Lightning-fast Boundary Element Method

JIONG CHEN, Inria, France
FLORIAN SCHÄFER, Georgia Institute of Technology, USA
MATHIEU DESBRUN, Inria/Ecole Polytechnique, France

Fig. 1. **Diffusion contours.** From *6.6M* salient line segments extracted from van Gogh's "*Irises*" (Getty Museum) acting as Dirichlet boundary conditions, we solve a boundary integral equation to diffuse these boundary colors through Laplace's equation, generating a 9600×7413 image (right) corresponding to 64M extrapolated pixels. With our lightning-fast inverse-LU preconditioner, a GMRES solver reaches a relative error below 0.001 in just *20 iterations*, compared to *4200 iterations* with regular Jacobi preconditioning. This corresponds to a wall-clock time speedup factor of over 200 (15 minutes vs. 2.1 days per color channel).

Boundary element methods (BEM) for solving linear elliptic partial differential equations have gained traction in a wide range of graphics applications: they eliminate the need for volumetric meshing by solving for variables exclusively on the domain boundary through a linear boundary integral equation (BIE). However, BEM often generate dense and ill-conditioned linear systems that lead to poor computational scalability and substantial memory demands for large-scale problems, limiting their applicability and efficiency in practice. In this paper, we address these limitations by generalizing the Kaporin-based approach to *asymmetric* preconditioning: we construct a sparse approximation of the inverse-LU factorization of *arbitrary* BIE matrices in a massively parallel manner. Our sparse inverse-LU factorization, when employed as a preconditioner for the generalized minimal residual (GMRES) method, significantly enhances the efficiency of BIE solves, often yielding orders-of-magnitude speedups in solving times.

CCS Concepts: • **Mathematics of computing → Solvers**; • **Computing methodologies → Computer graphics**.

Additional Key Words and Phrases: Boundary integral equation, Inverse LU decomposition, Screening effect, Preconditioned GMRES

Authors' addresses: Jiong Chen (jiong.chen@inria.fr): Inria Saclay (IP Paris), Palaiseau, France; Florian Schäfer (florian.schaefer@cc.gatech.edu): School of Computational Science and Engineering, Georgia Tech, Atlanta, USA; Mathieu Desbrun (mathieu.desbrun@inria.fr): Inria Saclay and LIX/DIX (IP Paris), Palaiseau, France.

ACM Reference Format:
Jiong Chen, Florian Schäfer, and Mathieu Desbrun. 2025. Lightning-fast Boundary Element Method. *ACM Trans. Graph.* 44, 4 (August 2025), 14 pages. https://doi.org/10.1145/3731196

1 Introduction

The Boundary Element Method (BEM) offers compelling advantages for solving linear partial differential equations. Its key benefits include requiring only a discretization of the boundaries (removing the heavy burden of constructing conforming volumetric meshes in 3D and dramatically reducing the number of unknowns to solve for), handling both interior and exterior problems with equal effectiveness (even for infinite domains with bounded boundaries), and achieving high convergence rates for both the solution and its derivatives throughout the domain. In practice, however, BEM faces significant scalability challenges in both memory *and* computational cost: the resulting boundary integral equations (BIE) lead to dense and often ill-conditioned linear systems [Yuan and Zhang 2019].

A recent collection of works promotes stochastic methods to circumvent some of these issues [Sawhney and Crane 2020; Sawhney et al. 2023; Miller et al. 2024; Sugimoto et al. 2023] by completely bypassing the need for linear solves. These methods leverage a stochastic interpretation of PDEs, computing their solution as the expectation of a large number of random walks. While allowing for rapid and localized previews of solutions and flexible handling of diverse boundary representations, their inherent noise and slow convergence rate often reinforce that traditional deterministic approaches remain essential for applications requiring controllable accuracy at reasonable computational costs.

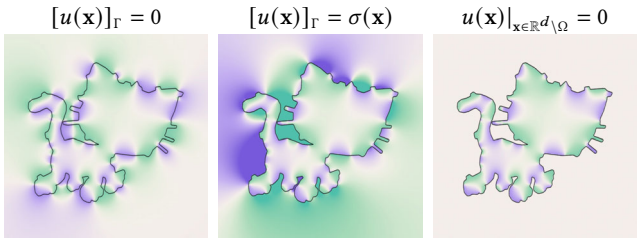


Fig. 2. **Variants of BIE.** A Dirichlet problem for Laplace's equation has a unique solution inside a kitten-shaped domain, but can differ outside based on whether one uses an indirect version with single-layer potential (left) or double-layer potential (middle), or the direct approach (right).

Recent work by Chen et al. [2024] made significant progress towards a fast deterministic approach by introducing an inverse Cholesky preconditioner that substantially speedups the Method of Fundamental Solutions (MFS) [Fairweather and Karageorghis 1998], achieving computational gains of several orders of magnitude compared to existing direct and iterative solvers. However, their approach only applies to *symmetric systems* and, thus, primarily addresses meshless discretization of BIE — while the majority of applications using BEM involves asymmetric systems.

Overview. The core focus of this work is therefore to extend the benefits of inverse preconditioning to general asymmetric linear systems arising from BEM for solving elliptic PDEs. In a nutshell, we present a method for constructing, in a massively parallel manner, an *inverse LU preconditioner* to any (symmetric or asymmetric) BIE matrix. When used to precondition an FMM-accelerated generalized minimal residual (GMRES) iterative method, *our resulting preconditioner enables one to solve large-scale BEM problems with low memory usage* as neither the construction nor the GMRES iterations ever require storing the full dense BIE matrix. Perhaps even more importantly, the efficiency of our preconditioning and its minimal computational overhead at runtime accelerates the GMRES iterative solve of the BIE in all the large-scale experiments we conducted, resulting at times in *orders-of-magnitude speedups* compared to existing solvers. We validate the scalability and efficiency of our method by demonstrating its use in a few BEM-based graphics applications.

2 Boundary element method in graphics: a review

We begin with a review of the boundary element method in order to provide context to our contributions and introduce our notations.

2.1 Boundary integral equation

For the following discussion, Laplace's equation will be used as an illustration — but any other linear operator for which we have explicit knowledge of a fundamental solution of the differential equation can be used. Given a domain $\Omega \subset \mathbb{R}^d$ with boundary $\Gamma \coloneqq \partial\Omega$, a solution to the Laplace equation $\triangle u(\mathbf{x}) = 0$ can be expressed through the sum of two boundary potentials, in an expression valid away from Γ called the *representation formula* [Sauter et al. 2011]:

$$\forall \mathbf{x} \in \mathbb{R}^d \setminus \Gamma,\ u(\mathbf{x}) = \int_\Gamma \frac{\partial G(\mathbf{x},\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}} \sigma(\mathbf{y})\, \mathrm{d}A_\mathbf{y} - \int_\Gamma G(\mathbf{x},\mathbf{y})\tau(\mathbf{y})\, \mathrm{d}A_\mathbf{y}, \quad (1)$$

where $G(\mathbf{x},\mathbf{y})$ is the Green's function (the fundamental solution of the partial differential equation for a Dirac impulse at boundary point $\mathbf{y}$), and $\partial G(\mathbf{x},\mathbf{y})/\partial \mathbf{n}_\mathbf{y}$ is its normal derivative as $\mathbf{n}_\mathbf{y}$ is the outward normal of the domain at $\mathbf{y}$. In this representation of a solution u, the second integral is often referred to as the *single-layer potential*, while the first integral is the *double-layer* (or dipole) *potential*. This general expression involves two *unknown* densities $\sigma(\mathbf{y})$ and $\tau(\mathbf{y})$ that need to be solved for. If we denote $[.]_\Gamma$ the jump of a field across the boundary Γ, the densities are respectively encoding the difference of values of the solution u and its normal derivative on both sides of the boundary Γ, *i.e.*, $\sigma(\mathbf{y}) \coloneqq [u(\mathbf{y})]_\Gamma$ and $\tau(\mathbf{y}) \coloneqq [\partial_{\mathbf{n}_\mathbf{y}} u(\mathbf{y})]_\Gamma$.

These two densities are found by solving what is referred to as the *Boundary Integral Equation* (BIE), which consists in ensuring that the representation formula leads to a solution u satisfying the desired boundary conditions on Γ — with either Dirichlet conditions enforcing particular values (and/or jump values) of u on Γ, or Neumann conditions enforcing the values of the normal derivative of u (and/or its jumps) across Γ, or a mix thereof [Sugimoto et al. 2023, Table 1]. The resulting boundary integral equation can be a Fredholm integral equation of the first kind involving only the Green's function as its kernel, or of the second kind with the normal derivative of the Green's function as its kernel, or even a combination of both if mixed boundary conditions are prescribed.

This general form of the representation formula allows us to deal with a variety of cases such as direct and indirect versions [Costabel 1987]. For instance, for prescribed Dirichlet boundary values on Γ, we can find solutions with no jumps across the boundary ($[u(\mathbf{x})]_\Gamma = 0$), no jumps in the normal derivatives ($[\partial_\mathbf{n} u(\mathbf{x})]_\Gamma = 0$ and $[u(\mathbf{x})]_\Gamma = \sigma(\mathbf{x})$), or vanishing outside the domain Ω ($u(\mathbf{x}) = 0\ \forall \mathbf{x} \in \mathbb{R}^d \setminus \Omega$), as illustrated in Fig. 2. We will elaborate on the exact forms of the BIE for specific applications in Sec. 4.

2.2 Discretization

In order to discretize Eq. (1) and the BIE, basis functions are introduced to parameterize the solution u and the unknown density functions. In all the examples presented in this paper, we discretize the boundary through simplicial elements (line segments in 2D, triangles in 3D) and use piecewise-constant basis functions for the sake of simplicity of exposition and to simplify the evaluation of kernels in a matrix-free manner. The boundary condition, be it Dirichlet or Neumann, is enforced at the centroid of each boundary element by substituting the solution with the representation formula, resulting in a linear system $\mathbf{Gs} = \mathbf{b}$ where the variable $\mathbf{s}$ contains the density values on boundary elements since the integrals in the original equation turn into a sum of local integrals over each elements. The resulting BIE matrix $\mathbf{G}$ is dense due to the global nature of the Green's function and its derivative. Note that while the Green's function is singular at its associated boundary sample point $\mathbf{y}_i$, it is only weakly singular in the sense that the integral of $G(\mathbf{x},\mathbf{y}_k)$ over an element T_k is finite; similarly, the normal derivative is strongly singular at a boundary sample point, but its integral over an element exists in the sense of the Cauchy principle value.

2.3 Existing numerical solvers

In Computer Graphics applications, solving the BIE is often achieved via direct solvers based on Singular Value Decomposition [Schreck et al. 2019] or LU decomposition [James and Pai 1999; De Goes and James 2017; Xia et al. 2020], and sometimes via Conjugate Gradient

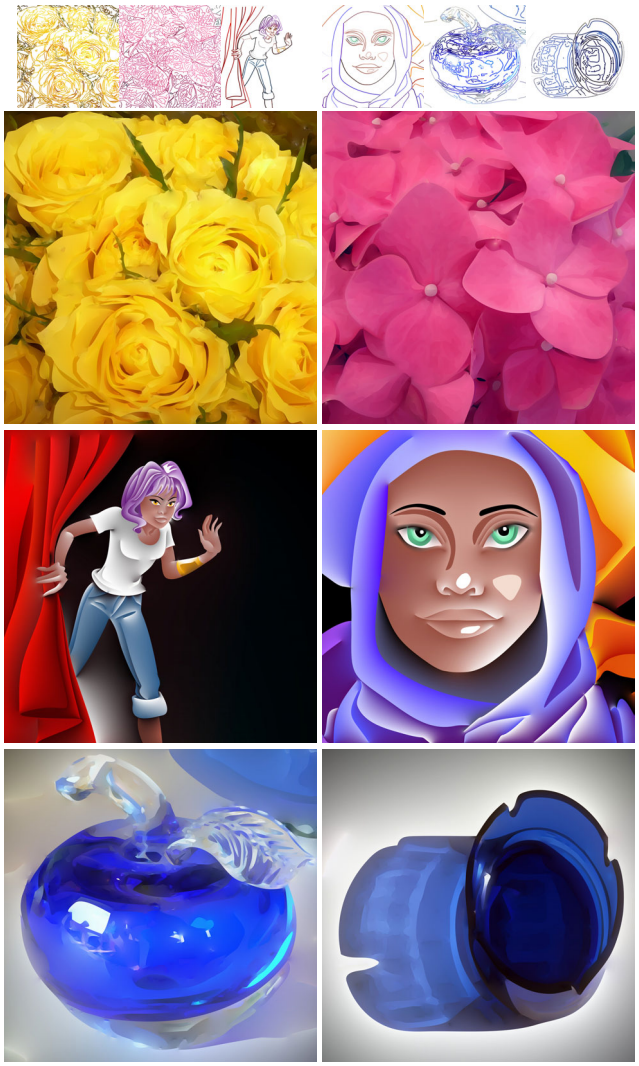


Fig. 3. **Diffusion curves.** Starting with the colored curves (top row) as input, we generate images at a minimum resolution of 4K in approximately 30 seconds to a few minutes (see Table 1), outperforming the Jacobi preconditioner by orders of magnitude (see Fig. 7).

(CG) or Generalized Minimal Residual method (GMRES) for which a Fast Multipole Method (FMM) based dense-matrix-vector multiplication can be employed to drastically improve efficacy [Zhong et al. 2019; Bang et al. 2023]. While direct solvers have a prohibitive cubic complexity, the iteration count of iterative solvers scales poorly in the number of boundary elements in many single-layer BEM contexts, limiting the sizes of problems one can solve efficiently; in fact, most numerical examples in BEM applications were not exceeding 20K degrees of freedom in practice to keep memory footprint and computational costs acceptable.

Efforts to construct fast preconditioners for BIE matrices to accelerate the convergence of FMM-optimized iterative solvers have also been limited. While early works only leveraged basis transformation [Steinbach and Wendland 1998] and ad-hoc multigrid preconditioning [Schippers 1985], more recent improvements involved $\mathcal{H}$-matrices allowing for fast LU factorizations [Kriemann 2013], or nested GMRES-based constructions [Amlani et al. 2019], with reported performance improvements around a factor two. Closely related to our method, the use of (factorized) sparse approximate inverse preconditioners (also known as FSAI) was pursued by [Kolotilina 1988; Vavasis 1992; Carpentieri et al. 2005; Kolotilina and Yeremin 1993]. However, Carpentieri et al. [2004] found that "classical factorized approximate inverses [...] show poor convergence behavior". We show in this paper that approximate inverse preconditioning can actually be tremendously helpful — not for sparse matrices as more commonly considered, but for dense BEM matrices — provided that appropriate ordering and sparsity are employed.

In the specific case of a symmetric and positive-definite BIE matrix, Chen et al. [2024] recently pioneered a much faster, yet accurate alternative: they leveraged Kaporin's variational definition of the inverse-Cholesky factor [Kaporin 1994] and its connection to the statistical screening effect [Stein 2002] observed by Schäfer et al. [2021a]. Although Kaporin's preconditioning was originally formulated for sparse matrices, it was proposed as a Kullback-Leibler optimal direct solver for dense matrices based on Gaussian process regression in [Schäfer et al. 2021a] and coincides with the popular Vecchia approximation [Vecchia 1988; Katzfuss and Guinness 2021] in spatial statistics. Chen et al. [2024] turned it into a massively-parallelizable MFS preconditioner, improving existing solvers by several orders of magnitude. While this contribution now renders MFS methods and the symmetric Galerkin formulation of BEM [Bonnet et al. 1998] able to efficiently handle very large problems (examples with up to 1M degrees of freedom were shown), it does *not address the more general formulations of BEM that generate asymmetric BIE matrices*, which are needed for many graphics applications such as diffusion images.

In addition to all these deterministic approaches, probabilistic methods have gained popularity recently. These methods compute solutions as expectation of a random walk simulation, and accommodate various types of boundary conditions. They are built on either the mean value property of harmonic functions (as in the Walk-on-Sphere method and its variants [Sawhney and Crane 2020; Sawhney et al. 2023; Miller et al. 2024]) or on stochastic evaluations of a Neumann series (as in the Walk-on-Boundary method [Sugimoto et al. 2023]). While their independent pointwise calculations enable massive parallelization, the resulting solution often lacks accuracy, exhibiting "salt-and-pepper" noise as they converge to the ground truth values at a sublinear rate (in the square root of the number of samples). Though useful for quick previews, visually-pleasing results requires boundary caching for path reuse [Miller et al. 2023] or post-processing through denoising [de Goes and Desbrun 2024], which undermines their initial appeal of being able to evaluate solutions in specific regions of interest without resorting to global computations.

2.4 Applications and their current bottlenecks

The Boundary Element Method has been widely adopted in graphics due to its ability to considerably reduce a problem's dimensionality and to deal with infinite domains: compared to the finite-element

method, only a surface needs to be discretized into a mesh to evaluate the solution everywhere in 3D, resulting in a substantial reduction in modeling effort and in the number of unknowns to solve for. Three kinds of second-order partial differential equations have been of particular interest: Laplace's equation, linear elasticity, and the Helmholtz equation. Among the three, Laplace's equation is especially prominent, with applications spanning geometry processing [Solomon et al. 2017], vector graphics [Sun et al. 2012; Bang et al. 2023], and fluid and ferrofluid flow simulation [Da et al. 2016; Huang and Michels 2020; Ni et al. 2024]. The use of BEM for linear elasticity began with the seminal work of James and Pai [1999] before being extended to include fracture modeling [Hahn and Wojtan 2015, 2016] and elastodynamics [Sugimoto et al. 2022]. Additionally, the wave nature of the Helmholtz equation lends itself well to modeling vibrational phenomena in graphics such as sound propagation [James et al. 2006; Umetani et al. 2016] or wave optics for rendering iridescence effects in thin fibers [Xia et al. 2020]. While all these applications benefited from the reduced dimensionality provided by BEM, only a small to moderate number of boundary elements or samples were typically employed in practice to allow for relatively fast dense linear solves using direct or iterative solvers as discussed in Sec. 2.3. Even if an FMM-accelerated evaluation is used, the dense BIE solve remains prohibitively expensive for very large problems, not only in terms of memory requirements, but also often in terms of computational complexity when ill-conditioned BIE systems (which often occur for Fredholm integral equations of the first kind [Yuan and Zhang 2019]) need to be solved. Our work proposes to build a fast BIE solver by quickly and efficiently preconditioning the BIE matrix to allow for large-scale applications of the BEM framework.

3 Constructing an Inverse LU Preconditioner

We now elaborate on our construction of a BIE matrix preconditioner, which will allow us to efficiently evaluate the unknown densities $\mathbf{s}$ from the non-symmetric linear system $\mathbf{Gs} = \mathbf{b}$ given boundary conditions assembled in $\mathbf{b}$. By approximating the inverse of the boundary integral operator $\mathbf{G}$ via LU factorization ($\mathbf{LU} \approx \mathbf{G}^{-1}$), a better conditioned system $\mathbf{UGLz} = \mathbf{Ub}$ can be solved for $\mathbf{z}$ instead, offering faster convergence of the GMRES solver with very small computational overhead as we only need to perform sparse matrix-vector multiplication. The final densities are then trivially obtained through forward-substitution via $\mathbf{s} = \mathbf{Lz}$.

3.1 Inverse LU factorization in closed-form

From inverse Cholesky factorization... Given a symmetric BIE matrix $\mathbf{G}$ of size $B \times B$, it may seem improbable to find a Cholesky factorization of its inverse without computing the inverse first. However, Kaporin [1994] proposed a simple construction of an *incomplete* inverse Cholesky factor, *i.e.*, a sparse approximation of the Cholesky factor of the inverse matrix. This method provides closed-form expressions for each column of the inverse factor. Although initially developed for sparse and symmetric differential operators, Kaporin's approach has proven most effective in other contexts, including in the approximation of Gaussian process regression [Katzfuss and Guinness 2021; Schäfer et al. 2021a] and in the massively-parallel evaluation of preconditioners for the Method of Fundamental Solutions and symmetric Galerkin BIE matrices [Chen et al. 2024]. However, many BEM approaches lead to non-symmetric BIE matrices, for which no good preconditioners have been proposed.

... to inverse LU factorization. Since the non-symmetric version of the Cholesky factorization is known as the LU factorization (factoring a matrix as the product of a lower triangular matrix $\mathbf{L}$ and an upper triangular matrix $\mathbf{U}$), we propose to extend Kaporin's construction to now express an incomplete inverse LU factorization of a non-symmetric BIE matrix $\mathbf{G}$ — an idea mentioned in [Kolotilina and Yeremin 1993] and explored by [Yeremin and Nikishin 2004], albeit for sparse matrices and without specific ordering or sparsity pattern. Given a lower-triangular sparsity pattern $\mathcal{S} \subseteq \{(i,j) |\ i \geq j\}$ specifying the desired nonzero coefficients (fill-ins) of $\mathbf{L}$, we denote the incomplete LU factors of the inverse of $\mathbf{G}$ as $\mathbf{L}_{\mathcal{S}}$ and $\mathbf{U}_{\mathcal{S}^\mathsf{T}}$, where we assumed the same sparsity pattern (up to transpose) for the upper triangular factor — note that we will later discuss that establishing a different sparsity pattern for each factor is possible as well, but would cost unnecessary memory consumption and computational overhead for little preconditioning gain. We propose to compute each column of $\mathbf{L}$ and each row of $\mathbf{U}$ independently through

$$\begin{cases} \mathbf{L}_{\mathcal{S}_j,j} = \dfrac{\mathbf{G}^{-1}_{\mathcal{S}_j,\mathcal{S}_j}\mathbb{e}_j}{\mathbb{e}_j^\mathsf{T}\mathbf{G}^{-1}_{\mathcal{S}_j,\mathcal{S}_j}\mathbb{e}_j}, & \text{(2a)} \\ \mathbf{U}^\mathsf{T}_{j,\mathcal{S}_j} = \mathbf{G}^{-\mathsf{T}}_{\mathcal{S}_j,\mathcal{S}_j}\mathbb{e}_j, & \text{(2b)} \end{cases}$$

where we adopted the notation of [Chen et al. 2024], *i.e.*, the sparsity pattern of the j^{th} column is denoted $\mathcal{S}_j = \{i\ |(i,j) \in \mathcal{S}\}$; $\mathbb{e}_j = [1, 0, \ldots, 0] \in \mathbb{R}^{|\mathcal{S}_j|}$ for $j = 1, ..., B$; and $\mathbf{G}_{\mathcal{S}_j,\mathcal{S}_j}$ denotes the submatrix indexed by $\mathcal{S}_j$ of the matrix $\mathbf{G}$, while $\mathbf{G}^{-\mathsf{T}}_{\mathcal{S}_j,\mathcal{S}_j}$ denotes the transpose of its inverse. Our normalization in Eq. (2a) follows the standard LU decomposition convention of a lower triangular factor with unit diagonal; and similar to the symmetric case, our factorization turns out to be the solution of a variational problem as shown in App. B.

From Eq. (2), one can notice that computing $\mathbf{L}_{\mathcal{S}_j,j}$ and $\mathbf{U}^\mathsf{T}_{\mathcal{S}_j,j}$ only requires a small linear system solve involving $\mathbf{G}_{\mathcal{S}_j,\mathcal{S}_j}$. As a result, each row and column of the inverse LU factors can be computed efficiently in a massively parallel manner, without *ever* assembling the entire BIE matrix in memory. To perform local solves, we propose a local UL factorization of $\mathbf{G}_{\mathcal{S}_j,\mathcal{S}_j} = \mathbb{U}_{\mathcal{S}_j,\mathcal{S}_j}\mathbb{L}_{\mathcal{S}_j,\mathcal{S}_j}$ which decomposes it into the product of an upper triangular matrix and a lower triangle matrix with a unit diagonal, the resulting factors $\mathbb{U}_{\mathcal{S}_j,\mathcal{S}_j}$ and $\mathbb{L}_{\mathcal{S}_j,\mathcal{S}_j}$ having dimensions $|\mathcal{S}_j| \times |\mathcal{S}_j|$; we can then evaluate our inverse LU factors efficiently since, by substituting this factorization into Eq. (2),

$$\boxed{\mathbf{L}_{\mathcal{S}_j,j} = \frac{\mathbb{L}^{-1}_{\mathcal{S}_j,\mathcal{S}_j}\mathbb{e}_j}{\mathbb{e}_j^\mathsf{T}\mathbb{L}^{-1}_{\mathcal{S}_j,\mathcal{S}_j}\mathbb{e}_j}, \quad \mathbf{U}^\mathsf{T}_{j,\mathcal{S}_j} = \mathbb{U}^{-\mathsf{T}}_{\mathcal{S}_j,\mathcal{S}_j}\mathbb{e}_j, \quad \forall j = 1, \ldots, B.} \tag{3}$$

Our use of a UL (instead of LU) decomposition of $\mathbf{G}_{\mathcal{S}_j,\mathcal{S}_j}$ improves computational efficiency as it allows to reuse local factorizations for processing multiple rows and columns belonging to the same supernode as we will discuss in Sec. 3.3. Similar to the Cholesky case, Eq. (2) implies a unit diagonal, that is,

$$\operatorname{diag}(\mathbf{U}_{\mathcal{S}^\mathsf{T}}\mathbf{G}\mathbf{L}_{\mathcal{S}}) = \mathbb{1}, \tag{4}$$

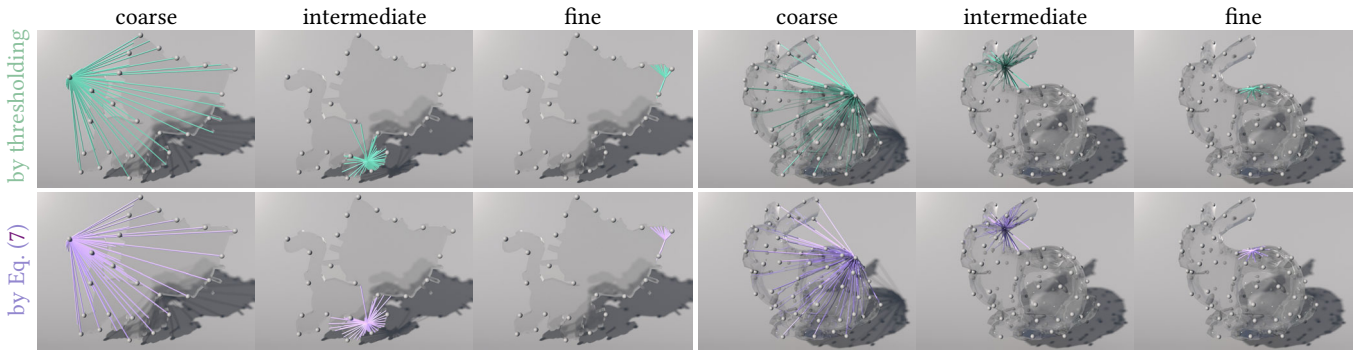


Fig. 4. **Multiscale structure of the sparsity pattern.** In order to validate the efficacy of our inverse LU approximation, we first reorder the boundary samples (centroids of boundary simplices) and compute the exact LU factorization of the inverse single-layer operator — a computationally feasible task for these small models. We then sparsify the columns of both LU factors individually by thresholding to zero any value smaller than 0.6% of the associated diagonal value to build a ground-truth sparsity, and we visualize the remaining non-zero connections emerging from points (corresponding to the leftover column values from these points) across different scales. These two comparisons demonstrate that the inverse factors are localized within each scale, and that our geometric sparsity construction, derived from the screening effect, captures the significant non-zero fill-ins, closely matching the ground-truth patterns in both 2D (left) and 3D (right). Note that only the coarsest boundary samples are displayed to maintain legibility.

which is easy to verify by directly evaluating a diagonal entry:

$$\forall j, \mathbf{U}_{j,\mathcal{S}_j}\mathbf{G}\,\mathbf{L}_{\mathcal{S}_j,j} = \mathbf{U}_{j,\mathcal{S}_j}\mathbf{G}_{\mathcal{S}_j,\mathcal{S}_j}\mathbf{L}_{\mathcal{S}_j,j} = \mathbf{e}_j^{\mathsf{T}}\mathbf{G}^{-1}_{\mathcal{S}_j,\mathcal{S}_j}\mathbf{G}_{\mathcal{S}_j,\mathcal{S}_j}\frac{\mathbf{G}^{-1}_{\mathcal{S}_j,\mathcal{S}_j}\mathbf{e}_j}{\mathbf{e}_j^{\mathsf{T}}\mathbf{G}^{-1}_{\mathcal{S}_j,\mathcal{S}_j}\mathbf{e}_j} = 1.$$

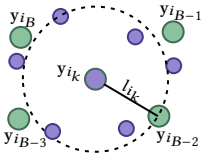
3.2 Ordering and sparsity selection

A critical step to the quality of preconditioners based on incomplete factorization is the pre-ordering of all degrees of freedom (and thus, of the rows and columns of the input matrix [Benzi 2002]) as it directly influences numerical performance and conditioning improvement. Similarly, determining which neighbors to consider to enforce sparsity but limit inaccuracy can be key in preconditioning efficacy, a matter left unaddressed by previous sparse inverse preconditioners. We discuss our choices next.

Ordering. To precondition a BIE matrix, we employ a reverse max-min ordering [Guinness 2018; Schäfer et al. 2021b] before computing the two factors defined in Eq. (3). Specifically, we reorder the boundary points using a farthest-point sampling strategy: at each step of a descending order $k = B, B-1, .., 1$, the next point is chosen as the farthest point from the previously-selected ones; that is, the k^{th} index i_k is defined as

$$i_k = \underset{p\in[1,..,B]\setminus\{i_{k+1},..,i_B\}}{\operatorname{argmax}} \min_{q\in\{i_{k+1},..,i_B\}} \operatorname{dist}(\mathbf{y}_p, \mathbf{y}_q), \tag{5}$$

where $\operatorname{dist}(\cdot,\cdot)$ is the Euclidean distance between two boundary samples. The initial index i_B is generically selected as the index of the point closest to a corner of the domain's bounding box. Although a brute-force approach to compute a farthest-point sampling has cubic complexity in the number B of boundary samples, an $O(B\log^2 B)$ algorithm was, for instance, proposed by [Schäfer et al. 2021b]. During the reordering of the boundary samples, a length scale (see inset) is also assigned to each boundary point, defined as



$$l_{i_k} = \min_{q\in\{i_{k+1},..,i_B\}} \operatorname{dist}(\mathbf{y}_{i_k}, \mathbf{y}_q), \tag{6}$$

encoding the spatial influence of a boundary point that is monotonically increasing in the reverse max-min ordering such that the coarser scale a point is at, the larger its support radius.

Multiscale sparsity pattern. From the length scales, we define the lower triangular sparsity pattern as in [Schäfer et al. 2021b; Chen et al. 2021], that is, we use

$$S_\rho = \{(i,j) \mid i \geq j \text{ and } \operatorname{dist}(\mathbf{y}_i, \mathbf{y}_j) < \rho \min(l_i, l_j)\}, \tag{7}$$

meaning that an element (i,j) in S_ρ will be forced to be zero if $\mathbf{y}_i$ and $\mathbf{y}_j$ are not within each other's support radius scaled by a parameter $\rho > 0$. This adjustable parameter provides a trade-off between accuracy and performance of our LU preconditioner: the smaller ρ is, the more sparse the approximations of the inverse LU factors will be — but the less accurate they will be.

Rationale behind ordering and sparsity choices. The non-locality of Green's functions for second-order PDEs implies long-range spatial interactions between boundary elements, hence the density of the BIE matrix and its ensuing challenges to numerical efficiency. But when performing Gaussian elimination, this non-locality allows the elimination of a degree of freedom to neutralize interactions

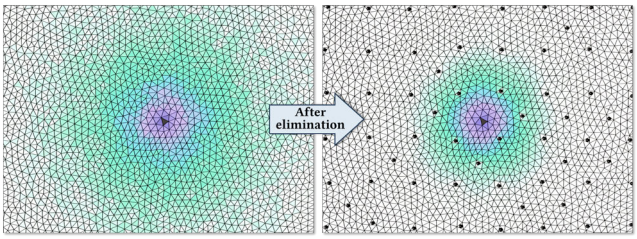


Fig. 5. **Screening long-range interactions.** We visualize a single row of the discretized single-layer operator corresponding to the central element (left) and the same row in the Schur complement matrix after eliminating a number of elements around the center (right). The elimination significantly weakens interaction strength between the central element and distant ones, subject to the screening effect of Green's function.

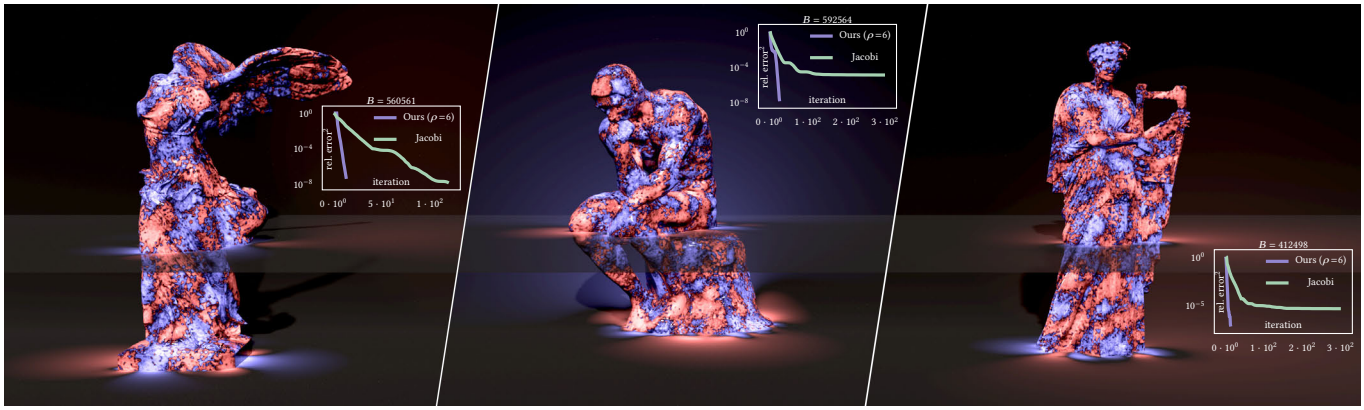


Fig. 6. **3D diffusion.** For a Dirichlet boundary condition based on noise functions, the single-layer BIE for this Laplace's equation results in a density of change that is diffused into the ambient space via the representation formula (Eq. (10)) and visualized through a cross section. The intricate geometry of these boundaries leads to large-scale BIEs with approximately 0.5M unknowns in these three examples. Despite the problem's size, our method achieves efficient GMRES convergence within 10 to 20 iterations, while a simple Jacobi preconditioner often fails to even converge in practice.

in the remaining matrix, see Fig. 5. In probabilistic terms, dependencies between degrees of freedom increase the complexity of the joint distribution, but the resulting redundancy decreases the complexity of its *conditional* distributions, which are essentially encoded by the inverse-Cholesky factors. This is the key idea behind the *screening effect* by which conditioning a smooth stochastic process on values near a target point weakens the target's correlation with more distant points. Intuitively, this happens because the information provided by the nearby points renders that of the distant points redundant. The screening effect has been observed and studied in Gaussian process (GP) regression and geostatistics for years [Stein 2002, 2011]. Ordering points progressively from fine scales to coarse scales distributes points evenly within each scale, thereby adequately exploiting the screening effect throughout the factorization. The conditional correlation length of each point is proportional to the density of the remaining points, resulting in a constant number of non-negligible entries per column. This is precisely what Schäfer et al. [2021a] exploited through max-min ordering and multiscale sparsity to approximate the sparse inverse of the Matérn covariance matrix, an approach later adapted to MFS systems in [Chen et al. 2024]. While these prior works were restricted to symmetric systems, the efficiency and accuracy afforded by the screening effect numerically extends to sparse inverse approximation of asymmetric BEM operators through our approach: Fig. 4 shows that the sparsity pattern constructed from Eq. (7) closely matches the ground-truth sparsity obtained by truncating the *exact* inverse LU factors of a *single-layer* operator (in this case, for Laplace's equation). We will also demonstrate in Sec. 4 the efficacy of our resulting preconditioner in multiple graphics-related scenarios.

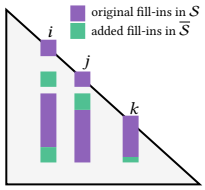
Fallback single-scale sparsity pattern. In the few occasions where the screening effect is too weak to bring conditioning benefits, we resort to a fallback sparsity pattern where now the length scales are all set *equal*: in this case, each boundary element has potentially much denser connections to others, thereby removing the benefits of establishing scales. We will clearly identify in Sec. 3.4 the few types of BEM problems for which this pattern offers a slight improvement over the multiscale pattern described above.

3.3 Supernodal implementation

As discussed in Sec. 3.1, computing each column of $\mathbf{L}$ (or $\mathbf{U}^\intercal$) involves assembling a local matrix extracted from the global BIE system and solving the associated small linear system with a unit vector as the right-hand side. This process incurs a memory cost of $O(\sum_j \|\mathcal{S}_j\|^2)$ and a computational cost of $O(\max_j \|\mathcal{S}_j\|^3)$, assuming sufficient threads are available for parallel solves. For boundaries with a large number of elements, the memory and computational costs of constructing our preconditioner would become prohibitively high. Yet, boundary samples that are close in both *space and scales* tend to have very similar interactions. This observation motivates the construction of *supernodes*, a usual recourse in compute-intensive tasks to reduce memory and numerical operations by avoiding duplicate storage and reusing factorization results [Stein et al. 2004; Guinness 2018; Schäfer et al. 2021a; Ferronato et al. 2015]. A supernode, denoted $\mathbb{J}$, represents a group of nearby points of similar scale to be treated concurrently, and are thus sets of selected indices. We identify supernodes $\{\mathbb{J}_k\}_k$ following the approach in [Chen et al. 2024, Algorithm 1]. Once these supernodes have been found, the sparsity pattern is extended to enable the *reuse of local factorizations*: for each supernode $\mathbb{J}$, we merge all non-zero fill-ins of its members from the original sparsity pattern into a new sparsity set:

$$\forall j \in \mathbb{J}, \quad \overline{\mathcal{S}}_j = \{i \mid i \geq j \text{ and } \exists\, k \in \mathbb{J} \text{ such that } (i,k) \in \mathcal{S}\}. \tag{8}$$

This padding process adds extra entries to each column to align the sparsity patterns within a supernode. As a result, for any $j,k \in \mathbb{J}$ with $k > j$, $\overline{\mathcal{S}}_k \subset \overline{\mathcal{S}}_j$ (see inset). Instead of storing a local BIE matrix for each column, we thus only need to store one for each supernode, significantly reducing memory usage as the number of supernodes is far smaller. The size of each local matrix is determined by the largest number of non-zero fill-ins within the supernode; for example, given a supernode $\mathbb{J} = \{i, j, k\}$ shown in the inset, the size of the local system is $|\overline{\mathcal{S}}_i| \times |\overline{\mathcal{S}}_i|$.



As mentioned in Sec. 3.1, we perform a local UL factorization of $\mathbf{G}_{\mathcal{S}_j,\mathcal{S}_j}$ to evaluate solve $\mathbf{L}_j$ and $\mathbf{U}_j^\intercal$. This is because *the bottom-right*

subblocks of $\mathbb{U}$ *and* $\mathbb{L}$ *corresponds to the UL factors of the bottom-right subblock of* $\mathbf{G}_{S_j,S_j}$: this property thus allows for a one-time factorization for all nodes within a supernode, permitting the reuse of the factorization results for triangular solves associated with each of its member columns, similar to what is done in [Chen et al. 2024, Algorithm 2]. Note that computing a UL variant is no more complex than performing a standard LU factorization: if we denote by $\mathcal{L}(.)$ (*resp.*, $\mathcal{U}(.)$) the lower (*resp.*, upper) triangular factor of a standard LU factorization as a function of the input matrix $\mathbb{G}$, then the local UL factors are directly deduced from the LU factors through

$$\mathbb{U} = \mathbb{P}_{\text{rev}}\mathcal{L}(\mathbb{P}_{\text{rev}}\mathbb{G}\mathbb{P}_{\text{rev}})\mathbb{P}_{\text{rev}}, \quad \mathbb{L} = \mathbb{P}_{\text{rev}}\mathcal{U}(\mathbb{P}_{\text{rev}}\mathbb{G}\mathbb{P}_{\text{rev}})\mathbb{P}_{\text{rev}}, \tag{9}$$

where $\mathbb{P}_{\text{rev}}$ is the exchange (*i.e.*, reverse permutation) matrix which reorders the sequence $\{1, 2, .., n\}$ to $\{n, n-1, .., 1\}$.

3.4 Discussion

We conclude our exposition of our preconditioner construction by discussing some of the choices we made when deriving our approach, and previewing the cases in which this preconditioner can be expected to be particularly efficient in practical uses.

Choice of sparsity patterns for $\mathbf{L}$ *and* $\mathbf{U}$. Notice that we use the same sparsity pattern for $\mathbf{L}$ as for $\mathbf{U}^\intercal$, when they could, theoretically, be different. This strategy was motivated by practical reasons. First off, two different sparsity patterns would necessitate constructing of separate supernode structures, storing two distinct sets of local Green submatrices for $\mathbf{L}$ and $\mathbf{U}$, and performing local factorizations independently, significantly degrading the efficiency of the preconditioner construction. Moreover, while an asymmetric notion of distance $\text{dist}(\cdot,\cdot)$ could be devised to induce anisotropic interactions between samples, computing the induced ordering efficiently would be potentially far more difficult. So while such an asymmetry between the two sparsity patterns of the inverse factors could potentially help to better deal with irregular sampling of the boundary Γ, it is unlikely to compensate for the drop in efficiency.

Expected efficacy. Our general preconditioner for BEM leverages the inherent sparsity of the inverse of the BIE matrix, and has roots in numerical homogenization [Chen et al. 2023], which promises gains in efficiency and scalability when combined with an FMM-accelerated iterative solver such as GMRES to solve boundary integral equations. Indeed, scalability is dramatically increased as the entire BIE matrix needs not even be assembled and stored in memory, allowing the handling of millions of DoFs in practice as we will demonstrate in Sec. 4. As for efficiency, one can also describe expected behaviors for the various types of BEM problems on which our approach can be used. Our approach exploits the *screening effect* (as discussed in Sec. 3.2), which holds particularly well in the case of elliptic problems with Dirichlet boundary conditions; we can thus expect our preconditioners to be particularly helpful in these cases — and Sec. 4 will demonstrate *orders of magnitude* improvements over existing solvers indeed in these cases. Various factors can, however, weaken the screening effect, lowering the efficiency of our preconditioner. First, the addition of integral/solution operators associated to different PDEs will weaken the screening effect, since the inverses of sums of elliptic solution operators are nonlocal. From a statistical perspective, this is because conditioning on the sum of two random processes does not give full information about either process and thus results in weaker screening. The most common example is the addition of a positive diagonal matrix (the solution operator of a zeroth order equation) to the BEM matrix, as in the case of the second-kind Fredholm equations composed of the pure double-layer potential. Fortunately, this case is typically leading to better conditioned systems [Steinbach and Wendland 2001]. While our preconditioner will be less effective, the GMRES-based linear solve will be quick to produce a solution anyway — and the use of single-scale sparsity pattern in this case may even improve conditioning efficiency (see Fig. 11). Second, a strong hyperbolic component in the PDE will also debase this screening effect at coarse scales, thus our preconditioner will help less. A third reason for which the efficiency of our preconditioner can decrease is if mixed boundary conditions are present, or if Neumann boundary conditions are set on a very complex and wavy boundaries: as the resulting BIE matrix mixes kernel values of highly-varying magnitudes, the screening effect is less pronounced; but unlike the first case, the condition number of the BIE matrix becomes worse, further hampering GMRES-based solves. Specifically, in this last case that involves mixed boundary conditions, ordering the density values of the double-layer potential *first* using a single scale *followed by* the density values of the single-layer potential with their associated multiscale length scales is then advised for maximum efficacy. Despite these potential factors diminishing our preconditioning efficacy, we will show through concrete applications that our method is always beneficial: even when dealing with theoretically very well-conditioned problems such as double-layer operators, factors such as the irregularity of the boundary mesh (in particular, for graded or anisotropic simplices) can drastically affect the condition number of the BIE matrix; and in this case, our preconditioner can still offer up to an order of magnitude acceleration in wall-clock timings. In sum, our preconditioner is all the more useful when the BIE solution is difficult to compute.

Table 1. **Timing statistics.** Problem sizes and timings for the examples shown in Fig. 3 are provided. For each scene, the cubic Bézier curves are discretized into B line segments, which are used to generate the final image at the indicated resolution. Timings for precomputation (pre.), BIE solves (for RGB channels) with the $B\times B$ matrix (slv.) and extrapolation (extrap.) of the solution through Eq. (10) are detailed, all in seconds. The error tolerance for BIE solves is set to 10^{-3}, and ρ is set to 7.5 for all these examples.

scene	#curves	B	image res.	t(pre.)	t(slv.)	t(extrap.)
Astray	417	171471	4096×4096	3.2	33.8	9.1
BehindCurtain	131	140751	4096×4096	2.1	20.8	8.1
PinkFlowers	2964	872901	6144×8192	51.8	213.6	38.3
BlueApple	984	285532	4096×4096	7.9	45.6	11.1
Zephir	87	368774	4096×4096	11.3	82.0	11.9
Roses	3906	944822	5136×6400	61.6	192.0	35.3

4 Applications and Results

We now discuss implementation details, before describing various experiments we conducted in order to demonstrate both the efficiency and scalability of our technique and to illustrate the different possible types of BIE that are relevant in graphics applications.

4.1 Implementation details

We implemented our approach described in Sec. 3 in CUDA and C++, so that the inverse LU factors of the preconditioner are computed

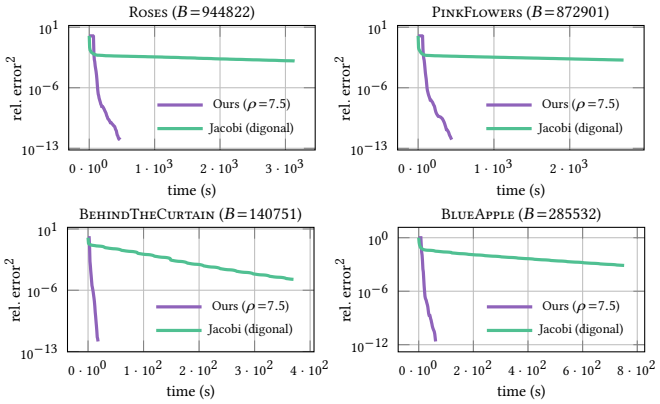
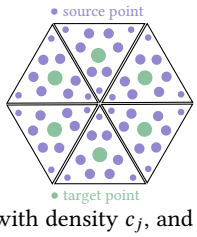


Fig. 7. **Comparison with Jacobi preconditioner.** We compare our preconditioner with a Jacobi preconditioner on four examples of Fig. 3. All precomputations (before the GMRES iterations) including ordering and sparsity patterns are included as evidenced by the flat beginning of our timing curves. The larger the problem (indicated by the number B of boundary elements), the larger the speedup.

on the GPU to best utilize parallelism, before being transferred to the CPU to precondition GMRES iterations. The restart number for GMRES is set to 40 by default. The matrix-vector product for preconditioning uses Intel MKL sparse BLAS routines for efficiency. Note that all examples in this paper were run on a desktop with an Intel Xeon w7-3555 CPU (256GB RAM) and an Nvidia RTX 5000 GPU with 32GB of memory. Our open-source code is available at https://gitlab.inria.fr/geomerix/public/lubie.

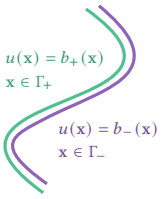
Boundary discretization. As pointed out in Sec. 2.2, we discretize the boundary using piecewise constant functions over simplicial elements. More formally, the boundary Γ is approximated by a set of B elements $\{T_k\}_{k=1..B}$ with centroids $\{\mathbf{y}_k\}_{k=1..B}$ and associated basis function $\{\varphi_k(\mathbf{y})\}_{k=1..B}$. The density function σ is then interpolated in a piecewise-constant manner as $\sigma(\mathbf{y}) = \sum_{k=1}^{B} \sigma_k \, \varphi_k(\mathbf{y})$ and similarly for τ. This choice simplifies our implementation as the data required to store and assemble local BIE matrices is more structured across all boundary elements. For singular integrals where the evaluation point is the centroid of an element, we use the closed-form expressions that are derived in App. A. For other cases, we employ quadrature rules: an 8-point Gauss-Legendre quadrature for line segments in 2D, and a 9-point quadrature scheme [Strang and Fix 2008, Table 4.1] for triangles in 3D.

Fast multipole method. We also accelerate boundary integral evaluations by incorporating FMM in our implementation for both BIE solves and solution evaluation. We opted for the FMM2D and FMM3D open-source libraries [Gimbutas et al. 2024a,b] to efficiently compute the summation $u_i = \sum_j G(\mathbf{x}_i, \mathbf{y}_j) c_j$ and its gradient, where $\mathbf{y}_j$ are source points with density c_j, and $\mathbf{x}_i$ are target points. To evaluate an integral, we specify as source points all the quadrature points of the boundary elements, while target points are set to the centroids of elements where boundary conditions are imposed (inset). Quadrature weights and element areas required for integral approximation are included in the density c_j, effectively treating them as charges at source points.



4.2 Diffusion curves and surfaces

Diffusion curves, introduced by Orzan et al. [2008], is an approach to generate smoothly-shaded images by diffusing colors, given on each side of prescribed curves, to the entire image at an arbitrary resolution. Since this diffusion process requires a BIE solve, Chen et al. [2024] tested their preconditioners on a few examples, demonstrating significant gains. However, to ensure a *symmetric* BIE matrix, they only performed *pixel-based* color constraints for which the Method of Fundamental Solutions is particularly convenient. However, diffusion images require an asymmetric system when 1D curves are prescribed as boundary conditions for colors. Indeed, the boundary Γ is defined as a set of cubic Bézier curves, and side colors $\{b_-(\mathbf{x}), b_+(\mathbf{x})\}_{\mathbf{x}\in\Gamma}$ are prescribed as Dirichlet boundary conditions (see inset). In order to generate an image, each Bézier curve is slightly offset along its normal in both directions, creating two displaced boundaries Γ_- and Γ_+ separated by a distance of about one pixel. The solution, continuous across the rest of the domain, is then represented using a single-layer potential integrated over the boundary $\Gamma = \Gamma_- \cup \Gamma_+$ via



$$\forall \mathbf{x} \in \mathbb{R}^2 \backslash \Gamma, \quad u(\mathbf{x}) = \int_\Gamma G(\mathbf{x}, \mathbf{y}) \sigma(\mathbf{y}) \mathrm{d}A_\mathbf{y}. \tag{10}$$

Imposing Dirichlet conditions yields the following BIE system

$$\boxed{\forall \mathbf{x} \in \Gamma, \quad \int_\Gamma G(\mathbf{x}, \mathbf{y}) \sigma(\mathbf{y}) \, \mathrm{d}A_\mathbf{y} = b(\mathbf{x}),} \tag{11}$$

where $b(\mathbf{x})$ encodes both $b_-(\mathbf{x})$ and $b_+(\mathbf{x})$. This is a Fredholm integral equation of the first kind, thus typically ill-conditioned.

Diffusion contours and curves. Whether one extracts salient contours from an input image using OpenCV library [Bradski 2000] (as done in Fig. 1) or directly uses a set of given diffusion curves (as in Fig. 3), one can discretize this diffusion equation by sampling the contours or curves uniformly with a pixel-based spacing. All the elements of the BIE matrix $\mathbf{G}$ can then be easily evaluated through integration over elements, leading to its asymmetry. We finally set the two-sided colors for each extracted contour or each curve as the right-hand side of the BIE system, and solve for the unknown densities σ on the sampled elements, rather than rasterizing the curves as in [Orzan et al. 2008].

Comparisons. While diffusing colors from boundary points symmetrizes the discretization and simplifies the linear solve of the BIE, it introduces artifacts when the boundary is undersampled (see Fig. 8). As for asymmetric problems, Chen et al. [2024] proposed to store the Green matrix $\mathbf{G}$ and factorize the least squares system $\mathbf{G}^\mathsf{T}\mathbf{G}$ to leverage their symmetric preconditioning; but now we can directly handle the BIE linear solve without even having to assemble the entire matrix $\mathbf{G}$. Our method thus enables the efficient processing of high-resolution diffusion images, as 4K or 8K images are now possible in seconds or minutes. Timing statistics for the BIE solve using GMRES are provided in Fig. 7 for several examples from Fig. 3, clearly demonstrating the effectiveness of our

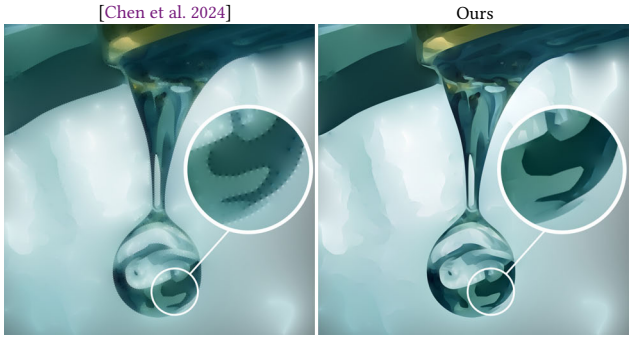


Fig. 8. **Diffusion points vs. diffusion curves.** To ensure a symmetric BIE matrix on which to apply their inverse Cholesky preconditioner, Chen et al. [2024] generated diffusion images from colored sample points. However, this approach introduces noticeable artifacts when the number of samples is insufficient (left), which can be prevented by diffusing colors along curves instead as our approach allows (right). Both approaches result in BIE matrices of size 4632×4632. For [Chen et al. 2024], the PCG solver requires 18 iterations to reach an error below 10^{-3}, taking 7.3 seconds in total. Our GMRES solver converges in 8 iterations for the same error threshold, taking 1.8 seconds in total. The resolution of the generated image is 1246×1246.

preconditioner compared to the Jacobi preconditioner which struggles to reduce the error — especially as the number of boundary elements gets large. Note that multigrid preconditioners for sparse matrices do not apply to these large dense matrices due to memory issues while $\mathcal{H}$-matrix based approaches suffer from unfavourable cost-accuracy trade-offs as reported in [Chen et al. 2024], and other existing preconditioners show only marginal improvement over Jacobi preconditioning as discussed in Sec. 2.3. Finally, we point out that while our work follows [Orzan et al. 2008] to impose two-sided Dirichlet boundary conditions through offsetting Bézier curves and employs only the single-layer potential, alternative representations also exist, such as Eq. (1) which incorporates the double-layer potential to describe discontinuities across boundaries [van de Gronde 2010; Bang et al. 2023]. Despite the distinction in representation, both approaches result in the Fredholm integral equations of the first kind that require to invert only the single-layer operator, making our preconditioner equally effective in both approaches. However, we found that incorporating the double-layer potential tends to introduce more obvious aliasing artifacts in practice (see Fig. 9), most likely due to its higher singularity near the boundary which requires additional efforts for anti-aliasing such as the adaptive subdivision described in [Bang et al. 2023].

3D Laplace's equation. We also tested our method in Fig. 6 by solving for heat radiation from a complex surface embedded in $\mathbb{R}^3$. While BEM is attractive due to offering a volumetric-mesh free approach and dimensionality reduction, large boundary meshes often challenge existing solvers: the size of the fine surface discretization required to capture geometric complexity often make solvers run out the memory, particularly when attempting to store the full BIE matrix; if enough memory is available, direct solvers or iterative solvers can be impractically slow to deal with such large-scale ill-conditioned BIE. Our preconditioner allows not only to keep the memory requirements low, but offers significant improvements to

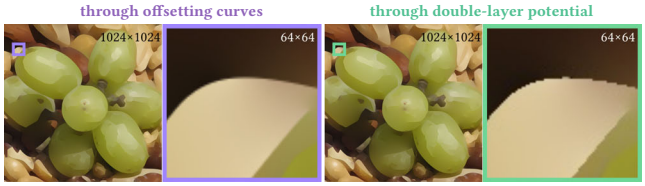


Fig. 9. **Offsetting curves vs. double-layer potential.** Compared to offsetting the curves (left) and using only the single-layer potential, introducing a double-layer potential through Eq. (1) (right) also enables color jumps across the boundary, with visually similar results. However, the latter tends to produce more aliasing artifacts as illustrated by the zoomed-in portion, likely due to being strongly (rather than weakly) singular near the boundary.

the GMRES solve, affording fast convergence when simple existing preconditioners often fail to converge altogether.

4.3 Magnetostatics

In ferrofluid simulation [Ni et al. 2024], the free-surface of a ferrofluid is submitted to a pressure force due to the magnetic field $\mathbf{H}$, which includes both the external magnetic field $\mathbf{H}_{\text{ext}}$ and the "scattered" magnetic field $\mathbf{H}_\Omega$ induced by the ferrofluid itself. Under the zero-current assumption, Ampère's law implies that $\mathbf{H}_\Omega$ is curl-free, hence it is the gradient of a scalar potential, *i.e.*, $\mathbf{H}_\Omega(\mathbf{x}) = -\nabla u(\mathbf{x})$. Ni et al. [2024] further show that Gauss's law implies that u is a harmonic function which satisfies Neumann boundary conditions $\partial_{\mathbf{n}} u(\mathbf{x})_+ + \chi \mathbf{H}_{\text{ext}} \cdot \mathbf{n} = (1+\chi)\partial_{\mathbf{n}} u(\mathbf{x})_-$ for $\mathbf{x} \in \Gamma$, where χ is the (constant) magnetic susceptibility of the ferrofluid.

The scalar field u being harmonic, we can represent it as a single-layer potential as in Eq. (10). While the potential is continuous across the boundary, its normal derivative will have a jump expressed as

$$\forall \mathbf{x} \in \Gamma,\ \partial_{\mathbf{n}} u(\mathbf{x})_\pm = \int_\Gamma \frac{\partial G(\mathbf{x},\mathbf{y})}{\partial \mathbf{n}_\mathbf{x}} \sigma(\mathbf{y}) \mathrm{d}A_\mathbf{y} \mp \frac{1}{2}\sigma(\mathbf{x}), \tag{12}$$

where the exterior and interior normal derivatives are distinguished by the subscript + and − respectively. By substituting this representation into the Neumann boundary condition, the BIE we need to solve for this ferrofluid context becomes:

$$\boxed{\forall \mathbf{x} \in \Gamma,\ \frac{2+\chi}{2\chi}\sigma(\mathbf{x}) + \int_\Gamma \frac{\partial G(\mathbf{x},\mathbf{y})}{\partial \mathbf{n}_\mathbf{x}} \sigma(\mathbf{y})\ \mathrm{d}A_\mathbf{y} = \mathbf{H}_{\text{ext}} \cdot \mathbf{n}.} \tag{13}$$

This equation falls into the category of Fredholm integral equations of the second kind, where the integral on the left-hand side corresponds to the adjoint double-layer potential [Steinbach 2007].

As explained in App. A, the double-layer operator has a vanishing diagonal when the boundary is discretized with piecewise constant elements, causing all diagonal entries in the BIE matrix to be identical. When the susceptibility χ is small, diagonal terms become dominant, improving the conditioning of the BIE system. In this regime, even simple fixed-point iterations can outperform Krylov subspace methods [Ni et al. 2024]. However, as χ increases, diagonal terms asymptotically approach $1/2$, drastically diminishing the effectiveness of fixed-point iterations [Ni et al. 2024, Fig. 13]. Despite this degradation for large χ values, the BIE system retains good properties; specifically, it remains a compact operator on Lipschitz boundaries [Costabel 1988; Steinbach and Wendland 2001], resulting in a spectrum that is significantly better behaved

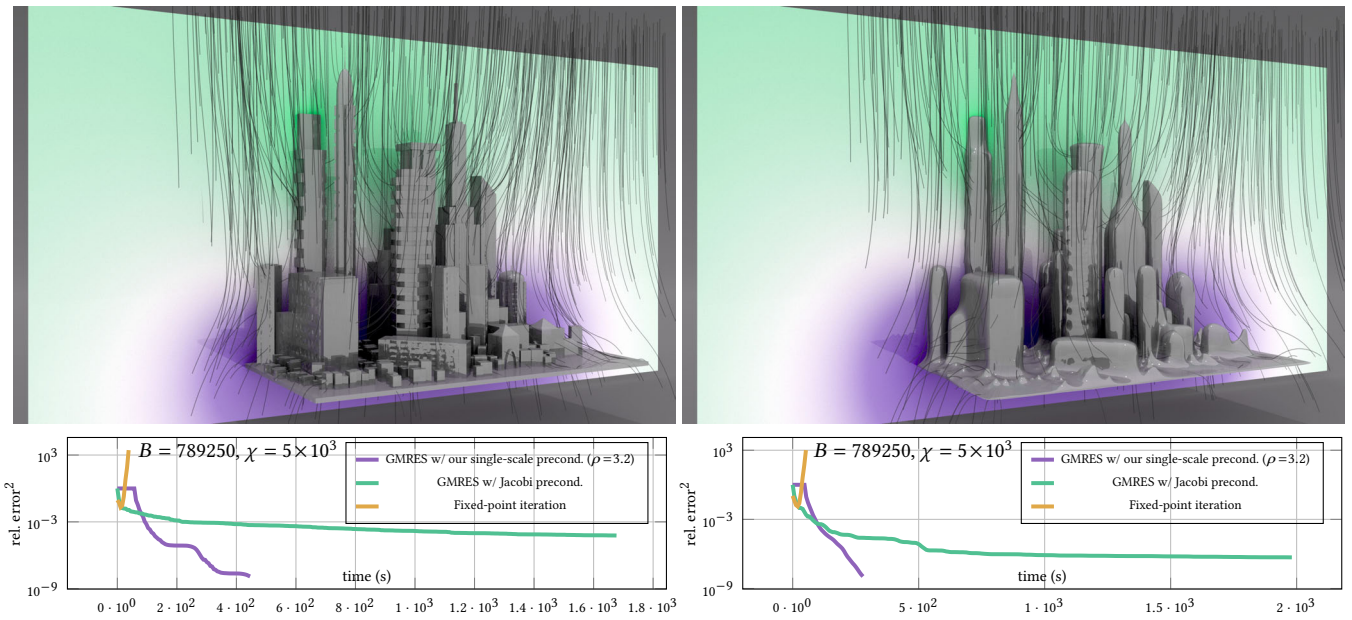


Fig. 10. **Magnetostatics.** In this example, an external magnetic field $\mathbf{H}_{\text{ext}} = (0, 1, 0)$ is applied and scattered by the input model. The resulting field $u(\mathbf{x})$ is visualized on a vertical slice of the 3D volume, with its gradient corresponding to the induced magnetic field $\mathbf{H}_\Omega$, and some of its streamlines are shown by tracing random particles advected in the total magnetic field $\mathbf{H} = \mathbf{H}_{\text{ext}} + \mathbf{H}_\Omega$. Due to sharp features and thin structures (left), the Jacobi preconditioner struggles to converge, while the fixed-point iteration diverges. In contrast, our method converges quite fast. Although smoothing the geometry (right) improves the efficiency of the Jacobi preconditioner, it remains significantly slower than our method. All precomputations for our method are included in the timings.

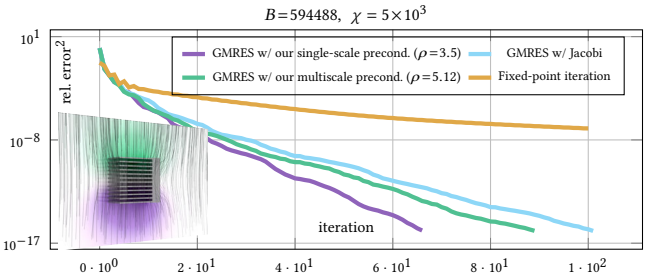


Fig. 11. **Single-scale vs. multiscale sparsity.** We tuned the value of ρ to achieve a comparable density of the inverse LU factors, with both single-scale and multiscale patterns exhibiting a density of approximately 0.0285%, albeit with slightly more fill-ins for the multiscale pattern. Nonetheless, the single-layer pattern tends to be more efficient in preconditioning the double-layer operator. Note that the fixed-point iteration completely loses its efficiency under such high susceptibility ($\chi = 5\times10^3$).

than that of a single-layer operator. Moreover, compared to the case of diffusion curves, the right-hand side vector of Eq. (13) is generally smoother: in diffusion curves, numerous discontinuities arise due to color jumps across the curves, inevitably introducing high-frequency components that challenges the solver; boundary normals are varying more smoothly in general. These properties suggest that GMRES, coupled with a simple Jacobi preconditioner, can remain reasonably effective as shown in Figs. 11 and 15.

Notice that our method loses some of its efficiency in this case: the strong singularity of the derivatives and the varying normal field both weaken the crucial screening effect on which our approach heavily relies. Case in point: our experiment demonstrates that using an identical length scale for all points to build the sparsity leads to a better convergence than the multiscale sparsity pattern for a given number of nonzero fill-ins (see Fig. 11). Furthermore, the benefit of applying the inverse preconditioner is less significant than for single-layer operators, indicating that the sparsity of its inverse has partially diminished. Nevertheless, our method always outperforms the Jacobi preconditioner. For smooth and regular meshes, the speedup may not reach orders of magnitude, but as the surface becomes more complex (*e.g.*, with many non-smooth features or multiscale thin structures), traditional methods tend to degrade substantially, while our approach demonstrates greater resilience to these geometric complexities, thus still leading to consequential performance improvements (see Fig. 10).

4.4 Mixed boundary conditions

Mixed boundary conditions are useful in various scenarios. For instance, one might pin down parts of the object boundary while applying traction to other parts when simulating an elastic body: this setup naturally leads to a mix of Dirichlet and Neumann boundary conditions [James and Pai 1999]. We thus discussed mixed boundary conditions in our BEM context next.

From representation formula to BIE. If we consider the case of Laplace's equation [Sawhney et al. 2023] for a solution that vanishes outside the domain Ω, the density functions of the single- and double-layer potentials reduce to the normal derivative of the solution and to the solution itself on the boundary respectively (Eq. (1)). Consequently, the solution within the domain can be evaluated through the representation formula:

$$u(\mathbf{x}) = -\int_\Gamma \frac{\partial G(\mathbf{x},\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}} u(\mathbf{y})\, \mathrm{d}A_\mathbf{y} + \int_\Gamma G(\mathbf{x},\mathbf{y}) \frac{\partial u(\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}}\, \mathrm{d}A_\mathbf{y}. \tag{14}$$

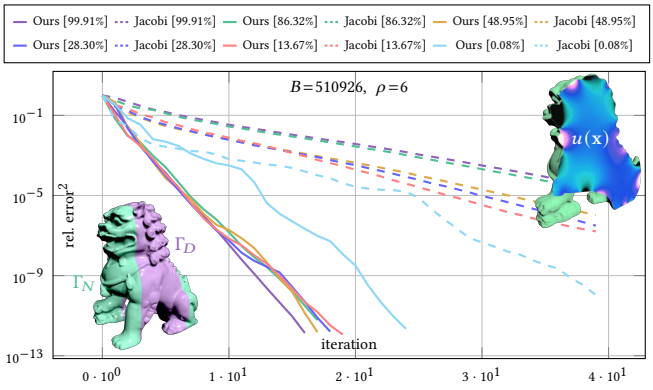


Fig. 12. **Mixed boundary condition.** As the percentage of Dirichlet boundary conditions (listed in brackets) increases, the amount of single-layer potential equations increases in the BIE matrix, and our preconditioner quickly demonstrates its efficacy even when the double-layer still dominates, *e.g.*, at more than 80%. In contrast, the Jacobi preconditioner becomes worse.

Since no additional density functions are introduced, this representation is referred to as the "direct approach" in BEM. Suppose $\Gamma_D \subset \Gamma$ is the subset of the boundary where we impose Dirichlet boundary condition $u|_{\Gamma_D} = b$, and $\Gamma_N = \Gamma \setminus \Gamma_D$ is the remaining part where to impose Neumann boundary condition $\partial u/\partial \mathbf{n}|_{\Gamma_N} = g$. The BIE amounts to ensure compatibility between boundary values and normal derivatives, by linking the unknown solution values on Γ_N and the unknown normal derivatives on Γ_D, yielding for any $\mathbf{x} \in \Gamma$,

$$\begin{aligned} &\frac{1-\chi_D(\mathbf{x})}{2} u(\mathbf{x}) + \int_{\Gamma_N} \frac{\partial G(\mathbf{x},\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}} u(\mathbf{y}) \mathrm{d}A_\mathbf{y} - \int_{\Gamma_D} G(\mathbf{x},\mathbf{y}) \frac{\partial u(\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}} \mathrm{d}A_\mathbf{y} \\ &= -\frac{\chi_D(\mathbf{x})}{2} b(\mathbf{x}) - \int_{\Gamma_D} \frac{\partial G(\mathbf{x},\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}} b(\mathbf{y}) \mathrm{d}A_\mathbf{y} + \int_{\Gamma_N} G(\mathbf{x},\mathbf{y}) g(\mathbf{y}) \mathrm{d}A_\mathbf{y}, \end{aligned} \tag{15}$$

where $\chi_D(\cdot)$ is Γ_D indicator function, *i.e.*, $\chi_D(\mathbf{x}) = 1$ if $\mathbf{x} \in \Gamma_D$ and $\chi_D(\mathbf{x}) = 0$ otherwise. After discretizing Eq. (15) with piecewise-constant functions u and $\partial u/\partial \mathbf{n}$ (assumed to be distinct discrete functions) for each centroid of the boundary elements of Γ, we get an asymmetric BIE linear system linking their coefficients, from which the final solution can be evaluated anywhere through Eq. (14).

Hybrid sparsity pattern. To address the strong screening effect of the single-layer potential and the attenuated screening effect of the double-layer potential, we adopt *a hybrid strategy that combines single-scale and multiscale patterns for the inverse factors* to precondition the BIE with mixed boundary conditions. Specifically, we first compute the reverse max-min ordering solely on the Dirichlet boundary (Γ_D), generating the sequence of degrees of freedom from i_B to $i_{B-|\Gamma_D|+1}$ as the coarse scale, characterized by their length scale l_{i_k}. Next, we put all Neumann boundary points to the sequence to the beginning as the fine scale with a uniform length scale $l_{i_{B-|\Gamma_D|+1}}$. With the ordering and associated length scales, we construct the sparsity pattern based on Eq. (7) and its supernodal extension as we discussed in Sec. 3.3.

Results and discussion. In Fig. 12, we illustrate the efficacy of our preconditioner with respect to the percentage of elements on the Dirichlet boundary (Γ_D), where $\partial u/\partial \mathbf{n}$ needs to be solved. For

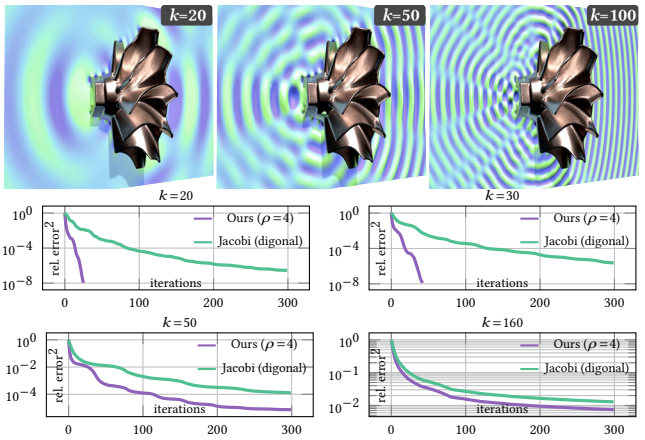


Fig. 13. **Helmholtz equation.** For the Helmholtz equation $\triangle u + k^2 u = 0$ using the single-layer BEM, a Dirichlet boundary condition (based on a fixed spherical wave function) is imposed on a turbine surface discretized with $16.2K$ triangular elements. The real part of the solution is displayed on a cross section (top). As the wavenumber k increases, our method gradually loses its superiority over the Jacobi preconditioner, due to the weakening of the screening effect caused by increasingly oscillatory kernels.

comparisons, we randomly sample a number of points $\{\mathbf{q}_i\}_i$ in the domain, and assign random values $\{\sigma_i\}_i$ to these points as charges. A harmonic function can be subsequently constructed as $f(\mathbf{x}) = \sum_i \sigma_i(\mathbf{q}_i)/\|\mathbf{x} - \mathbf{q}_i\|$. We then sample the function or its normal derivative on Γ_D and Γ_N, respectively, to initialize the mixed boundary conditions. Fig. 12 shows that our method always leads to significant speedup even when the Neumann boundary condition becomes dominant, *e.g.*, when it is used for more than 80% of the boundary. It is also worth mentioning that even in the presence of mixed boundary conditions, it is technically possible to formulate a *symmetric* system by incorporating an another hypersingular BIE [Steinbach 2007, Equation 7.19] for the Neumann boundary, alongside Eq. (15) for the Dirichlet boundary, and further discretizing the two with a symmetric Galerkin approach. In practice, however, computing the integral of hypersingular kernels and adopting FMM with the four different kernels involved is complex, whereas simply forgoing symmetry is far simpler.

4.5 Beyond Laplace's equation

While we mostly focused on Laplace's equation until now, it bears mentioning that the Green's functions for linear elasticity equations, despite being vector-valued, also scale as $O(1/r)$ in 3D, thus making our approach applicable to them as well. While this is also true for the *low-wavenumber* Helmholtz equation, this particular case introduces additional challenges. Even through a symmetric Galerkin discretization, the inverse Cholesky preconditioner of Chen et al. [2024] is not applicable, as the resulting BIE matrix $\mathbf{G}$, while symmetric, is not Hermitian. One possible workaround that the authors mentioned is to solve the system in a least-squares sense. However, this approach incurs significant memory and precomputation overheads, mostly for two reasons. First, assembling the local matrix $(\mathbf{G}^\mathsf{T}\mathbf{G})_{s_j,s_j}$ to compute the inverse factors becomes considerably more expensive: each matrix entry requires the evaluation of the

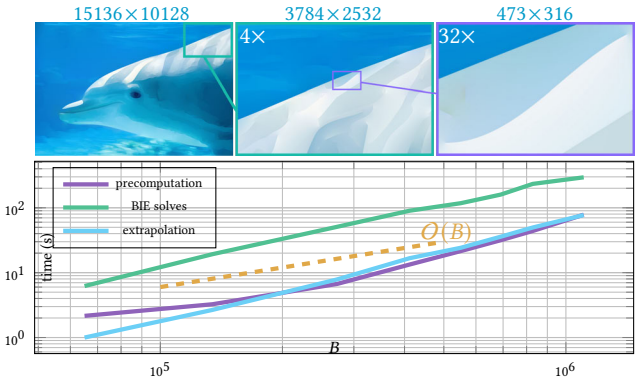


Fig. 14. **Scalability.** In this diffusion curve example, we sampled the Bézier curves at various resolutions to generate BIEs with varying sizes. The timing plot illustrates that the time required for precomputation, BIE solves (with error tolerance of 10^{-3}), and extrapolation (through Eq. (10)) all scale near-linearly with the number of boundary elements. This is achieved through the scalability of FMM and the efficacy of our preconditioner — the number of iterations only increases from 7 to 13 as B grows from 65K to 1.1M. Consequently, our method can handle very large problems. For instance, the top diffused image contains 153M pixels colored by 1.1M boundary elements. Even when zoomed in 32×, the image shows no noticeable aliasing artifacts.

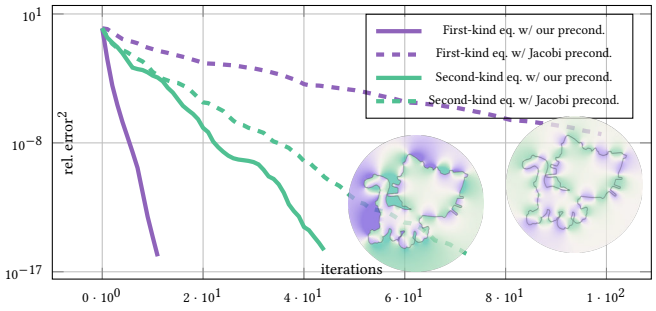


Fig. 15. **Solving single vs. double potentials.** We compare the performance of a BIE solve for a single-layer vs. a double-layer operator, using Jacobi and our inverse-LU preconditioning for the problem in Fig. 2, which of course result in the same solution within the kitten-shaped domain. The resulting error curves during GMRES solves (in log-log scale) are representative of the examples presented throughout the paper: although Fredholm equations of the first kind are quite ill-posed, our method significantly accelerates their solves, achieving orders-of-magnitude speedups; on the other hand, Fredholm equations of the second kind have inherently better spectral properties, but the effect of our preconditioning is less pronounced, resulting in slightly less efficient solves. Therefore, we recommend formulating any BEM problem with a single-layer potential whenever possible.

dense inner product $\sum_{k=1}^{B} \mathbf{G}_{ik}\mathbf{G}_{kj}$. This step can be over 100 times slower than the actual factorization plus iterative solves even on a small 10K system (see [Chen et al. 2024, Fig. 11]). Second, the iterative solver now requires *two* dense matrix-vector multiplications per iteration to compute $\mathbf{G}^{\mathsf{T}}\mathbf{G}\mathbf{x}$. So unless the symmetric preconditioner accelerates convergence by at least a factor two, these additional costs will outweigh the benefits. Such acceleration is unlikely in practice however, since the least-squares problem squares the condition number of the original matrix, further hindering convergence.

In contrast, our novel LU-based preconditioning applies directly, avoiding these difficulties and remaining effective for *low-frequency* Helmholtz problems. However, when the wavenumber k increases, the associated Green's function $G(\mathbf{x},\mathbf{y}) = \frac{\exp(ik\|\mathbf{x}-\mathbf{y}\|)}{4\pi\|\mathbf{x}-\mathbf{y}\|}$ becomes more oscillatory, gradually weakening the screening effect and degrading the performance of our method (see Fig. 13), as expected: high-frequency Helmholtz problems are notoriously difficult.

5 Conclusion

In this paper, we have formulated a lightning-fast approach to scaling up the linear solves involved in boundary element methods. In particular, we showed that the BIE matrix never needs to be fully stored in memory contrary to previous approaches such as [Chen et al. 2024] on dealing with least-squares problems, offering far improved scalability. The resulting preconditioner, based on sparse LU factors approximating the inverse of the BIE matrix, was also proven to significantly accelerate GMRES solves — from about a factor two in the worst case presented in this paper, to orders of magnitude for large examples like Fig. 1 since our solver exhibits a near-linear complexity (Fig. 14) compared to the cubic complexity of direct solvers and without steep increase in the required number of iterations as the discretization is refined.

Take-home message. Fredholm integral equations of the first kind involving a single-layer potential are often considered very ill-posed [Yuan and Zhang 2019] and thus numerically challenging. However, the efficient preconditioner for iterative solvers we propose in this paper is not only exploiting the sparsity of the inverse BIE operator, but also the screening effect of the associated PDEs, which significantly alleviates these numerical difficulties and offers an appealing numerical tool for boundary element methods. From a practitioner perspective, if a boundary problem can be formulated with a single-layer operator dominating the BIE, our approach is guaranteed to be highly beneficial. In contrast, for second-kind Fredholm integral equations, which are inherently well-conditioned, simpler methods may often suffice, especially when the boundary geometry is simple and its discretization is regular: in such cases, the performance gains of our method are typically less pronounced, as illustrated in Fig. 15. Nonetheless, when solving problems with multiple right-hand side vectors so that our precomputations can be done once and for all, the cumulative time savings achieved by our method can still be considerable.

Future work. Developing a more effective preconditioner for BIEs that are Fredholm integral equations of the second kind would be a valuable complement to our work. It would, however, require finding another property, akin to the screening effect, which could be leveraged instead. Another avenue of improvement is the problem-adaptive selection of the sparsity pattern, akin to the recent work of Huan et al. [2023]. Additionally, we believe that for rectangular systems requiring the solution of a least-squares problem, an efficient inverse preconditioner could be constructed based this time on QR factorization. The use of QR decomposition could further embrace randomized algorithms [Rokhlin and Tygert 2008] to reduce memory and computational cost while maintaining a high quality of preconditioning. Finally, improving the implementation of our preconditioner through advanced GPU-level optimization would

likely further improve our timings. Potential improvements would undoubtedly strengthen the capability of boundary-based methods for solving an increasingly diverse range of problems.

Acknowledgments

We sincerely thank the reviewers for their thoughtful and constructive feedback, and Xingze Tian for assisting with the parsing of diffusion curve files from Orzan et al. [2008] (Figs. 3, 8, 9 and 14). Meshes were provided by Scan The World (WingedVictory), Musée Rodin (Thinker), Geoffrey Marchal (Erato), Artec 3D (Turbine), and Thingi10K [Zhou and Jacobson 2016] (City). FS acknowledges support from the Office of Naval Research under award number N00014-23-1-2545 (Untangling Computation). MD benefited from the generous support of the MediTwin consortium (funded by the French government as part of *France 2030*), Ansys, and from an Inria chair.

References

Faisal Amlani, Stéphanie Chaillat, and Adrien Loseille. 2019. An efficient preconditioner for adaptive Fast Multipole accelerated Boundary Element Methods to model time-harmonic 3D wave propagation. *Comput. Methods Appl. Mech. Eng.* 352 (2019), 189–210. https://doi.org/10.1016/j.cma.2019.04.026

Seungbae Bang, Kirill Serkh, Oded Stein, and Alec Jacobson. 2023. An Adaptive Fast-Multipole-Accelerated Hybrid Boundary Integral Equation Method for Accurate Diffusion Curves. *ACM Trans. Graph.* 42, 6, Article 215 (2023). https://doi.org/10.1145/3618374

Michele Benzi. 2002. Preconditioning techniques for large linear systems: a survey. *Journal of computational Physics* 182, 2 (2002), 418–477.

Marc Bonnet, Giulio Maier, and Castrenze Polizzotto. 1998. Symmetric Galerkin Boundary Element Methods. *Applied Mechanics Reviews* 51, 11 (1998), 669.

G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

Bruno Carpentieri, Iain S Duff, Luc Giraud, and M Magolu monga Made. 2004. Sparse symmetric preconditioners for dense linear systems in electromagnetism. *Numerical linear algebra with applications* 11, 8-9 (2004), 753–771.

Bruno Carpentieri, Iain S Duff, Luc Giraud, and Guillaume Sylvand. 2005. Combining fast multipole techniques and an approximate inverse preconditioner for large electromagnetism calculations. *SIAM Journal on Scientific Computing* 27, 3 (2005), 774–792.

Jiong Chen, Florian Schäfer, Jin Huang, and Mathieu Desbrun. 2021. Multiscale Cholesky preconditioning for ill-conditioned problems. *ACM Trans. Graph. (SIGGRAPH)* 40, 4 (2021), 1–13.

Jiong Chen, Florian T Schäfer, and Mathieu Desbrun. 2024. Lightning-fast Method of Fundamental Solutions. *ACM Trans. Graph.* 43, 4, Article 77 (2024).

Yifan Chen, Houman Owhadi, and Florian Schäfer. 2023. Sparse Cholesky factorization for solving nonlinear PDEs via Gaussian processes. *arXiv:2304.01294* (2023).

Martin Costabel. 1987. Principles of boundary element methods. *Computer Physics Reports* 6, 1 (1987), 243–74.

Martin Costabel. 1988. Boundary integral operators on Lipschitz domains: elementary results. *SIAM journal on Mathematical Analysis* 19, 3 (1988), 613–626.

Fang Da, David Hahn, Christopher Batty, Chris Wojtan, and Eitan Grinspun. 2016. Surface-only liquids. *ACM Trans. Graph. (SIGGRAPH)* 35, 4 (2016), 1–12.

Fernando de Goes and Mathieu Desbrun. 2024. Stochastic Computation of Barycentric Coordinates. *ACM Transactions on Graphics* 43, 4 (2024), 13.

Fernando De Goes and Doug L James. 2017. Regularized Kelvinlets: sculpting brushes based on fundamental solutions of elasticity. *ACM Trans. Graph. (SIGGRAPH)* 36, 4 (2017), 1–11.

Graeme Fairweather and Andreas Karageorghis. 1998. The method of fundamental solutions for elliptic boundary value problems. *Advances in computational mathematics* 9 (1998), 69–95.

Massimiliano Ferronato, Carlo Janna, and Giuseppe Gambolati. 2015. A novel factorized sparse approximate inverse preconditioner with supernodes. *Procedia Computer Science* 51 (2015), 266–275.

Zydrunas Gimbutas, Leslie Greengard, Jeremy Magland, Manas Rachh, and Vladimir Rokhlin. 2024a. FMM2D: Fast multipole methods in two dimensions. Flat Iron Institute. https://github.com/flatironinstitute/FMM2D.

Zydrunas Gimbutas, Leslie Greengard, Jeremy Magland, Manas Rachh, and Vladimir Rokhlin. 2024b. FMM3D: Fast multipole methods in three dimensions. Flat Iron Institute. https://github.com/flatironinstitute/FMM3D.

Joseph Guinness. 2018. Permutation and Grouping Methods for Sharpening Gaussian Process Approximations. *Technometrics* 60, 4 (2018), 415–429.

David Hahn and Chris Wojtan. 2015. High-resolution brittle fracture simulation with boundary elements. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–12.

David Hahn and Chris Wojtan. 2016. Fast approximations for boundary element based brittle fracture simulation. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–11.

Stephen Huan, Joseph Guinness, Matthias Katzfuss, Houman Owhadi, and Florian Schäfer. 2023. Sparse Cholesky factorization by greedy conditional selection. *arXiv preprint arXiv:2307.11648* (2023).

Libo Huang and Dominik L Michels. 2020. Surface-only ferrofluids. *ACM Trans. Graph. (SIGGRAPH)* 39, 6 (2020), 1–17.

Doug L James, Jernej Barbič, and Dinesh K Pai. 2006. Precomputed acoustic transfer: output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Trans. Graph. (SIGGRAPH)* 25, 3 (2006), 987–995.

Doug L. James and Dinesh K. Pai. 1999. ArtDefo: Accurate Real Time Deformable Objects. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. 65–72. https://doi.org/10.1145/311535.311542

I. E. Kaporin. 1994. New convergence results and preconditioning strategies for the conjugate gradient method. *Numer. Linear Algebra Appl.* 1, 2 (1994), 179–210. https://doi.org/10.1002/nla.1680010208

Matthias Katzfuss and Joseph Guinness. 2021. A General Framework for Vecchia Approximations of Gaussian Processes. *Statist. Sci.* 36, 1 (2021), 124 – 141. https://doi.org/10.1214/19-STS755

Liliya Yurievna Kolotilina. 1988. Explicit preconditioning of systems of linear algebraic equations with dense matrices. *Journal of Soviet Mathematics* 43 (1988), 2566–2573.

Liliya Yurievna Kolotilina and Aleksey Yuryevich Yeremin. 1993. Factorized sparse approximate inverse preconditionings I. Theory. *SIAM J. Matrix Anal. Appl.* 14, 1 (1993), 45–58.

Ronald Kriemann. 2013. $\mathcal{H}$-LU factorization on many-core systems. *Comput. Visual Sci.* 16 (2013), 105–117. https://doi.org/10.1007/s00791-014-0226-7

Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. 2023. Boundary Value Caching for Walk on Spheres. *ACM Trans. Graph.* 42, 4, Article 82 (July 2023), 11 pages. https://doi.org/10.1145/3592400

Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. 2024. Walkin'robin: Walk on stars with robin boundary conditions. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–18.

Xingyu Ni, Ruicheng Wang, Bin Wang, and Baoquan Chen. 2024. An Induce-on-Boundary Magnetostatic Solver for Grid-Based Ferrofluids. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–14.

Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. 2008. Diffusion curves: a vector representation for smooth-shaded images. *ACM Trans. Graph. (SIGGRAPH)* 27, 3 (2008), 1–8.

Constantine Pozrikidis. 2002. *A practical guide to boundary element methods with the software library BEMLIB*. CRC Press.

Vladimir Rokhlin and Mark Tygert. 2008. A fast randomized algorithm for overdetermined linear least-squares regression. *Proceedings of the National Academy of Sciences* 105, 36 (2008), 13212–13217.

Stefan A Sauter, Christoph Schwab, Stefan A Sauter, and Christoph Schwab. 2011. *Boundary element methods*. Springer.

Rohan Sawhney and Keenan Crane. 2020. Monte Carlo geometry processing: A grid-free approach to PDE-based methods on volumetric domains. *ACM Trans. Graph. (SIGGRAPH)* 39, 4 (2020).

Rohan Sawhney, Bailey Miller, Ioannis Gkioulekas, and Keenan Crane. 2023. Walk on Stars: A Grid-Free Monte Carlo Method for PDEs with Neumann Boundary Conditions. *ACM Trans. Graph.* 42, 4, Article 80 (July 2023), 20 pages. https://doi.org/10.1145/3592398

Florian Schäfer, Matthias Katzfuss, and Houman Owhadi. 2021a. Sparse Cholesky Factorization by Kullback–Leibler Minimization. *SIAM J. Sci. Comput* 43, 3 (2021), A2019–A2046. https://doi.org/10.1137/20M1336254

Florian Schäfer, Timothy John Sullivan, and Houman Owhadi. 2021b. Compression, inversion, and approximate PCA of dense kernel matrices at near-linear computational complexity. *Multiscale Model. Simul.* 19, 2 (2021), 688–730.

H. Schippers. 1985. Multigrid methods for boundary integral equations. *Numer. Math.*, 46 (1985), 351–363. https://doi.org/10.1007/BF01389491

Camille Schreck, Christian Hafner, and Chris Wojtan. 2019. Fundamental solutions for water wave animation. *ACM Trans. Graph. (SIGGRAPH)* 38, 4 (2019), 1–14.

Justin Solomon, Amir Vaxman, and David Bommes. 2017. Boundary element octahedral fields in volumes. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1.

Michael L. Stein. 2002. The screening effect in kriging. *The Annals of Statistics* 30, 1 (2002), 298–323.

Michael L. Stein. 2011. 2010 Rietz Lecture: When does the screening effect hold? *The Annals of Statistics* 39, 6 (2011), 2795–2819. http://www.jstor.org/stable/41713599

Michael L. Stein, Zhiyi Chi, and Leah J Welty. 2004. Approximating likelihoods for large spatial data sets. *J. R. Stat. Soc., B: Stat. Methodol.* 66, 2 (2004), 275–296.

Olaf Steinbach. 2007. *Numerical approximation methods for elliptic boundary value problems: finite and boundary elements*. Springer Science & Business Media.

Olaf Steinbach and Wolfgang L. Wendland. 1998. The construction of some efficient preconditioners in the boundary element method. *Adv. Comput. Math.* 9 (1998),

191–216.
Olaf Steinbach and Wolfgang L. Wendland. 2001. On C. Neumann's Method for Second-Order Elliptic Systems in Domains with Non-smooth Boundaries. *J. Math. Anal. Appl.* 262, 2 (2001), 733–748. https://doi.org/10.1006/jmaa.2001.7615
Gilbert Strang and George Fix. 2008. *An Analysis of the Finite Element Methods, New Edition.* Wellesley-Cambridge Press, Philadelphia, PA. https://doi.org/10.1137/1.9780980232707 arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9780980232707
Ryusuke Sugimoto, Christopher Batty, and Toshiya Hachisuka. 2022. Surface-Only Dynamic Deformables using a Boundary Element Method. In *Comput. Graph. Forum*, Vol. 41. 75–86.
Ryusuke Sugimoto, Terry Chen, Yiti Jiang, Christopher Batty, and Toshiya Hachisuka. 2023. A Practical Walk-on-Boundary Method for Boundary Value Problems. *ACM Trans. Graph.* 42, 4, Article 81 (2023). https://doi.org/10.1145/3592109
Xin Sun, Guofu Xie, Yue Dong, Stephen Lin, Weiwei Xu, Wencheng Wang, Xin Tong, and Baining Guo. 2012. Diffusion curve textures for resolution-independent texture mapping. *ACM Trans. Graph. (SIGGRAPH)* 31, 4 (2012), 1–9.
Nobuyuki Umetani, Athina Panotopoulou, Ryan Schmidt, and Emily Whiting. 2016. Printone: interactive resonance simulation for free-form print-wind instrument design. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–14.
JJ van de Gronde. 2010. *A high quality solver for diffusion curves.* Ph. D. Dissertation. Faculty of Science and Engineering.
Stephen A Vavasis. 1992. Preconditioning for boundary integral equations. *SIAM journal on matrix analysis and applications* 13, 3 (1992), 905–925.
Aldo V Vecchia. 1988. Estimation and model identification for continuous spatial processes. *J. R. Stat. Soc., B: Stat. Methodol.* 50, 2 (1988), 297–312. https://doi.org/10.1111/j.2517-6161.1988.tb01729.x
Mengqi Xia, Bruce Walter, Eric Michielssen, David Bindel, and Steve Marschner. 2020. A wave optics based fiber scattering model. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–16.
A Yu Yeremin and AA Nikishin. 2004. Factorized-sparse-approximate-inverse preconditionings of linear systems with unsymmetric matrices. *Journal of Mathematical Sciences* 121 (2004), 2448–2457.
Di Yuan and Xinming Zhang. 2019. An overview of numerical methods for the first kind Fredholm integral equation. *SN Applied Sciences* 1 (2019), 1–12.
Deyun Zhong, Ju Zhang, and Liguan Wang. 2019. Fast Implicit Surface Reconstruction for the Radial Basis Functions Interpolant. *App. Sci.* 24, 9 (2019), 5335–5349. https://doi.org/10.3390/app9245335
Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797* (2016).

A Computing singular integrals

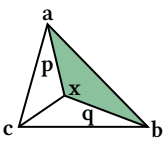
When an evaluation point $\mathbf{x}$ lies on one of the boundary elements, the integrals of the Green's functions and their normal derivatives become both singular. To ensure better accuracy, we employ analytical evaluation instead of resorting to numerical quadrature rules for these singular integrals. To begin with, the normal derivative of the Green's function is strongly singular, and its integral exists only in the sense of the Cauchy principal value. For piecewise constant elements, this integral vanishes as the normal vector is always orthogonal to the tangent vector $\mathbf{x}-\mathbf{y}$ [Pozrikidis 2002, Sec. 5.2.3]. Consequently, the discrete double-layer operator has a zero diagonal. In contrast, the case of the Green's function is more involved. Since it is weakly singular, its integral exists in the usual sense. In the following, we detail the calculation of the singular integral of the Green's function of the Laplace operator in 2D and 3D respectively — most other operators can be handled in a similar fashion.

Over a 2D line segment. In 2D, the Green's function of the Laplace operator is given by $-\log(\|\mathbf{x}-\mathbf{y}\|)/2\pi$. For a point $\mathbf{x}$ on a boundary element $(\mathbf{a},\mathbf{b})$, the integral can be decomposed into two parts: the integration I_1 over $(\mathbf{a},\mathbf{x})$ and I_2 over $(\mathbf{x},\mathbf{b})$. Through the parameterization $\mathbf{y}=\mathbf{x}+t\mathbf{p}$ for $t\in(0,1)$ and $\mathbf{p}=\mathbf{b}-\mathbf{x}$, the integral over $(\mathbf{x},\mathbf{b})$ is easily evaluated through

$$I_2=\int_{\mathbf{x}}^{\mathbf{b}}\log(\|\mathbf{x}-\mathbf{y}\|)\,\mathrm{d}t=\|\mathbf{p}\|\int_0^1\log(\|\mathbf{p}\|t)\,\mathrm{d}t=\|\mathbf{p}\|(\log(\|\mathbf{p}\|)-1)).$$

The other integral I_1 is dealt with in a similar fashion.

Over a 3D triangle. The 3D Laplace Green's function is $1/4\pi r$. For a triangular element $\triangle_{\mathbf{abc}}$ and an evaluation point $\mathbf{x}$ located on this triangle, we similarly decompose the integral over $\triangle_{\mathbf{abc}}$ into the sum of the three integrals I_1, I_2, I_3 over the domains $\triangle_{\mathbf{xab}}$, $\triangle_{\mathbf{xbc}}$, and $\triangle_{\mathbf{xca}}$ respectively, formed by the evaluation point and the three triangle vertices. Let us focus on the computation of one of the three subdomains, as one can proceed similarly for each of them. For $\triangle_{\mathbf{xab}}$, we define $\mathbf{p}=\mathbf{a}-\mathbf{x}$ and $\mathbf{q}=\mathbf{b}-\mathbf{x}$ (see inset). A point $\mathbf{y}$ within $\triangle_{\mathbf{xab}}$ is parameterized using barycentric coordinates (α,β) as $\mathbf{y}=\mathbf{x}+\alpha\mathbf{p}+\beta\mathbf{q}$ where $\alpha,\beta\geq 0$ and $\alpha+\beta\leq 1$. The integral over $\triangle_{\mathbf{xab}}$ is then evaluated as:



$$\begin{aligned}I_1&=\int_{\triangle_{\mathbf{xab}}}\frac{1}{\|\mathbf{x}-\mathbf{y}\|}\,\mathrm{d}A_y=2|\triangle_{\mathbf{xab}}|\int_0^1\mathrm{d}\alpha\int_0^{1-\alpha}\mathrm{d}\beta\,\frac{1}{\|\alpha\mathbf{p}+\beta\mathbf{q}\|}\\&=2|\triangle_{\mathbf{xab}}|\int_0^1\mathrm{d}\alpha\int_0^{1-\alpha}\mathrm{d}\beta\frac{1}{\sqrt{A\alpha^2+B\alpha\beta+C\beta^2}},\end{aligned}$$

where the constant A, B, C are defined as $\mathbf{p}^\mathsf{T}\mathbf{p}$, $2\mathbf{p}^\mathsf{T}\mathbf{q}$ and $\mathbf{q}^\mathsf{T}\mathbf{q}$ respectively. Now we can compute the above integral in polar coordinates, using the transformation $\alpha=\rho\cos(\theta)$ and $\beta=\rho\sin(\theta)$, where $\rho\geq 0$ and $\theta\in[0,\pi/2]$. The area element transforms as $\mathrm{d}A=\rho\mathrm{d}\rho\mathrm{d}\theta$. Substituting these terms into the integral yields:

$$I_1=2|\triangle_{\mathbf{xab}}|\int_0^{\frac{\pi}{2}}\frac{\mathrm{d}\theta}{\sqrt{A\cos^2(\theta)+B\sin(\theta)\cos(\theta)+C\sin^2(\theta)}}\int_0^{R(\theta)}\mathrm{d}\rho,$$

where the integration radius $R(\theta)$ along the direction θ is given as $R(\theta)=1/(\sin(\theta)+\cos(\theta))$. Finally, we reach the expression:

$$\begin{aligned}I_1&=\frac{2|\triangle_{\mathbf{xab}}|\left(\tanh^{-1}\left(\frac{2C-B}{2\sqrt{C}\sqrt{A-B+C}}\right)-\tanh^{-1}\left(\frac{B-2A}{2\sqrt{A}\sqrt{A-B+C}}\right)\right)}{\sqrt{A-B+C}}\\&=\frac{2|\triangle_{\mathbf{xab}}|\left(\tanh^{-1}\left(\frac{\mathbf{q}\cdot(\mathbf{p}-\mathbf{q})}{\|\mathbf{q}\|\|\mathbf{p}-\mathbf{q}\|}\right)-\tanh^{-1}\left(\frac{\mathbf{p}\cdot(\mathbf{q}-\mathbf{p})}{\|\mathbf{p}\|\|\mathbf{q}-\mathbf{p}\|}\right)\right)}{\|\mathbf{p}-\mathbf{q}\|}.\end{aligned}$$

B A variational definition of Eq. (2)

It can be easily verified (by forming the Lagrangian and deriving its optimality conditions through variations of $\mathbf{U}_\mathcal{S}$ and $\mathbf{L}_\mathcal{S}$) that our definition of the inverse LU factors given by Eq. (2) is the optimal solution of an extremization problem:

$$\begin{aligned}&\text{extremize }\ \mathrm{Tr}\big((\mathbf{I}-\mathbf{U}_\mathcal{S}\overline{\mathbf{U}})(\mathbf{I}-\overline{\mathbf{L}}\mathbf{L}_\mathcal{S})\big)\\&\text{s.t. }\ \mathrm{diag}(\mathbf{U}_\mathcal{S}\mathbf{G}\mathbf{L}_\mathcal{S})=\mathbb{1}\text{ and }\mathrm{diag}(\mathbf{L}_\mathcal{S})=\mathbb{1},\end{aligned}$$

where $\overline{\mathbf{U}}$ and $\overline{\mathbf{L}}$ represent the global UL factors of $\mathbf{G}$ such that $\mathbf{G}=\overline{\mathbf{U}}\overline{\mathbf{L}}$. Note that if we remove the unit diagonal constraint for $\mathbf{L}$, then we get an expression exactly equivalent to the Kaporin's solution for symmetric matrices $\mathbf{G}$. However, for asymmetric systems, this simplified form of the variational problem would require access to the diagonal terms $\{\overline{\mathbf{U}}_{j,j},\overline{\mathbf{L}}_{j,j}\}_j$ of global inverse $\mathbf{G}^{-1}$ and would also involve square roots which may breakdown factorization due to the presence of negative values. We thus add this constraint in our case to bypass these two issues.