GPU Optimizations for the Hierarchical Poincaré-Steklov Scheme

Anna Yesypenko and Per-Gunnar Martinsson

1 Introduction

We describe methods for solving boundary value problems of the form

$$\begin{cases} \mathcal{A}u(x) = f(x), & x \in \Omega, \\ u(x) = g(x), & x \in \partial\Omega, \end{cases} \tag{1}$$

where $\mathcal{A}$ is a second order elliptic differential operator, and Ω is domain in two dimensions with boundary $\partial\Omega$. For the sake of concreteness, we will focus on the case where $\mathcal{A}$ is a variable coefficient Helmholtz operator

$$\mathcal{A}u(x) = -\Delta u(x) - \kappa^2 b(x)u(x), \tag{2}$$

where κ is a reference wavenumber, and where $b(x)$ is a smooth non-negative function that typically satisfies $0 \le b(x) \le 1$. Upon discretizing (1), one obtains a linear system

$$\mathbf{Au} = \mathbf{f} \tag{3}$$

involving a sparse coefficient matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$. The focus of this work is on efficiently solving the sparse system (3) for the Hierarchical Poincaré-Steklov (HPS) discretization. HPS is a multi-domain spectral collocation scheme that allows for relatively high choices of p, while interfacing well with sparse direct solvers. For (1) discretized with HPS with local polynomial order p, the cost of factorizing $\mathbf{A}$ directly is

$$T_{\text{build}} = O\left(\underbrace{p^4 N}_{\text{leaf operations}} + \underbrace{N^{3/2}}_{\text{direct solver}} \right). \tag{4}$$

Anna Yesypenko, Per-Gunnar Martinsson
Oden Institute, e-mail: annayesy@utexas.edu, pgm@oden.utexas.edu

After the leaf operations are complete, the cost to factorize the system directly has no pre-factor dependence on p. The pre-factor cost of the leaf operations, however, has long been viewed as prohibitively expensive. This manuscript describes simple GPU optimizations using batched linear algebra that substantially accelerate the leaf operations and shows compelling results for p up to 42. We also demonstrate that the choice of p does not have substantial effects on the build time for the direct factorization stage, allowing p to be chosen based on physical considerations instead of practical concerns.

High order discretization is crucial in resolving variable-coefficient scattering phenomena due to the well known “pollution effect” that generally requires the number of points per wavelength to increase, the larger the computational domain is. The pollution effect is very strong for low order discretizations, but quickly gets less problematic as the discretization order increases [2, 7]. HPS is less sensitive to pollution because the scheme allows for high choices of local polynomial order p [11, 16]. Combining HPS discretization with efficient sparse direct solvers provides a powerful tool for resolving challenging scattering phenomena to high accuracy, especially for situations where no efficient preconditioners are known to exist (e.g. trapped rays, multiple reflections, backscattering) [9].

2 HPS Discretization and interfacing with sparse direct solvers

We next discuss the HPS discretization and efficient methods to interface the resulting sparse linear system with direct solvers. We introduce the HPS briefly for the simple model problem (1), and refer the reader to [1, 17, 13] for details and extensions. An important limitation of the discretization is that we assume the solution is smooth and that the coefficients in the operator $\mathcal{A}$ of (1) are smooth as well.

The domain Ω is partitioned into non-overlapping subdomains. The discretization is described by two parameters, a and p, which are the element size and local polynomial order, respectively. On each subdomain, we place a $p \times p$ tensor product mesh of Chebyshev points. Internal to each subdomain, the PDE is enforced locally via spectral differentiation and direct collocation. On element boundaries, we enforce that the flux between adjacent boundaries is continuous. On each subdomain of p^2 nodes, the spectral differentiation operators lead to a dense matrix of interactions of size $p^2 \times p^2$. To improve efficiency of sparse direct solvers for HPS discretizations, we “eliminate” the dense interactions of nodes interior to each subdomain. This process is referred to as “static condensation” [4, 12]. The remaining active nodes are on the boundaries between subdomains. As a result of the leaf elimination, we produce a smaller system

$$\tilde{\mathbf{A}}\tilde{\mathbf{u}} = \tilde{\mathbf{f}}. \tag{5}$$

of size $\approx N/p$ with equivalent body load $\tilde{\mathbf{f}}$ on the active nodes located on the boundaries between subdomains, as shown in Figure 2.

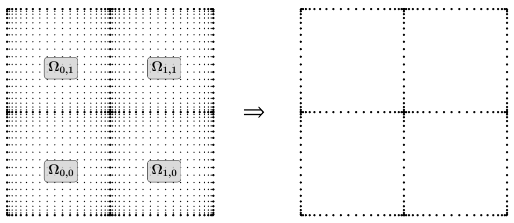


Fig. 1 Prior to interfacing with sparse direct solvers, we do static condensation to produce an equivalent system (5) to solve on the remaining active nodes. The original grid has N points, and remaining grid has $\approx N/p$ points.

Due to the domain decomposition used in HPS, the leaf operations required to produce the equivalent system (5) can be done embarrassingly in parallel. The leaf operations require independent dense linear algebraic operations (e.g., LU factorization, matrix-matrix multiply) on N/p^2 systems, each of size $p^2 \times p^2$, resulting in an overall cost of $O(p^4 N)$. For p up to about 42, these operations can be efficiently parallelized with batched linear algebra (BLAS). However, for larger p, methods that produce a sparser equivalent system may be more appropriate [3, 10].

Overhead costs can make achieving high arithmetic intensity for many small parallel tasks a challenge. However, batched BLAS offers a solution. It is highly optimized software for parallel operations on matrices that are small enough to fit in the top levels of the memory hierarchy (i.e., smaller than the L2 cache size) [8]. The framework groups small inputs into larger "batches" to automatically achieve good parallel performance on high-throughput architectures such as GPUs.

The technique we present is most readily applicable to the case where the same discretization order p is used on every discretization patch. However, it would not be too difficult to allow p to be chosen from a fixed set of values (say $p \in \{6, 10, 18, 36\}$ or something similar). This would enable many of the advantages of hp-adaptivity, while still enabling batching to accelerate computations.

Remark 1 Since the leaf computations are *very* efficient, we saved memory and reduced communication by not explicitly storing the factorizations of the local spectral differentiation matrices. Instead, these are reformed and refactored after each solve involving the reduced system (5).

We combined the fast leaf factorization procedure with two methods for solving the reduced system (5). The first option for solving (5) uses a black-box sparse direct solver with the nested dissection (ND) ordering. ND is a based on a multi-level graph partitioning of nodes and produces a sparse factorization with minimal fill-in [5, 6]. In 2D, sparse factorization using the ND ordering requires $O\left(N^{3/2}\right)$ time to build and $O\left(N \log N\right)$ time to apply.

As a second option for solving (5), we used a scheme we refer to as SlabLU, which is a simplified two-level scheme (as opposed to standard hierarchical schemes) that is designed for ease of parallelization [18]. To be precise, SlabLU uses a decomposition of the domain into elongated “slab” subdomains, as shown in Figure 2. With this decomposition, the linear system (5) has the block form

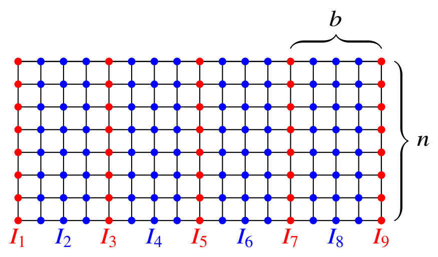


Fig. 2 Domain decomposition used in SlabLU. The even-numbered nodes correspond to the nodes interior to each subdomain. The odd-numbered nodes correspond to interfaces between slabs. The slab partitioning is chosen so that interactions between slab interiors are zero. The slabs have width of b points.

$$\begin{bmatrix} \tilde{\mathbf{A}}_{11} & \tilde{\mathbf{A}}_{12} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \tilde{\mathbf{A}}_{21} & \tilde{\mathbf{A}}_{22} & \tilde{\mathbf{A}}_{23} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \tilde{\mathbf{A}}_{32} & \tilde{\mathbf{A}}_{33} & \tilde{\mathbf{A}}_{34} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}}_{43} & \tilde{\mathbf{A}}_{44} & \tilde{\mathbf{A}}_{45} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{u}}_1 \\ \tilde{\mathbf{u}}_2 \\ \tilde{\mathbf{u}}_3 \\ \tilde{\mathbf{u}}_4 \\ \vdots \end{bmatrix} = \begin{bmatrix} \tilde{f}_1 \\ \tilde{\mathbf{f}}_2 \\ \tilde{f}_3 \\ \tilde{f}_4 \\ \vdots \end{bmatrix}. \tag{6}$$

The nodes internal to each slab are eliminated by computing sparse factorizations of the diagonal blocks $\tilde{\mathbf{A}}_{22}, \tilde{\mathbf{A}}_{44}, \ldots$ in parallel. This results in another block tridiagonal coefficient matrix $\mathbf{T}$ that has much smaller blocks than $\tilde{\mathbf{A}}$ (and half as many). The blocks of $\mathbf{T}$ are dense, but can be represented efficiently using data sparse formats such as the $\mathcal{H}$-matrix format of Hackbusch. The corresponding $\mathcal{H}$-matrices have significantly low ranks, due to the thinness of the slabs. The blocks of $\mathbf{T}$ can be rapidly constructed in $\mathcal{H}$-matrix format using the black-box randomized compression techniques described in [14].

The reduced linear system with blocks having $\mathcal{H}$-matrix structure can in principle be solved efficiently using rank-structured linear algebra. However, we found that for 2D problems, it is most efficient to relinquish the rank structure and simply convert all blocks to a dense format before factorizing the block tridiagonal system. (In 3D, this simplistic approach is possible only for small problems.) With a choice of slab width b that grows slowly with the problem size as $b \sim n^{2/3}$, the resulting two-level scheme has complexity $O(N^{5/3})$ to factorize $\tilde{\mathbf{A}}$ directly and $O\left(N^{7/6}\right)$ complexity to apply the computed factors to solve (5). SlabLU is simple scheme that leverages high concurrency and batched BLAS to achieve high performance on modern hybrid architectures. Despite the asymptotically higher costs, SlabLU performs favorably compared to multi-level nested dissection schemes in its build time and memory footprint, as we demonstrate in Section 3. [18] provides details on SlabLU.

3 Numerical experiments

We demonstrate the effectiveness of the HPS discretization combined with sparse direct solvers in solving high-frequency Helmholtz equations. The experiments were conducted on a desktop computer equipped with a 16-core Intel i9-12900k CPU and 128GB of memory, and a NVIDIA RTX 3090 GPU with 24GB of memory.

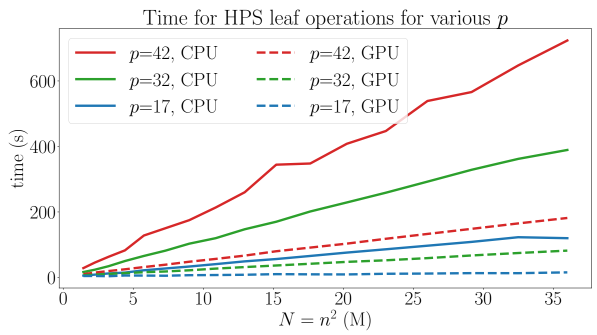


Fig. 3 Leaf operations for HPS require $O(p^4 N)$ operations, though the practical scaling for parallel operations has a small constant prefactor for p up to 42. Parallel HPS leaf operations are further accelerated on GPUs, with a speed-up of at least 4x.

We show that GPU optimizations enable efficient leaf operations for various local polynomial orders, cf. Figure 3. After the leaf operations, we directly factorize the reduced system (5) using efficient sparse direct solvers. We demonstrate that the choice of p does not significantly affect the time to factorize $\tilde{\mathbf{A}}$. Having the freedom to choose p allows the user to resolve highly oscillatory PDEs to high-order accuracy without worrying about how the choice may affect the cost of solving (3) directly.

To demonstrate the effectiveness of the HPS discretization resolving oscillatory solutions to high accuracy, we report results for a PDE with a known analytic solution

$$\begin{cases} -\Delta u(x) - \kappa^2 u(x) = 0, & x \in \Omega = [0,1]^2, \\ u(x) = u_{\text{true}}(x), & x \in \partial\Omega, \end{cases} \tag{7}$$

The true solution u_{true} is given by $u_{\text{true}} = J_0\left(\kappa |x - (-0.1, 0.5)|\right)$, where $x \mapsto J_0(\kappa|x|)$ is the free-space fundamental solution to the Helmholtz equation. We discretize (7) using HPS for various choices of p and set the wavenumber κ to increase with N to maintain 10 points per wavelength with increasing problem size. After applying a direct solver to solve (5) on the reduced HPS grid, we re-factorize the linear systems on interior leaf nodes to calculate the solution $\mathbf{u}_{\text{calc}}$ on the full HPS grid. The leaf solve requires time $O(p^4 N)$ but is particularly efficient using the GPU optimizations described. The reported build times and solve times include the leaf operations. We report the relative error with respect to the residual of the discretized system (3). When a true solution is known, we also report the relative error with respect to the true solution $\mathbf{u}_{\text{true}}$ evaluated on the collocation points of the full HPS grid

$$\text{relerr}_{\text{res}} = \frac{\|\mathbf{A}\mathbf{u}_{\text{calc}} - \mathbf{f}\|_2}{\|\mathbf{f}\|_2}, \qquad \text{relerr}_{\text{true}} = \frac{\|\mathbf{u}_{\text{calc}} - \mathbf{u}_{\text{true}}\|_2}{\|\mathbf{u}_{\text{true}}\|_2}. \tag{8}$$

3.1 Comparison of sparse direct solvers

The system (5) is solved using two different sparse direct solvers, SuperLU and SlabLU. SuperLU is a black-box solver that finds an appropriate ordering of the system to minimize fill-in while increasing concurrency by grouping nodes into super-nodes [15]. We accessed SuperLU through the Scipy interface (version 1.8.1) and called it with the COLAMD ordering. This version of Scipy uses the CPU only. Not many GPU-aware sparse direct solvers are widely available, though this is an active area of research. SuperLU uses a pivoting scheme that can exchange rows between super-nodes to attain almost machine precision accuracy in the residual of the computed solutions.

SlabLU, on the other hand, uses an ordering based on a decomposition of the domain into slabs that has a limited pivoting scheme. Despite this limitation, SlabLU can achieve 10 digits of accuracy in the residual, which also gives high-order true relative accuracy, depending on the choice of p. SlabLU is a simple two-level framework that achieves large speedups over SuperLU by leveraging batched BLAS and GPU optimizations. Figure 4 provides a comparison between SuperLU and SlabLU in factorizing $\tilde{\mathbf{A}}$ to solve (5). The figure shows that SlabLU is faster to build and requires a smaller memory footprint than SuperLU, where the memory footprint refers to how much main memory is required to store the sparse factorization of $\tilde{\mathbf{A}}$. Figure 5 presents a comparison of the accuracy of computed solutions when using SlabLU and SuperLU.

Next we demonstrate the ability of HPS, combined with SlabLU, for various p to solve Helmholtz problems of size up to $900\lambda \times 900\lambda$ (for which N=81M) to high-order accuracy. Figure 6 reports build and solve times for various choices of p, and Figure 7 reports the accuracy of the calculated solutions.

3.2 Convergence for scattering problems for various *p*

We will now demonstrate the ability of HPS, combined with SlabLU as a sparse direct solver, to solve complex scattering phenomena on various 2D domains. For the presented PDEs, we will show how the accuracy of the calculated solution converges to a reference solution depending on the choice of p in the discretization. Specifically, we will solve the BVP (1) with the variable-coefficient Helmholtz operator (2) for various Dirichlet data on smooth and rectangular domains.

We fix the PDE and refining the mesh to compare calculated solutions to a reference solution obtained on a fine mesh with high p, as the exact solution is unknown. The relative error is calculated by comparing $\mathbf{u}_{\text{calc}}$ to the reference solution $\mathbf{u}_{\text{ref}}$ at a small number of collocation points $\{x_j\}_{j=1}^M$ using the l_2 norm

$$\text{relerr}_{\text{approx}} = \frac{\|\mathbf{u}_{\text{calc}} - \mathbf{u}_{\text{ref}}\|_2}{\|\mathbf{u}_{\text{ref}}\|_2}. \tag{9}$$

We demonstrate the convergence on a unit square domain $\Omega = [0,1]^2$ with a variable coefficient field b_{crystal} corresponding to a photonic crystal, shown in Figure 8. The convergence plot is presented in Figure 9.

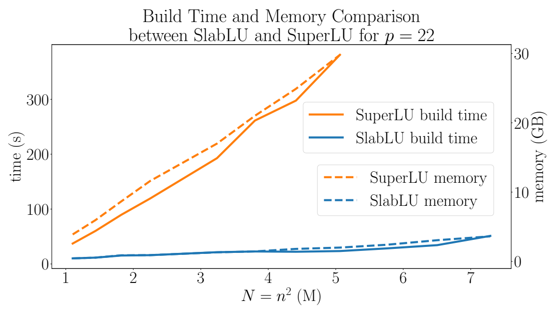


Fig. 4 Build time and memory footprint comparison between SuperLU and SlabLU for (7) discretized with HPS (p=22), where κ is increased to maintain 10 points per wavelength. For N=5.06M, the SlabLU build time is faster by a factor of 16x and the memory footprint is less by a factor of 14x.

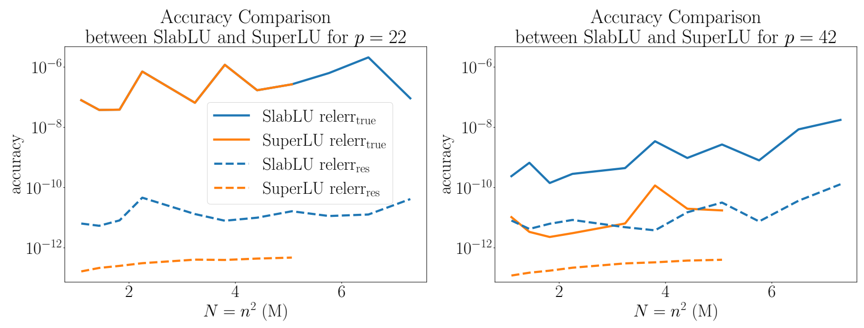


Fig. 5 Accuracy comparison between SuperLU and SlabLU for (7) discretized with HPS (p=22,42), where κ is increased to maintain 10 points per wavelength. SuperLU uses a more sophisticated pivoting scheme to achieve high accuracy in the residual error. For p=22, SlabLU and SuperLU both resolve the solution to 6 digits in the true relative accuracy. For p=42, SuperLU is able to achieve 10 digits in the true relative accuracy, while SlabLU achieves 8 digits.

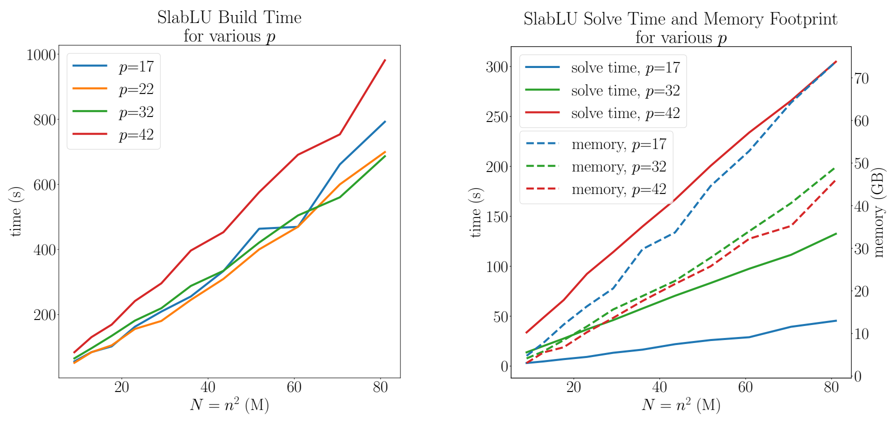


Fig. 6 Build time and solve time for HPS with various p for (7) where κ is increased with N to maintain 10 points per wavelength. The choice of p does not substantially affect the time needed to factorize the sparse linear system with SlabLU. As p increases, the memory footprint required to store the factorization decreases and the solve time increases.

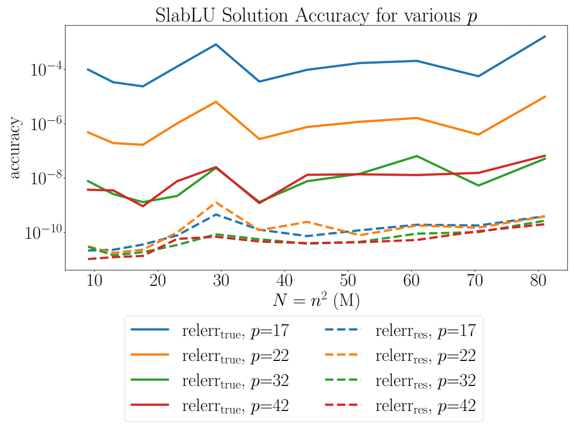
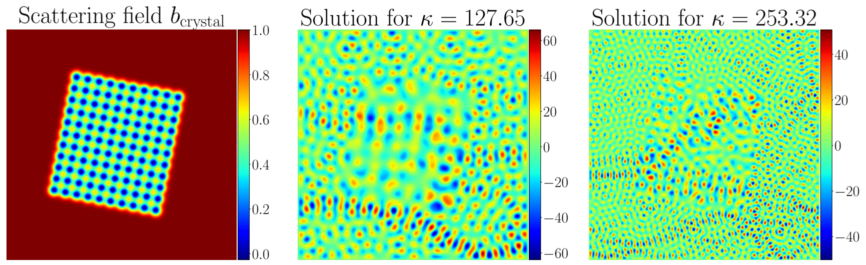


Fig. 7 Solution accuracy for HPS with various p for (7) where κ is increased with N to maintain 10 points per wavelength. Regardless of the choice of p, SlabLU resolves the solution to at least 10 digits of relative accuracy in the residual. With increasing p, one can calculate solutions with higher relative accuracy, compared to the true solution of the PDE.



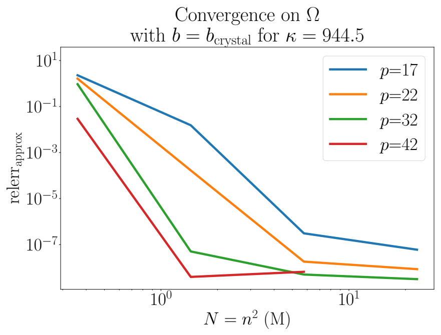


Fig. 8 Solutions of variable-coefficient Helmholtz problem on square domain Ω with Dirichlet data given by $u \equiv 1$ on $\partial\Omega$ for various wavenumbers κ. The scattering field is b_{crystal}, which is a photonic crystal that blocks the wave from propagating.

Fig. 9 Convergence on square domain Ω for reference solution $\mathbf{u}_{\text{ref}}$ on HPS discretization for N=36M with $p = 42$.

Next, we show the convergence on a curved domain Ψ with a constant-coefficient field $b \equiv 1$, where Ψ is given by an analytic parameterization over a reference square $\Omega = [0, 1]^2$. The domain Ψ is parametrized as

$$\Psi = \left\{ \left(x_1, \frac{x_2}{\psi(x_1)} \right) \text{ for } (x_1, x_2) \in \Omega = [0,1]^2 \right\}, \text{ where } \psi(z) = 1 - \frac{1}{4}\sin(z). \quad (10)$$

Using the chain rule, (2) on Ψ takes the following form on Ω

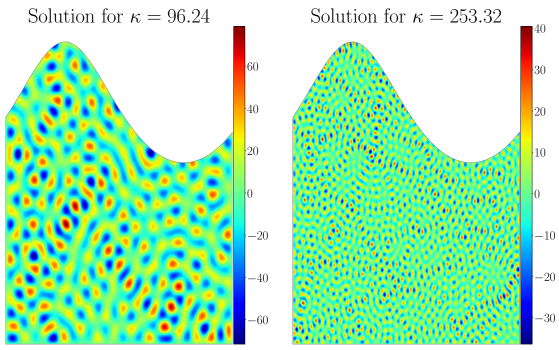


Fig. 10 Solutions of constant-coefficient Helmholtz problem on curved domain Ψ with Dirichlet data given by $u \equiv 1$ on $\partial\Psi$ for various wavenumbers κ. The solutions are calculated parametrizing Ψ in terms of a reference square domain Ω as (10) and solving (11) on Ω.

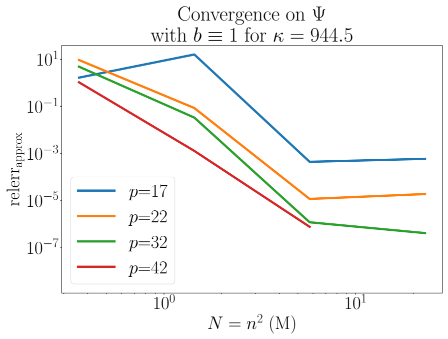


Fig. 11 Convergence on curved domain Ψ for reference solution $\mathbf{u}_{\text{ref}}$ on HPS discretization for N=36M with $p = 42$. The solutions exhibit mildly singular behavior near the corners, and choosing high orders of p aids in the convergence.

$$\begin{aligned}&-\frac{\partial^2 u}{\partial x_1^2} - 2\frac{\psi'(x_1)x_2}{\psi(x_1)}\frac{\partial^2 u}{\partial x_1 \partial x_2} - \left(\left(\frac{\psi'(x_1)x_2}{\psi(x_1)}\right)^2 + \psi(x_1)^2\right)\frac{\partial^2 u}{\partial x_2^2} \\ &\qquad - \frac{\psi''(x_1)x_2}{\psi(x_1)}\frac{\partial u}{\partial x_2} - \kappa^2 u = 0, \qquad (x_1, x_2) \in \Omega.\end{aligned} \tag{11}$$

The solutions on Ψ are shown in Figure 10, and the convergence plot is presented in Figure 11.

4 Conclusions

HPS is a high-order convergent discretization scheme that interfaces well with sparse direct solvers. In this manuscript, we describe GPU optimizations of the scheme that enable rapid and memory-efficient direct solutions of (3) for resulting linear systems. First, we perform the leaf operations in parallel using batched BLAS. Then, we factorize a smaller system (5) of size $\approx N/p$ using sparse direct solvers, where p denotes the local order of convergence, which we show can be chosen as high as 42. The numerical results feature comparisons between sparse direct solvers and demonstrate that SlabLU can factorize systems corresponding to domains of size up to $900\lambda \times 900\lambda$ (for which N=81M) in less than 20 minutes. The approach is effective in resolving challenging scattering problems on various domains to high accuracy.

The techniques described are currently being implemented for three dimensional problems. The parallelizations described are immediately applicable. The scaling with p deteriorates from $O(p^4N)$ to $O(p^6N)$, which limits how large p can be chosen. However, initial numerical experiments demonstrate that $p = 15$ remains viable on current hardware, which is high enough for most applications.

Acknowledgements The work reported was supported by the Office of Naval Research (N00014-18-1-2354), by the National Science Foundation (DMS-1952735 and DMS-2012606), and by the Department of Energy ASCR (DE-SC0022251).

References

1. Babb, T., Gillman, A., Hao, S., and Martinsson, P.-G. An Accelerated Poisson Solver based on Multidomain Spectral Discretization. *BIT Numerical Mathematics* **58**, 851–879 (2018).
2. Bériot, H., Prinn, A., and Gabard, G. Efficient Implementation of High-Order Finite Elements for Helmholtz Problems. *International Journal for Numerical Methods in Engineering* **106**(3), 213–240 (2016).
3. Brubeck, P. D. and Farrell, P. E. A Scalable and Robust Vertex-Star Relaxation for High-Order FEM. *SIAM Journal on Scientific Computing* **44**(5), A2991–A3017 (2022).
4. Cockburn, B. Static Condensation, Hybridization, and the Devising of the HDG Methods. In: *Building Bridges: Connections and Challenges in Modern Approaches to Numerical Partial Differential Equations*, 129–177. Springer (2016).
5. Davis, T. A. *Direct Methods for Sparse Linear Systems*, vol. 2. SIAM (2006).
6. Davis, T. A., Rajamanickam, S., and Sid-Lakhdar, W. M. A Survey of Direct Methods for Sparse Linear Systems. *Acta Numerica* **25**, 383 – 566 (2016).
7. Deraemaeker, A., Babuška, I., and Bouillard, P. Dispersion and Pollution of the FEM Solution for the Helmholtz Equation in One, Two and Three Dimensions. *International Journal for Numerical Methods in Engineering* **46**(4), 471–499 (1999).
8. Dongarra, J., Hammarling, S., Higham, N. J., Relton, S. D., Valero-Lara, P., and Zounon, M. The Design and Performance of Batched BLAS on Modern High-Performance Computing Systems. *Procedia Computer Science* **108**, 495–504 (2017).
9. Ernst, O. G. and Gander, M. J. Why It Is Difficult to Solve Helmholtz Problems with Classical Iterative Methods. *Numerical Analysis of Multiscale Problems* 325–363 (2012).
10. Fortunato, D., Hale, N., and Townsend, A. The Ultraspherical Spectral Element Method. *Journal of Computational Physics* **436**, 110087 (2021).
11. Gillman, A. and Martinsson, P.-G. A Direct Solver with O(N) Complexity for Variable Coefficient Elliptic PDEs Discretized via a High-Order Composite Spectral Collocation Method. *SIAM Journal on Scientific Computing* **36**(4), A2023–A2046 (2014).
12. Guyan, R. J. Reduction of Stiffness and Mass Matrices. *AIAA journal* **3**(2), 380–380 (1965).
13. Hao, S. and Martinsson, P.-G. A Direct Solver for Elliptic PDEs in Three Dimensions based on Hierarchical Merging of Poincaré–Steklov Operators. *Journal of Computational and Applied Mathematics* **308**, 419–434 (2016).
14. Levitt, J. and Martinsson, P.-G. Linear-Complexity Black-Box Randomized Compression of Hierarchically Block Separable Matrices. *arXiv preprint arXiv:2205.02990* (2022).
15. Li, X. S. and Shao, M. A Supernodal Approach to Incomplete LU Factorization with Partial Pivoting. *ACM Transactions on Mathematical Software (TOMS)* **37**(4), 1–20 (2011).
16. Martinsson, P.-G. A direct solver for variable coefficient elliptic PDEs discretized via a composite spectral collocation method. *Journal of Computational Physics* **242**, 460–479 (2013).
17. Martinsson, P.-G. *Fast Direct Solvers for Elliptic PDEs*, *CBMS-NSF Conference Series*, vol. CB96. SIAM (2019).
18. Yesypenko, A. and Martinsson, P.-G. SlabLU: A Two-Level Sparse Direct Solver for Elliptic PDEs. *arXiv preprint arXiv:2211.07572* (2022).