

A kernel-independent precorrected-FFT method

Tristan Goodwill¹ and Owen Melia²

Abstract

The precorrected-FFT has been used to rapidly evaluate N -body sums arising in electromagnetics and acoustics settings. It has, however, been historically limited to kernels that satisfy a Green’s identity. In this work, we demonstrate that, using an improved version of the method of proxy points, the precorrected-FFT can be used to efficiently compute N -body sums with a wide variety of kernels. This algorithm requires no analytic information beyond pointwise access to the kernel. Our examples include kernels from non-local PDEs, Matérn kernels, anisotropic kernels, and the Green’s functions of high order PDEs. We also provide a detailed algorithm for choosing the proxy points and spreading grid, as implemented in the new `pcfft` package. We show these algorithmic choices result in favorable scaling when compared to other kernel-independent fast summation methods. Finally, we use the algorithm to accelerate a wide variety of integral equations solvers involving non-standard kernels, with applications in surface scattering, geophysics, and microbial community formation.

Keywords: Precorrected FFT, N-Body sums, Proxy shells

AMS subject classifications: 65R20, 65D30, 65F55

1 Introduction

In this work, we describe how the precorrected-FFT method [58, 59] can be used to compute N -body sums of the form

$$u(\mathbf{x}_i) = \sum_{\mathbf{y}_j \neq \mathbf{x}_i} K(\mathbf{x}_i - \mathbf{y}_j) \sigma_j \quad (1.1)$$

for a very general class of kernels. Here, the kernel $K : \mathbb{R}^d \rightarrow \mathbb{C}$ is a smooth function away from the origin, and the source $\mathbf{y}_j$ and target $\mathbf{x}_i$ points are general point clouds in $\mathbb{R}^d$, where the ambient dimension d is 2 or 3. N -body sums are ubiquitous in scientific computing, appearing in the numerical discretization of integral equation formulation of partial differential equations (PDEs) [41, 51, 56, 29], Gaussian processes [60, 1, 49, 70], spatial statistics [62, 14], molecular dynamics [44, 8], and many other settings. These applications often require repeated calculation of eq. (1.1), for very large N , which denotes the number of source points $\{\mathbf{y}_j\}_{j=1,\dots,N}$ and target points $\{\mathbf{x}_i\}_{i=1,\dots,N}$. A naïve method of evaluating eq. (1.1) requires $O(N^2)$ work, while “fast algorithms” for this problem, our method included, approximate the sum using a method requiring $O(N)$ or $O(N \log(N))$ work. It is a nontrivial problem to approximate the N -body sum, because the kernel K is non-smooth (or possibly diverging) when $\mathbf{x}_i$ approaches $\mathbf{y}_j$, which prevents the use of global function approximation methods. At the same time, these kernels often decay slowly in $\|\mathbf{x}_i - \mathbf{y}_j\|$, meaning interaction distances are large and all N^2 terms in the sum must be addressed. The precorrected-FFT method [58, 59] achieves this by approximating the potential from general source points with an equivalent set of charges on a regular grid, allowing the sum to be diagonalized with a pair of fast Fourier transform (FFT) and inverse transform (IFFT) operations. The nonuniform to regular approximation is only valid for sufficiently separated interactions, so interactions between nearby points must be “precorrected”. While past work has focused on applying this method to individual PDE kernels, here we propose a “kernel-independent” version of the algorithm which can automatically and efficiently apply to a broad class of problems given only pointwise access to the kernel. We also show how the method can be applied to a more general class of kernels using a recently-developed extension [3, 30] of the method of proxy points [72]. To emphasize the generality of the kernel-independent precorrected-FFT method, we provide numerical examples solving integral and integro-differential equations arising in biophysics, geophysics, and differential geometry.

¹Department of Mathematics, University of Florida, Gainesville, FL (tgoodwill@ufl.edu)

²Center for Computational Mathematics, Flatiron Institute, New York, NY (omelia@flatironinstitute.org)

The precorrected-FFT method is algorithmically similar to grid-based Ewald methods [22, 36, 16, 44, 8], which also spread nonuniform points to a regular grid, apply a convolution in Fourier space, and compute nearby interactions directly. For both precorrected-FFT and grid-based Ewald methods, the computationally intensive step is a FFT/IFFT pair, meaning both methods scale like $O(N \log(N))$. The conceptual difference between our proposed method and grid-based Ewald methods becomes apparent when one imagines the implementation. The prototypical five step Ewald method [44] can only be implemented *with a-priori knowledge of the kernel’s decay in Fourier space*, which determines the number of required Fourier modes, and, indirectly, the number of regular gridpoints. The method we develop is complementary but serves a different purpose: a method which can automatically and efficiently determine the number of gridpoints and Fourier modes required for a user-specified error tolerance, using only pointwise access to the kernel. We envision this will be particularly useful for applications in integral equation formulations of PDEs, in settings where the Fourier transform of the integro-differential operator is complicated [3, 33], or the kernel changes during the runtime of the algorithm [75, 68].

The precorrected-FFT method was originally developed by Phillips and White [58, 59] for the analysis of integrated circuits. It has gained popularity in circuit analysis [37]. Tissari and Rahola [63] use the method to accelerate forward solvers in a magnetoencephalography inverse problem in brain imaging. One of the authors of the present work has used the method in studies of surface wave interactions with floating plates and membranes [3]. The precorrected-FFT method has also been used to accelerate computations in nonlinear hydrodynamics [71, 39], acoustic wave propagation in shallow water [42], acoustic scattering by impenetrable obstacles [11, 57, 43], electromagnetic scattering by perfect conductors [54, 53], and electromagnetic analysis of thin multilayered conductors [55]. The precorrected-FFT method has been combined with a fast multipole method for large-scale aerodynamic simulations of aircraft [69].

Fast algorithms for kernel summation rely on analytic properties of the kernel K to form structured approximations of eq. (1.1) which can be exploited computationally. For instance, fast multipole methods [31, 13, 73] rely on a multipole or similar series expansion of K , which allows for rapid approximate interactions between sufficiently separated points. These methods incur complexity $O(N)$, and can adapt to highly-nonuniform distributions of points. Jiang and Greengard’s [38, 40] recent dual-space multilevel kernel framework performs hierarchical kernel splits into smooth and local kernels, efficiently handling highly-nonuniform point distributions.

While the fast multipole method enjoys an optimal computational complexity and adaptability to challenging distributions of points, grid-based Ewald methods are often preferable because the constant factors implicit in the $O(N \log N)$ estimate are quite small, FFT software is highly-optimized for modern computer architectures, and many point distributions of interest are relatively benign enough that multiresolution schemes are not needed. In this paper, we show our kernel-independent precorrected-FFT method enjoys the same qualities, which lead to large speedups when used in integral equation solvers.

The remainder of this paper is organized in follows. In section 2, we describe the kernel-independent precorrected FFT algorithm in detail and give theoretical and empirical evidence for our choice of proxy points in various nonstandard kernels. A MATLAB implementation of the method as described in this section is available at <https://github.com/meliao/pcfft>. We summarize the complexity of the algorithm and run a series of scalability studies in section 3. In section 4, we provide a series of numerical examples where the precorrected-FFT is used to accelerate a wide variety of integral equations solvers. Finally, we then conclude with some closing remarks in section 5.

2 Algorithm description

Fast algorithms for computing N -body sums with generic point locations tend to come in one of two categories. The first, category, which includes the fast multipole method [13, 31] and Ewald splitting schemes [22, 36, 16, 44, 8], leverage an analytical understanding of the kernel, and typically explicit expressions, to achieve their speed. Leveraging this understanding gives these methods good computational speed, but comes at the cost of being kernel-specific. The other category includes

fast algorithms that leverage the fact that for most kernels of interest the function $K(\mathbf{x} - \mathbf{y})$ can be well approximated by a low rank function for $(\mathbf{y}, \mathbf{x})$ in large swathes of the computational domain, typically for $\mathbf{y}$ in a ball and $\mathbf{x}$ outside a larger ball. Sometimes this takes the form of polynomial series approximations (e.g. [25]), but a more popular method for translation-invariant kernels K is to build an approximation of the form

$$K(\mathbf{x} - \mathbf{y}) \approx \sum_{\mathbf{z}_i \in \bar{\mathcal{B}}(\mathbf{y})} K(\mathbf{x} - \mathbf{z}_i) \chi_i(\mathbf{y}), \quad (2.1)$$

where $\bar{\mathcal{B}}(\mathbf{y})$ is a collection points near $\mathbf{y}$. This approximation can be thought of as “spreading” out the charge density $\sigma_j \rightarrow \{\chi_i(\mathbf{y})\}$ and is the basis of a number of algorithms, including recursive skeletonization [35, 47, 50], kernel-independent FMMs [46, 34], and the precorrected-FFT method [58, 59]. In the precorrected-FFT method, we lay down a regular grid of points $\mathbf{z}_i$ that cover the computational domain and choose $\mathcal{N}(\mathbf{y})$ to be a subset of that grid living near $\mathbf{y}$. In section 2.2 we describe how we choose the subset and the equivalent weights $\chi_i(\mathbf{y})$ so that this approximation eq. (2.1) holds for $\mathbf{x}$ that is well-separated from $\mathbf{y}$. A major contribution of the present work is establishing this approximation for kernels K which do not satisfy a Green’s identity.

If we neglect, for now, the fact that eq. (2.1) only holds for some $\mathbf{x}$, we arrive at the approximation

$$\mathbf{u} \approx \mathbf{K}_{\mathbf{x} \leftarrow \mathbf{z}} \mathbf{S}_{\mathbf{y}} \boldsymbol{\sigma},$$

where $(\mathbf{K}_{\mathbf{x} \leftarrow \mathbf{z}})_{i,j} = K(\mathbf{x}_i - \mathbf{z}_j)$ is dense, and $\mathbf{S}_{\mathbf{y}}$ is a sparse matrix of the spreading weights $\chi_i(\mathbf{y}_j)$. An equivalent approximation to eq. (2.1) will hold for $\mathbf{x}$. Nesting the two approximations yields a sum of the form:

$$\begin{aligned} \mathbf{u}(\mathbf{x}_k) &= \sum_{\mathbf{y}_l \neq \mathbf{x}_k} K(\mathbf{x}_k - \mathbf{y}_l) \sigma_l \\ &\approx \sum_{\mathbf{y}_l \neq \mathbf{x}_k} \left(\sum_{\mathbf{z}_i \in \bar{\mathcal{B}}(\mathbf{y}_l)} \left(\sum_{\mathbf{z}_j \in \bar{\mathcal{B}}(\mathbf{x}_k)} K(\mathbf{z}_j - \mathbf{z}_i) \tilde{\chi}_j(\mathbf{x}_k) \right) \chi_i(\mathbf{y}_l) \right) \sigma_l, \end{aligned}$$

where again, we are temporarily ignoring the “well-separated” condition for eq. (2.1). This nested summation can be organized into the composition of the “spreading” operation, a kernel summation on the approximation points $\mathbf{z}_i$, and the adjoint of another “spreading” operation:

$$\mathbf{u} \approx \mathbf{S}_{\mathbf{x}}^\top \mathbf{K}_{\mathbf{z} \leftarrow \mathbf{z}} \mathbf{S}_{\mathbf{y}} \boldsymbol{\sigma} \quad (2.2)$$

The advantage of this expression is that the only remaining dense matrix, $\mathbf{K}_{\mathbf{z} \leftarrow \mathbf{z}}$, has a multilevel Toeplitz structure because the $\mathbf{z}_i$ ’s form a d -dimensional grid. This allows us to apply it quickly using a zero-padded fast Fourier transform [26]:

$$\mathbf{K}_{\mathbf{z} \leftarrow \mathbf{z}} \mathbf{v} = \mathcal{F}^{-1} \widehat{\mathbf{K}_{\mathbf{z} \leftarrow \mathbf{z}}} \mathcal{F} \mathbf{v},$$

where $\mathcal{F}$ is a zero-padded d -dimensional fast Fourier transform, $\mathcal{F}^{-1}$ is its inverse, and $\widehat{\mathbf{K}_{\mathbf{z} \leftarrow \mathbf{z}}}$ is the Fourier transform of K evaluated on the regular grid. Figure 1 shows the structure of the operators used to compute the approximate N -body sum in eq. (2.2).

Of course, the approximations required to arrive at eq. (2.2) are only valid when $\mathbf{x}$ and $\mathbf{y}$ are sufficiently well-separated. To that end, we introduce another sparse matrix, $\mathbf{A}$, which corrects the interactions of nearby points where eq. (2.1) was used erroneously. Thus, our approximation for general sets of source and target points is

$$\mathbf{u} \approx \mathbf{S}_{\mathbf{x}}^\top \mathbf{K}_{\mathbf{z} \leftarrow \mathbf{z}} \mathbf{S}_{\mathbf{y}} \boldsymbol{\sigma} + \mathbf{A} \boldsymbol{\sigma}. \quad (2.3)$$

Algorithm 1 describes how we evaluate the N -body sum once the matrices $\mathbf{S}_{\mathbf{x}}$, $\mathbf{S}_{\mathbf{y}}$, $\widehat{\mathbf{K}_{\mathbf{z} \leftarrow \mathbf{z}}}$, and $\mathbf{A}$ are pre-computed. We describe the process of computing $\mathbf{A}$ in section 2.1. We discuss our method for choosing the regular grid $\mathbf{z}_i$ and spreading weights $\chi_i(\mathbf{x})$ in section 2.2. We test these spreading algorithms in section 2.3 and introduce a method for applying the same approximation to derivatives of the kernel in section 2.4.

Algorithm 1: PCFFT APPLY

Input: spreading matrices S_x, S_y ; padded discrete Fourier transform of kernel $\widehat{K}_{z \leftarrow z}$; correction matrix A ; source weights σ .

- 1 Spread sources to the regular grid $S_y \sigma$
- 2 Evaluate discrete Fourier transform of sources $\text{IFFT} \left(\widehat{K}_{z \leftarrow z} \cdot \text{FFT} (S_y \sigma) \right)$
- 3 Interpolate from regular grid to targets $S_x^\top \text{IFFT} \left(\widehat{K}_{z \leftarrow z} \cdot \text{FFT} (S_y \sigma) \right)$
- 4 Correct near interactions $A \sigma$
- 5 $u \leftarrow S_x^\top \text{IFFT} \left(\widehat{K}_{z \leftarrow z} \cdot \text{FFT} (S_y \sigma) \right) + A \sigma$

Result: u

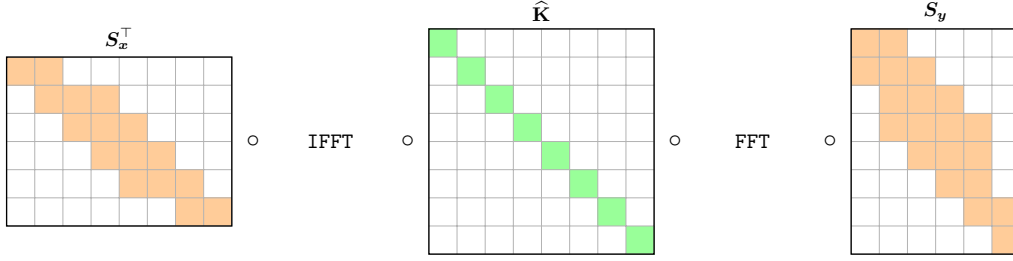


Figure 1: The matrix decomposition in eq. (2.2), read right to left: S_y (orange) is sparse with contiguous nonzeros per row; $\widehat{K}$ (green) is diagonal, holding the Fourier transform of K on the grid; and $S_x^\top$ (orange) is sparse with contiguous nonzeros per column.

2.1 Corrections for nearby targets

The first term in eq. (2.3) applies a far-field approximation to all of the pairs of points (x_i, y_j) , but is only valid for those pairs of points which are “well-separated”. To identify these points, we split the computational domain up into regular sized “spreading bins” with side length $2h$. For any point y in a spreading bin $\mathcal{B}_k \subset \mathbb{R}^d$, we will spread y to all the regular grid points that live in a “spreading box”, $\bar{\mathcal{B}}_k$, which is twice the size of the spreading bin; see fig. 2a. We will say that a point x is well-separated from $\mathcal{B}_k$ if it is at least a distance $\rho_1 = 2\sqrt{d}h$ from the center of $\mathcal{B}_k$. In other words, we will demand that eq. (2.1) is valid for x a distance ρ_1 from the center of $\mathcal{B}_k$. Equation (2.2) will therefore be valid when every point in $\bar{\mathcal{B}}(x_k)$ is well separated from every point in $\bar{\mathcal{B}}(y_l)$. We can correct the errors made by eq. (2.2) by subtracting the contribution of this approximation for all pairs of points belonging to spreading bins whose centers lie within $2\rho_1$ of each other (fig. 2b) and then adding the correct contributions.

Remark 1. A careful examination of fig. 2b will show that we can actually use eq. (2.2) for boxes that are slightly closer than $2\rho_1$. We give ourselves the extra margin for stability, as it does not substantially increase the cost of the algorithm.

In order to compute these contributions, we let

$$\mathcal{N}_k = \{z \in \bar{\mathcal{B}}_j \mid \bar{\mathcal{B}}_j \text{ is near } \mathcal{B}_k\}$$

be the neighborhood of the spreading bin $\mathcal{B}_k$. To evaluate the contribution of nearby points to eq. (2.2), we construct a template for evaluating the kernel from a spreading box $\bar{\mathcal{B}}$ to a neighborhood $\mathcal{N}$:

$$(K_{\mathcal{N} \leftarrow \bar{\mathcal{B}}})_{i,j} = K(z_j - z_i),$$

where the points $\{z_j\}$ are the regular discretization points in $\mathcal{N}_k$ and the points $\{z_i\}$ are the regular discretization points in $\bar{\mathcal{B}}_k$. Due to the translation invariance of the kernel $K(\cdot)$, we only need to

precompute this matrix once, and we can use it for all bins $\mathcal{B}_k$. We can now compute the correction matrix $\mathbf{A}$:

$$\mathbf{A}(\mathcal{I}_k, \mathcal{J}_k) = \mathbf{D}_k - \mathbf{S}_x(:, \mathcal{I}_k)^\top \mathbf{K}_{\mathcal{N}_k \leftarrow \bar{\mathcal{B}}} \mathbf{S}_y(:, \mathcal{J}_k),$$

where $\mathcal{I}_k$ is the set of indices for target points $\mathbf{x}_i$ in neighborhood $\mathcal{N}_k$, $\mathcal{J}_k$ is the set of indices for source points $\mathbf{y}_j$ in spreading bin $\mathcal{B}_k$, $\mathbf{D}_k$ is the matrix of direct interactions between these points, and all other entries of $\mathbf{A}$ are zero. Algorithm 2 summarizes this process of computing the correction matrix.

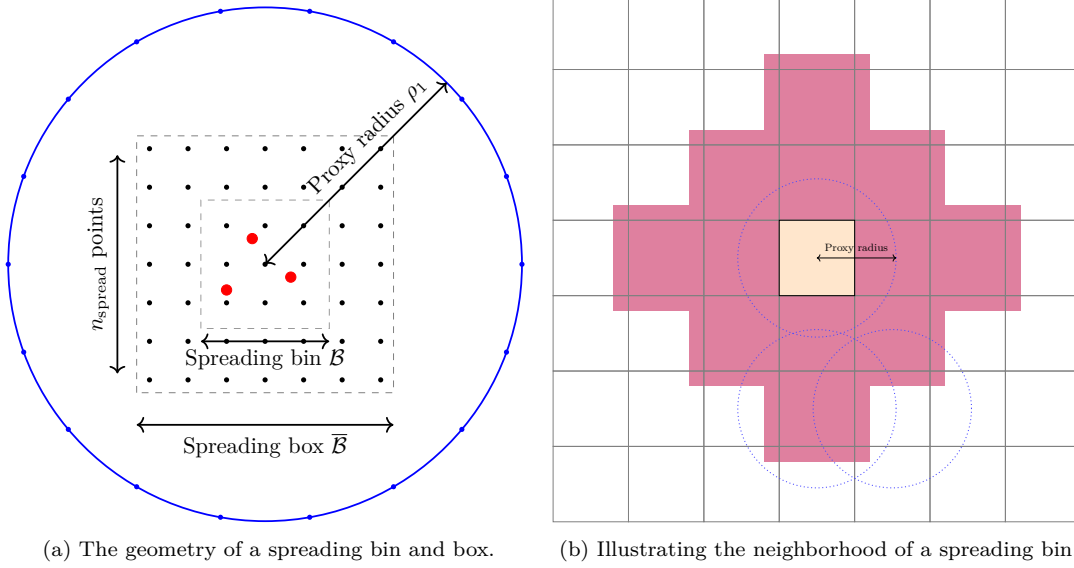


Figure 2: Figure 2a: the geometry of the spreading operation. Source or target points (red dots) belonging to the spreading bin are spread to the regular grid points (black dots) in the spreading box. The proxy surface is a constant radius ρ_1 away from the center of the spreading box. In this subfigure, $n_{\text{shell}} = 1$, $n_{\text{proxy}} = 18$, and $n_{\text{spread}} = 7$. Figure 2b: our criteria for nearness is whether the inner proxy surfaces of two spreading bins intersect. The proxy surfaces of a few bins are shown in blue. The spatial support of bin $\mathcal{B}_k$ is shown in orange, and the spatial support of its neighborhood $\mathcal{N}_k$ is shown in pink.

Algorithm 2: COMPUTE_CORRECTION

Input: Target points $\{\mathbf{x}_i\}_{i=1, \dots, N_{\text{src}}}$; source points $\{\mathbf{y}_j\}_{j=1, \dots, N_{\text{targ}}}$; spreading matrices $\mathbf{S}_y, \mathbf{S}_x$; neighborhood template $\mathbf{K}_{\mathcal{N}_k \leftarrow \bar{\mathcal{B}}}$; kernel $K(\cdot)$.

- 1 Initialize $\mathbf{A}$ to zero.
- 2 **for** Bin $\mathcal{B}_k \in \{\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_{N_{\text{bin}}}\}$ **do**
- 3 Let $\mathcal{I}_k$ index the target points in the neighborhood of $\mathcal{B}_k$.
- 4 Let $\mathcal{J}_k$ index the source points in spreading box $\bar{\mathcal{B}}_k$.
- 5 Compute exact interactions $\mathbf{D}_{i,j} = K(\mathbf{x}_i, \mathbf{y}_j)$ for each $\mathbf{x}_i \in \mathcal{I}_k$ and $\mathbf{y}_j \in \mathcal{J}_k$.
- 6 Compute approximate interactions $\mathbf{S}_x(:, \mathcal{J}_k)^\top \mathbf{K}_{\mathcal{N}_k \leftarrow \bar{\mathcal{B}}} \mathbf{S}_y(:, \mathcal{I}_k)$.
- 7 Add and subtract these matrices to the sparse matrix

$$\mathbf{A}(\mathcal{I}_k, \mathcal{J}_k) \leftarrow \mathbf{D} - \mathbf{S}_x(:, \mathcal{J}_k)^\top \mathbf{K}_{\mathcal{N}_k \leftarrow \bar{\mathcal{B}}} \mathbf{S}_y(:, \mathcal{I}_k).$$

Result: $\mathbf{A}$

2.2 Finding the spreading parameters

This section discusses how we automatically choose how many discretization points $\mathbf{z}_j$ live in each $\mathcal{N}(\mathbf{y}_k)$ and how to find the equivalent charges χ_j so that eq. (2.2) holds for well-separated points.

This method is designed so that for any distribution of source points $\mathbf{y}_i$ in a spreading bin $\mathcal{B}_k$, we can automatically map charges $\boldsymbol{\sigma}$ to equivalent charges $\boldsymbol{\chi}$ supported on the pre-selected grid points $\mathbf{z}_j \in \overline{\mathcal{B}_k}$.

First, we choose h , the half side length of the spreading bin so that each points has around n_{nbr} points in its bin. This parameter can be chosen to balance the cost of the FFTs against the cost of the sparse corrections $\mathbf{A}$ (discussed below in section 3). We find h by constructing an adaptive hyperoctree of the union of source and target points such that no leaf box has more than $n_{\text{nbr}}/2^d$ points. We then set h to be

$$h = \begin{cases} \frac{\bar{h}}{4\sqrt{\pi}} \left(\frac{N_{\text{pts}}}{\hat{n}} \right)^{1/2} & d = 2, \\ \frac{\bar{h}}{4\sqrt{\pi}} \left(\frac{3\sqrt{\pi}}{4} \right)^{1/3} \left(\frac{N_{\text{pts}}}{\hat{n}} \right)^{1/3} & d = 3, \end{cases},$$

where $\bar{h}$ is the mean leaf box side length, and $\hat{n}$ is the mean occupancy of each leaf box. In our implementation, we construct the adaptive hyperoctree using the utility provided in the **FLAM** package [34].

Once we have determined h , and so the bins $\mathcal{B}_k$, we turn to finding spreading weights $\boldsymbol{\chi}$. Given a user-specified error tolerance ϵ , source points $\mathbf{y} \in \mathcal{B}_k$, and target points $\mathbf{x}$ that are well separated from the spreading bin, we want weights $\boldsymbol{\chi}$ which satisfy

$$\frac{\|\mathbf{K}_{\mathbf{x} \leftarrow \mathbf{z}} \boldsymbol{\chi} - \mathbf{K}_{\mathbf{x} \leftarrow \mathbf{y}} \boldsymbol{\sigma}\|_2}{\|\mathbf{K}_{\mathbf{x} \leftarrow \mathbf{y}} \boldsymbol{\sigma}\|_2} < \epsilon. \quad (2.4)$$

To find these weights, we introduce a set of “proxy points”, $\mathbf{p}_j$, living outside of the spreading box $\overline{\mathcal{B}_k}$. The proxy points give us a set of target points, at which the kernel evaluations must match between the original charges and the equivalent charges. This constraint can be written as the linear problem for $\boldsymbol{\chi}$:

$$\mathbf{K}_{\mathbf{p} \leftarrow \mathbf{z}} \boldsymbol{\chi} = \mathbf{K}_{\mathbf{p} \leftarrow \mathbf{y}} \boldsymbol{\sigma}. \quad (2.5)$$

We choose $\boldsymbol{\chi}$ as the minimum ℓ_2 norm least squares solution of this equation by computing a complete orthogonal decomposition of $\mathbf{K}_{\mathbf{p} \leftarrow \mathbf{z}}$, thresholding with tolerance ϵ , and computing the least-squares solution.

The idea of using proxy points has a long history in the development of kernel-independent FMMs and broader classes of kernel matrix approximation methods; see [72] for an overview. The use of proxy points is also standard in pre-corrected FFT algorithms. Phillips and White [58, 59] use proxy points supported on a sphere to define a set of linear constraints, which is solved in a least-squares sense via the pseudoinverse computed with the SVD. Error analysis is provided for Helmholtz and Laplace kernels. Nie et al. [53] uses the same method applied to the electric field integral equation kernel, which includes a derivative. Bruno and Kunyansky [11] use proxy points on a sphere for the single- and double-layer Helmholtz kernels. Ye et al. [72] analyze the accuracy of the proxy point method for the Laplace kernel when the proxy points are chosen uniformly on the proxy surface; in this case they show the approximation error converges exponentially with the number of proxy points.

In all of these works, the proxy points are arranged on a single curve or surface. A PDE stability theorem or Green’s identity can be used to show that this is enough to guarantee accuracy of eq. (2.1) outside of that shell. Indeed, if the proxy points form good interpolation nodes (we use equispaced points when $d = 2$ and Fibonacci spiral points when $d = 3$), eq. (2.5) will guarantee that

$$K(\mathbf{x} - \mathbf{y}) - \sum_{\mathbf{z}_i \in \mathcal{N}(\mathbf{y})} K(\mathbf{x} - \mathbf{z}_i) \chi_i(\mathbf{y}), \quad (2.6)$$

is a solution of the PDE with small Dirichlet data determined by the interpolation error. This will guarantee that it is small for all $\mathbf{x}$ outside the shell. For kernels based on higher order PDEs, PDE stability theorems require more than just Dirichlet boundary data. We discuss how we handle this case in section 2.4.

For kernels that do not arise from a Green’s function of an elliptic PDE, we can not expect sufficient accuracy when using a single proxy surface. Instead, we enforce agreement on a set of n_{shell} proxy surfaces, with radii ρ_k . We choose ρ_1 in the same way as described above and choose the remaining shells spaced so that $1/\rho_k$ are second-kind Chebyshev nodes. If $\{x_k\}_{k=2,\dots,n_{\text{shell}}}$ are the second-kind Chebyshev points on the interval $(-1, 1)$, then the radii are

$$\rho_k = \frac{2\rho_1}{x_k + 1} = \frac{2\rho_1}{\cos\left(\frac{(k-1)\pi}{n_{\text{shell}}-1}\right) + 1}, \quad k = 2, \dots, n_{\text{shell}}.$$

This choice is motivated by [30, Thrm. 4], where the authors phrase the approximation eq. (2.1) as interpolation problem of each angular Fourier mode of $K(\cdot - \mathbf{y})$. They use this framing to show that if K is some combination of derivatives of $\log(\cdot)$ in $\mathbb{R}^3$, then choosing the shells to be reciprocal Chebyshev nodes guarantees spectral convergence in the approximation eq. (2.1) as n_{shell} grows, up to the error determined by the number of nodes on each shell, n_{proxy} . Their proof technique can be directly extended to other kernels that have multipole expansions in a lower dimension and we find this choice works well for all of the kernels considered in this work. We add the extra shell at ρ_1 for stability and to provide uniformity with the single shell case. We find our choice is an improvement over the method suggested by Askham et al. [3], who choose ρ_k so that $1/\rho_k$ are equally-spaced.

Ultimately, our algorithm has three kernel-specific parameters: the number of grid points per box in each direction, n_{spread} ; the number of proxy points on each shell, n_{proxy} ; and the number of proxy shells, n_{shell} . See fig. 2a for an example of these parameters. For individual kernels, analysis of multipole expansions allow us to derive how these parameters scale with tolerance ϵ , using arguments similar to those in [30]. These estimates are listed in table 1 for a few representative examples.

Kernel	d	n_{spread}	n_{proxy}	n_{shell}
$\log(r)$	2	$O(\log \epsilon ^{1/2})$	$O(\log \epsilon)$	$O(1)$
$\log(r)$	3	$O(\log \epsilon)$	$O(\log \epsilon ^2)$	$O(\log \epsilon)$
$1/r$	3	$O(\log \epsilon ^{2/3})$	$O(\log \epsilon ^2)$	$O(1)$

Table 1: The dependence of n_{spread} , n_{proxy} , and n_{shell} on ϵ for a few representative kernels.

While estimates exist for particular kernels, we use a purely empirical approach to choosing $(n_{\text{spread}}, n_{\text{proxy}}, n_{\text{shell}})$ in this work, which we find to be robust and efficient. First, we choose n_{proxy} so that $K(\mathbf{p} - \mathbf{y}_j)$ is a well resolved function of $\mathbf{p}$ on the shell of radius ρ_1 for a few sources $\mathbf{y}_j$ by checking the Fourier series or spherical harmonic tails in two or three dimensions, respectively. We do this using 117 random $\mathbf{y}_j$ inside a box with half side length h , as well as its corners. We then compute the coefficients of a series expansion of $K(\mathbf{p} - \mathbf{y})$ in a Fourier series ($d = 2$) or a spherical harmonic series ($d = 3$); we ensure there are enough discretization points $\mathbf{p}$ to capture the tails of this series. In both cases, we give ourselves a margin to account for the Lesbegue constant of the interpolation by multiplying the minimal n_{proxy} by $(4/3)^{d-1}$, which corresponds to enforcing accuracy if we chop off the last quarter of the orders in the Fourier or spherical harmonic series expansion of $K(\mathbf{p} - \mathbf{y})$.

Once we have determined n_{proxy} , we sweep through possible pairs $(n_{\text{spread}}, n_{\text{shell}})$ looking for an allowable pair. For each proposed n_{spread} , we sweep up in n_{shell} to see if eq. (2.4) is satisfied for 500 targets points with random radii in $(\rho_1, 2\rho_1)$ and spread uniformly in angle. If none are found for n_{shell} less than some maximum (typically 20), we increase n_{spread} by one and then sweep through n_{shell} again. If the kernel is known to satisfy a Green’s identity, then we force $n_{\text{shell}} = 1$ and simply sweep in n_{spread} . Our algorithm for computing $(n_{\text{spread}}, n_{\text{proxy}}, n_{\text{shell}})$ is summarized in algorithm 3.

Remark 2. We will see below (e.g. in fig. 4b) that this algorithm can occasionally lead to large n_{shell} when a given n_{spread} is barely enough to achieve a tolerances. We have found that this large n_{shell} can be avoided by checking if there is a smaller number of shells that works with $n_{\text{spread}} + 1$ grid points in each direction. In practice, we use algorithm 3 as it yields the smallest n_{spread} , which has a bigger impact on the code speed.

Algorithm 3: NSPREAD_NPROXY_NSHELL

Input: Spreading bin sidelength h ; proxy radius ρ ; requested tolerance ϵ ; kernel $K(\cdot)$; `multishell` flag.

- 1 If `multishell`, $n_{\max} \leftarrow 20$, else $n_{\max} \leftarrow 1$.
- 2 $\mathbf{y} \leftarrow$ 117 points randomly sampled uniform on $[-h, h]^d$ and append the corners.
- 3 Find n_{proxy} that resolves $K(\mathbf{p} - \mathbf{y})$ on the first proxy shell.
- 4 $n_{\text{spread}} \leftarrow 1$.
- 5 $n_{\text{shell}} \leftarrow 1$.
- 6 While eq. (2.4) is unsatisfied, increase n_{shell} . If $n_{\text{shell}} > n_{\max}$, $n_{\text{spread}} \leftarrow n_{\text{spread}} + 1$ and return to step 5. If $n_{\text{spread}} > 100$, abort.

Result: n_{spread} ; n_{proxy} ; n_{shell} .

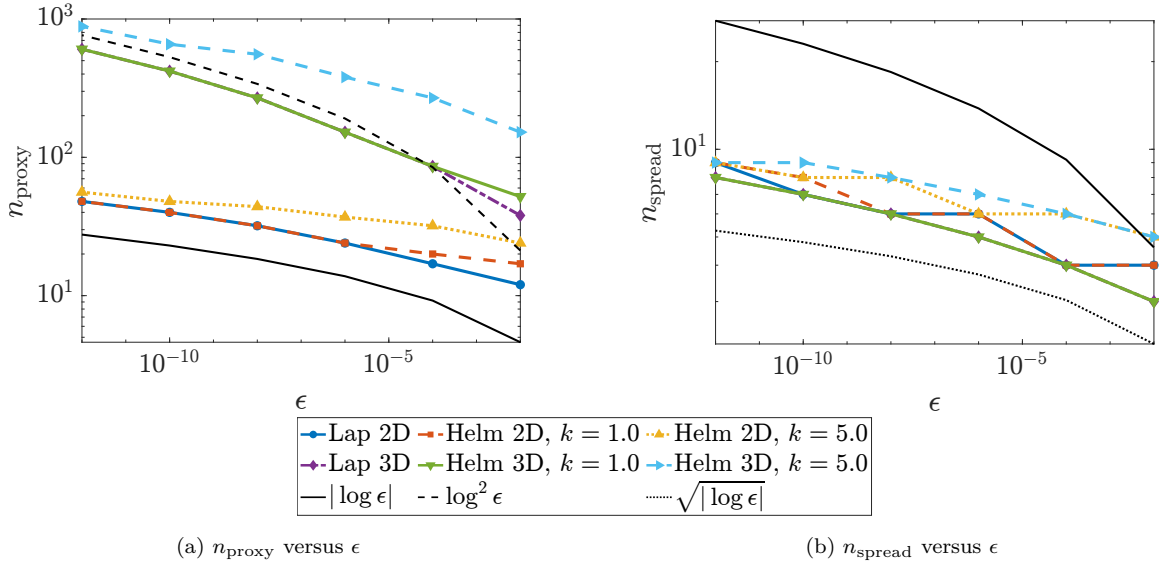


Figure 3: The selected n_{proxy} (3a) and n_{spread} (3b) for the Laplace and Helmholtz Green's functions in 2D and 3D, as functions of the tolerance ϵ .

2.3 Proxy point tests

In this section, we test algorithm 3 and demonstrate how n_{spread} , n_{proxy} , and n_{shell} depend on ϵ for a variety of different kernels. In all of the experiments we choose the half side length $h = 1$. In fig. 3, we show how n_{proxy} and n_{spread} depend on the tolerance for the Laplace and Helmholtz Green's functions in 2D and 3D. As predicted in table 1, we see that n_{proxy} scales as $|\log \epsilon|$ in 2D and $\log^2 \epsilon$ in 3D, and that n_{spread} grows a rate between $\sqrt{|\log \epsilon|}$ and $|\log \epsilon|$. We also see that n_{spread} increases with the frequency of the Helmholtz problem. In fig. 4 we repeat this experiment, but we swap the domain dimensions. This time the kernels do not satisfy Green's identities and so we run the algorithms in multi-shell mode. The figure shows that n_{proxy} and n_{shell} behave in a similar dimension-dependent way. We also see that the number of shells grows roughly as $|\log \epsilon|$ as the tolerance shrinks and that three dimensional kernels in 2D generally need a few more shells. The figures also show that the interaction rank with these kernels is slightly higher in this experiment than the previous one, as is exhibited by the increased n_{spread} and number of proxy points. The large n_{shell} at $\epsilon \sim 10^{-4}$ can be prevented using the approach described in remark 2.

Having tested the algorithm's performance on the Laplace and Helmholtz Green's functions, we now move to a broader class of kernels. In fig. 5 we run an equivalent test for the Matérn kernel in 2D, which is written in terms of the modified Bessel function of the second kind as $C_\nu(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} (2\sqrt{\nu}r)^\nu K_\nu(\sqrt{2\nu}r)$ and is a standard kernel in geospatial statistics (see [14] and the

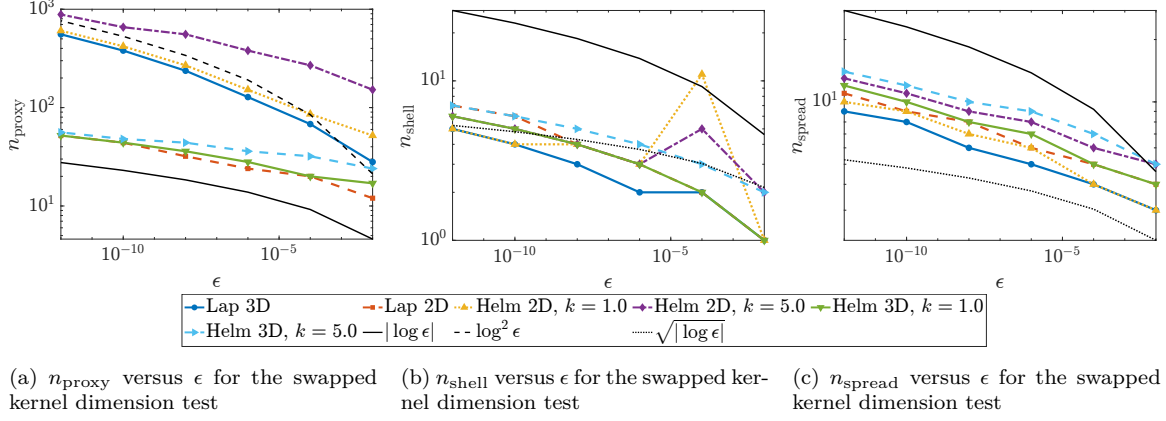


Figure 4: The selected n_{proxy} (4a), n_{shell} (4b), and n_{spread} (4c) for the Laplace and Helmholtz Green's functions in 2D and 3D in the swapped dimension, as functions of the tolerance ϵ . For example, the figure legend "Lap 2D" indicates $K(\mathbf{x} - \mathbf{y}) = \frac{1}{\|\mathbf{x} - \mathbf{y}\|}$ for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$.

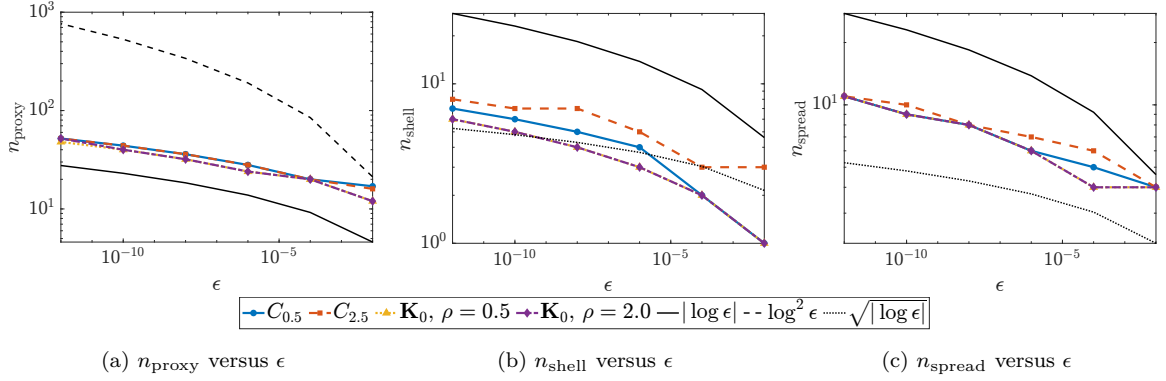


Figure 5: The selected n_{proxy} (5a), n_{shell} (5b), and n_{spread} (5b) for the Matérn kernel C_ν and Struve function $\mathbf{K}_0(\rho r)$ in 2D, as functions of the tolerance ϵ .

references therein) and the Struve function $\mathbf{K}_0(\rho r)$ [17, eqn. 11.2.5] in 2D, which occurs in surface wave problems (see [4]). In fig. 6 we also test Gaussian kernels with covariance

$$C_{2D} = \frac{1}{16} \begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix} \quad \text{and} \quad C_{3D} = \frac{1}{16} \begin{pmatrix} 2 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

in 2D and 3D respectively, and $f(r) = \sin r / (r + r_1^2)$. Both tests show that the number of shells again grows slowly with decreasing tolerance and that n_{proxy} grows logarithmically. Figure 8c shows our method attains high accuracy on these kernels which do not arise from the Green's function of an elliptic PDE.

As our final experiment in this vein, we test our algorithm on higher order equations, namely equations defined by increasing powers of the Laplacian. This time a single shell is sufficient, but we also enforce agreement of some radial derivatives ($n - 1$ derivatives for Δ^n) at our proxy points. Figure 7 shows that the higher order equations require a slightly higher n_{proxy} and n_{spread} . The spike in n_{proxy} for the smallest ϵ is caused by the error in the finite difference scheme we used to compute the radial derivatives.

2.4 Derivatives of the kernel

Many N -body calculations involve source or target derivatives of the kernel. Indeed, almost all boundary integral equation methods require them. Part of the power of the proxy point method,

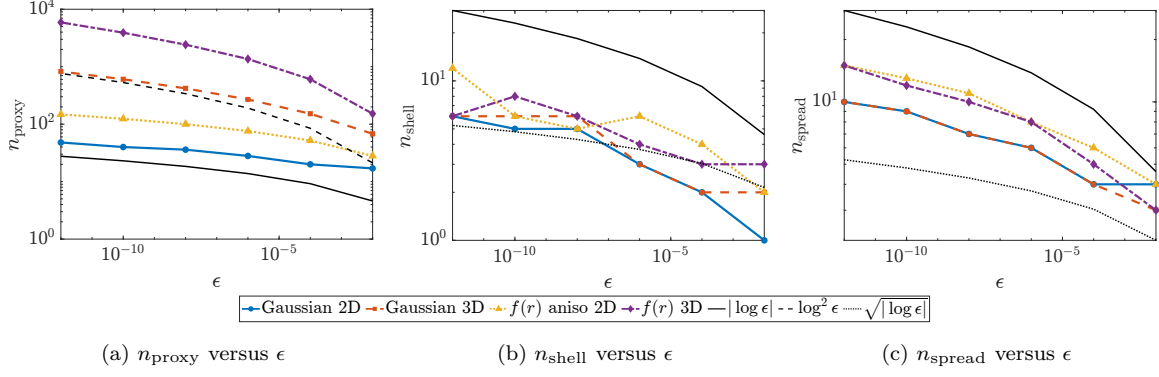


Figure 6: The selected n_{proxy} (6a), n_{shell} (6b), and n_{spread} (6c) for an anisotropic Gaussian and $f(r) = \frac{\sin r}{r+r_1^2}$. Figure 8c shows the accuracy of the N -body sum for these examples.

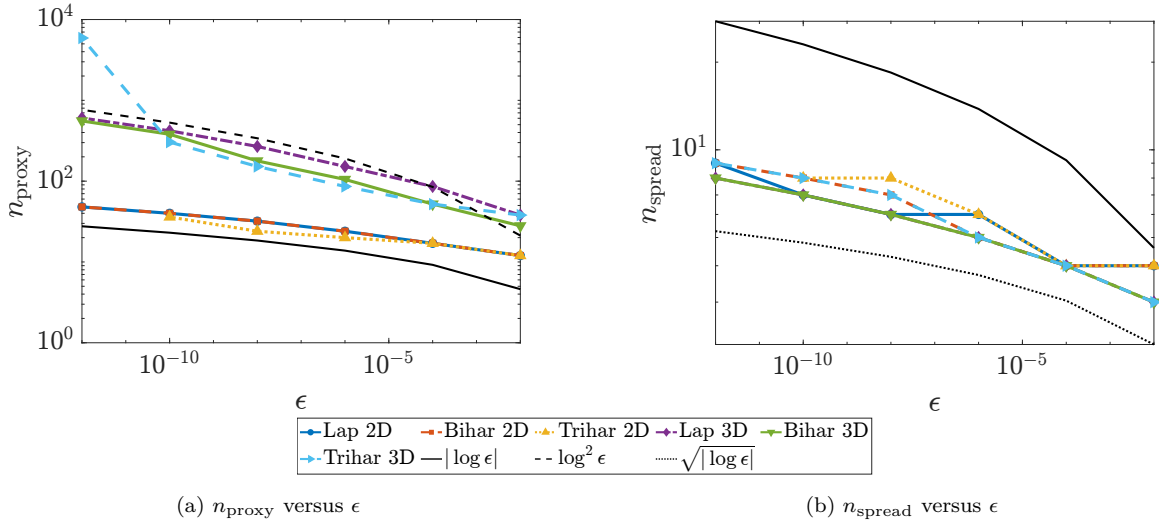


Figure 7: The selected n_{proxy} (7a) and n_{spread} (7b) for the Green's functions for the Laplace, biharmonic, and triharmonic problems in 2D and 3D, as functions of the tolerance ϵ . Because these kernels all arise from Green's functions of elliptic PDEs, we use $n_{\text{shell}} = 1$.

is that when approximation eq. (2.1) holds for the original kernel $K(\mathbf{x}, \mathbf{y})$, we can expect a similar approximation to hold for the derivative of the kernel $K'(\mathbf{x}, \mathbf{y}) = \mathbf{n}(\mathbf{y}) \cdot \nabla_{\mathbf{y}} K(\mathbf{x}, \mathbf{y})$. More formally, let $\mathcal{W}_k$ be the set of points $\mathbf{x}$ that are at least ρ_1 away from the center of spreading bin $\mathcal{B}_k$. Then the set of functions f such that

$$\text{dist}_{C_b(\mathcal{W}_k)}(f, \text{span}\{K(\cdot - \mathbf{z}_i) \mid \mathbf{z}_i \in \overline{\mathcal{B}_k}\}) \leq \epsilon$$

is a closed subset of $C_b(\mathcal{W}_k)$ (the set of bounded continuous functions) in the sup norm. Thus, if K is decays at infinity and is smooth enough that

$$\lim_{h \rightarrow 0} \frac{K(\mathbf{x} - (\mathbf{y} + h\mathbf{n}(\mathbf{y}))) - K(\mathbf{x} - \mathbf{y})}{h}$$

converges to $\mathbf{n}(\mathbf{y}) \cdot \nabla_{\mathbf{y}} K(\mathbf{x} - \mathbf{y})$ in $C_b(\mathcal{W}_k)$, then its derivative will also admit the low rank factorization. This result gives us the following theorem.

Theorem 1. *If K is a kernel that is bounded outside the ball B and admits a low rank factorization*

of the form eq. (2.1) in $C_b(\mathcal{W}_k)$, then for all $\epsilon > 0$ there exists a spreading grid such that

$$\sup_{\mathbf{x} \in \mathcal{W}_k} \left| \mathbf{n}(\mathbf{y}) \cdot \nabla_{\mathbf{y}} K(\mathbf{x} - \mathbf{y}) - \sum_{\mathbf{z}_i \in \mathcal{N}(\mathbf{y})} K(\mathbf{x} - \mathbf{z}_i) \tilde{\chi}_i(\mathbf{y}) \right| < \epsilon \quad (2.7)$$

for some weights $\tilde{\chi}_i(\mathbf{y})$, for all $\mathbf{y} \in \mathcal{B}_k$. Similar expressions hold for other derivatives of K .

For kernels that grow at infinity, the same argument works if we weight the norm of $C(\mathcal{W}_k)$ appropriately. Numerically, we first determine the number of proxy points required without any derivatives using algorithm 3, then find the weights $\tilde{\chi}_i$ by enforcing eq. (2.7) at those proxy points. After they are known, we can compute N -body sums using a very slight modification of algorithm 2 and eq. (2.3).

To test the effect of these derivatives on the accuracy of our algorithm, we run two experiments. In both, we measure the relative ℓ_∞ error in the N -body calculation for 2,000 sources and 2,017 targets with a uniform random distribution in the unit box. In the first experiment (fig. 8a), we use our algorithm to compute N -body sums for derivatives of the Laplace Green's function, G_L ,

$$\begin{aligned} k_S(\mathbf{x}, \mathbf{y}) &= G_L(\mathbf{x} - \mathbf{y}), \quad k_D(\mathbf{x}, \mathbf{y}) = -\mathbf{n}(\mathbf{y}) \cdot \nabla G_L(\mathbf{x} - \mathbf{y}), \\ \text{and} \quad k_{D'}(\mathbf{x}, \mathbf{y}) &= -\mathbf{n}(\mathbf{x}) \cdot \nabla(\mathbf{n}(\mathbf{y}) \cdot \nabla G_L(\mathbf{x} - \mathbf{y})). \end{aligned}$$

We see that the derivatives can decrease the accuracy, likely because the spread charges are larger, but the error still scales with the tolerance.

As a second experiment, we explore the accuracy in the kernels that arise when using a standard integral equation method for the flexural wave equation with clamped plate boundary conditions [52, 45, 23]. If Ω is a compact subset of $\mathbb{R}^2$ with a smooth boundary, then these equations can be stated as

$$\begin{cases} (\Delta^2 - k^4)u = 0 & \text{in } \mathbb{R}^2 \setminus \Omega, \\ u = g_1 & \text{on } \partial\Omega, \\ \partial_{\mathbf{n}} u = g_2 & \text{on } \partial\Omega, \end{cases} \quad (2.8)$$

where $\mathbf{n}$ is the unit normal to $\partial\Omega$, and g_1 and g_2 are Dirichlet and Neumann boundary data, respectively. The integral equation is built out of the flexural Green's function, which satisfies

$$(\Delta^2 - k^4)G_F = \delta.$$

The kernels that occur in the integral equation formulation are

$$\begin{aligned} k_{\text{BC}} &= \begin{bmatrix} G_F \\ \partial_{\mathbf{n}(\mathbf{x})} G_F \end{bmatrix}, \quad k_{\text{rep}} = \begin{bmatrix} \partial_{\mathbf{n}(\mathbf{y})}^3 G_F + 3\partial_{\mathbf{n}(\mathbf{y})} \partial_{\boldsymbol{\tau}(\mathbf{y})}^2 G_F & -\partial_{\mathbf{n}(\mathbf{y})}^2 G_F + \partial_{\boldsymbol{\tau}(\mathbf{y})}^2 G_F \end{bmatrix}, \\ \text{and} \quad k_{\text{BIE}} &= \begin{bmatrix} \partial_{\mathbf{n}(\mathbf{y})}^3 G_F + 3\partial_{\mathbf{n}(\mathbf{y})} \partial_{\boldsymbol{\tau}(\mathbf{y})}^2 G_F & -\partial_{\mathbf{n}(\mathbf{y})}^2 G_F + \partial_{\boldsymbol{\tau}(\mathbf{y})}^2 G_F \\ \partial_{\mathbf{n}(\mathbf{x})} \partial_{\mathbf{n}(\mathbf{y})}^3 G_F + 3\partial_{\mathbf{n}(\mathbf{x})} \partial_{\mathbf{n}(\mathbf{y})} \partial_{\boldsymbol{\tau}(\mathbf{y})}^2 G_F & -\partial_{\mathbf{n}(\mathbf{x})} \partial_{\mathbf{n}(\mathbf{y})}^2 G_F + \partial_{\mathbf{n}(\mathbf{x})} \partial_{\boldsymbol{\tau}(\mathbf{y})}^2 G_F \end{bmatrix}, \end{aligned}$$

where $\boldsymbol{\tau}$ is the unit tangent to $\partial\Omega$. In fig. 8b we see that we achieve accuracy that scales with the tolerance for all of these combinations of derivatives, though the accuracy is not always below the tolerance.

3 Complexity

Here we will discuss the complexity of the precorrected-FFT method as it scales with N_{pts} . In the previous section, we saw that the spreading parameters ($n_{\text{spread}}, n_{\text{proxy}}, n_{\text{shell}}$) scale at most polylogarithmically with ϵ . These spreading parameters are purely a function of ϵ and the kernel with no dependence on N_{pts} . Here we analyze the complexity for fixed settings of the user inputs ϵ, n_{nbr} , and $K(\cdot)$.

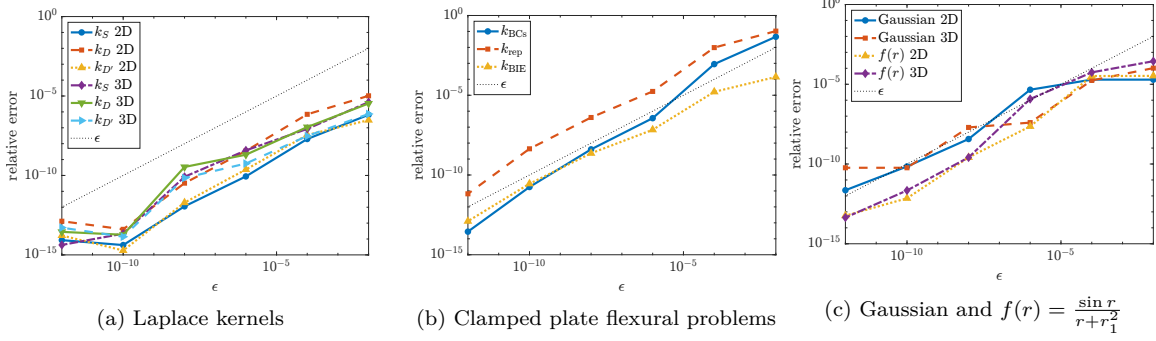


Figure 8: Relative ℓ_∞ error in the N -body sum versus the proxy tolerance ϵ for derivatives of the Green's function for Laplace's equation in 2D and 3D (8a), in a representation for the flexural boundary value problem with clamped plate boundary conditions and wavenumber $k = 10$ (8b), and the anisotropic examples (8c).

The complexity of executing algorithm 1 is $O(N_{\text{pts}} \log N_{\text{pts}})$. This can be seen because $\mathbf{A}$ has $O(N_{\text{pts}} n_{\text{nbr}})$ entries, the regular grid has $O(N_{\text{pts}} n_{\text{spread}}^d / n_{\text{nbr}})$ grid points, and each sparse spreading operator has $O(N_{\text{pts}} n_{\text{spread}}^d)$ nonzero entries. The dominant cost in algorithm 1 is therefore the FFT/IFFT pair, so the overall complexity scales as $O(N_{\text{pts}} \log N_{\text{pts}})$. The following theorem shows that the precomputation steps have complexity $O(N_{\text{pts}} \log N_{\text{pts}})$ as well.

Theorem 2 ([59]). *Suppose $n_{\text{nbr}}, n_{\text{spread}}, n_{\text{proxy}}$, and n_{shell} are fixed. For points spread uniformly in a compact subset of $\mathbb{R}^d$, the pre-computation cost of before we can run Algorithm 1 is*

$$O(N_{\text{pts}} n_{\text{spread}}^d / n_{\text{nbr}} \cdot \log(N_{\text{pts}} n_{\text{spread}}^d / n_{\text{nbr}}) + N_{\text{pts}} n_{\text{spread}}^d (n_{\text{proxy}} n_{\text{shell}} + n_{\text{spread}}^d + n_{\text{nbr}})).$$

Further, the cost a single evaluation of the N -body calculation using Algorithm 1 is

$$O(N_{\text{pts}} n_{\text{spread}}^d / n_{\text{nbr}} \cdot \log(N_{\text{pts}} n_{\text{spread}}^d / n_{\text{nbr}}) + N_{\text{pts}} n_{\text{nbr}}).$$

Proof. The precomputation involves four mains steps.

1. Determining n_{proxy} and n_{spread} through Algorithm 3, which is independent of N_{pts} , but can have non-trivial cost for generic kernels.
2. Computing the spreading operators $\mathbf{S}_y$ and $\mathbf{S}_x$ requires solving N_{pts} least-squares problems with the same $n_{\text{spread}}^d \times n_{\text{proxy}}$ matrix. Since N_{pts} is large, the dominant cost will be $O(N_{\text{pts}} \cdot n_{\text{proxy}} \cdot n_{\text{shell}} \cdot n_{\text{spread}}^d)$.
3. Computing the near-field corrections will cost $O(N_{\text{pts}} \cdot n_{\text{nbr}})$ to get the correct interactions and $O(N_{\text{pts}} \cdot n_{\text{spread}}^d \cdot (n_{\text{nbr}} + n_{\text{spread}}^d))$ to subtract the incorrect value given by the spreading operators.
4. Since the points are spread quasi-uniformly the number of spreading bins will be $O(N_{\text{pts}} / n_{\text{nbr}})$. Evaluating the Fourier transform of the kernel will therefore scale like

$$O(N_{\text{pts}} n_{\text{spread}}^d / n_{\text{nbr}} \cdot \log(N_{\text{pts}} n_{\text{spread}}^d / n_{\text{nbr}})),$$

as the number of grid points scale with the number of bins and the number of grid points per box, n_{spread}^d .

Adding up the costs of these steps gives the result.

In a single application of Algorithm 1, we apply two sparse matrices with $O(N_{\text{pts}} \cdot n_{\text{nbr}})$ entries, use two FFTs, and apply a diagonal matrix. Summing these costs gives the cost of a single matrix apply listed above. $\square$

The above theorem indicates that the algorithm is fastest when $n_{\text{spread}}^d \approx n_{\text{nbr}} n_{\text{shell}}$, though the optimal choice will depend on the computer architecture and memory movement considerations. It is easy to see that the memory required to store $\mathbf{S}_{\mathbf{y}}$ and $\mathbf{S}_{\mathbf{x}}$ are $N_{\text{pts}} n_{\text{spread}}^d$, the cost to store the corrections is $O(N_{\text{pts}} n_{\text{nbr}})$, and the cost to store the grid variables is $O(N_{\text{pts}} n_{\text{spread}}^d / n_{\text{nbr}})$. Adding these up gives the following theorem.

Theorem 3. *The memory cost to use Algorithm 1 is $O(N_{\text{pts}} n_{\text{spread}}^d + N_{\text{pts}} n_{\text{nbr}} + N_{\text{pts}} n_{\text{spread}}^d / n_{\text{nbr}})$.*

3.1 Scalability

In this section, we measure the performance of our method as N_{pts} grows, and compare the performance with other fast N -body summation methods.

Compute environment We perform our tests on a MacBook Pro, with an Apple M4 Max processor, 64GB of random-access memory, and MacOS version 26.5. For all experiments, we use the Apple Accelerate framework as the default BLAS implementation. Most experiments are driven from the MATLAB compute environment version R2025b. The 2D and 3D scaling experiments are conducted using a single thread, and the other experiments and numerical examples are conducted using multi-threading.

2D scaling First, we experiment in two spatial dimensions, when K is the single-layer kernel from the freespace Helmholtz equation:

$$K(\mathbf{x}, \mathbf{y}) = \frac{i}{4} H_0^{(1)}(k \|\mathbf{x} - \mathbf{y}\|),$$

where $H_0^{(1)}$ is the Hankel function of the first kind and order 0, and the wavenumber $k = 50$. We choose a “quasi-uniform” distribution of source and target points by successively adding smooth starfish-shaped curves to the domain $[-10, 10]^2$. Each starfish-shaped curve is discretized with about 1,200 points. We compute eq. (1.1) with tolerance $\epsilon = 10^{-5}$. As we increase the number of source and target points, we measure two notions of runtime. First, we measure the time required to compute the full sum eq. (1.1). Second, we measure the time required to apply a new charge vector $\boldsymbol{\sigma}$, after any necessary precomputations are finished. This notion of runtime is pertinent in an integral equation setting, when eq. (1.1) must be evaluated for dozens or hundreds of new charge vectors $\boldsymbol{\sigma}$ over the course of an iterative algorithm.

Figure 9 shows the result of this comparison experiment. We compare the performance of different methods: PCFFT, our pure-MATLAB implementation of our kernel-independent pre-corrected FFT method; FMM2D [27], a Fortran code called from MATLAB implementing the fast multipole method for the Helmholtz kernel; and IFMM [34] a pure-MATLAB implementation the interpolative fast multipole method [48], run with default flags and far-field proxy functions. Among these methods, FMM2D is the only one designed with *a priori* knowledge of the Helmholtz kernel; the other methods are “kernel independent”. FMM2D’s low total runtimes observed in fig. 9a can be attributed to the fact that this method does not need to perform expensive precomputation steps to compress the kernel, which are required by the other methods. Figure 9b shows that once the pre-computation is completed PCFFT is able to apply the charge vector an order of magnitude faster than other methods. Preliminary experiments were conducted with BBFMM2D [67], a C++ implementation of the black-box fast multipole method [25], but this method was unable to accurately compress the highly-oscillatory Helmholtz kernel for this problem.

3D scaling In three spatial dimensions, we run two similar comparisons. First, we investigate approximating sums with a three-dimensional single-layer Laplace kernel,

$$K(\mathbf{x}, \mathbf{y}) = \frac{1}{4\pi \|\mathbf{x} - \mathbf{y}\|},$$

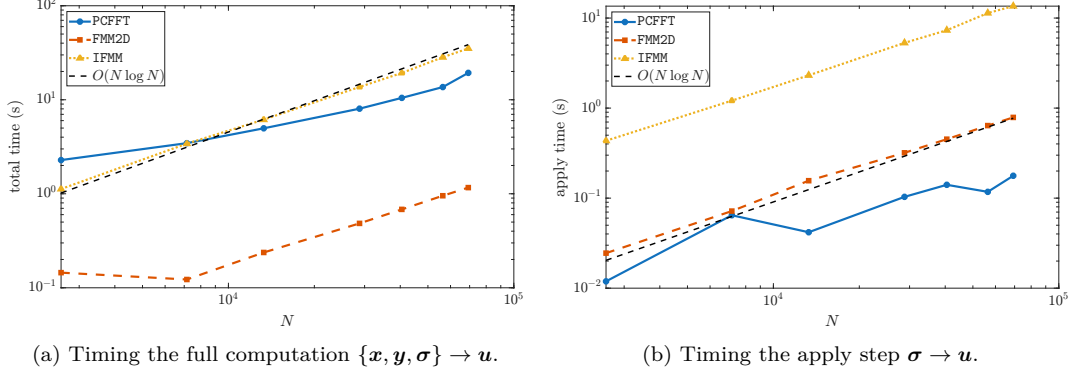


Figure 9: Comparing 2D fast summation algorithms computing an N -body sum with a Helmholtz kernel and error tolerance $\epsilon = 10^{-5}$. Figure 9a shows the total time to compute the N -body sum. The PCFFT method’s run time is dominated by the precomputation of the correction matrix. Figure 9b shows the time spent applying the charge vector σ . PCFFT is much faster in this stage because it relies on optimized FFT codes and sparse linear algebra.

with uniformly-sampled source and target point distributions over the box $[-1, 1]^3$. This experiment allows us to compare the scaling of our method with other highly-optimized methods designed for fast electrostatic calculations. Second, we investigate approximating sums with a three-dimensional single-layer Helmholtz kernel,

$$K(x, y) = \frac{e^{ik\|x-y\|}}{ik\|x-y\|},$$

with source and target point distributions generated by a triangulation of the surface shown in fig. 14. We start with a coarse triangulation and generate larger problem sizes by randomly choosing patches to subdivide. This experiment allows us to measure the scaling of our method in realistic integral equation settings.

We compare the performance of different methods: PCFFT and IFMM, which provide a single interface for both two- and three-dimensional problems; FMM3D [5], a Fortran code called from MATLAB implementing the fast multipole method for the Helmholtz kernel; ESP [32], a C++/Fortran code implementing a fast Ewald summation method [44]; DMK [32], a C++/Fortran code implementing the dual-space multilevel kernel method [38]; and BBFMM3D [66], a C++ implementation of the black-box fast multipole method.

In fig. 10, we show the results of the 3D scaling experiment with the Laplace kernel. We see the optimized codes designed specifically for the Laplace kernel are much more performant than the kernel-independent codes, PCFFT, IFMM, and BBFMM. Again, PCFFT has the fastest apply step of these algorithms. In fig. 11, we show the results of the 3D scaling experiment with the Helmholtz kernel. In this setting, the differences in apply step timings is more stark: after a precomputation step, PCFFT can apply the compressed kernel nearly an order of magnitude faster than the other methods. In an integral equation setting, where repeated applications of the kernel operator are required, the cost of the precomputation step will be quickly amortized. In the 3D Helmholtz scaling test, we performed preliminary experiments with BBFMM3D but found it could not accurately compress this oscillatory kernel.

Boundary integral equation solve In this experiment, we verify the claim that our method is able to improve the runtime of algorithms solving boundary integral equations. We solve a Helmholtz scattering problem with sound-hard scatterers. Let $\Omega \subset \mathbb{R}^2$ denote the union of scatterers, which have piecewise smooth boundary $\partial\Omega$. We use the same starfish-shaped scatterers as in the previous

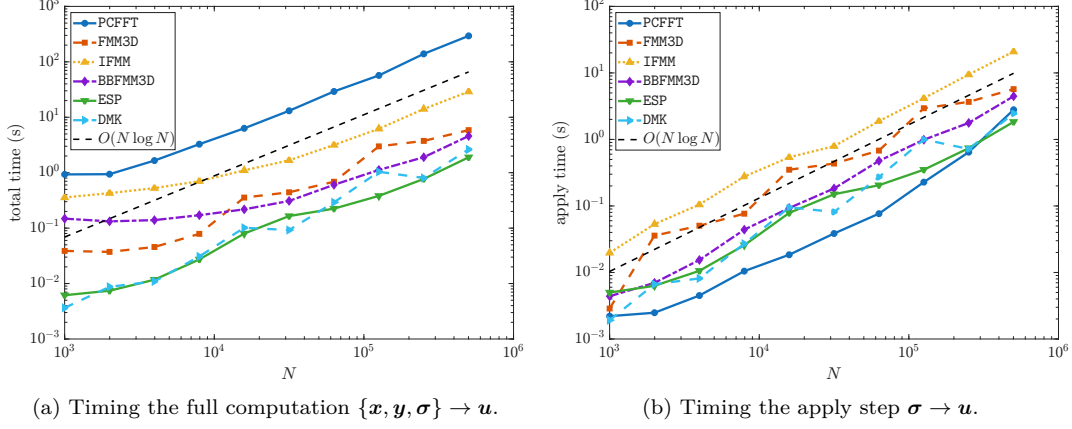


Figure 10: Scaling comparison for the 3D Laplace kernel. Figure 10a shows the total time to compute the N -body sum. FMM3D, DMK, and ESP are all highly-optimized codes specifically designed for the Laplace kernel, and are much faster than the others because they avoid expensive precomputation steps. Figure 10b shows time spent applying the charge vector σ . PCFFT is much faster than the others in this stage because algorithm 1 relies on optimized FFT codes and sparse linear algebra.

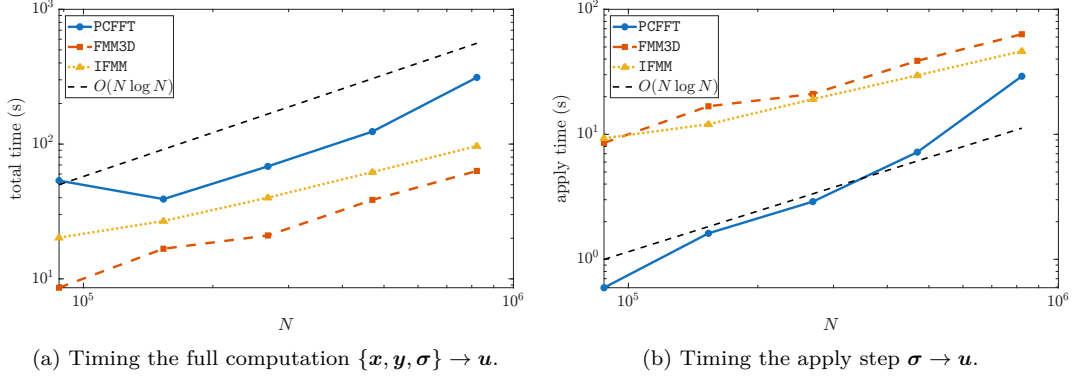


Figure 11: Scaling comparison for the 3D Helmholtz kernel. Comparing 3D fast summation algorithms computing an N -body sum with a Helmholtz kernel and error tolerance $\epsilon = 10^{-5}$. Figure 11a: total time to compute the N -body sum. The PCFFT method's time is dominated by the precomputation of the correction matrix. Figure 11b: time spent applying the charge vector σ . PCFFT is much faster in this stage because it relies on optimized FFT codes and sparse linear algebra.

2D scaling experiment. We compute a solution $u(x)$ of the following partial differential equation:

$$\begin{cases} \Delta u(x) + k^2 u(x) = 0, & x \in \mathbb{R}^2 \setminus \Omega, \\ \partial_n u(x) = g(x), & x \in \partial\Omega, \\ \sqrt{r} \left(\frac{\partial u}{\partial r} - iku \right) \rightarrow 0, & r = \|x\| \rightarrow \infty. \end{cases} \quad (3.1)$$

We solve this problem using the single layer representation

$$u(x) = \int_{\partial\Omega} G_k(x, y) \sigma(y) dy,$$

where $G_k(x, y) = \frac{i}{4} H_0^{(1)}(\|x - y\|)$ is the 2D Helmholtz Green's function. This representation solves eq. (3.1) provided σ solves the integral equation

$$-\frac{\sigma(x)}{2} + \int_{\partial\Omega} \partial_{n(x)} G_k(x, y) \sigma(y) dy = g(x) \quad (3.2)$$

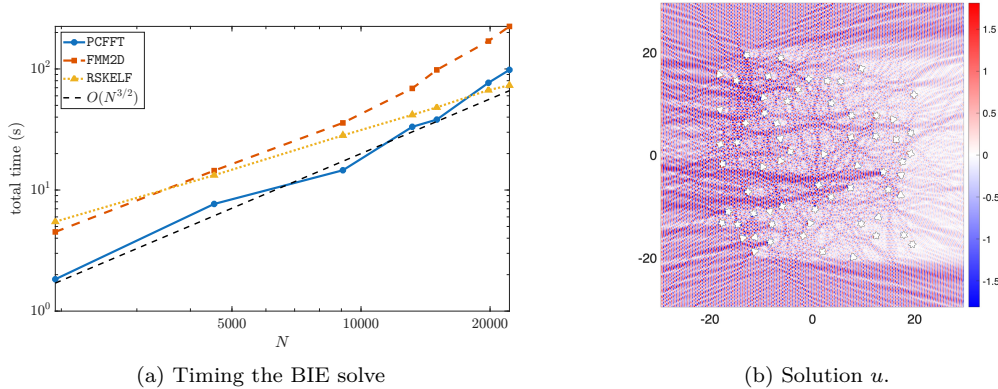


Figure 12: Comparing the solution times of a BIE formulation of a Helmholtz scattering problem. Figure 12a: total time to solve the boundary integral equation eq. (3.2). All methods use all 16 cores on a Macbook Pro M4. Figure 12b: visualizing the imaginary part of the solution at the largest problem size. Different problem sizes are generated by adding starfish scatterers to the domain.

on $\partial\Omega$. To generate a variety of problem sizes in this setting, we add more and more starfish-shaped scatterers to the domain. The addition of each scatterer both increases the number of discretization points, and increases the conditioning of the problem as multiple-scattering effects are increased.

In this experiment, we use both PCFFT and FMM2D to accelerate the application of the boundary integral operator eq. (3.2), and solve the resulting linear system iteratively using GMRES. We also measure the runtime of RSKELF, a recursive skeletonization based direct solver which computes a factored inverse of the operator in eq. (3.2). Figure 12 shows the results of this scaling experiment. At $N_{\text{pts}} = 13,168$, the iterative methods require 526 GMRES iterations and PCFFT is faster than the direct solver. At the two largest problem sizes, the direct solver computes the solution faster than both iterative methods, which require 804 and 916 GMRES iterations on these problem sizes, respectively.

4 Examples

We now demonstrate our algorithm on a few applications, focusing on integral equation settings where Fourier analysis based fast N -body algorithms are unavailable.

4.1 Eigendecomposition of the flexural clamped plate biharmonic problem

We use the kernels introduced in section 2.4 to find eigenvalues and eigenfunctions of the biharmonic operator with clamped plate boundary conditions. That is, for a given 2D domain Ω , we search for pairs k and u which solve eq. (2.8). We use three domains Ω , corresponding to the shape of the A-23 iceberg, which was pushed into the Weddell Sea and was suddenly broken apart by storms between July and September, 2025. We construct Ω_1, Ω_2 , and Ω_3 by tracing high-resolution satellite imagery of the iceberg in July, August, and September of 2025, respectively [18, 19, 20]. This experiment gives us insight to how the eigenfunctions and eigenvalues change with the changing iceberg shape.

A popular class of methods for finding these eigenvalues involve looking for poles of the function

$$S(k) = 1/(f^* F_k^{-1} g),$$

where F_k is a discrete system matrix of the system at wavenumber k . Methods of this form include contour integration methods, such as those developed in [6, 15], and methods that directly look for roots of $S(k)$ [12, 21] and related functions (see [9, 74]). In this example, we use the latter method, building a Chebyshev expansion of $S(k)$. For F_k is the boundary integral operator corresponding to the clamped plate biharmonic problem in the left-hand side of eq. (2.8), with wavenumber k . The

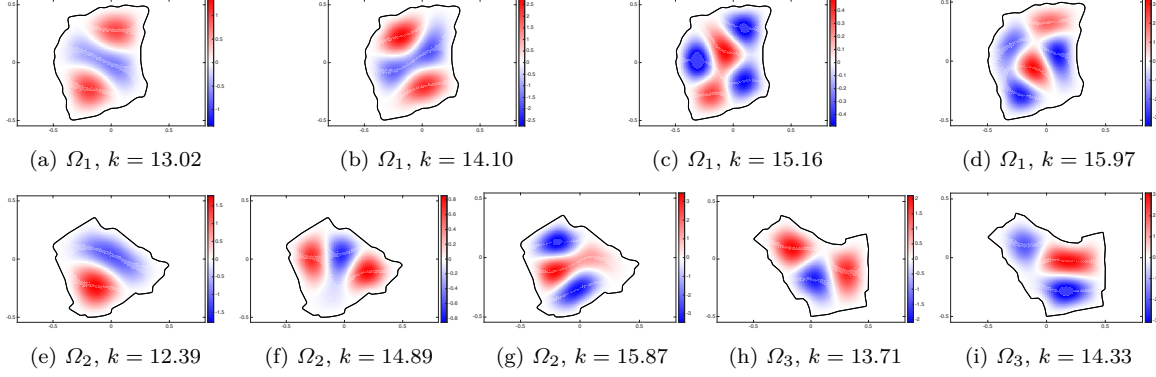


Figure 13: Eigenfunctions of the clamped plate biharmonic operator on three different iceberg geometries. Each subfigure shows an individual eigenfunction. As solutions of eq. (2.8) with zero boundary data, both u and $\partial_{\mathbf{n}}u$ must vanish on the boundary, and $\Delta^2 u$ must hold on the interior. Statistics of the computation required to find these eigenfunctions are described in table 2.

operator F_k will be the map from auxiliary densities to boundary data $g = (g_1, g_2)$. Thus, F_k^{-1} maps from boundary data to auxiliary densities defining the solution. If k is an eigenvalue of the operator, the resonance will cause a pole in $S(k)$.

To find the poles of $S(k)$, we set a search interval $k \in [12, 16]$, discretize the interval with 80 Chebyshev points, and evaluate $S(k)$ at these points. At each value of k , we compute $F_k^{-1}g$ by solving the resulting boundary integral equation iteratively using GMRES, where the matrix-vector products are accelerated using PCFFT. Given samples of $S(k)$ at Chebyshev points, we fit a Chebyshev series for $1/S(k)$ and find the zeros of this function. The roots of $1/S(k)$ are poles of $S(k)$ and, equivalently, resonant wavenumbers of eq. (2.8). Once we have found an eigenvalue k , we find an auxiliary density in the nullspace of F_k^{-1} . Because we are at a resonance, we expect to incur a large iteration complexity to solve this problem, so we use the RSKELF direct solver, implemented in FLAM [34].

We conduct the search for eigenvalues and eigenfunctions using three iceberg geometry shapes. Table 2 contains statistics of this experiment. Figure 13 shows the recovered eigenfunctions on the three different geometries. Even with a relatively un-optimized algorithm for finding roots of $S(k)$, we are able to solve these nonlinear eigenvalue problems quite efficiently.

Geometry	Eigenvalues	Search interval	N_{pts}	$n_{\text{iter.}}$	$T_{\text{res.}}$	$T_{\text{dir.}}$
Ω_1	$\{13.02, 14.10, 15.16, 15.97\}$	$[12, 16]$	4,928	48.7	11.9	32.9
Ω_2	$\{12.39, 14.89, 15.87\}$	$[12, 16]$	7,104	56.5	17.4	92.6
Ω_3	$\{13.71, 14.33\}$	$[12, 16]$	3,280	59.6	9.4	27.5

Table 2: Statistics of the eigendecomposition experiment. For each geometry, the eigenvalues found by our search are shown in the second column. The third column shows the interval over which we searched for eigenvalues. The remaining columns are N_{pts} : number of discretization points along the boundary, $n_{\text{iter.}}$: average number of GMRES iterations needed to solve each clamped plate boundary value problem, $T_{\text{res.}}$: average time (seconds) to solve each boundary integral equations when sampling the resolvent $S(k)$, and $T_{\text{dir.}}$: average time (seconds) required by the direct solver to compute each eigenfunction. Visualizations of the eigenfunctions are provided in fig. 13.

4.2 Surface Helmholtz

As our next example, we solve the Helmholtz-Beltrami problem on the “chainlink torus” surface Γ depicted in fig. 14:

$$(\Delta_{\Gamma} + k(\mathbf{x})^2)u(\mathbf{x}) = f(\mathbf{x}) \quad \text{on } \Gamma, \quad (4.1)$$

where Δ_Γ is the Laplace-Beltrami operator. We solve this equation using a modification of the parametrix method introduced in [30]. In short, we suppose that

$$u(\mathbf{x}) = \int_\Gamma \frac{1}{2\pi} \log(\mathbf{x} - \mathbf{y}) \sigma(\mathbf{y}) d\mathbf{y},$$

for some unknown density σ . The calculations in [30] show that this u will solve the Helmholtz-Beltrami equation provided

$$\sigma(\mathbf{x}) + \int_\Gamma R(\mathbf{x}, \mathbf{y}) \sigma(\mathbf{y}) d\mathbf{y} = f(\mathbf{x}) \quad (4.2)$$

for all $\mathbf{x} \in \Gamma$, where the remainder kernel is

$$R(\mathbf{x}, \mathbf{y}) = \frac{1}{2\pi} \left[\left(\frac{\mathbf{n}(\mathbf{x}) \cdot (\mathbf{x} - \mathbf{y})}{\|\mathbf{x} - \mathbf{y}\|^2} \right)^2 - 2H(\mathbf{x}) \frac{\mathbf{n}(\mathbf{x}) \cdot (\mathbf{x} - \mathbf{y})}{\|\mathbf{x} - \mathbf{y}\|^2} + k^2(\mathbf{x}) \log(\mathbf{x} - \mathbf{y}) \right],$$

and $H(\mathbf{x})$ is the mean curvature of Γ . In [30], the authors solve the same problem using a Hankel function with a characteristic frequency instead of the logarithm. We choose this representation as it is real-valued and requires a smaller n_{spread} . We discretize the integral equation using the collocation scheme `fmm3dbie` [2]. In short, this package splits the surface into high order Koornwinder patches discretized with Vioreanu-Rokhlin nodes [65]. For every patch, the targets are split into a near-field where special quadrature is used, and a far-field where a consistent oversampled smooth quadrature rule is used. To apply the operators with our algorithm, we simply use the oversampled nodes as the sources and the original discretization as the target. We then add a sparse matrix of corrections to apply the near-field quadratures.

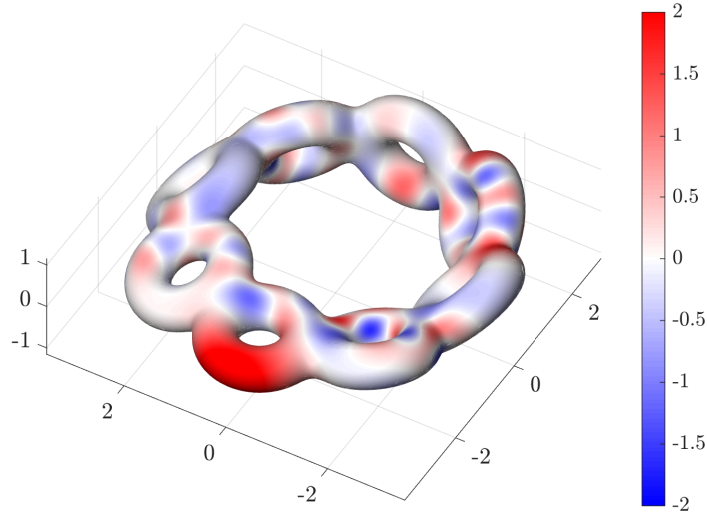


Figure 14: An example solution of the Helmholtz-Beltrami equation eq. (4.1). The “chainlink torus” surface Γ is a smooth surface triangulated with 7,797 patches.

For our experiment, we used $k(\mathbf{x}) = 4 - 4\cos(2\mathbf{x}_2)$; the source $f(\mathbf{x})$ and wavenumber $k(\mathbf{x})$ are shown in fig. 15. An analytic solution test indicates that the solution is accurate to 4 digits. The surface was discretized with 350,865 points belonging to the 7,797 8th order patches. It took 26 seconds to generate the quadratures. For the far-field quadrature `fmm3dbie` oversamples the surface to 14th order, resulting 674,924 nodes. The precorrected-FFT precomputation with these sources and targets took 181 seconds for the remainder and 180 seconds for the parametrix. Each application of the system matrix took 1.50 seconds. The solve required 365 GMRES iterations which took 899 seconds. Overall the entire solver used 48 GB of memory.

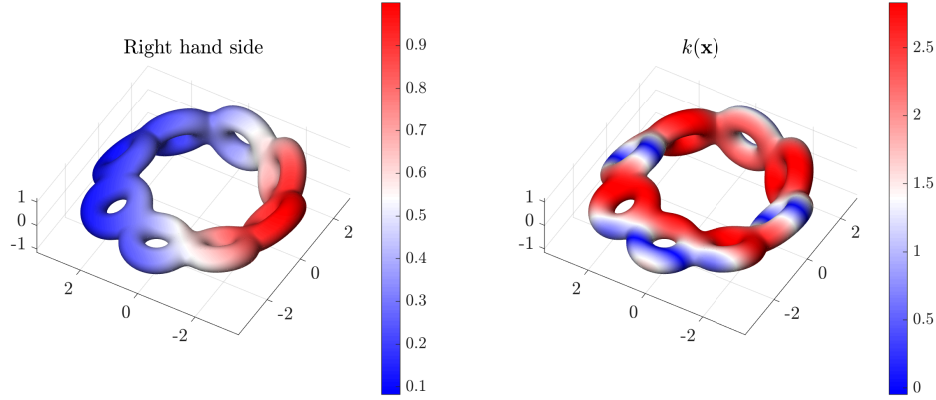


Figure 15: The example right hand side source function $f(\mathbf{x})$ (left) and wavenumber $k(\mathbf{x})$ (right) used to generate the Helmholtz-Beltrami solution in fig. 14.

4.3 Lens optimization

As a final example, we consider a simple lens design problem inspired by [64]. In that work, 3D printing is used to build a lens out of a dielectric material with a variable density. In this example we consider a simpler 2D lens and model the field using a scalar Helmholtz equation. We model this as a compact region Ω with a local wavenumber $k(\mathbf{x}) = \sqrt{k_1^2 + q(\mathbf{x})}$ that is customizable between some maximum and minimum allowable values. Outside of the lens, we suppose that the field has wavenumber k_0 .

Our goal will be to choose the wavenumber so that an incident planewave u^{in} is focused to yield the largest possible intensity at the focal point $\mathbf{x}_0$. The total field can be split $u = u^{\text{in}} + u^{\text{sc}}$, where the scattered field satisfies the Sommerfeld radiation conditions and the equations

$$\begin{cases} \Delta u^{\text{sc}}(\mathbf{x}) + k_0^2 u^{\text{sc}}(\mathbf{x}) = 0, & \mathbf{x} \text{ in } \mathbb{R}^2 \setminus \Omega, \\ \Delta u^{\text{sc}}(\mathbf{x}) + (q(\mathbf{x}) + k_1^2) u^{\text{sc}}(\mathbf{x}) = -(q(\mathbf{x}) + k_1^2 - k_0^2) u^{\text{in}}(\mathbf{x}), & \mathbf{x} \text{ in } \Omega, \\ [[u^{\text{sc}}]](\mathbf{x}) = 0, & \mathbf{x} \text{ on } \partial\Omega, \\ [[\partial_{\mathbf{n}} u^{\text{sc}}]](\mathbf{x}) = 0, & \mathbf{x} \text{ on } \partial\Omega, \end{cases} \quad (4.3)$$

where $[[u^{\text{sc}}]]$ and $[[\partial_{\mathbf{n}} u^{\text{sc}}]]$ are the jump in the scattered field and its normal derivative across $\partial\Omega$ respectively. This is equivalent to looking for the weak (i.e. H_{loc}^2) solution of the equation

$$(\Delta + k^2(\mathbf{x})) u^{\text{sc}}(\mathbf{x}) = -(k(\mathbf{x})^2 - k_0^2) u^{\text{in}}(\mathbf{x})$$

in all of $\mathbb{R}^2$ that satisfies the Sommerfeld radiation conditions. We will work to find a $q(\mathbf{x})$ that optimizes the intensity at the focal point $\mathbf{x}_0$:

$$\mathcal{F}[q] := |u^{\text{inc}}(\mathbf{x}_0) + u^{\text{sc}}(\mathbf{x}_0)|^2$$

where $\mathbf{x}_0$ is some point outside of Ω . For this example, we choose Ω to be the interior of the squircle satisfying

$$x_1^2 + x_2^2 - 0.9^2 x_1^2 x_2^2 = 1$$

which is shown in fig. 16. The focal point is chosen as $\mathbf{x}_0 = (4, 0)$. We represent q using a global 6th order tensor product Chebyshev basis, supported on the square that just contains the region Ω . To improve the conditioning, we use a QR factorization (weighted by the square root of the quadrature weights in Ω) to orthogonalize the basis on Ω .

To solve this optimization problem efficiently, we must efficiently compute the Fréchet derivative of the scattered field with respect to perturbations of q . If $D[u^{\text{sc}}]p$ is the Fréchet derivative of u^{sc} with respect to a perturbation $q(\mathbf{x}) \rightarrow q(\mathbf{x}) + hp(\mathbf{x})$, it will be the outgoing weak solution of

$$(\Delta + k^2(\mathbf{x}))(D[u^{\text{sc}}]p)(\mathbf{x}) = -p(\mathbf{x})(u^{\text{in}}(\mathbf{x}) + u^{\text{sc}}(\mathbf{x}))$$

(see [7]). The strong form of this equation is identical to eq. (4.3) with a new volumetric source in Ω . To simplify the computation, we introduce w , the weak solution of

$$(\Delta + k^2(\mathbf{x}))w(\mathbf{x}) = \delta(\mathbf{x} - \mathbf{x}_0)$$

(i.e. the field generated by a point source at the focal point $\mathbf{x}_0$). A straightforward integration by parts argument [7] then shows that

$$\begin{aligned} (D[u^{\text{sc}}]p)(\mathbf{x}_0) &= v(\mathbf{x}_0) = \int_{\mathbb{R}^2} (D[u^{\text{sc}}]p)(\mathbf{x})\delta(\mathbf{x} - \mathbf{x}_0) d\mathbf{x} = \int_{\mathbb{R}^2} (D[u^{\text{sc}}]p)(\mathbf{x})(\Delta + k^2(\mathbf{x}))w(\mathbf{x}) d\mathbf{x} \\ &= \int_{\mathbb{R}^2} (\Delta + k^2(\mathbf{x}))(D[u^{\text{sc}}]p)(\mathbf{x})w(\mathbf{x}) d\mathbf{x} = - \int_{\Omega} p(\mathbf{x})(u^{\text{in}}(\mathbf{x}) + u^{\text{sc}}(\mathbf{x}))w(\mathbf{x}) d\mathbf{x}. \end{aligned}$$

The advantage of this approach is that we can compute the partial derivative of the field at $\mathbf{x}_0$ with respect to all of the Chebyshev coefficients defining q with two solves, one to find u^{sc} and one to find w , followed by one inner product per basis function. After some applications of the chain rule, this approach allows us to compute the gradient of the objective function with respect to the coefficients defining q . This in turn allows us to optimize the objective function $\mathcal{F}[q]$ using BFGS [10, 24, 61, 28] and a backtracking line search with the initial condition $q(\mathbf{x}) \equiv 0$.

In order to solve eq. (4.3) at each step, we represent the scattered field using a volume potential σ supported in Ω and layer potentials η and ρ supported on $\partial\Omega$:

$$u^{\text{sc}}(\mathbf{x}) = \int_{\partial\Omega} G_{k_0}(\mathbf{x} - \mathbf{y})\eta(\mathbf{y}) d\mathbf{y} + \int_{\partial\Omega} \partial_{\mathbf{n}(\mathbf{y})} G_{k_0}(\mathbf{x} - \mathbf{y})\rho(\mathbf{y}) d\mathbf{y} \quad \mathbf{x} \in \mathbb{R}^2 \setminus \Omega,$$

where $G_{k_0}(\mathbf{x}) = \frac{i}{4} H_0^{(1)}(k_0 \|\mathbf{x}\|)$ is the Green's function for the exterior problem, and

$$u^{\text{sc}}(\mathbf{x}) = \int_{\Omega} G_{k_1}(\mathbf{x} - \mathbf{y})\sigma(\mathbf{y}) d\mathbf{y} + \int_{\partial\Omega} G_{k_1}(\mathbf{x} - \mathbf{y})\eta(\mathbf{y}) d\mathbf{y} + \int_{\partial\Omega} \partial_{\mathbf{n}(\mathbf{y})} G_{k_1}(\mathbf{x} - \mathbf{y})\rho(\mathbf{y}) d\mathbf{y} \quad \mathbf{x} \in \Omega,$$

where G_{k_1} is the Green's function for the prototypical interior wavenumber. Plugging this representation into eq. (4.3) yields a Fredholm second kind equation that can be solved for σ, η, ρ . We then discretize Ω using `fmm3dbie` and $\partial\Omega$ using `chunkie`. We apply the operator quickly using two precorrected-FFTs, one at wavenumber k_1 , and a small one involving only boundary nodes at wavenumber k_0 . Since k_0 and k_1 are fixed, the quadrature corrections, spreading operators, and near-field corrections can be pre-computed and reused at each optimization step.

It is worth remarking that eq. (4.3) can also be solved by using the much simpler representation $u^{\text{sc}}(\mathbf{x}) = \int_{\Omega} G_{k_0}(\mathbf{x} - \mathbf{y})\sigma(\mathbf{y}) d\mathbf{y}$ for all $\mathbf{x} \in \mathbb{R}^2$, but we have found that our representation requires substantially fewer GMRES iterations for small q (33 versus 66 iterations for $(k_0, k_1) = (4, 6)$ and $q(\mathbf{x}) \equiv -1$ in Ω), likely because our representation uses a more representative wavenumber k_1 inside Ω . We have found that the decreased number of iterations outweighs the increased cost associated with adding discretization points on the boundary and the second small precorrected-FFT per GMRES iteration.

With our integral equation solver in hand, we run our BFGS optimizer to find an improved lens with $k_0 = 4$, $k_1 = 6$, and q restricted to live in $[-10, 10]$. We stopped the optimization after 132 iterations, when the norm of the gradient first dropped below 10^{-3} . The resulting field intensity and wavenumber are shown in fig. 16. The intensity at $\mathbf{x}_0 = (4, 0)$ is $|u^{\text{in}}(\mathbf{x}_0) + u^{\text{sc}}(\mathbf{x}_0)|^2 = 5.1$. For comparison, the best constant wavenumber $k \in [5, 7] = k_1 + [-1, 1]$ was 5, which generated an intensity of 4.79. The discretization has 20,138 unknowns and used an average of 60.2 GMRES iterations per solve with a tolerance of 10^{-9} . The total time to solve the optimization problem was 2,041 seconds.

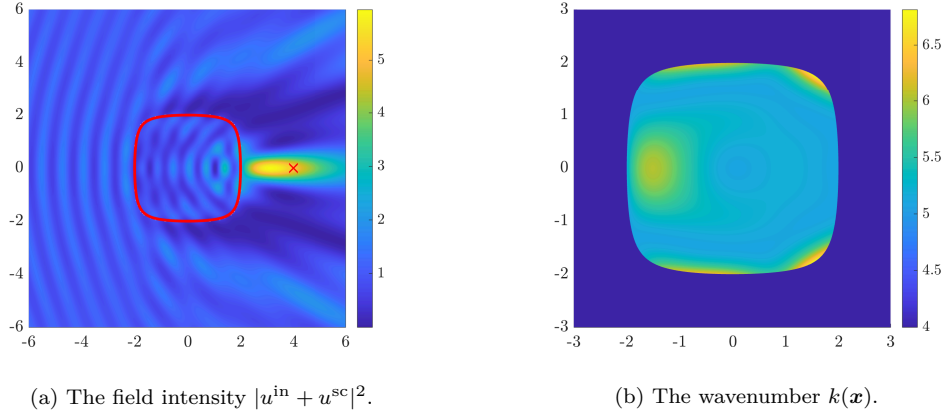


Figure 16: The field intensity $|u^{\text{in}} + u^{\text{sc}}|^2$ (16a) and wavenumber $k(\mathbf{x})$ (16b) for the lens chosen to maximize the intensity at $\mathbf{x}_0 = (4, 0)$ when u^{in} is a plane wave coming from the left.

5 Conclusion

In this work, we present a modification of the precorrected-FFT method that allows it to be used for a generic kernels. The method uses a series of proxy shells to build spreading operators that map point charges to equivalent charges on a grid. We also present and test an algorithm for systematically choosing the proxy points and spreading grids without the need for analytical knowledge of the kernel. The method is therefore flexible and can trivially be applied to new problem. To demonstrate the effectiveness of the new flexible version of the precorrected-FFT, we use it to solve accelerate integral equation approach for a variety of PDE problems.

So far, this work has been limited to scalar-valued kernels. While the presented algorithm should extend directly to vectored-valued problems, such as those considered in [63], efficient implementations remains an open question. In order to offset the extra cost for these problems, we plan to explore GPU implementations of the algorithm.

Acknowledgments

The authors would like to thank Jeremy Hoskins, Travis Askham, and Manas Rachh for many helpful discussions on the algorithm and for suggesting that we complete this work. They would also like to thank Shidong Jiang for his suggestions of good benchmark problems and Alex Barnett and Vicente Gomez Herrera for help conversations. T. Goodwill was supported in part by an AMS-Simons Travel Grant. The Flatiron Institute is a division of the Simons Foundation.

References

- [1] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, and M. O’Neil. Fast direct methods for Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):252–265, Feb. 2016. ISSN 1939-3539. doi: 10.1109/TPAMI.2015.2448083.
- [2] T. Askham, L. G. J. Hoskins, L. Lu, M. O. M. Rachh, F. Vico, and V. Rokhlin. fmm3dbie: FMM-accelerated boundary integral equation solvers. URL <https://github.com/fastalgorithms/fmm3dbie>.
- [3] T. Askham, T. Goodwill, J. G. Hoskins, P. Nekrasov, and M. Rachh. Surface layers and linearized water waves: a boundary integral equation framework. (arXiv:2510.22748), Oct. 2025. doi: 10.48550/arXiv.2510.22748. URL <http://arxiv.org/abs/2510.22748>. arXiv:2510.22748 [math].

- [4] T. Askham, J. G. Hoskins, P. Nekrasov, and M. Rachh. Integral equations for flexural-gravity waves: analysis and numerical methods, 2025. URL <https://arxiv.org/abs/2501.00887>.
- [5] T. Askham, Z. Gimbutas, L. Greengard, L. Lu, J. Magland, D. Malhotra, M. O’Neil, M. Rachh, V. Rokhlin, and F. Vico. FMM3D, Feb. 2026. URL <https://github.com/flatironinstitute/fmm3d>.
- [6] W.-J. Beyn. An integral method for solving nonlinear eigenvalue problems. *Linear Algebra and its Applications*, 436(10):3839–3863, 2012.
- [7] C. Borges, A. Gillman, and L. Greengard. High resolution inverse scattering in two dimensions using recursive linearization. *SIAM Journal on Imaging Sciences*, 10(2):641–664, 2017.
- [8] E. Boström, A.-K. Tornberg, and L. af Klinteberg. Fast Ewald summation using prolate spheroidal wave functions, Feb. 2026.
- [9] J. P. Boyd. Computing zeros on a real interval through chebyshev expansion and polynomial rootfinding. *SIAM Journal on Numerical Analysis*, 40(5):1666–1682, 2002.
- [10] C. G. Broyden. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970.
- [11] O. P. Bruno and L. A. Kunyansky. A fast, high-order algorithm for the solution of surface scattering problems: basic implementation, tests, and applications. *Journal of Computational Physics*, 169(1):80–110, 2001.
- [12] O. P. Bruno, M. Santana, and L. N. Trefethen. Evaluation of resonances: Adaptivity and AAA rational approximation of randomly scalarized boundary integral resolvents. *SIAM Journal on Scientific Computing*, 48(3):A1260–A1283, June 2026. ISSN 1064-8275, 1095-7197. doi: 10.1137/24M1690680.
- [13] J. Carrier, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm for particle simulations. *SIAM Journal on Scientific and Statistical Computing*, 9(4):669–686, July 1988. ISSN 0196-5204, 2168-3417. doi: 10.1137/0909044.
- [14] J. Chen, L. Wang, and M. Anitescu. A fast summation tree code for Matérn kernel. *SIAM Journal on Scientific Computing*, 36(1):A289–A309, Jan. 2014. ISSN 1064-8275, 1095-7197. doi: 10.1137/120903002.
- [15] M. J. Colbrook and A. Townsend. Avoiding discretization issues for nonlinear eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 46(1):648–675, 2025.
- [16] T. Darden, D. York, and L. Pedersen. Particle mesh Ewald: An $N \cdot \log(N)$ method for Ewald sums in large systems. *The Journal of Chemical Physics*, 98(12):10089–10092, June 1993. ISSN 0021-9606, 1089-7690. doi: 10.1063/1.464397.
- [17] DLMF. *NIST Digital Library of Mathematical Functions*. <https://dlmf.nist.gov/>, Release 1.2.7 of 2026-06-15. URL <https://dlmf.nist.gov/>. F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain, eds.
- [18] J. Doris. Iceberg A-23DE press release, 2025. URL <https://usicecenter.gov/PressRelease/IcebergA23DE>.
- [19] J. Doris. Iceberg A-23GH press release, Aug. 2025. URL <https://usicecenter.gov/PressRelease/IcebergA23GH>.
- [20] J. Doris. Iceberg A-23I press release, 2025. URL <https://usicecenter.gov/PressRelease/IcebergA23I>.

- [21] C. Effenberger and D. Kressner. Chebyshev interpolation for nonlinear eigenvalue problems. *BIT Numerical Mathematics*, 52(4):933–951, 2012.
- [22] P. P. Ewald. Die berechnung optischer und elektrostatischer gitterpotentiale. *Annalen der Physik*, 369(3):253–287, Jan. 1921. ISSN 0003-3804, 1521-3889. doi: 10.1002/andp.19213690304.
- [23] P. Farkas. Mathematical foundations for fast methods for the biharmonic equation. *Ann Arbor, MI*, 1989.
- [24] R. Fletcher. A new approach to variable metric algorithms. *The computer journal*, 13(3): 317–322, 1970.
- [25] W. Fong and E. Darve. The black-box fast multipole method. *Journal of Computational Physics*, 228(23):8712–8725, Dec. 2009. ISSN 0021-9991. doi: 10.1016/j.jcp.2009.08.031.
- [26] M. Frigo and S. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, Feb. 2005. ISSN 0018-9219. doi: 10.1109/JPROC.2004.840301.
- [27] Z. Gimbutas, L. Greengard, M. O’Neil, M. Rachh, and V. Rokhlin. FMM2D, Oct. 2023. URL <https://github.com/flatironinstitute/fmm2d>.
- [28] D. Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26, 1970.
- [29] T. Goodwill and M. O’Neil. A parametrix method for elliptic surface PDEs. *Pure and Applied Analysis*, 7(1):171–217, Mar. 2025. ISSN 2578-5885, 2578-5893. doi: 10.2140/paa.2025.7.171.
- [30] T. Goodwill, J. Hoskins, Z. Gimbutas, and B. Wu. A parametrix for the surface Stokes equation, Feb. 2026.
- [31] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, Dec. 1987. ISSN 0021-9991. doi: 10.1016/0021-9991(87)90140-9.
- [32] L. Greengard, S. Jiang, and R. Blackwell. DMK. URL <https://github.com/flatironinstitute/dmk>.
- [33] V. G. Herrera and S. Weady. Instability of microbial droplets growing on viscous substrates. *Journal of Fluid Mechanics*, 1037:A45, 2026.
- [34] K. L. Ho. FLAM: Fast linear algebra in MATLAB - algorithms for hierarchical matrices. *Journal of Open Source Software*, 5(51):1906, July 2020. ISSN 2475-9066. doi: 10.21105/joss.01906.
- [35] K. L. Ho and L. Greengard. A fast direct solver for structured linear systems by recursive skeletonization. *SIAM Journal on Scientific Computing*, 34(5):A2507–A2532, 2012.
- [36] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles*. CRC Press, Boca Raton, Mar. 1988. ISBN 978-0-367-80693-4. doi: 10.1201/9780367806934.
- [37] H. Hu, D. Blaauw, V. Zolotov, K. Gala, Min Zhao, R. Panda, and S. Sapatnekar. Fast on-chip inductance simulation using a precorrected-FFT method. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(1):49–66, Jan. 2003. ISSN 0278-0070. doi: 10.1109/TCAD.2002.805719.
- [38] S. Jiang and L. Greengard. A dual-space multilevel kernel-splitting framework for discrete and continuous convolution. *Communications on Pure and Applied Mathematics*, 78(5):1086–1143, 2025. ISSN 1097-0312. doi: 10.1002/cpa.22240.
- [39] S.-c. Jiang, B. Teng, Y. Gou, and D.-z. Ning. A precorrected-FFT higher-order boundary element method for wave-body problems. *Engineering Analysis with Boundary Elements*, 36(3):404–415, Mar. 2012. ISSN 0955-7997. doi: 10.1016/j.enganabound.2011.08.002.

- [40] L. a. Klinteberg, L. Greengard, S. Jiang, and A.-K. Tornberg. Fast summation of Stokes potentials using a new kernel-splitting in the DMK framework. *Journal of Computational Physics*, page 114892, 2026.
- [41] R. Kress. *Linear Integral Equations*, volume 82 of *Applied Mathematical Sciences*. Springer New York, New York, NY, 1999. ISBN 978-1-4612-6817-8 978-1-4612-0559-3. doi: 10.1007/978-1-4612-0559-3.
- [42] C. Li and J. Lian. Development and application of a pre-corrected fast Fourier transform accelerated multi-layer boundary element method for the simulation of shallow water acoustic propagation. *Applied Sciences*, 10(7):2393, Jan. 2020. ISSN 2076-3417. doi: 10.3390/app10072393.
- [43] W. Li and F. Wang. Precorrected-FFT Accelerated Singular Boundary Method for High-Frequency Acoustic Radiation and Scattering. *Mathematics*, 10(2):238, Jan. 2022. ISSN 2227-7390. doi: 10.3390/math10020238.
- [44] J. Liang, L. Lu, A. Barnett, L. Greengard, and S. Jiang. Accelerating fast Ewald summation with prolates for molecular dynamics simulations, May 2025.
- [45] A. E. Lindsay, B. Quaife, and L. Wendelberger. A boundary integral equation method for mode elimination and vibration confinement in thin plates with clamped points. *Advances in Computational Mathematics*, 44(4):1249–1273, 2018.
- [46] D. Malhotra and G. Biros. Pvfmm: A parallel kernel independent fmm for particle and volume potentials. *Communications in Computational Physics*, 18(3):808–830, 2015.
- [47] P.-G. Martinsson and V. Rokhlin. A fast direct solver for boundary integral equations in two dimensions. *Journal of Computational Physics*, 205(1):1–23, 2005.
- [48] P. G. Martinsson and V. Rokhlin. An accelerated kernel-independent fast multipole method in one dimension. *SIAM Journal on Scientific Computing*, 29(3):1160–1178, Jan. 2007. ISSN 1064-8275, 1095-7197. doi: 10.1137/060662253.
- [49] V. Minden, A. Damle, K. L. Ho, and L. Ying. Fast spatial Gaussian process maximum likelihood estimation via skeletonization factorizations. *Multiscale Modeling & Simulation*, Nov. 2017. doi: 10.1137/17M1116477.
- [50] V. Minden, K. L. Ho, A. Damle, and L. Ying. A recursive skeletonization factorization based on strong admissibility. *Multiscale Modeling & Simulation*, 15(2):768–796, 2017.
- [51] J.-C. Nédélec. *Acoustic and Electromagnetic Equations*, volume 144 of *Applied Mathematical Sciences*. Springer New York, New York, NY, 2001. ISBN 978-1-4419-2889-4 978-1-4757-4393-7. doi: 10.1007/978-1-4757-4393-7.
- [52] P. Nekrasov, Z. Su, T. Askham, and J. G. Hoskins. Boundary integral formulations for flexural wave scattering in thin plates. *Journal of Computational Physics*, 542:114355, 2025. ISSN 0021-9991. doi: <https://doi.org/10.1016/j.jcp.2025.114355>.
- [53] X. Nie, L.-W. Li, N. Yuan, and T. S. Yeo. Fast analysis of scattering by arbitrarily shaped three-dimensional objects using the precorrected-FFT method. *Microwave and Optical Technology Letters*, 34(6):438–442, 2002.
- [54] X.-C. Nie, L.-W. Li, and N. Yuan. Precorrected-FFT algorithm for solving combined field integral equations in electromagnetic scattering. *Journal of Electromagnetic Waves and Applications*, 16(8):1171–1187, Jan. 2002. ISSN 0920-5071. doi: 10.1163/156939302X00697.
- [55] V. Okhmatovski, M. Yuan, I. Jeffrey, and R. Phelps. A three-dimensional precorrected FFT algorithm for fast method of moments solutions of the mixed-potential integral equation in layered media. *IEEE Transactions on Microwave Theory and Techniques*, 57(12):3505–3517, Dec. 2009. ISSN 0018-9480, 1557-9670. doi: 10.1109/TMTT.2009.2033882.

- [56] M. O’Neil. Second-kind integral equations for the Laplace-Beltrami problem on surfaces in three dimensions. *Advances in Computational Mathematics*, 44(5):1385–1409, Oct. 2018. ISSN 1572-9044. doi: 10.1007/s10444-018-9587-7.
- [57] S. Peng and C.-F. Wang. Precorrected-FFT method on graphics processing units. *IEEE Transactions on Antennas and Propagation*, 61(4):2099–2107, Apr. 2013. ISSN 0018-926X, 1558-2221. doi: 10.1109/TAP.2012.2232892.
- [58] J. R. Phillips. Error and complexity analysis for a collocation-grid-projection plus precorrected-FFT algorithm for solving potential integral equations with Laplace or Helmholtz kernels. Sept. 1996.
- [59] J. R. Phillips and J. K. White. A precorrected-FFT method for electrostatic analysis of complicated 3-D structures. *IEEE Transactions on computer-aided design of integrated circuits and systems*, 16(10):1059–1072, 1997.
- [60] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Nov. 2005. ISBN 978-0-262-25683-4. doi: 10.7551/mitpress/3206.001.0001.
- [61] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970.
- [62] M. L. Stein. *Interpolation of Spatial Data*. Springer Series in Statistics. Springer New York, New York, NY, 1999. ISBN 978-1-4612-7166-6 978-1-4612-1494-6. doi: 10.1007/978-1-4612-1494-6.
- [63] S. Tissari and J. Rahola. A precorrected-FFT method to accelerate the solution of the forward problem in magnetoencephalography. *Physics in Medicine & Biology*, 48(4):523, Feb. 2003. ISSN 0031-9155. doi: 10.1088/0031-9155/48/4/308.
- [64] F. Vico, L. Jiménez, M. Cabedo-Fabrés, C. Bachiller, and A. Voronov. Designing x-band lenses with the adjoint method and 3-d printing techniques. *IEEE Open Journal of Antennas and Propagation*, 6(4):1061–1070, 2025.
- [65] B. Viooreanu and V. Rokhlin. Spectra of multiplication operators as a numerical tool. *SIAM Journal on Scientific Computing*, 36(1):A267–A288, 2014.
- [66] R. Wang. BBFMM3D, 2018. URL <https://github.com/ruoxi-wang/BBFMM3D>.
- [67] R. Wang and S. Ambikasaran. BBFMM2D, 2013. URL <https://github.com/SAFRAN-LAB/BBFMM2D>.
- [68] Z. Wang, J. G. Hoskins, M. Rachh, and A. H. Barnett. The peak heat flux conjecture for the first Dirichlet eigenmode of convex planar domains, Mar. 2026.
- [69] D. Willis, J. Peraire, and J. K. White. FastAero – a precorrected FFT – fast multipole tree steady and unsteady potential flow solver. Jan. 2005.
- [70] A. Wilson and H. Nickisch. Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1775–1784. PMLR, June 2015.
- [71] H. Yan and Y. Liu. An efficient high-order boundary element method for nonlinear wave-wave and wave-body interactions. *Journal of Computational Physics*, 230(2):402–424, 2011.
- [72] X. Ye, J. Xia, and L. Ying. Analytical low-rank compression via proxy point selection. *SIAM Journal on Matrix Analysis and Applications*, July 2020. doi: 10.1137/19M1247838.
- [73] L. Ying, G. Biros, and D. Zorin. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *Journal of Computational Physics*, 196(2):591–626, May 2004. ISSN 0021-9991. doi: 10.1016/j.jcp.2003.11.021.

- [74] L. Zhao and A. Barnett. Robust and efficient solution of the drum problem via nystrom approximation of the fredholm determinant. *SIAM Journal on Numerical Analysis*, 53(4): 1984–2007, 2015.
- [75] L. Zhao and A. Barnett. Robust and efficient solution of the drum problem via Nyström Approximation of the Fredholm determinant. *SIAM Journal on Numerical Analysis*, 53(4): 1984–2007, Jan. 2015. ISSN 0036-1429, 1095-7170. doi: 10.1137/140973992.