorer dependence on the rank of the blocks. In either case the overall cost of constructing the H-matrix scales nearly linearly with N. Moreover, the construction process only uses the approximation $\widetilde{A}$. It does not require expensive application of A.

4. Theory.

Here we show that the error in $\widetilde{A}$ is controlled by the failure of A to be locally translation-invariant with respect to a locally expanded cover, $\{U_k^E\}_{k=1}^r$, created by unioning each U_k with its neighbors:

$$U_k^E \coloneqq \bigcup_{j\in \mathrm{nbrs}(k)} U_j.$$

This provides an a priori error estimate for the approximation, and shows that the approximation will not contain boundary artifacts.

Let F_k be the following functions that measure how much the impulse response of A at p_k fails to represent the impulse response of A at x (see Figure 1):

$$F_k\left[y,x\right] \coloneqq A\left[y-x+p_k,p_k\right] - A\left[y,x\right]. \tag{22}$$

We aggregate these F_k to form a function F which measures, pointwise, how much A fails to be locally translation-invariant with respect to the cover $\{U_k^E\}_{k=1}^r$. Specifically, we define

$$F\left[y,x\right] \coloneqq \max_{k:(y,x)\in\mu_k^E} \left|F_k\left[y,x\right]\right|, \tag{23}$$

where the sets

$$\mu_k^E \coloneqq \{(y,x) : x\in U_k^E, y\in \mathbf{\Omega}, y-x+p_k\in\mathbf{\Omega}\} \tag{24}$$

are defined to be all $(y,x)\subset \mathbf{\Omega}\times\mathbf{\Omega}$ such that $x\in U_k^E$, and $F_k\left[y,x\right]$ is well-defined without resorting to extension by zero. In Theorem 5 we will show that

$$\|\widetilde{A}-A\| \le \|F\|. \tag{25}$$

If we instead maximized over $k : x\in U_k^E$ rather than $k:(y,x)\in\mu_k^E$ in (23), then the right-hand side of bound (25) would be undefined, because evaluating $\|F\|$ requires evaluating $A\left[y-x+p_k,p_k\right]$, and $y-x+p_k$ may be outside of $\mathbf{\Omega}$ even if x, y, and p_k are in $\mathbf{\Omega}$. Extending A by zero would make $\|F\|$ well-defined, and would make the theory simple, but then the bound would be unnecessarily large due to boundary artifacts. Achieving bound (25) while maximizing over $k:(y,x)\in\mu_k^E$ in (23) requires the boundary extension procedure of section 2.5, and is the reason why proving bound (25) will require several pages rather than a few lines.

A multistep path leads to Theorem 5. In Proposition 3 we show that $\widetilde{A}$ can be reinterpreted as a weighted sum involving the original (not extended) impulse response functions φ_k, but with weighting functions that form a partition of unity on $\mathbf{\Omega}\times\mathbf{\Omega}$, and are supported in the sets μ_k^E. Proposition 3 relies on a lemma about the functions $v_k^{(j)}$ used in our impulse response extension procedure (Lemma 2), which in turn relies on a lemma about Minkowski sums of boxes (Lemma 1). After establishing these prerequisites, in Proposition 4 we show that $\widetilde{A}-A$ can be represented as a weighted sum of the F_k functions, with the same weighting functions as in Proposition 3. Finally, we use Proposition 4 and the properties of these weighting functions to prove bound (25) in Theorem 5.

Lemma 1. *If* $\mathbf{S}$ *and* $\mathbf{T}$ *are boxes, and* $\mathbf{S}$ *is at least as large as* $\mathbf{T}$ *in the sense that* $s^i_{\max}-s^i_{\min}\ge t^i_{\max}-t^i_{\min}$ *for* $i=1,\dots,d$, *then* $\mathbf{S}+\mathbf{T}=\mathbf{S}+\mathrm{corners}(\mathbf{T})$.

Lemma 2. *We have*

$$\sum_{j\in\mathrm{nbrs}(k)} v_k^{(j)}\left[z\right] = \begin{cases} 1, & z\in\mathbf{\Omega}-U_k,\\ 0, & \textit{otherwise.}\end{cases} \tag{26}$$

Proof. By construction,

$$\sum_{j \in \operatorname{nbrs}(k)} v_k^{(j)}[z] = \begin{cases} 1, & z \in \operatorname{supp}(c_k), \\ 0, & \text{otherwise}, \end{cases}$$

and $\operatorname{supp}(c_k) = \bigcup_{j \in \operatorname{nbrs}(k)} (\mathbf{\Omega} - p_j)$. We now show that $\mathbf{\Omega} - U_k = \bigcup_{j \in \operatorname{nbrs}(k)} (\mathbf{\Omega} - p_j)$. To that end, recall that U_k is the union of leaf boxes $\mathbf{C}_i$ that contain p_k. Thus

$$\mathbf{\Omega} - U_k = \mathbf{\Omega} - \bigcup_{\mathbf{C}_i \in \operatorname{cells}(p_k)} \mathbf{C}_i = \bigcup_{\mathbf{C}_i \in \operatorname{cells}(p_k)} (\mathbf{\Omega} - \mathbf{C}_i).$$

Since $\mathbf{C}_i \subset \mathbf{\Omega}$, we see that $\mathbf{\Omega}$ is at least as large as $-\mathbf{C}_i$ (in the sense of Lemma 1). Applying Lemma 1 to $\mathbf{\Omega} - \mathbf{C}_i$ and performing algebraic manipulations yields

$$\bigcup_{\mathbf{C}_i \in \operatorname{cells}(p_k)} (\mathbf{\Omega} - \mathbf{C}_i) = \bigcup_{\mathbf{C}_i \in \operatorname{cells}(p_k)} (\mathbf{\Omega} - \operatorname{corners}(\mathbf{C}_i)) = \mathbf{\Omega} - \bigcup_{\mathbf{C}_i \in \operatorname{cells}(p_k)} \operatorname{corners}(\mathbf{C}_i).$$

Furthermore, by definition the union of all corners of leaf cells containing a point is the union of all neighboring points, so we have

$$\mathbf{\Omega} - \bigcup_{\mathbf{C}_i \in \operatorname{cells}(p_k)} \operatorname{corners}(\mathbf{C}_i) = \mathbf{\Omega} - \bigcup_{j \in \operatorname{nbrs}(k)} p_j = \bigcup_{j \in \operatorname{nbrs}(k)} (\mathbf{\Omega} - p_j),$$

which, with the chain of set equalities in previous lines, implies the desired result. □

PROPOSITION 3. *Let*

$$W_k[y,x] := \sum_{j \in \operatorname{nbrs}(k)} w_j[x]\, v_j^{(k)}[y-x]. \tag{27}$$

1. *The entries of* $\widetilde{A}$ *can be written as:*

$$\widetilde{A}[y,x] = \sum_{k=1}^{r} W_k[y,x]\, \varphi_k[y-x].$$

2. *The functions* $\{W_k\}_{k=1}^r$ *form a partition of unity:*

$$\sum_{k=1}^{r} W_k[y,x] = 1 \quad \text{for all } (y,x) \in \mathbf{\Omega} \times \mathbf{\Omega}.$$

3. *The partition of unity is subordinate to the cover* $\{\mu_k^E\}_{k=1}^r$*:*

$$\operatorname{supp}(W_k) \subset \mu_k^E.$$

Proof.

1. Substituting the definition of φ_k^E from (16) into the definition of $\widetilde{A}$ from (12) then performing algebraic manipulations, we have

$$\begin{aligned} \widetilde{A}[y,x] &= \sum_{k=1}^{r} w_k[x] \sum_{j \in \operatorname{nbrs}(k)} v_k^{(j)}[y-x]\, \varphi_j[y-x] \\ &= \sum_{k=1}^{r} \sum_{j \in \operatorname{nbrs}(k)} w_k[x]\, v_k^{(j)}[y-x]\, \varphi_j[y-x] \\ &= \sum_{j=1}^{r} \sum_{k \in \operatorname{nbrs}(j)} w_k[x]\, v_k^{(j)}[y-x]\, \varphi_j[y-x] = \sum_{j=1}^{r} W_j[y,x]\, \varphi_j[y-x]. \end{aligned}$$

Going from the second to the third line we used the fact that

$$\sum_{a\in X}\sum_{\{b:b\in X, b\sim a\}} f(a,b) = \sum_{b\in X}\sum_{\{a:a\in X, a\sim b\}} f(a,b)$$

for any symmetric relation $\sim$. Note the switch of k and j.

2. Using the definition of W_k in (27), we have

$$\begin{aligned}\sum_{k=1}^{r} W_k\left[y,x\right] &= \sum_{k=1}^{r}\sum_{j\in\text{nbrs}(k)} w_j\left[x\right] v_j^{(k)}\left[y-x\right]\\ &= \sum_{j=1}^{r}\sum_{k\in\text{nbrs}(j)} w_j\left[x\right] v_j^{(k)}\left[y-x\right] = \sum_{j=1}^{r} w_j\left[x\right]\left(\sum_{k\in\text{nbrs}(j)} v_j^{(k)}\left[y-x\right]\right).\end{aligned}$$

If $x\in U_j$ and $y\in\mathbf{\Omega}$, then Minkowski set arithmetic implies $y-x\in\mathbf{\Omega}-U_j$, so (26) in Lemma 2 implies

$$\sum_{k\in\text{nbrs}(j)} v_j^{(k)}\left[y-x\right] = 1.$$

Since $\operatorname{supp}(w_j)\subset U_j$, this implies

$$\sum_{j=1}^{r} w_j\left[x\right]\left(\sum_{k\in\text{nbrs}(j)} v_j^{(k)}\left[y-x\right]\right) = \sum_{j=1}^{r} w_j\left[x\right] = 1.$$

Thus $\sum_{k=1}^{r} W_k\left[y,x\right] = 1$ as required.

3. From the definition of $v_k^{(j)}$ in (15), either $\operatorname{supp}(v_k^{(j)}) = (\mathbf{\Omega}-p_j)\setminus(\mathbf{\Omega}-p_k)$ when $k\neq j$, or $\operatorname{supp}(v_k^{(j)}) = \mathbf{\Omega}-p_j$ when $k=j$. In either case $\operatorname{supp}(v_k^{(j)})\subset\mathbf{\Omega}-p_j$. Thus

$$(y-x\notin\mathbf{\Omega}-p_j) \implies \left(v_k^{(j)}\left[y-x\right]=0\right),$$

which is equivalent to the statement

$$(y-x+p_j\notin\mathbf{\Omega}) \implies \left(v_k^{(j)}\left[y-x\right]=0\right). \tag{28}$$

Since W_k consists of a sum of terms, each term containing $v_j^{(k)}\left[y-x\right]$, statement (28) implies (note the swap of k,j)

$$(y-x+p_k\notin\mathbf{\Omega}) \implies \left(W_k\left[y,x\right]=0\right). \tag{29}$$

Additionally, since each w_j in the sum defining W_k is supported in the blocky neighborhood U_j, and since the union of these blocky neighborhoods U_j is U_k^E, we have

$$\left(x\notin U_k^E\right) \implies \left(W_k\left[y,x\right]=0\right). \tag{30}$$

Altogether, (29), (30), and the definition of μ_k^E in (24) imply $\operatorname{supp}(W_k)\subset\mu_k^E$. □

PROPOSITION 4. *The pointwise error in our product-convolution approximation takes the following form:*

$$\widetilde{A}\left[y,x\right] - A\left[y,x\right] = \sum_{k:(y,x)\in\mu_k^E} W_k\left[y,x\right] F_k\left[y,x\right]. \tag{31}$$

Proof. From Proposition 3 and the fact that $\varphi_k[z] = A[z + p_k, p_k]$, we know that

$$\widetilde{A}[y,x] = \sum_{k=1}^{r} W_k[y,x]\, A[y - x + p_k, p_k].$$

Hence the pointwise error in the approximation takes the following form:

$$\begin{aligned}\widetilde{A}[y,x] - A[y,x] &= \sum_{k=1}^{r} W_k[y,x]\, A[y - x + p_k, p_k] - A[y,x]\\ &= \sum_{k=1}^{r} W_k[y,x]\left(A[y - x + p_k, p_k] - A[y,x]\right)\\ &= \sum_{k=1}^{r} W_k[y,x]\, F_k[y,x] = \sum_{k:(y,x)\in\mu_k^E} W_k[y,x]\, F_k[y,x].\end{aligned}$$

Going from the first line to the second line we used the partition of unity property of W_k from Proposition 3. Going from the second to the third line we used the definition of F_k. In the last equality on the third line we used the fact that $\operatorname{supp}(W_k) \subset \mu_k^E$. □

THEOREM 5. *We have*

$$\|\widetilde{A} - A\| \leq \|F\|. \tag{32}$$

Proof. Using the result of Proposition 4, the fact that W_k form a partition of unity, and the definition of F yields the pointwise error bound

$$\begin{aligned}|\widetilde{A}[y,x] - A[y,x]| &= \left|\sum_{k:(y,x)\in\mu_k^E} W_k[y,x]\, F_k[y,x]\right|\\ &\leq \max_{k:(y,x)\in\mu_k^E} |F_k[y,x]| = F[y,x].\end{aligned}$$

The overall bound, (32), follows directly from the definition of the norm and this pointwise bound. □

Remark 6. Let

$$T[y,x] := A[y + x, x]$$

be the spatially varying impulse response (SVIR) function (see, e.g., [11] for a more in-depth discussion of the SVIR). Under the change of variables $h := p - x$, $\xi := y - x$, we may express the failure of local translation invariance in terms of the SVIR as follows:

$$A[y - x + p, p] - A[y,x] = -\left(T[\xi, p + h] - T[\xi, p]\right).$$

If x is near p, then h is small, so

$$T[\xi, p + h] - T[\xi, p] \approx \frac{dT}{dp}(\xi, p)h.$$

Hence, if our scheme is applied to a discretization of a continuous operator, the smoother the function $x \mapsto T[y,x]$ is, the better our scheme will perform.

5. Numerical examples.

We numerically test our scheme on a spatially varying blur operator (section 5.1), on the nonlocal component of the Schur complement associated with restricting the Poisson operator to an internal interface (section 5.2), and on the data misfit Hessian for an advection-diffusion inverse problem (section 5.3). For the spatially varying blur operator, our scheme refines towards the boundary between blur kernels and refines almost nowhere else, therefore, outperforming the standard nonadaptive scheme which refines everywhere uniformly. For the Poisson interface Schur complement, our scheme is mesh scalable: it requires roughly the same convolution rank (number of terms in (12)) to achieve a desired error tolerance regardless of how fine the mesh is. For the Hessian, our scheme is data scalable: it requires roughly the same convolution rank to achieve a desired error tolerance regardless of how informative the data are about the unknown parameter in the inverse problem. For both the Poisson Schur complement and the Hessian, we show that our scheme, in combination with H-matrix methods, can be used to build good preconditioners. Additionally, we find that the randomized a posteriori error estimator achieves good performance with only a handful of random samples: our scheme performs almost as well with $q = 5$ as it does with $q = 100$.

For H-matrices, we use the standard coordinate splitting nested-bisection binary cluster tree,[6] and the standard diameter-less-than-distance admissibilty condition.[7]

5.1. Spatially varying blur.

Problem setup. Let a be the following spatially varying blurring kernel,

$$a(s,t) := \exp\left(-\frac{s^2+t^2}{2\sigma^2(s,t)}\right), \quad \text{where} \quad \sigma(s,t) = \begin{cases} 0.1, & s^2+t^2 < 0.5, \\ 0.2, & s^2+t^2 \geq 0.5. \end{cases}$$

Here A is the matrix generated by sampling a on $[-1,1]^2$ with a 75×75 equally spaced regular grid.

Results. Figure 6(a) compares the product-convolution approximation of A using our adaptive scheme, versus the standard product-convolution approximation of A using an equally spaced regular grid of sample points, with bilinear interpolation of impulse response functions, no adaptivity, and no boundary extension procedure. Our adaptive scheme converges much faster than the regular grid scheme.

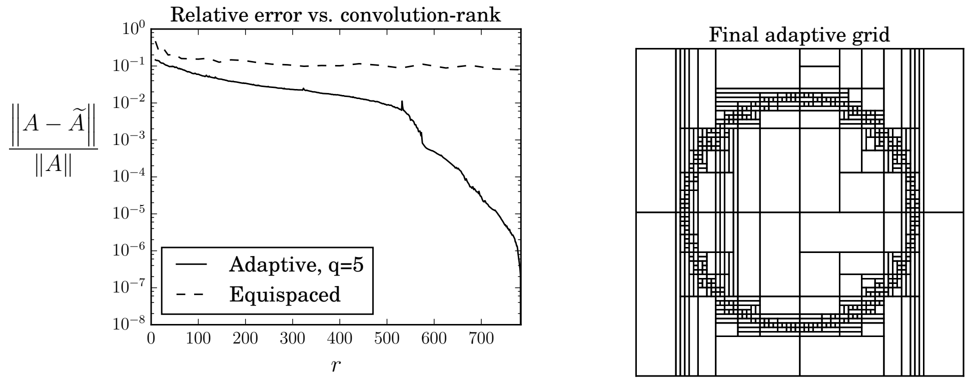
Figure 6(b) shows the final grid generated by our adaptive scheme, in which the boundary of the circle $s^2 + t^2 = 1$ is fully resolved with 2×2 cells. The error in the adaptive procedure is zero (within machine epsilon) for this final grid.

5.2. Poisson interface Schur complement.

Problem setup. Here we consider the discretized (negative) Laplace operator $K \approx -\Delta$ on the interior of the cube, $(-1,1)^3$. To build K, we discretize the Laplace operator on the whole cube, $[-1,1]^3$ with piecewise linear finite elements on a regular $n \times n \times n$ mesh of tetrahedra, so that there are $(n+1)^3$ mesh grid points. Then we exclude rows and columns from the resulting matrix that correspond to boundary degrees of freedom. The resulting $(n-1)^3 \times (n-1)^3$ matrix, K, is the coefficient

[6] Degrees of freedom are split into two equally sized clusters by a hyperplane normal to widest coordinate direction for that cluster. Then each cluster is split into two smaller clusters in the same way, and so on, recursively. The splitting continues until the number of degrees of freedom in a cluster is less than 32.

[7] We mark a block of the matrix as low rank (admissible) if the distance between the degree of freedom cluster associated with the rows of the block and the diameter of the degree of freedom cluster associated with the columns of the block is less than or equal to the diameter of the smaller of the two degree of freedom clusters.



(a) (b)

FIG. 6. Spatially varying blur: *Product-convolution approximation of the spatially varying blur operator defined in section* 5.1. (a) *Convergence of our adaptive scheme, compared to convergence of standard product-convolution approximation with an equispaced regular grid of sample points and local bilinear interpolant weighting functions.* (b) *Final grid generated by our adaptive scheme.*

matrix for the linear system that would need to be solved to determine the solution on the interior degrees of freedom for the Poisson problem in the cube with Dirichlet boundary conditions.

Let "i" denote the degrees of freedom on the interface hyperplane at $z = 0$ that separates[8] the degrees of freedom in the top half of the cube from the bottom half of the cube. Let "t" denote the degrees of freedom in the top half of the cube ($z > 0$), and let "b" denote degrees of freedom in the bottom half of the cube ($z < 0$), not including the interface in both cases. Denote the associated blocks of K by K_{it}, K_{tt}, K_{ti}, K_{ib} and so forth. We use our adaptive product-convolution scheme to approximate the operator

$$A := K_{it}K_{tt}^{-1}K_{ti} + K_{ib}K_{bb}^{-1}K_{bi}.$$

The matrix $-A$ is the nonlocal component of the Schur complement for degrees of freedom on the interface hyperplane, i.e., the matrix

$$S := K_{ii} - K_{it}K_{tt}^{-1}K_{ti} - K_{ib}K_{bb}^{-1}K_{bi}.$$

Matrix entries of A are not directly available; we apply A to vectors by performing matrix-vector products with K_{bi}, K_{ib}, K_{ti}, and K_{it}, and solving linear systems with K_{tt} and K_{bb} as the coefficient matrices. After approximating A with $\widetilde{A}$ using our product-convolution scheme, we also construct the Schur complement approximation

$$\widetilde{S} := K_{ii} - \widetilde{A}.$$

Such Schur complement approximations could be constructed recursively. One would subdivide the top and bottom subdomains, then subdivide the subdivisions, and so on. Approximations of Schur complements at deeper levels of the recursion would be used when constructing approximations at shallower levels. Here we only present results for one subdivision.

[8] We choose n even so that the interface is at $z = 0$, rather than being slightly offset.

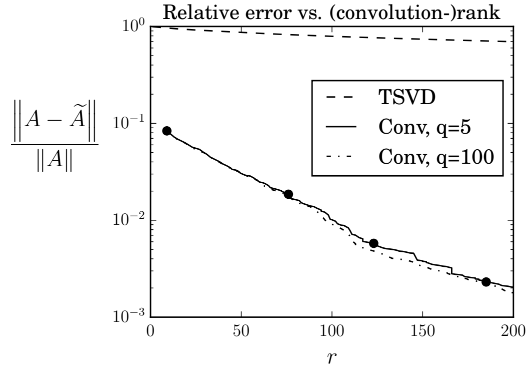


FIG. 7. Poisson Schur complement: *Relative error in TSVD low-rank approximation ("TSVD") compared to our product-convolution approximation ("Conv") as the (convolution) rank, r, changes. We show convergence curves for our scheme using both $q = 5$ and $q = 100$ random samples for the a posteriori error estimator. Black dots correspond to the adaptive grids visualized in Figure* 8.

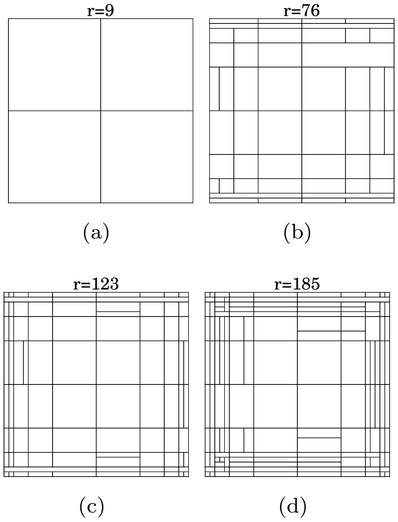


FIG. 8. Poisson Schur complement: *Intermediate stages of adaptive grid refinement corresponding to black dots in Figure* 7.

Results. Figure 7 compares the convergence of our scheme to truncated SVD ("TSVD") approximation for $n = 40$ ($N = (n-1)^2 = 1521$). Since the Poisson Schur complement is high rank, TSVD performs poorly. In contrast, our scheme performs well: at $r = 200$ our scheme has less than 0.03% error, whereas TSVD has approximately 69% error. Figure 7 also shows that our scheme performs well even when we use a small number of random samples for the a posteriori error estimator: the convergence curve for $q = 5$ is almost identical to the convergence curve for $q = 100$. Figure 8 displays the adaptive meshes from four different stages of the adaptive refinement process from Figure 7. Our scheme adaptively refines towards the boundary, then the corners. This is expected since boundary effects are the only source of translation-invariance failure.

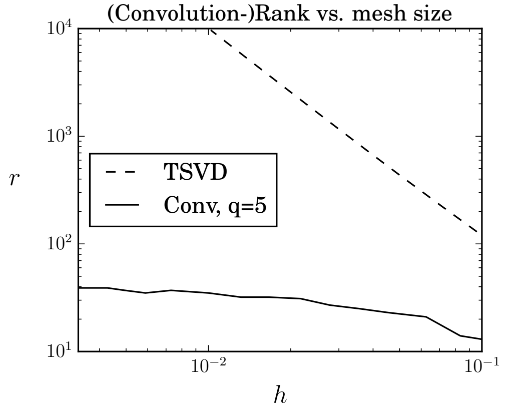


FIG. 9. Poisson Schur complement: *The (convolution) rank, r, required to achieve a relative approximation error of* 5%*, for a variety of mesh sizes, h. TSVD indicates truncated SVD low rank approximation, and Conv indicates our product-convolution scheme.*

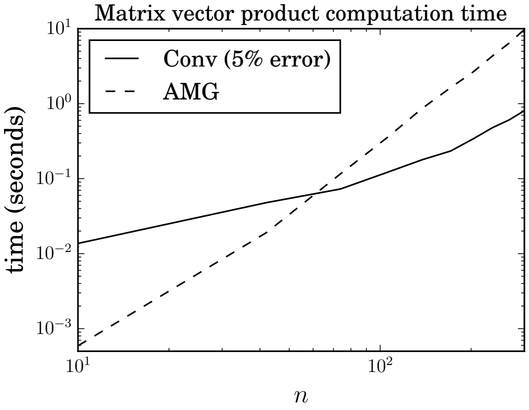


FIG. 10. Poisson Schur complement: *The time required to apply $\widetilde{A}$ to a vector using the FFT to compute the convolutions (Conv), compared to the time required to apply A to a vector, using multigrid to apply the matrices K_{tt}^{-1} and K_{bb}^{-1} to vectors ("AMG"). For our product-convolution scheme, the average slope between $n = 171$ and $n = 300$ (from the final upturn to the end, containing* 5 *equally spaced n) is* 2.2*, suggesting an asymptotic cost of $O(n^{2.2})$ (theory predicts $O(n^2 \log n)$). For algebraic multigrid, the average slope between $n = 171$ and $n = 300$ is* 3.2*, suggesting an asymptotic cost of $O(n^{3.2})$ (theory predicts $O(n^3)$).*

Figure 9 compares our scheme to TSVD on a sequence of progressively finer meshes, from $h \approx 0.1$ to $h \approx 0.01$, where h is the distance between adjacent gridpoints in the mesh. The curves show the (convolution) rank, r, required to achieve a relative error tolerance of 5%. The rank for TSVD grows with the number of degrees of freedom on the top surface ($r \sim O(1/h^2)$), offering little improvement over directly building a dense matrix representation of A column-by-column. In contrast, the convolution rank for our scheme remains small for all h considered.

Figure 10 compares the time required to apply A to a vector versus the time required to apply $\widetilde{A}$ to a vector. When applying A to vectors, we solve the necessary linear systems with K_{tt} and K_{bb} as coefficient operators using PyAMG's [49] rootnode algebraic multigrid. When applying $\widetilde{A}$ to vectors, we use the FFT, as discussed in

Table 1

Poisson Schur complement: Comparison of condition numbers for the Poisson interface Schur complement for a range of $n \times n \times n$ meshes. S is the unpreconditioned Schur complement. $\widetilde{S}$ is the approximate Schur complement generated by replacing the nonlocal terms, A, within the Schur complement, with our convolution aproximation, $\widetilde{A}$, with a 5% relative error tolerance. The last column shows r, the convolution rank of $\widetilde{A}$.

n	cond (S)	cond $\left(\widetilde{S}^{-1}S\right)$	r
10	10.3	1.1	9
20	21.3	1.2	20
30	32.2	1.3	27
40	43.0	1.4	28
50	53.8	1.5	31
60	64.5	1.5	33
70	75.3	1.8	32
80	86.1	1.8	35
90	96.9	1.8	35
100	107.7	1.9	35

section 3.2. For large n, applying $\widetilde{A}$ to a vector is much cheaper than applying A to a vector.

In Table 1 we compare the condition number of the Schur complement, S, with the condition numbers of the preconditioned Schur complement, $\widetilde{S}^{-1}S$ for $n\times n\times n$ meshes ranging from $n = 10$ to $n = 100$. Here $\widetilde{S}^{-1}$ is constructed by converting $\widetilde{S}$ to H-matrix format, then inverting it using H-matrix arithmetic. Here, we use a tolerance of 10^{-6} for the low-rank approximations performed during H-matrix construction and arithmetic. The condition number of the (unpreconditioned) Schur complement grows as $O(1/h)$, where $h \approx 1/n$ is the mesh size. In contrast, the preconditioned Schur complement remains extremely well conditioned: the largest value of cond$(\widetilde{S}^{-1}S)$ is 1.9 for all meshes considered.

5.3. Advection-diffusion inverse problem Hessian.

Problem setup. In this section we approximate the data misfit portion of the Hessian for an advection-diffusion inverse problem in which an unknown initial concentration, m, of a contaminant, u, is inferred from time series data, y, of the contaminant flowing through a boundary, Γ. Specifically, consider the following PDE:

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{1}{\mathrm{Pe}}\Delta u - \left(\begin{smallmatrix}0\\1\end{smallmatrix}\right)\cdot \nabla u, & t \in [0,1], \\ u = m, & t = 0, \end{cases} \tag{33}$$

where Pe is the Peclet number. The region of interest and support of m is the unit square, $\Omega = [0,1]^2$, and the desired unbounded domain for the PDE is $\mathbb{R}^2$. To simulate the effect of having an unbounded domain, we extend the computational domain beyond $[0,1]^2$ on all sides and use Neumann boundary conditions on the outer, larger, domain. We use y to denote the known noisy time series observations of u on the top boundary: $y(x,t) = u(x,t) + \zeta,\ x \in \Gamma,\ t \in (0,1]$, where $\Gamma := [0,1] \times \{1\}$ and ζ is 1% i.i.d. Gaussian noise.

The inverse problem is, given y, determine m. This is commonly formulated as a least squares optimization problem of the following form:

$$\min_{m}\ J(m) + R(m), \tag{34}$$

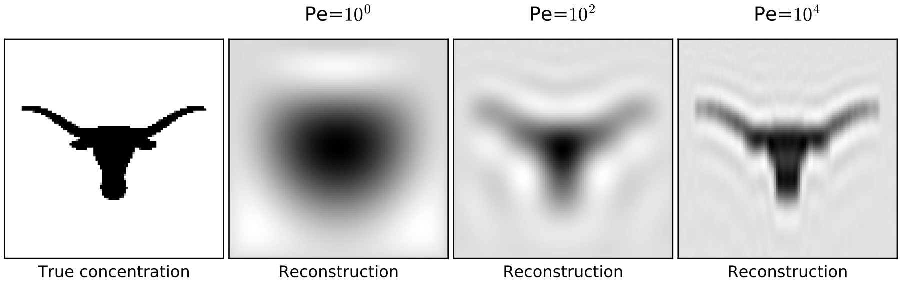


FIG. 11. Advection-diffusion inverse problem Hessian: *True initial concentration m (left image) and optimal reconstructions for Peclet numbers 10^0, 10^2, and 10^4 (all other images, left to right). The regularization parameter used for the reconstruction is $\alpha = 10^{-3}$, which satisfies the Morozov discrepancy principle within a 5% tolerance for all Peclet numbers considered.*

where J is the data misfit

$$J(m) := \frac{1}{2}\int_0^1 \int_\Gamma (u(m) - y)^2 \, dx \; dt,$$

and $R(m)$ is a quadratic regularization term. Here $u(m)$ denotes the solution of (33) as a function of m. We use Laplacian regularization, $R(m) = \frac{\alpha}{2} m^T \Delta m$, where Δ is a discretization of the Laplacian operator with zero Dirichlet boundary conditions, and $\alpha = 10^{-3}$ is the regularization parameter. This value of α was chosen since it satisfies the Morozov discrepancy principle [46] to within a 5% tolerance for all Peclet numbers considered. For discretization, we use piecewise linear finite elements defined on a regular rectilinear 100×100 mesh of triangles, with 100 time steps. We use backward Euler time stepping and streamline upwind/Petrov–Galerkin stabilization [16].

We use an image of the University of Texas "Hook'em Horns" logo as the initial concentration, m. The sharp edges in this image are computationally expensive to recover using existing methods. The solutions to the inverse problem for Peclet numbers in the range 10^0 to 10^5 are shown in Figure 11.

The results we present are twofold. First, we show that our product-convolution scheme can be used to approximate the discretized version of the operator

$$A := \frac{d^2 J}{dm^2},$$

which is the Hessian of the data misfit. Second, we use the convolution approximation of A to build a preconditioner for the overall Hessian,

$$H := A + \frac{d^2 R}{dm^2}.$$

We show that the preconditioner is effective even if the Peclet number is large.

Preconditioning the Hessian. Since y depends linearly on m, the solution to (34) is the solution to a linear system with H as the coefficient matrix. Although this inverse problem is linear, Newton methods for solving nonlinear advection-diffusion inverse problems require solving linear systems with similar Hessians as coefficient operators. The Hessian of the regularization, $\frac{d^2 R}{dm^2}$, is a differential operator with known entries, and thus it is easy to manipulate. In contrast, A is dense and its matrix entries are not directly available. We can only apply A to vectors using an adjoint-based framework (see [2]). This requires solving a pair of advection-diffusion equations: a state equation of the form (33) forward in time, and the adjoint of (33) backward in time. Explicitly forming A is thus prohibitively expensive: a pair of PDEs would need to be solved for every column of A.

While Krylov methods can be used to solve linear systems with the Hessian as the coefficient operator in a matrix-free manner, good general purpose preconditioners have not been available (see [3] for a discussion of these issues). But now our convolution-product scheme allows us to build a good preconditioner as follows: first we form a product-convolution approximation of A, then convert it to an H-matrix format, then symmetrize it, then add a small amount of identity regularization, then combine it with $\frac{d^2R}{dm^2}$, then finally invert the combined H-matrix with fast H-matrix arithmetic. In detail, we form the following approximation to the inverse of the Hessian, which we use as a preconditioner:

$$P^{-1} := \left((\widetilde{A} + \widetilde{A}^T)/2 + \tau \left\| A \right\| I + \frac{d^2R}{dm^2} \right)^{-1} \approx \left(A + \frac{d^2R}{dm^2} \right)^{-1}. \tag{35}$$

Here $\tau \left\| A \right\| I$ is a small amount of additional regularization (I is the identity matrix). We use $\tau = 0.0025$. Matrix addition, scaling, and inversion in (35) are performed with H-matrix arithmetic. Here, we use a fixed rank of 20 for the low-rank approximations performed during H-matrix construction and arithmetic.

Data scalability. The Peclet number, Pe, controls the ratio of advection to diffusion. As Pe increases, the rank of A increases [30], making the inverse problem more difficult to solve with existing methods. This increase in the rank corresponds to an increase in the informativeness of the data about the parameter in the inverse problem—eigenvectors of A corresponding to large eigenvalues represent modes of the parameter that are well informed by the data, whereas eigenvectors of A corresponding to small eigenvalues represent modes of the parameter that are poorly informed by the data (see [4] for a discussion of these issues). As a result, for an approximation of A to be data scalable (perform well regardless of how informative the data are about the parameter), the cost of constructing the approximation must not grow as Pe increases.

Results. Figure 12 compares the convergence of our product-convolution scheme CONV to TSVD low rank approximation when Pe $= 10^4$. Our scheme performs better than TSVD: at $r = 100$ our scheme has less than 1% error whereas TSVD has approximately 71% error. Like the Poisson problem, the convergence curve for $q = 5$ is almost identical to the convergence curve for $q = 100$. Figure 13 shows the adaptive meshes from four different stages of the adaptive refinement process from Figure 12. Our scheme chooses to adaptively refine in the direction of the vertical flow, prioritizing refinement near the top surface. We expect similar results would hold for inverse problems involving nonvertical, nonuniform flow if the convolution grid were aligned with the streamlines of the flow.

Figure 14 compares our scheme to TSVD for a sequence of increasing Peclet numbers, from Pe $= 10^1$ to Pe $= 10^5$. The curves show the (convolution) rank, r, required to achieve a relative error tolerance of 10% (estimated using $q = 5$ random adjoint samples). Whereas the required rank for TSVD grows dramatically as Pe increases, the required convolution rank for our scheme remains constant.

Figure 15 shows the convergence of Krylov methods for solving the Hessian linear system using GMRES with our preconditioner (“GMRES-CONV”), compared to conjugate gradient with regularization preconditioning (“CG-REG”), for Pe $= 10^4$.

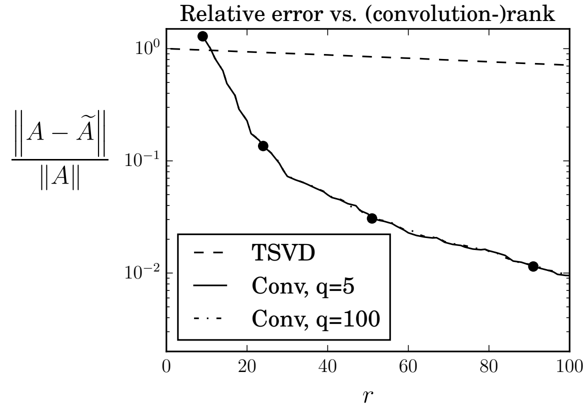


FIG. 12. Advection-diffusion inverse problem Hessian: *Relative error in the TSVD low-rank approximation compared to our product-convolution approximation (Conv) as the (convolution) rank, r, changes. We show convergence curves for our scheme using both $q = 5$ and $q = 100$ random samples for the a posteriori error estimator. Black dots correspond to the adaptive grids visualized in Figure* 13.

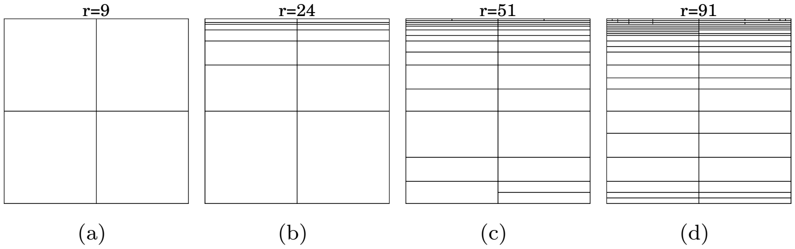


FIG. 13. Advection-diffusion inverse problem Hessian: *Intermediate stages of adaptive grid refinement corresponding to black dots in Figure* 12.

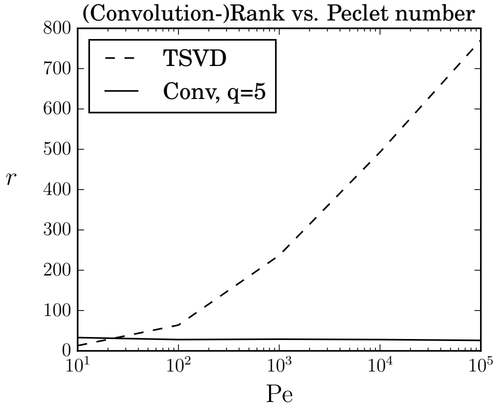


FIG. 14. Advection-diffusion inverse problem Hessian: *The (convolution) rank, r, required to achieve a relative approximation error of* 10% *for a variety of Peclet numbers,* Pe. *TSVD indicates truncated SVD low rank approximation, and Conv indicates our product-convolution scheme.*

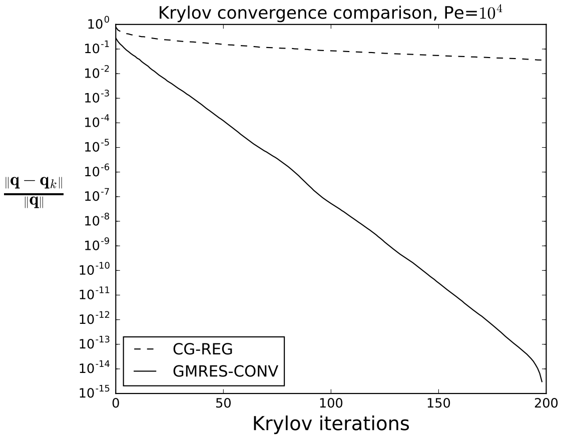


Fig. 15. Advection-diffusion inverse problem Hessian: *Convergence of conjugate gradient with regularization preconditioning ("CG-REG"), compared to GMRES with our product-convolution preconditioner,* (35) *('GMRES-CONV'), for solving the Hessian linear system. Here* $\mathrm{Pe} = 10^4$*, and the product-convolution approximation is accurate to* 5% *relative error.*

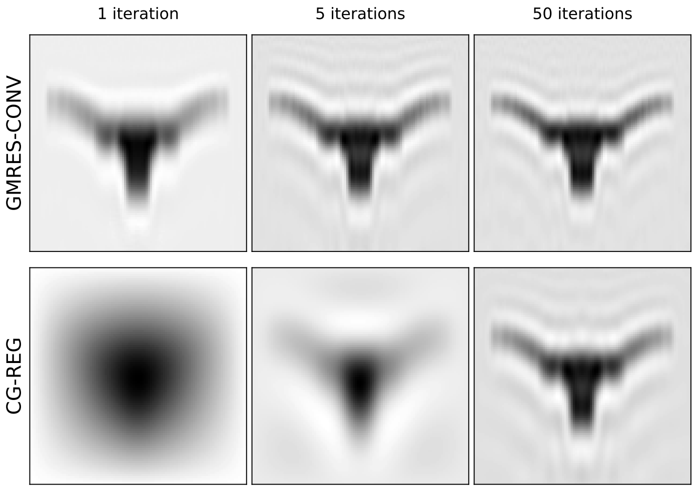


Fig. 16. Advection-diffusion inverse problem Hessian: *Comparison of parameter reconstructions associated with terminating the Krylov solver after* 1, 5, *and* 50 *iterations, for both GMRES with our preconditioner (GMRES-CONV), and conjugate gradient with regularization preconditioning (CG-REG). Here* $\mathrm{Pe} = 10^4$*, and the product-convolution approximation is accurate to* 5% *relative error.*

Here the product-convolution approximation is computed to a 5% relative error tolerance. Our preconditioner substantially outperforms regularization preconditioning, coverging rapidly even though the Peclet number is large. In Figure 16, we show intermediate reconstructions associated with 1, 5, and 50 Krylov iterations, for both GMRES-CONV and CG-REG. CG-REG first reconstructs large-scale features of m, then medium-scale features, then small-scale features, while GMRES-CONV reconstructs features of m at all scales simultaneously. Even one iteration of GMRES-CONV yields a visually reasonable reconstruction.

6. Conclusions. In this paper we presented a matrix-free adaptive grid product-convolution operator approximation scheme. The efficiency of our scheme depends on the degree to which the operator being approximated is locally translation-invariant. As a result, our scheme is well suited for approximating or preconditioning operators that arise in Schur complement methods for solving PDEs, reduced Hessians in PDE-constrained optimization and inverse problems, integral operators, covariance operators with spatially varying kernels, and Dirichlet-to-Neumann maps or other Poincaré–Steklov operators in multiphysics problems. These operators are often dense, implicitly defined, and high rank, making them difficult to approximate with standard techniques. Our scheme is best suited to moderate accuracy requirements (say, 80% to 99% accuracy).

Our scheme improves on existing product-convolution schemes by providing an automated method for performing adaptivity, and by addressing issues related to boundaries. Once constructed, the approximation can be manipulated efficiently and accessed in ways that the original operator cannot: matrix entries of the approximation can be computed at $O(1)$ cost, the approximation (or *blocks* of the approximation) can be efficiently applied to vectors with the FFT, and the approximation can be efficiently converted to H-matrix format. Once in H-matrix format, it can be factorized, inverted, or otherwise manipulated with fast H-matrix arithmetic. Since our scheme is best suited to moderate accuracy requirements, the resulting H-matrix can be exploited to construct a good preconditioner.

We tested our scheme numerically on a spatially varying blur operator, on the nonlocal component of an interface Schur complement for the Poisson operator, and on the data misfit Hessian for an advection-diffusion inverse problem. We saw that our scheme outperformed existing methods in all cases. Additionally, we found that the scheme performs well even when only a handful of random samples are used to construct the a posteriori error estimator used in the adaptive refinement procedure.

Acknowledgments. We thank J.J. Alger, Benjamin Babcock, Joe Bishop, Andrew Potter, Georg Stadler, Umberto Villa, and Hongyu Zhu for helpful discussions. We thank the anonymous reviewers for their helpful and in-depth comments, which have helped us improve this manuscript considerably.

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, Space Telescope Science Institute, Baltimore, MD, 1994, pp. 72–78.

[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 Supercomputing (SC2005), Seattle, IEEE, Piscataway, NJ, 2005.

[3] V. Akçelik, G. Biros, O. Ghattas, J. Hill, D. Keyes, and B. van Bloeman Waanders, *Parallel algorithms for PDE-constrained optimization*, in Parallel Processing for Scientific Computing, Software Environ. Tools 20, M. Heroux, P. Raghaven, and H. D. Simon, eds., SIAM, Philadelphia, 2006.

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

[5] H. Avron and S. Toledo, *Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix*, J. ACM, 58 (2011), 8.

[6] I. Babuska and J. Melenk, *The partition of unity method*, in Internat. J. Numer. Methods Engrg., 40 (1996), pp. 727–758.

[7] R. E. Bank and T. Dupont, *An optimal order process for solving finite element equations*, Math. Comp., 36 (1981), pp. 35–51.

[8] J. Bardsley, S. Jefferies, J. Nagy, and R. Plemmons, *A computational method for the restoration of images with an unknown, spatially-varying blur*, Opt. Express, 14 (2006), pp. 1767–1782.

[9] M. Bebendorf, *Approximation of boundary element matrices*, Numer. Math., 86 (2000), pp. 565–589.

[10] R. Bélanger-Rioux and L. Demanet, *Compressed absorbing boundary conditions via matrix probing*, SIAM J. Numer. Anal., 53 (2015), pp. 2441–2471.

[11] J. Bigot, P. Escande, and P. Weiss, *Estimation of linear operators from scattered impulse responses*, Appl. Comput. Harmon. Anal., 2017.

[12] J. E. Bishop, *A displacement-based finite element formulation for general polyhedra using harmonic shape functions*, Internat. J. Numer. Methods Engrg., 97 (2014), pp. 1–31.

[13] S. Börm, *Efficient Numerical Methods for Non-local Operators: $H2$-Matrix Compression, Algorithms and Analysis*, EMS Tracts Math 14, European Mathematical Society, Zurich, 2010.

[14] S. Börm and L. Grasedyck, *Hybrid cross approximation of integral operators*, Numer. Math., 101 (2005), pp. 221–249.

[15] D. Braess and W. Hackbusch, *A new convergence proof for the multigrid method including the V-cycle*, SIAM J. Numer. Anal., 20 (1983), pp. 967–975.

[16] A. N. Brooks and T. J. Hughes, *Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations*, Comput. Methods Appl. Mech. Engrg., 32 (1982), pp. 199–259.

[17] T. Bui-Thanh, C. Burstedde, O. Ghattas, J. Martin, G. Stadler, and L. C. Wilcox, *Extreme-scale UQ for Bayesian inverse problems governed by PDEs*, in SC12: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, Piscataway, NJ, 2012.

[18] 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 J. Sci. Comput., 35 (2013), pp. A2494–A2523.

[19] D. Calvetti, P. J. Hadwin, J. Huttunen, J. P. Kaipio, and E. Somersalo, *Artificial boundary conditions and domain truncation in electrical impedance tomography. Part* II: *Stochastic extension of the boundary map*, Inverse Probl. Imaging, 9 (2015), pp. 767–789.

[20] D. Calvetti, P. J. Hadwin, J. M. Huttunen, D. Isaacson, J. P. Kaipio, D. McGivney, E. Somersalo, and J. Volzer, *Artificial boundary conditions and domain truncation in electrical impedance tomography. Part* I: *Theory and preliminary results*, Inverse Probl. Imaging, 9 (2015), pp. 749–766.

[21] D. Calvetti, B. Lewis, and L. Reichel, *Restoration of images with spatially variant blur by the GMRES method*, in Advanced Signal Processing Algorithms, Architectures, and Implementations X, F. T. Luk, ed., Proc. SPIE 4116, SPIE, Bellingham, WA, 2000, pp. 364–374.

[22] R. H. Chan and M. K. Ng, *Conjugate gradient methods for Toeplitz systems*, SIAM Rev., 38 (1996), pp. 427–482.

[23] P. Chen, U. Villa, and O. Ghattas, *Taylor approximation and variance reduction for PDE-constrained optimal control under uncertainty*, J. Comput. Phys., 385 (2019), pp. 163–186.

[24] H. Cheng, Z. Gimbutas, P.-G. Martinsson, and V. Rokhlin, *On the compression of low rank matrices*, SIAM J. Sci. Comput., 26 (2005), pp. 1389–1404.

[25] J. Chiu and L. Demanet, *Matrix probing and its conditioning*, SIAM J. Numer. Anal., 50 (2012), pp. 171–193.

[26] T. Cui, J. Martin, Y. M. Marzouk, A. Solonen, and A. Spantini, *Likelihood-informed dimension reduction for nonlinear inverse problems*, Inverse Problems, 30 (2014), 114015.

[27] L. Denis, E. Thiébaut, F. Soulez, J.-M. Becker, and R. Mourya, *Fast approximations of shift-variant blur*, Int. J. Comput. Vis., 115 (2015), pp. 253–278.

[28] P. Escande and P. Weiss, *Approximation of integral operators using product-convolution expansions*, J. Math. Imaging Vision, 58 (2017), pp. 333–348.

[29] D. Fish, J. Grochmalicki, and E. Pike, *Scanning singular-value-decomposition method for restoration of images with space-variant blur*, J. Opt. Soc. Amer. A, 13 (1996), pp. 464–469.

[30] 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 J. Sci. Comput., 33 (2011), pp. 407–432.

[31] R. C. Flicker and F. J. Rigaut, *Anisoplanatic deconvolution of adaptive optics images*, J. Opt. Soc. Amer. A, 22 (2005), pp. 504–513.

[32] M. Gentile, F. Courbin, and G. Meylan, *Interpolating point spread function anisotropy*, Astron. Astrophy., 549 (2013), A1.

[33] E. Gilad and J. Von Hardenberg, *A fast algorithm for convolution integrals with space and time variant kernels*, J. Comput. Phys., 216 (2006), pp. 326–336.

[34] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, *A theory of pseudoskeleton approximations*, Linear Algebra Appl., 261 (1997), pp. 1–21.

[35] W. Hackbusch, *A sparse matrix arithmetic based on H-matrices. Part* I: *Introduction to H-matrices*, Computing, 62 (1999), pp. 89–108.

[36] W. Hackbusch, *Hierarchical Matrices: Algorithms and Analysis*, Springer Ser. Comput. Math. 49, Springer, Berlin, 2015.

[37] W. Hackbusch, B. Khoromskij, and S. A. Sauter, *On H^2-matrices*, in Lectures on Applied Mathematics: Proceedings of the Symposium Organized by the Sonderforschungsbereich 438 on the Occasion of Karl-Heinz Hoffmann's 60th Birthday, Munich, 1999, Springer, Berlin, 2000, pp. 9–30.

[38] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.

[39] M. Hirsch, S. Sra, B. Schölkopf, and S. Harmeling, *Efficient filter flow for space-variant multiframe blind deconvolution*, in 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Piscataway, NJ, 2010, pp. 607–614.

[40] T. Hrycak, S. Das, G. Matz, and H. G. Feichtinger, *Low complexity equalization for doubly selective channels modeled by a basis expansion*, IEEE Trans. Signal Process., 58 (2010), pp. 5706–5719.

[41] 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*, J. Comput. Phys., 296 (2015), pp. 348–368, http://doi.org/10.1016/j.jcp.2015.04.047.

[42] J. E. Jones and P. S. Vassilevski, *AMGe based on element agglomeration*, SIAM J. Sci. Comput., 23 (2001), pp. 109–133.

[43] P. Le Tallec and A. Patra, *Non-overlapping domain decomposition methods for adaptive hp approximations of the Stokes problem with discontinuous pressure fields*, Comput. Methods Appl. Mech. Engrg., 145 (1997), pp. 361–379.

[44] L. Lin, J. Lu, and L. Ying, *Fast construction of hierarchical matrix representation from matrix–vector multiplication*, J. Comput. Phys., 230 (2011), pp. 4071–4087.

[45] M. W. Mahoney, *Randomized algorithms for matrices and data*, Found. Trends Mach. Learn., 3 (2011), pp. 123–224.

[46] V. A. Morozov, *Methods for Solving Incorrectly Posed Problems*, Springer, New York, 1984.

[47] J. G. Nagy and D. P. O'Leary, *Restoring images degraded by spatially variant blur*, SIAM J. Sci. Comput., 19 (1998), pp. 1063–1082.

[48] J. Ng, R. Prager, N. Kingsbury, G. Treece, and A. Gee, *Wavelet restoration of medical pulse-echo ultrasound images in an EM framework*, IEEE Trans. Ultrason. Ferroelect. Freq. Control, 54 (2007), pp. 550–568.

[49] L. N. Olson and J. B. Schroder, *PyAMG: Algebraic Multigrid Solvers in Python Release* 4.0, https://github.com/pyamg/pyamg (2018).

[50] 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 J. Sci. Comput., 36 (2014), pp. A1525–A1555.

[51] C. Preza and J.-A. Conchello, *Depth-variant maximum-likelihood restoration for three-dimensional fluorescence microscopy*, J. Opt. Soc. Amer. A, 21 (2004), pp. 1593–1601.

[52] A. Rogers and J. D. Fiege, *Strong gravitational lens modeling with spatially variant point-spread functions*, Astrophys. J., 743 (2011), 68.

[53] Y. Saad and M. Sosonkina, *Distributed Schur complement techniques for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 1337–1356.

[54] A. Spantini, A. Solonen, T. Cui, J. Martin, L. Tenorio, and Y. Marzouk, *Optimal low-rank approximations of Bayesian linear inverse problems*, SIAM J. Sci. Comput., 37 (2015), pp. A2451–A2487.

[55] H. Trussell and B. Hunt, *Sectioned methods for image restoration*, IEEE Trans. Acoust. Speech Signal Process., 26 (1978), pp. 157–164.

[56] H. J. Trussell and S. Fogel, *Identification and restoration of spatially variant motion blurs in sequential images*, IEEE Trans. Image Process., 1 (1992), pp. 123–126.

[57] E. Tyrtyshnikov, *Incomplete cross approximation in the mosaic-skeleton method*, Computing, 64 (2000), pp. 367–380.

[58] 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.