A direct solver for variable coefficient elliptic PDEs discretized via a composite spectral collocation method

P.G. Martinsson, Department of Applied Mathematics, University of Colorado at Boulder

Abstract: A numerical method for variable coefficient elliptic problems on two-dimensional domains is presented. The method is based on high-order spectral approximations and is designed for problems with smooth solutions. The resulting system of linear equations is solved using a direct solver with $O(N^{1.5})$ complexity for the pre-computation and $O(N \log N)$ complexity for the solve. The fact that the solver is direct is a principal feature of the scheme, and makes it particularly well suited to solving problems for which iterative solvers struggle; in particular for problems with highly oscillatory solutions. Numerical examples demonstrate that the scheme is fast and highly accurate. For instance, using a discretization with 12 points per wave-length, a Helmholtz problem on a domain of size 100×100 wavelengths was solved to ten correct digits. The computation was executed on a standard laptop; it involved 1.6M degrees of freedom and required 100 seconds for the pre-computation, and 0.3 seconds for the actual solve.

1. Introduction

1.1. **Problem formulation.** The paper describes a numerical method for solving boundary value problems of the form

$$\begin{cases} A\,u(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}) & \boldsymbol{x} \in \Gamma, \end{cases} \tag{1.1}$$

where Ω is a rectangle in the plane with boundary Γ, and where A is a variable coefficient elliptic partial differential operator

$$[Au](\boldsymbol{x}) = -c_{11}(\boldsymbol{x})[\partial_1^2 u](\boldsymbol{x}) - 2c_{12}(\boldsymbol{x})[\partial_1\partial_2 u](\boldsymbol{x}) - c_{22}(\boldsymbol{x})[\partial_2^2 u](\boldsymbol{x}) + c_1(\boldsymbol{x})[\partial_1 u](\boldsymbol{x}) + c_2(\boldsymbol{x})[\partial_2 u](\boldsymbol{x}) + c(\boldsymbol{x})\,u(\boldsymbol{x}). \tag{1.2}$$

The methodology is based on a composite spectral discretization and can be modified to handle a range of different domains, including curved ones. The primary limitation of the method is that it requires the solution u to be smooth in Ω.

The key advantage of the proposed method is that it is based on an efficient *direct* solver, and is therefore capable of solving problems for which iterative methods require very large numbers of iterations. In particular, we will demonstrate that the method can solve variable coefficient Helmholtz problems of the form

$$\begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2\,(1 - b(\boldsymbol{x}))\,u(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}) & \boldsymbol{x} \in \Gamma, \end{cases} \tag{1.3}$$

where the Helmholtz parameter κ is a real number, and where b is a given smooth scattering potential. It is known [4, 23] that this equation is quite challenging to solve accurately when κ is large and the solution is highly oscillatory. The method described in this paper is capable of solving (1.3) on domains of size 200×200 wave-length on a laptop computer, to ten digits of accuracy or more.

1.2. **Summary of proposed methodology.** We solve (1.1) by combining a spectral multidomain technique similar to the one proposed by Pfeiffer *et al* in [26] with a hierarchical direct solver that is closely related to the classical nested dissection technique of George [7].

Discretization: The equation (1.1) is discretized via a composite spectral scheme. The domain Ω is split into small square (or rectangular) patches. On each patch, the solution u is represented via tabulation on a tensor product grid of Chebyshev points, see Figure 1. The elliptic operator is

approximated via a spectral differentiation matrix acting on each local grid, and then equation (1.1) is enforced strongly via collocation at all tabulation nodes in the interior of each patch. To glue patches together, continuity of both the potential u and its normal derivative are enforced at the spectral interpolation nodes on the boundaries between patches.

Direct solver: To solve the linear system arising from the spectral discretization, we organize the rectangular patches into a binary tree of successively larger patches. The solver then involves two stages, one with an upwards pass through the tree, and one with a downwards pass:

(1) A *build stage* where an approximation to the solution operator for (1.1) is computed. This is done via a single sweep of the hierarchical tree, going from smaller patches to larger. For each leaf in the tree, a local solution operator, and an approximation to the Dirichlet-to-Neumann (DtN) map for the patch are constructed. For a parent node in the tree, a local solution operator and a local DtN operator are computed from an equilibrium equation formed using the DtN operators of the children of the patch. The build stage requires $O(N^{1.5})$ operations and $O(N \log N)$ memory.
(2) A *solve stage* that takes as input a vector of Dirichlet data tabulated on Γ, and constructs tabulated values of u at all internal grid points. The solve stage involves a single downwards sweep through the hierarchical tree of patches, going from larger patches to smaller. The solve stage has asymptotic complexity $O(N \log N)$, and is in practice *very* fast.

Remark 1.1. Like most direct solvers, the scheme requires more memory than iterative techniques, both in that memory scales as $O(N \log N)$ rather than $O(N)$, and in that the constants suppressed by the "big-O" notation are larger. However, for 2D problems even modest modern computers can handle large-scale problems, as we demonstrate in Section 6.

1.3. **Prior work.** The direct solver outlined in Section 1.2 is an evolution of a sequence of direct solvers for integral equations dating back to work by Starr & Rokhlin [32] and later [22, 12, 9, 16, 3]. The common idea is to build a global solution operator by splitting the domain into a hierarchical tree of patches, build a local solution operator for each "leaf" patch, and then build solution operators for larger patches via a hierarchical merge procedure in a sweep over the tree from smaller to larger patches. In the context of integral equations, the "solution operator" is a type of scattering matrix while in the present context, the solution operator is a DtN operator.

The proposed method is also somewhat similar in its objective to the direct solver for Poisson's equation described by Greengard & Etheridge [6], but the solver of [6] requires the kernel of the solution operator to be known analytically, which is not the case in the present context.

The work on direct solvers for the dense systems arising from integral equations was inspired by earlier work on direct solvers for sparse systems arising from finite difference and finite element discretizations of elliptic PDEs such as the classical nested dissection method of George [7] and the multifrontal methods by Duff and others [5]. While the direct solver described in this paper was arrived at via a different path, it is possible it is functionally similar to the solver that would result if such classical nested dissection techniques were applied to construct an LU-factorization of the block sparse system arising from a collocation discretization of a spectral composite method. Both methods would have complexities $O(N^{1.5})$ and $O(N \log N)$ for the build and solve stages, respectively.

The discretization technique we use is similar to earlier work on multidomain pseudospectral methods, see, e.g., [18, 36], and in particular Pfeiffer *et al* [26]. A difference is that we immediately eliminate all degrees of freedom associated with nodes that are internal to each patch (i.e. nodes not shared with any other patch) and formulate a linear system that involves only the boundary nodes. We demonstrate that this system is very amenable to efficient direct solvers, which enables us to solve problems with highly oscillatory solutions. (In contrast, [26] relies on iterative solvers.)

The technique proposed here is also similar to the *Spectral Element Method (SEM)* [25, 28] in that both methods represent solutions to the differential equation via tabulation of function values

on spectral composite meshes (as illustrated in Fig. 1). SEM is commonly used for solving wave-equations via time-stepping [14, 37, 15], but appears to be less popular for solving time-harmonic problems; we have not found computational examples in the literature involving highly oscillatory solutions in the variable coefficient regime. Mehdizadeh and Paraschivoiu [23] report results for problems up to size 17×17 wave-lengths, and report that convergence in the iterative solver is a problem, even though different pre-conditioners were tested. Kwan and Shen [19] describe a direct solver for elliptic PDEs discretized via SEM that could potentially overcome these convergence problems, but their method requires the PDE to be *separable* and they apply it only to constant coefficient problems. Ainsworth & Wajid present an error analysis of SEM for constant coefficient Helmholtz in [1], and for related work for the case of *hp*-FEM discretizations see Sauter & Melenk [24]. Shen and co-workers have analyzed spectral methods for the constant coefficient Helmholtz equation at high wave-numbers [30, 31]; for alternative spectral approaches to such problems, see also [13] and [10] and the references therein.

An early version of this manuscript was published on arXiv as [21].

1.4. **Generalizations.** The discretization scheme and the direct solver described in Section 1.2 can be extended and improved in several directions. For instance, the method can easily be applied to non-rectangular domains such as L-shapes and curved domains, as illustrated in Sections 6.3 and 6.4. For situations where the solution is non-oscillatory (or only moderately oscillatory), the scheme can be improved to attain optimal $O(N)$ complexity for both the build and the solve stages, see Section 7.1. It can be modified to handle the free space scattering problem

$$-\Delta u(\boldsymbol{x}) - \kappa^2 \left(1 - b(\boldsymbol{x})\right) u(\boldsymbol{x}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \mathbb{R}^2, \tag{1.4}$$

coupled with appropriate radiation conditions at infinity as long as the "scattering potential" b is smooth and has compact support, see Section 7.2. It can handle local mesh refinements near points where the solution is singular or has sharp gradients, and can be extended to 3D, see Section 7.3.

1.5. **Outline.** The paper is organized as follows: Section 2 introduces notation and lists some classical material on spectral interpolation and differentiation. Section 3 describes how to compute the solution operator and the DtN operator for a leaf in tree (which is discretized via a single tensor-product grid of Chebyshev nodes). Section 4 describes how the DtN operator for a larger patch consisting of two small patches can be computed if the DtN operators for the smaller patches are given. Section 5 describes the full hierarchical scheme. Section 6 reports the results of some numerical experiments. Section 7 describes how the scheme can be extended to more general situations.

2. Preliminaries — spectral differentiation

This section introduces notation for spectral differentiation on tensor product grids of Chebyshev nodes on the square domain $[-a, a]^2$. This material is classical, see, e.g., Trefethen [33].

Let p denote a positive integer. The *Chebyshev nodes* on $[-a, a]$ are the points

$$t_i = a \cos((i-1)\pi/(p-1)), \qquad i = 1,\, 2,\, 3,\, \ldots,\, p.$$

Let $\{\boldsymbol{x}_k\}_{k=1}^{p^2}$ denote the set of points of the form $(t_i,\, t_j)$ for $1 \leq i, j \leq p$. Let $\mathcal{P}_p$ denote the linear space of sums of tensor products of polynomials of degree $p-1$ or less. $\mathcal{P}_p$ has dimension p^2. Given a vector $\mathbf{u} \in \mathbb{R}^{p^2}$, there is a unique function $u \in \mathcal{P}_p$ such that $u(\boldsymbol{x}_k) = \mathbf{u}(k)$ for $1 \leq k \leq p^2$. (A reason Chebyshev nodes are of interest is that for any fixed $\boldsymbol{x} \in [-a, a]^2$, the map $\mathbf{u} \mapsto u(\boldsymbol{x})$ is stable.) Now define $\mathbf{D}$, $\mathbf{E}$, and $\mathbf{L}$ as the unique $p^2 \times p^2$ matrices such that

$$[\mathbf{Du}](k) = [\partial_1 u](\boldsymbol{x}_k), \qquad k = 1,\, 2,\, 3,\, \ldots,\, p^2, \tag{2.1}$$

$$[\mathbf{Eu}](k) = [\partial_2 u](\boldsymbol{x}_k), \qquad k = 1,\, 2,\, 3,\, \ldots,\, p^2, \tag{2.2}$$

$$[\mathbf{Lu}](k) = [-\Delta u](\boldsymbol{x}_k), \qquad k = 1,\, 2,\, 3,\, \ldots,\, p^2. \tag{2.3}$$

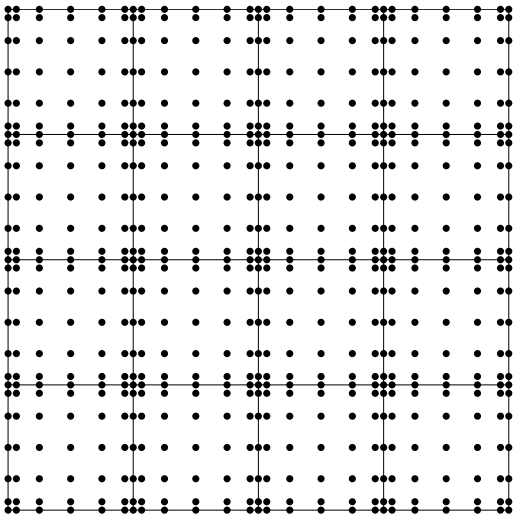


FIGURE 1. The box $\Omega = [0,1]^2$ is split into 4×4 leaf boxes, and a Cartesian grid of Chebyshev nodes is placed on each leaf box. The figure shows local grids of size 7×7 for clarity; in actual computations, local grids of size 21×21 were typically used.

3. Leaf computation

This section describes the construction of a discrete approximation to the Dirichlet-to-Neumann operator associated with the an elliptic equation such as (1.1) for a square patch Ω. We in this section assume that the patch is small enough that it can readily be handled via a "single-domain" (non-composite) spectral method using a tensor product grid of Chebyshev nodes on Ω. In addition to the DtN operator, we also construct a solution operator to (1.1) that maps the Dirichlet data on the nodes on the boundary of Ω to the value of u at all internal interpolation nodes.

For notational simplicity, we restrict attention to the variable coefficient Helmholtz problem (1.3).

3.1. **Notation.** Let Ω denote a square patch. Let $\{\boldsymbol{x}_k\}_{k=1}^{p^2}$ denote the nodes in a tensor product grid of $p \times p$ Chebyshev nodes. Partition the index set

$$\{1,\, 2,\, \ldots,\, p^2\} = I_{\mathrm{e}} \cup I_{\mathrm{i}}$$

in such a way that I_{e} contains all nodes on the boundary of Ω, and I_{i} denotes the set of interior nodes, see Figure 2(a). Let u be a function that satisfies (1.3) on Ω and let

$$\mathbf{u} = [u(\boldsymbol{x}_k)]_{k=1}^{p^2}, \qquad \mathbf{v} = [\partial_1 u(\boldsymbol{x}_k)]_{k=1}^{p^2}, \qquad \mathbf{w} = [\partial_2 u(\boldsymbol{x}_k)]_{k=1}^{p^2},$$

denote the vectors of samples of u and its partial derivatives. We define the short-hands

$$\mathbf{u}_{\mathrm{i}} = \mathbf{u}(I_{\mathrm{i}}), \qquad \mathbf{v}_{\mathrm{i}} = \mathbf{v}(I_{\mathrm{i}}), \qquad \mathbf{w}_{\mathrm{i}} = \mathbf{w}(I_{\mathrm{i}}), \qquad \mathbf{u}_{\mathrm{e}} = \mathbf{u}(I_{\mathrm{e}}), \qquad \mathbf{v}_{\mathrm{e}} = \mathbf{v}(I_{\mathrm{e}}), \qquad \mathbf{w}_{\mathrm{e}} = \mathbf{w}(I_{\mathrm{e}}).$$

Let $\mathbf{L}$, $\mathbf{D}$, and $\mathbf{E}$ denote spectral differentiation matrices corresponding to the operators $-\Delta$, ∂_1, and ∂_2, respectively (see Section 2). We use the short-hand

$$\mathbf{D}_{\mathrm{i,e}} = \mathbf{D}(I_{\mathrm{i}}, I_{\mathrm{e}})$$

to denote the part of the differentiation matrix $\mathbf{D}$ that maps exterior nodes to interior nodes, etc.

3.2. **Equilibrium condition.** The operator (1.3) is approximated via the matrix

$$\mathbf{A} = \mathbf{L} - \kappa^2\big(\mathbf{I} - \mathrm{diag}(\mathbf{b})\big),$$

where $\mathbf{b}$ denotes the vector of pointwise values of b,

$$\mathbf{b} = [b(\boldsymbol{x}_k)]_{k=1}^{p^2}.$$

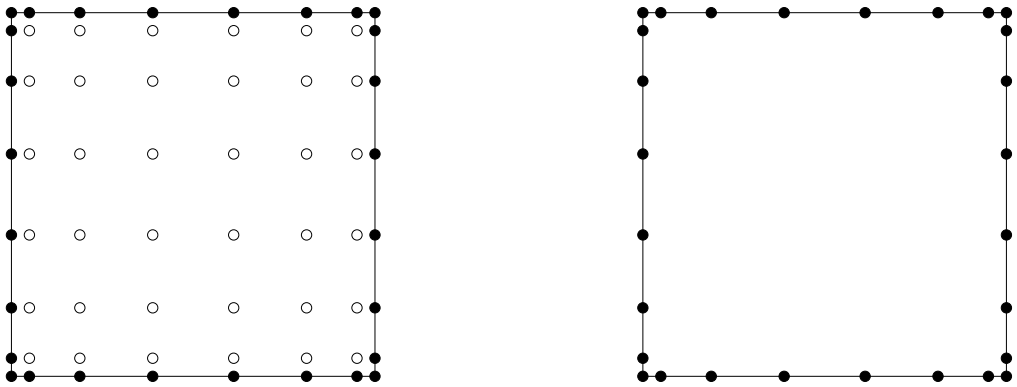


FIGURE 2. Notation for the leaf computation in Section 3. (a) A leaf before elimination of interior (white) nodes. (b) A leaf after elimination of interior nodes.

The equation we enforce on Ω is that the vector $\mathbf{A}\,\mathbf{u}$ should evaluate to zero *at all internal nodes*,

$$\mathbf{A}_{\mathrm{i,i}}\,\mathbf{u}_{\mathrm{i}} + \mathbf{A}_{\mathrm{i,e}}\,\mathbf{u}_{\mathrm{e}} = \mathbf{0}, \tag{3.1}$$

where

$$\mathbf{A}_{\mathrm{i,i}} = \mathbf{A}(I_{\mathrm{i}}, I_{\mathrm{i}}), \qquad \mathbf{A}_{\mathrm{i,e}} = \mathbf{A}(I_{\mathrm{i}}, I_{\mathrm{e}}).$$

Solving (3.1) for $\mathbf{u}_{\mathrm{i}}$, we obtain

$$\mathbf{u}_{\mathrm{i}} = \mathbf{U}\,\mathbf{u}_{\mathrm{e}}, \tag{3.2}$$

where

$$\mathbf{U} = -\big(\mathbf{A}_{\mathrm{i,i}}\big)^{-1}\,\mathbf{A}_{\mathrm{i,e}}. \tag{3.3}$$

3.3. **Constructing the DtN operator.** Let $\mathbf{V}$ and $\mathbf{W}$ denote the matrices that map boundary values of the potential to boundary values of $\partial_1 u$ and $\partial_2 u$. These are constructed as follows: Given the potential $\mathbf{u}_{\mathrm{e}}$ on the boundary, we reconstruct the potential $\mathbf{u}_{\mathrm{i}}$ in the interior via (3.2). Then, since the potential is known on all Chebyshev nodes in Ω, we can determine the gradient on the boundary $\{\mathbf{v}_{\mathrm{e}}, \mathbf{w}_{\mathrm{e}}\}$ via spectral differentiation on the entire domain. To formalize, we find

$$\mathbf{v}_{\mathrm{e}} = \mathbf{D}_{\mathrm{e,e}}\,\mathbf{u}_{\mathrm{e}} + \mathbf{D}_{\mathrm{e,i}}\,\mathbf{u}_{\mathrm{i}} = \mathbf{D}_{\mathrm{e,e}}\,\mathbf{u}_{\mathrm{e}} + \mathbf{D}_{\mathrm{e,i}}\,\mathbf{U}\,\mathbf{u}_{\mathrm{e}} = \mathbf{V}\,\mathbf{u}_{\mathrm{e}},$$

where

$$\mathbf{V} = \mathbf{D}_{\mathrm{e,e}} + \mathbf{D}_{\mathrm{e,i}}\,\mathbf{U}. \tag{3.4}$$

An analogous computation for $\mathbf{w}_{\mathrm{e}}$ yields

$$\mathbf{W} = \mathbf{E}_{\mathrm{e,e}} + \mathbf{E}_{\mathrm{e,i}}\,\mathbf{U}. \tag{3.5}$$

4. Merge operation

Let Ω denote a rectangular domain consisting of the union of the two smaller rectangular domains,

$$\Omega = \Omega_\alpha \cup \Omega_\beta,$$

as shown in Figure 3. Moreover, suppose that approximations to the DtN operators for Ω_α and Ω_β are available. (Represented as matrices that map boundary values of u to boundary values of $\partial_1 u$ and $\partial_2 u$.) This section describes how to compute a solution operator $\mathbf{U}$ that maps the value of a function u that is tabulated on the boundary of Ω to the values of u on interpolation nodes on the internal boundary, as well as operators $\mathbf{V}$ and $\mathbf{W}$ that map boundary values of u on the boundary of Ω to values of the $\partial_1 u$ and $\partial_2 u$ tabulated on the boundary.

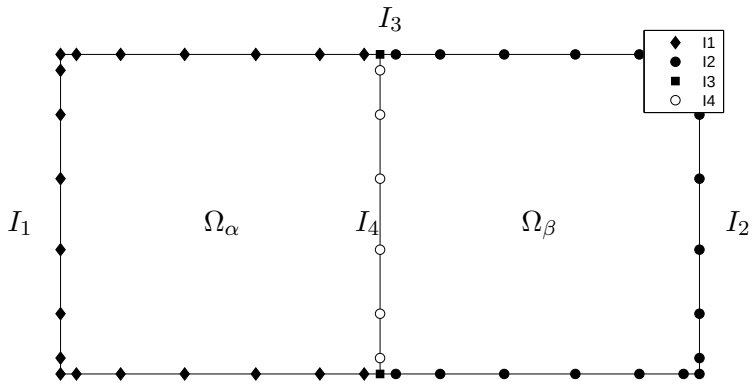


FIGURE 3. Notation for the merge operation described in Section 4. The rectangular domain Ω is formed by two squares Ω_α and Ω_β. The sets I_1, I_2, and I_3 form the exterior nodes (black), while I_4 consists of the interior nodes (white).

4.1. **Notation.** Let Ω denote a box with children Ω_α and Ω_β. For concreteness, let us assume that Ω_α and Ω_β share a vertical edge. We partition the points on $\partial\Omega_\alpha$ and $\partial\Omega_\beta$ into four sets:

- I_1 Boundary nodes of Ω_α that are not boundary nodes of Ω_β.
- I_2 Boundary nodes of Ω_β that are not boundary nodes of Ω_α.
- I_3 The two nodes that are boundary nodes of Ω_α, of Ω_β, and also of the union box Ω.
- I_4 Boundary nodes of both Ω_α and Ω_β that are *not* boundary nodes of the union box Ω.

Figure 3 illustrates the definitions of the I_j's. Let u denote a function such that

$$-\Delta u(\boldsymbol{x}) - \kappa^2 \left(1 - b(\boldsymbol{x})\right) u(\boldsymbol{x}) = 0, \qquad \boldsymbol{x} \in \Omega,$$

and let $\mathbf{u}_j$, $\mathbf{v}_j$, $\mathbf{w}_j$ denote the values of u, $\partial_1 u$, and $\partial_2 u$, restricted to the nodes in the set "j". Moreover, set

$$\mathbf{u}_{\text{i}} = \mathbf{u}_4, \qquad \text{and} \qquad \mathbf{u}_{\text{e}} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix}. \tag{4.1}$$

Finally, let $\mathbf{V}^\alpha$, $\mathbf{W}^\alpha$, $\mathbf{V}^\beta$, $\mathbf{W}^\beta$ denote the operators that map potential values on the boundary to values of $\partial_1 u$ and $\partial_2 u$ on the boundary for the boxes Ω_α and Ω_β. We partition these matrices according to the numbering of nodes in Figure 3,

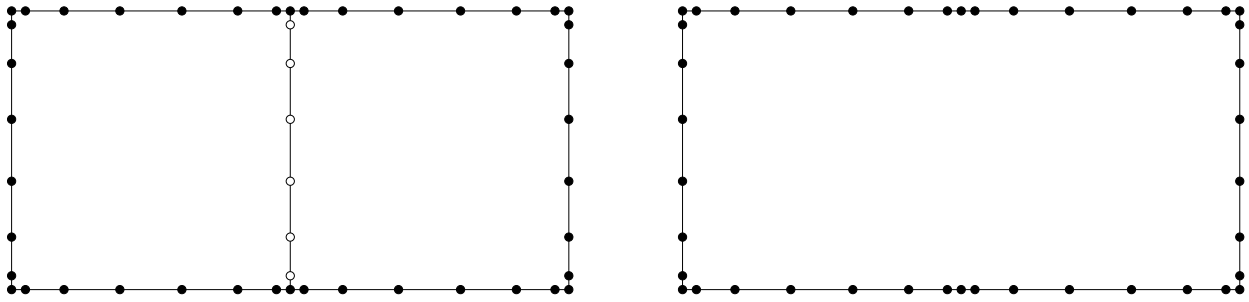
$$\begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{V}^\alpha_{1,1} & \mathbf{V}^\alpha_{1,3} & \mathbf{V}^\alpha_{1,4} \\ \mathbf{V}^\alpha_{3,1} & \mathbf{V}^\alpha_{3,3} & \mathbf{V}^\alpha_{3,4} \\ \mathbf{V}^\alpha_{4,1} & \mathbf{V}^\alpha_{4,3} & \mathbf{V}^\alpha_{4,4} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_3 \\ \mathbf{u}_4 \end{bmatrix}, \qquad \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_3 \\ \mathbf{w}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{W}^\alpha_{1,1} & \mathbf{W}^\alpha_{1,3} & \mathbf{W}^\alpha_{1,4} \\ \mathbf{W}^\alpha_{3,1} & \mathbf{W}^\alpha_{3,3} & \mathbf{W}^\alpha_{3,4} \\ \mathbf{W}^\alpha_{4,1} & \mathbf{W}^\alpha_{4,3} & \mathbf{W}^\alpha_{4,4} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_3 \\ \mathbf{u}_4 \end{bmatrix}, \tag{4.2}$$

and

$$\begin{bmatrix} \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{V}^\beta_{2,2} & \mathbf{V}^\beta_{2,3} & \mathbf{V}^\beta_{2,4} \\ \mathbf{V}^\beta_{3,2} & \mathbf{V}^\beta_{3,3} & \mathbf{V}^\beta_{3,4} \\ \mathbf{V}^\beta_{4,2} & \mathbf{V}^\beta_{4,3} & \mathbf{V}^\beta_{4,4} \end{bmatrix} \begin{bmatrix} \mathbf{u}_2 \\ \mathbf{u}_3 \\ \mathbf{u}_4 \end{bmatrix}, \qquad \begin{bmatrix} \mathbf{w}_2 \\ \mathbf{w}_3 \\ \mathbf{w}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{W}^\beta_{2,2} & \mathbf{W}^\beta_{2,3} & \mathbf{W}^\beta_{2,4} \\ \mathbf{W}^\beta_{3,2} & \mathbf{W}^\beta_{3,3} & \mathbf{W}^\beta_{3,4} \\ \mathbf{W}^\beta_{4,2} & \mathbf{W}^\beta_{4,3} & \mathbf{W}^\beta_{4,4} \end{bmatrix} \begin{bmatrix} \mathbf{u}_2 \\ \mathbf{u}_3 \\ \mathbf{u}_4 \end{bmatrix}. \tag{4.3}$$

4.2. **Equilibrium condition.** Suppose that we are given a tabulation of boundary values of a function u that satisfies (1.3) on Ω. In other words, we are given the vectors $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$. We can then reconstruct the values of the potential on the interior boundary (tabulated in the vector $\mathbf{u}_4$) by using information in (4.2) and (4.3). Simply observe that there are two equations specifying the normal derivative across the internal boundary (tabulated in $\mathbf{v}_4$), and combine these equations:

$$\mathbf{V}^\alpha_{4,1}\mathbf{u}_1 + \mathbf{V}^\alpha_{4,3}\mathbf{u}_3 + \mathbf{V}^\alpha_{4,4}\mathbf{u}_4 = \mathbf{V}^\beta_{4,2}\mathbf{u}_2 + \mathbf{V}^\beta_{4,3}\mathbf{u}_3 + \mathbf{V}^\beta_{4,4}\mathbf{u}_4.$$



(a) (b)

FIGURE 4. Merge operation for two small boxes to form a new large box. (a) Before elimination of interior (white) nodes. (b) After elimination of interior nodes.

Solving for $\mathbf{u}_4$ we get

$$\mathbf{u}_4 = \big(\mathbf{V}^\alpha_{4,4} - \mathbf{V}^\beta_{4,4}\big)^{-1}\big(\mathbf{V}^\beta_{4,2}\mathbf{u}_2 + \mathbf{V}^\beta_{4,3}\mathbf{u}_3 - \mathbf{V}^\alpha_{4,1}\mathbf{u}_1 - \mathbf{V}^\alpha_{4,3}\mathbf{u}_3\big). \tag{4.4}$$

Now set

$$\mathbf{U} = \big(\mathbf{V}^\alpha_{4,4} - \mathbf{V}^\beta_{4,4}\big)^{-1}\big[-\mathbf{V}^\alpha_{4,1} \;\big|\; \mathbf{V}^\beta_{4,2} \;\big|\; \mathbf{V}^\beta_{4,3} - \mathbf{V}^\alpha_{4,3}\big], \tag{4.5}$$

to find that (4.4) is (in view of (4.1)) precisely the desired formula

$$\mathbf{u}_\mathrm{i} = \mathbf{U}\,\mathbf{u}_\mathrm{e}. \tag{4.6}$$

The net effect of the merge operation is to eliminate the interior tabulation nodes in Ω_α and Ω_β so that only boundary nodes in the union box Ω are kept, as illustrated in Figure 4.

4.3. **Constructing the DtN operators for the union box.** We will next build a matrix $\mathbf{V}$ that constructs the derivative $\partial_1 u$ on $\partial\Omega$ given values of u on $\partial\Omega$. In other words

$$\begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix} = \mathbf{V} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix}.$$

To this end, observe from (4.2) and (4.3) that

$$\mathbf{v}_1 = \mathbf{V}^\alpha_{1,1}\,\mathbf{u}_1 + \mathbf{V}^\alpha_{1,3}\,\mathbf{u}_3 + \mathbf{V}^\alpha_{1,4}\,\mathbf{u}_4 = \mathbf{V}^\alpha_{1,1}\,\mathbf{u}_1 + \mathbf{V}^\alpha_{1,3}\,\mathbf{u}_3 + \mathbf{V}^\alpha_{1,4}\,\mathbf{U}\,\mathbf{u}_\mathrm{e} \tag{4.7}$$

$$\mathbf{v}_2 = \mathbf{V}^\beta_{2,2}\,\mathbf{u}_2 + \mathbf{V}^\beta_{2,3}\,\mathbf{u}_3 + \mathbf{V}^\beta_{2,4}\,\mathbf{u}_4 = \mathbf{V}^\beta_{2,2}\,\mathbf{u}_2 + \mathbf{V}^\beta_{2,3}\,\mathbf{u}_3 + \mathbf{V}^\beta_{2,4}\,\mathbf{U}\,\mathbf{u}_\mathrm{e}. \tag{4.8}$$

Equations (4.2) and (4.3) provide two different formulas for $\mathbf{v}_3$, either of which could be used. For numerical stability, we use the average of the two:

$$\mathbf{v}_3 = \frac{1}{2}\big(\mathbf{V}^\alpha_{3,1}\mathbf{u}_1 + \mathbf{V}^\alpha_{3,3}\mathbf{u}_3 + \mathbf{V}^\alpha_{3,4}\mathbf{u}_4 + \mathbf{V}^\beta_{3,2}\mathbf{u}_2 + \mathbf{V}^\beta_{3,3}\mathbf{u}_3 + \mathbf{V}^\beta_{3,4}\mathbf{u}_4\big) \tag{4.9}$$

$$= \frac{1}{2}\big(\mathbf{V}^\alpha_{3,1}\mathbf{u}_1 + \mathbf{V}^\alpha_{3,3}\mathbf{u}_3 + \mathbf{V}^\alpha_{3,4}\,\mathbf{U}\,\mathbf{u}_\mathrm{e} + \mathbf{V}^\beta_{3,2}\mathbf{u}_2 + \mathbf{V}^\beta_{3,3}\mathbf{u}_3 + \mathbf{V}^\beta_{3,4}\,\mathbf{U}\,\mathbf{u}_\mathrm{e}\big). \tag{4.10}$$

Combining (4.7) – (4.10) we obtain

$$\begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix} = \left(\begin{bmatrix} \mathbf{V}^\alpha_{1,1} & \mathbf{0} & \mathbf{V}^\alpha_{1,3} \\ \mathbf{0} & \mathbf{V}^\beta_{2,2} & \mathbf{V}^\beta_{2,3} \\ \frac{1}{2}\mathbf{V}^\alpha_{3,1} & \frac{1}{2}\mathbf{V}^\beta_{3,2} & \frac{1}{2}\mathbf{V}^\alpha_{3,3} + \frac{1}{2}\mathbf{V}^\beta_{3,3} \end{bmatrix} + \begin{bmatrix} \mathbf{V}^\alpha_{1,4} \\ \mathbf{V}^\beta_{2,4} \\ \frac{1}{2}\mathbf{V}^\alpha_{3,4} + \frac{1}{2}\mathbf{V}^\beta_{3,4} \end{bmatrix} \mathbf{U} \right) \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix}.$$

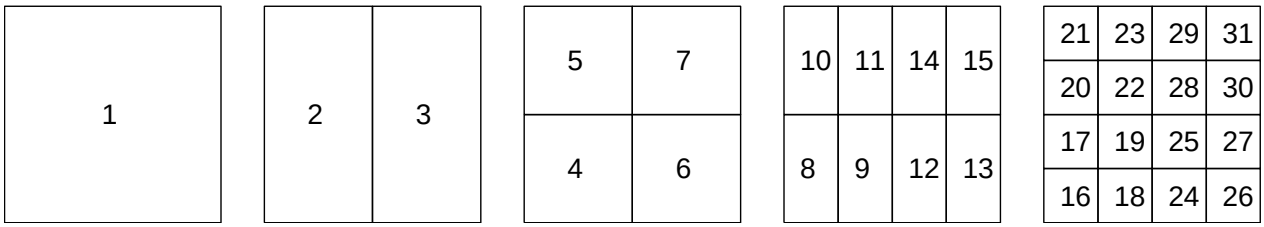


FIGURE 5. The square domain Ω is split into 4×4 leaf boxes. These are then gathered into a binary tree of successively larger boxes as described in Section 5.1. One possible enumeration of the boxes in the tree is shown, but note that the only restriction is that if box τ is the parent of box σ, then $\tau < \sigma$.

In other words,

$$\mathbf{V} = \begin{bmatrix} \mathbf{V}^{\alpha}_{1,1} & \mathbf{0} & \mathbf{V}^{\alpha}_{1,3} \\ \mathbf{0} & \mathbf{V}^{\beta}_{2,2} & \mathbf{V}^{\beta}_{2,3} \\ \frac{1}{2}\mathbf{V}^{\alpha}_{3,1} & \frac{1}{2}\mathbf{V}^{\beta}_{3,2} & \frac{1}{2}\mathbf{V}^{\alpha}_{3,3} + \frac{1}{2}\mathbf{V}^{\beta}_{3,3} \end{bmatrix} + \begin{bmatrix} \mathbf{V}^{\alpha}_{1,4} \\ \mathbf{V}^{\beta}_{2,4} \\ \frac{1}{2}\mathbf{V}^{\alpha}_{3,4} + \frac{1}{2}\mathbf{V}^{\beta}_{3,4} \end{bmatrix} \mathbf{U}. \tag{4.11}$$

An analogous computation for $\mathbf{w}_{\mathrm{e}}$ yields

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}^{\alpha}_{1,1} & \mathbf{0} & \mathbf{W}^{\alpha}_{1,3} \\ \mathbf{0} & \mathbf{W}^{\beta}_{2,2} & \mathbf{W}^{\beta}_{2,3} \\ \frac{1}{2}\mathbf{W}^{\alpha}_{3,1} & \frac{1}{2}\mathbf{W}^{\beta}_{3,2} & \frac{1}{2}\mathbf{W}^{\alpha}_{3,3} + \frac{1}{2}\mathbf{W}^{\beta}_{3,3} \end{bmatrix} + \begin{bmatrix} \mathbf{W}^{\alpha}_{1,4} \\ \mathbf{W}^{\beta}_{2,4} \\ \frac{1}{2}\mathbf{W}^{\alpha}_{3,4} + \frac{1}{2}\mathbf{W}^{\beta}_{3,4} \end{bmatrix} \mathbf{U}. \tag{4.12}$$

5. The full hierarchical scheme

5.1. **The algorithm.** Now that we know how to construct the DtN operator for a leaf (Section 3), and how to merge the DtN operators of two neighboring patches to form the DtN operator of their union (Section 4), we are ready to describe the full hierarchical scheme for solving (1.3).

First we partition the domain Ω into a collection of square (or possibly rectangular) boxes, called *leaf boxes*. These should be small enough that a small spectral mesh with $p\times p$ nodes (for, say, $p = 20$) accurately interpolates both any potential solution u of (1.3) and its partial derivatives $\partial_1 u$, $\partial_2 u$, and $-\Delta u$. Let $\{\boldsymbol{x}_k\}_{k=1}^{N}$ denote the points in this mesh. (Observe that nodes on internal boundaries are shared between two or four local meshes.) Next construct a binary tree on the collection of boxes by hierarchically merging them, making sure that all boxes on the same level are roughly of the same size, cf. Figure 5. The boxes should be ordered so that if τ is a parent of a box σ, then $\tau < \sigma$. We also assume that the root of the tree (i.e. the full box Ω) has index $\tau = 1$.

With each box τ, we define two index vectors I^{τ}_{i} and I^{τ}_{e} as follows:

I^{τ}_{e} A list of all indices of the nodes on the boundary of τ.

I^{τ}_{i} For a leaf τ, I^{τ}_{i} is a list of all interior nodes of τ.
For a parent τ, I^{τ}_{i} is a list of all its interior nodes that are not interior nodes of its children.

Next we execute a pre-computation in which for every box τ, we construct the following matrices:

$\mathbf{U}^{\tau}$ The matrix that maps the values of $\mathbf{u}$ on the boundary of a box to the values of $\mathbf{u}$ on the interior nodes of the box. In other words, $\mathbf{u}(I^{\tau}_{\mathrm{i}}) = \mathbf{U}^{\tau}\,\mathbf{u}(I^{\tau}_{\mathrm{e}})$.

$\mathbf{V}^{\tau}$ The matrix that maps the values of $\mathbf{u}$ on the boundary of a box to the values of $\mathbf{v}$ (tabulating du/dx_1) on the boundary of a box. In other words, $\mathbf{v}(I^{\tau}_{\mathrm{e}}) = \mathbf{V}^{\tau}\,\mathbf{u}(I^{\tau}_{\mathrm{e}})$.

$\mathbf{W}^{\tau}$ The matrix that maps the values of $\mathbf{u}$ on the boundary of a box to the values of $\mathbf{w}$ (tabulating du/dx_2) on the boundary of a box. In other words, $\mathbf{w}(I^{\tau}_{\mathrm{e}}) = \mathbf{W}^{\tau}\,\mathbf{u}(I^{\tau}_{\mathrm{e}})$.

To this end, we scan all boxes in the tree, going from smaller to larger. For any leaf box τ, a dense matrix $\mathbf{A}^\tau$ of size $p^2 \times p^2$ that locally approximates the differential operator in (1.3) is formed. Then the matrices $\mathbf{U}^\tau$, $\mathbf{V}^\tau$, and $\mathbf{W}^\tau$ are constructed via formulas (3.3), (3.4), and (3.5). For a parent box τ with children σ_1 and σ_2, the matrices $\mathbf{U}^\tau$, $\mathbf{V}^\tau$, and $\mathbf{W}^\tau$ are formed from the DtN operators encoded in the matrices $\mathbf{V}^{\sigma_1}$, $\mathbf{W}^{\sigma_1}$, $\mathbf{V}^{\sigma_2}$, $\mathbf{W}^{\sigma_2}$ using the formulas (4.5), (4.11), and (4.12). The full algorithm is summarized in Figure 6. An illustrated cartoon of the merge process is provided in Appendix A.

Once all the matrices $\{\mathbf{U}^\tau\}_\tau$ have been formed, it is a simple matter to construct a vector $\mathbf{u}$ holding approximations to the solution u of (1.3). The nodes are scanned starting with the root, and then proceeding down in the tree towards smaller boxes. When a box τ is processed, the value of $\mathbf{u}$ is known for all nodes on its boundary (i.e. those listed in I_{e}^τ). The matrix $\mathbf{U}^\tau$ directly maps these values to the values of $\mathbf{u}$ on the nodes in the interior of τ (i.e. those listed in I_{i}^τ). When all nodes have been processed, all entries of $\mathbf{u}$ have been computed. Figure 7 summarizes the solve stage.

Remark 5.1. Every interior meshpoint $\boldsymbol{x}_k$ belongs to the index vector I_{i}^τ for precisely one node τ. In other words $\bigcup_\tau I_{\text{i}}^\tau$ forms a disjoint union of the interior mesh points.

Remark 5.2. The way the algorithms are described, we compute for each node τ matrices $\mathbf{V}^\tau$ and $\mathbf{W}^\tau$ that allow the computation of both the normal and the tangential derivative at any boundary node, given the Dirichlet data on the boundary. This is done for notational convenience only. In practice, any rows of $\mathbf{V}^\tau$ and $\mathbf{W}^\tau$ that correspond to evaluation of tangential derivatives need never be evaluated since tangential derivatives do not enter into consideration at all.

Remark 5.3. The merge stage is exact when performed in exact arithmetic. The only approximation involved is the approximation of the solution u on a leaf by its interpolating polynomial.

5.2. **Complexity analysis.** The analysis of the asymptotic cost of the algorithm in Section 5.1 closely mimics the analysis of the classical nested dissection algorithm [17, 7]. For simplicity, we analyze the simplest situation in which a square domain is divided into 4^L leaf boxes, each holding a spectral cartesian mesh with $p \times p$ points. The total number of unknowns in the system is then roughly $4^L\, p^2$ (to be precise, $N = 4^L\,(p-1)^2 + 2^{L+1}\,(p-1) + 1$).

Cost of leaf computation: Evaluating the formulas (3.3), (3.4), and (3.5) requires dense matrix algebra on matrices of size roughly $p^2 \times p^2$. Since there are about N/p^2 leaves, the total cost is

$$T_{\text{leaf}} \sim \frac{N}{p^2} \times \left(p^2\right)^3 \sim N\, p^4.$$

Cost of the merge operations: For an integer $\ell \in \{0,\, 1,\, 2,\, \ldots,\, L\}$, we refer to "level ℓ" as the collection of boxes whose side length is $2^{-\ell}$ times the side of the full box Ω (so that $\ell = 0$ corresponds to the root and $\ell = L$ corresponds to the set of leaf boxes). To form the matrices $\mathbf{U}^\tau$, $\mathbf{V}^\tau$, and $\mathbf{W}^\tau$ for a box on level ℓ, we need to evaluate each of the formulas (4.5), (4.11), and (4.12) three times, with each computation involving matrices of size roughly $2^{-\ell} N^{0.5} \times 2^{-\ell} N^{0.5}$. Since there are 4^ℓ boxes on level ℓ, we find that the cost of processing level ℓ is

$$T_\ell \sim 4^\ell \times \left(2^{-\ell}\, N^{0.5}\right)^3 \sim 2^{-\ell}\, N^{1.5}.$$

Adding the costs at all levels, we get

$$T_{\text{merge}} \sim \sum_{\ell=0}^{L-1} T_\ell \sim \sum_{\ell=0}^{L-1} 2^{-\ell}\, N^{1.5} \sim N^{1.5}.$$

Cost of solve stage: The cost of processing a non-leaf node on level ℓ is simply the cost of a matrix-vector multiply involving the dense matrix $\mathbf{U}^\tau$ of size $2^{-\ell} N^{0.5} \times 2^{-\ell} N^{0.5}$. For a leaf, $\mathbf{U}^\tau$ is of size

PRE-COMPUTATION

This program constructs the global Dirichlet-to-Neumann operator for (1.3).
It also constructs all matrices $\mathbf{U}^\tau$ required for constructing u at all interior nodes.
It is assumed that if node τ is a parent of node σ, then $\tau < \sigma$.

---

for $\tau = N_{\text{boxes}},\ N_{\text{boxes}} - 1,\ N_{\text{boxes}} - 2,\ \ldots,\ 1$
  if (τ is a leaf)
    $\mathbf{b}_{\text{i}}^\tau = [b(\boldsymbol{x}_j)]_{j \in I_{\text{i}}^\tau}$
    $\mathbf{U}^\tau = -\big(-\mathbf{L}_{\text{i,i}} - \kappa^2 \text{diag}(\mathbf{b}_{\text{i}}^\tau)\big)^{-1}\, \mathbf{L}_{\text{i,e}}$
    $\mathbf{V}^\tau = \mathbf{D}_{\text{e,e}} + \mathbf{D}_{\text{e,i}}\, \mathbf{U}^\tau$
    $\mathbf{W}^\tau = \mathbf{E}_{\text{e,e}} + \mathbf{D}_{\text{e,i}}\, \mathbf{U}^\tau$
  else
    Let σ_1 and σ_2 be the children of τ.
    Partition $I_{\text{e}}^{\sigma_1}$ and $I_{\text{e}}^{\sigma_2}$ into vectors I_1, I_2, I_3, and I_4 as shown in Figure 3.
    if (σ_1 and σ_2 are side-by-side)
      $\mathbf{U}^\tau = \big(\mathbf{V}_{4,4}^{\sigma_1} - \mathbf{V}_{4,4}^{\sigma_2}\big)^{-1} \big[-\mathbf{V}_{4,1}^{\sigma_1} \ \big|\ \mathbf{V}_{4,2}^{\sigma_2} \ \big|\ \mathbf{V}_{4,3}^{\sigma_2} - \mathbf{V}_{4,3}^{\sigma_1}\big]$
    else
      $\mathbf{U}^\tau = \big(\mathbf{W}_{4,4}^{\sigma_1} - \mathbf{W}_{4,4}^{\sigma_2}\big)^{-1} \big[-\mathbf{W}_{4,1}^{\sigma_1} \ \big|\ \mathbf{W}_{4,2}^{\sigma_2} \ \big|\ \mathbf{W}_{4,3}^{\sigma_2} - \mathbf{W}_{4,3}^{\sigma_1}\big]$
    end if

$$\mathbf{V}^\tau = \begin{bmatrix} \mathbf{V}_{1,1}^{\sigma_1} & \mathbf{0} & \mathbf{V}_{1,3}^{\sigma_1} \\ \mathbf{0} & \mathbf{V}_{2,2}^{\sigma_2} & \mathbf{V}_{2,3}^{\sigma_2} \\ \frac{1}{2}\mathbf{V}_{3,1}^{\sigma_1} & \frac{1}{2}\mathbf{V}_{3,2}^{\sigma_2} & \frac{1}{2}\mathbf{V}_{3,3}^{\sigma_1} + \frac{1}{2}\mathbf{V}_{3,3}^{\sigma_2} \end{bmatrix} + \begin{bmatrix} \mathbf{V}_{1,4}^{\sigma_1} \\ \mathbf{V}_{2,4}^{\sigma_2} \\ \frac{1}{2}\mathbf{V}_{3,4}^{\sigma_1} + \frac{1}{2}\mathbf{V}_{3,4}^{\sigma_2} \end{bmatrix} \mathbf{U}^\tau.$$

$$\mathbf{W}^\tau = \begin{bmatrix} \mathbf{W}_{1,1}^{\sigma_1} & \mathbf{0} & \mathbf{W}_{1,3}^{\sigma_1} \\ \mathbf{0} & \mathbf{W}_{2,2}^{\sigma_2} & \mathbf{W}_{2,3}^{\sigma_2} \\ \frac{1}{2}\mathbf{W}_{3,1}^{\sigma_1} & \frac{1}{2}\mathbf{W}_{3,2}^{\sigma_2} & \frac{1}{2}\mathbf{W}_{3,3}^{\sigma_1} + \frac{1}{2}\mathbf{W}_{3,3}^{\sigma_2} \end{bmatrix} + \begin{bmatrix} \mathbf{W}_{1,4}^{\sigma_1} \\ \mathbf{W}_{2,4}^{\sigma_2} \\ \frac{1}{2}\mathbf{W}_{3,4}^{\sigma_1} + \frac{1}{2}\mathbf{W}_{3,4}^{\sigma_2} \end{bmatrix} \mathbf{U}^\tau.$$

    Delete $\mathbf{V}^{\sigma_1}$, $\mathbf{W}^{\sigma_1}$, $\mathbf{V}^{\sigma_2}$, $\mathbf{W}^{\sigma_2}$.
  end if
end for

FIGURE 6. Pre-computation

SOLVER

This program constructs an approximation $\mathbf{u}$ to the solution u of (1.3).
It assumes that all matrices $\mathbf{U}^\tau$ have already been constructed in a pre-computation.
It is assumed that if node τ is a parent of node σ, then $\tau < \sigma$.

$\mathbf{u}(k) = f(\boldsymbol{x}_k)$ for all $k \in I_{\text{e}}^1$.
for $\tau = 1,\ 2,\ 3,\ \ldots,\ N_{\text{boxes}}$
  $\mathbf{u}(I_{\text{i}}^\tau) = \mathbf{U}^\tau\, \mathbf{u}(I_{\text{i}}^\tau)$.
end for

FIGURE 7. Solve stage.

roughly $p^2 \times p$. Therefore

$$T_{\text{leaf}} \sim \frac{N}{p^2} \times p^2\, p + \sum_{\ell=0}^{L-1} 4^\ell \times \left(2^{-\ell} N^{0.5}\right)^2 \sim N\, p + N\, L \sim N\, p + N\, \log(N).$$

Remark 5.4. In terms of *practical efficiency*, the direct solver proposed excels for problems where iterative solvers require many iterations, such as the equations with highly oscillatory solutions investigated in Sections 6.1 – 6.4. In situations where iterative solvers converge rapidly, we expect the proposed solver to be competitive for moderate size problems, but for large scale problems, its $O(N^{1.5})$ complexity will eventually render it uncompetitive (unless acceleration as outlined in Section 7.1 is implemented). Note however that the solve stage is $O(N \log N)$ with a very small scaling constant, which makes the direct solver of high interest in situations where the same equation needs to be solved for a sequence of data functions. Finally we note that the direct solver requires somewhat more memory than an iterative method, although this tends to be unproblematic in 2D.

Remark 5.5. The solver *parallelizes* very well since all patches are processed independently of their neighbors; the only information required to process a patch comes from its children. For very large problems, the build time will be dominated by the dense matrix operations required at the top levels (the largest patches). These are dicey to parallelize well, but since they concern standard matrix operations, the top levels can be executed efficiently using standardized packages [2, 27].

5.3. **Problem of resonances.** The scheme presented in Section 5.1 will fail if one of the patches in the hierarchical partitioning is resonant in the sense that there exist non-trivial solutions to the Helmholtz equation that have zero Dirichlet data at the boundary of the patch. (Observe that this is a property of the "physics" of the problem, and not a mathematical or numerical artifact.) In this case, the Neumann data for the patch is not uniquely determined by the Dirichlet data, and the DtN operator cannot exist. In practice, this problem will of course arise even if we are merely *close* to a resonance, and will be detected by the discovery that the inverse matrix in the formulas (3.3) and (4.5) for the solution operator $\mathbf{U}^\tau$ is ill-conditioned.

It is our experience from working even with domains a couple of hundreds of wave-lengths across that resonances are very rare; one almost never encounters the problem. Nevertheless, it is important to monitor the conditioning of the formulas (3.3) and (4.5) to ensure the accuracy of the final answer. Should a problem be detected, the easiest solution would be to simply start the computation over with a different tessellation of the domain Ω. Very likely, this will resolve the problem.

While the problem of resonances can be managed, it would clearly be preferable to formulate a variation of the scheme that is inherently not vulnerable in this regard. We are currently investigating several different possibilities, including using the so called "total wave" approach suggested by Yu Chen of New York University (private communication). The idea is to maintain for each leaf not an operator that maps Dirichlet data to Neumann data, but rather a collection of matching pairs of Dirichlet and Neumann data (represented as vectors of tabulated values on the boundary). This scheme is easy to implement, but it would likely get prohibitively expensive in 3D, and does not seem to easily lend itself to acceleration techniques like those described in Section 7.1.

6. Numerical experiments

This section reports the results of some numerical experiments with the method described in Section 5.1. The method was implemented in Matlab and the experiments executed on a Lenovo W510 laptop with a quad core Intel i7 Q720 processor with 1.6GHz clockspeed, and 16GB of RAM.

The speed and memory requirements of the algorithm were investigated by solving the special case where $b = 0$ in (1.3), and the Dirichlet data is set to equal a known analytic solution. The results are reported in Section 6.1. We also report the errors incurred in the special case; these represent a best case estimate of the errors since the equation solved is particularly benign. To get a more realistic estimate of the errors in the method, we also applied it to three situation in which exact solutions are not known: A problem with variable coefficients in Section 6.2, a problem on an L-shaped domain in Section 6.3, a curved domain in Section 6.4, and a convection-diffusion problem in Section 6.5.

6.1. **Constant coefficient Helmholtz.** We solved the basic Helmholtz equation

$$\begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2\, u(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}) & \boldsymbol{x} \in \Gamma, \end{cases} \tag{6.1}$$

where $\Omega = [0,1]^2$ and $\Gamma = \partial\Omega$. The boundary data were in this first set of experiments chosen as the restriction to Γ of an exact solution

$$u_{\text{exact}}(\boldsymbol{x}) = Y_0(\kappa|\boldsymbol{x} - \hat{\boldsymbol{x}}|), \tag{6.2}$$

where $\hat{\boldsymbol{x}} = (-0.2,\, 0.4)$, and where Y_0 is the 0'th Bessel function of the second kind. This experiment serves two purposes. The first is to systematically measure the speed and memory requirements of the method described in Section 5.1. The second is to get a sense of what errors can be expected in a "best case" scenario with a very smooth solution. Observe however that the situation is by no means artificial since the smoothness of this case is exactly what one encounters when the solver is applied to a free space scattering problem as described in Section 7.2.

The domain Ω was discretized into $n \times n$ patches, and on each patch a $p \times p$ Cartesian mesh of Chebyshev nodes was placed. The total number of degrees of freedom is then

$$N = \big(n(p-1)\big)^2 + 2n(p-1) + 1.$$

We tested the method for $p \in \{6,\, 11,\, 21,\, 41\}$. For each fixed p, the method was executed for several different mesh sizes n. The wave-number κ was chosen to keep a constant of 12 points per wavelength, or $\kappa = 2\pi\, n(p-1)/12$.

Since the exact solution is known in this case, we computed the direct error measure

$$E_{\text{pot}} = \max_{k\,:\,\boldsymbol{x}_k \in \Omega} \big| u_{\text{computed}}(\boldsymbol{x}_k) - u_{\text{exact}}(\boldsymbol{x}_k) \big|, \tag{6.3}$$

where $\{\boldsymbol{x}_k\}_{k=1}^N$ is the set of all mesh points. We also computed the maximum error in the gradient of u on the boundary as computed via the $\mathbf{V}$ and $\mathbf{W}$ operators on the root box,

$$E_{\text{grad}} = \max_{k\,:\,\boldsymbol{x}_k \in \Gamma} \big\{ \big| v_{\text{computed}}(\boldsymbol{x}_k) - [\partial_1 u_{\text{exact}}](\boldsymbol{x}_k) \big|,\, \big| w_{\text{computed}}(\boldsymbol{x}_k) - [\partial_2 u_{\text{exact}}](\boldsymbol{x}_k) \big| \big\}. \tag{6.4}$$

Table 1 reports the following variables:

N_{wave}	The number of wave-lengths along one side of Ω.
t_{build}	The time in seconds required to execute the build stage in Figure 6.
t_{solve}	The time in seconds required to execute the solve stage in Figure 7.
M	The amount of RAM used in the build stage in MB.

The table also reports the memory requirements in terms of number of the number of double precision reals that need to be stored per degree of freedom in the discretization.

The high-order version of the method ($p = 41$) was also capable of performing a high accuracy solve with only six points per wave-length. The results are reported in Table 2.

We see that increasing the spectral order is very beneficial for improving accuracy. However, the speed deteriorates and the memory requirements increase as p grows. Choosing $p = 21$ appears to be a good compromise.

For comparison, Table 3 describes the computational results obtained when using a non-composite method with a $p \times p$ tensor product grid of Chebyshev nodes. For a domain of size 5.1×5.1 wave-lengths, this method performs very well. For an intermediate domain of size 25.1×25.1 wave-lengths, this method just barely converges before we run out of memory. For a large domain of size 100.1×100.1 wave-lengths, we attain no accuracy on the largest grid our computer could handle.

Table 4 shows how the accuracy in the spectral composite method depends on the order p of the local grids, and on the panel size $h = 1/n$ as p and n are increased for a fixed wave-number. We observe good convergence and stability for both h and p refinement. Figure 8 shows the time required for the experiments reported in Table 4. We see that for a small problem (of size 5.1×5.1 wave-lengths), the single-panel spectral method excels. On the other hand, for the large problem

p	N	N_{wave}	t_{build} (sec)	t_{solve} (sec)	E_{pot}	E_{grad}	M (MB)	M/N (reals/DOF)
6	6561	6.7	0.28	0.0047	8.02105e-03	3.06565e-01	2.8	56.5
6	25921	13.3	0.96	0.0184	1.67443e-02	1.33562e+00	12.7	64.2
6	103041	26.7	4.42	0.0677	3.60825e-02	5.46387e+00	56.2	71.5
6	410881	53.3	20.23	0.2397	3.39011e-02	1.05000e+01	246.9	78.8
6	1640961	106.7	88.73	0.9267	7.48385e-01	4.92943e+02	1075.0	85.9
11	6561	6.7	0.16	0.0019	2.67089e-05	1.08301e-03	2.9	58.0
11	25921	13.3	0.68	0.0073	5.30924e-05	4.34070e-03	13.0	65.7
11	103041	26.7	3.07	0.0293	1.01934e-04	1.60067e-02	57.4	73.0
11	410881	53.3	14.68	0.1107	1.07747e-04	3.49637e-02	251.6	80.2
11	1640961	106.7	68.02	0.3714	2.17614e-04	1.37638e-01	1093.7	87.4
21	6561	6.7	0.23	0.0011	2.56528e-10	1.01490e-08	4.4	87.1
21	25921	13.3	0.92	0.0044	5.24706e-10	4.44184e-08	18.8	95.2
21	103041	26.7	4.68	0.0173	9.49460e-10	1.56699e-07	80.8	102.7
21	410881	53.3	22.29	0.0727	1.21769e-09	3.99051e-07	344.9	110.0
21	1640961	106.7	99.20	0.2965	1.90502e-09	1.24859e-06	1467.2	117.2
21	6558721	213.3	551.32	20.9551	2.84554e-09	3.74616e-06	6218.7	124.3
41	6561	6.7	1.50	0.0025	9.88931e-14	3.46762e-12	7.9	157.5
41	25921	13.3	4.81	0.0041	1.58873e-13	1.12883e-11	32.9	166.4
41	103041	26.7	18.34	0.0162	3.95531e-13	5.51141e-11	137.1	174.4
41	410881	53.3	75.78	0.0672	3.89079e-13	1.03546e-10	570.2	181.9
41	1640961	106.7	332.12	0.2796	1.27317e-12	7.08201e-10	2368.3	189.2

TABLE 1. Results from an experiment with a constant coefficient Helmholtz problem on a square. The boundary data were picked so that the analytic solution was known; as a consequence, the solution is smooth, and can be smoothly extended across the boundary. The wave-number was chosen to keep a constant of 12 discretization points per wave-length.

p	N	N_{wave}	t_{build} (sec)	t_{solve} (sec)	E_{pot}	E_{grad}	M (MB)	M/N (reals/DOF)
41	6561	13.3	1.30	0.0027	1.54407e-09	1.78814e-07	7.9	157.5
41	25921	26.7	4.40	0.0043	1.42312e-08	2.35695e-06	32.9	166.4
41	103041	53.3	17.54	0.0199	1.73682e-08	5.84193e-06	137.1	174.4
41	410881	106.7	72.90	0.0717	2.28475e-08	1.51575e-05	570.2	181.9
41	1640961	213.3	307.37	0.3033	4.12809e-08	5.51276e-05	2368.3	189.2

TABLE 2. This table illustrates the same situation as Table 1, but now κ is increased twice as fast (so that we keep only 6 points per wave-length).

(of size 100.1×100.1 wave-lengths), the single panel spectral method attains no accuracy at all. In contrast, the high order spectral composite methods rapidly converge to 10 digits of accuracy.

Remark 6.1. In the course of executing the numerical examples, the instability problem described in Section 5.3 was detected precisely once (for $p = 16$ and 12 points per wave length).

6.2. **Variable coefficient Helmholtz.** We solved the equation

$$\begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2 \left(1 - b(\boldsymbol{x})\right) u(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}) & \boldsymbol{x} \in \Gamma, \end{cases} \tag{6.5}$$

p	t_{build} (sec)	t_{solve} (sec)	M (MB)	E_{pot} $N_{\text{wave}} = 5.1$	$N_{\text{wave}} = 25.1$	$N_{\text{wave}} = 100.1$
21	0.0	0.0003	2.0	1.48e-01	2.51e-01	6.46e-02
41	0.4	0.0026	35.3	3.39e-12	6.96e-01	1.76e-02
61	2.6	0.0166	184.9	6.42e-13	1.73e+00	2.00e-02
81	11.9	0.0461	594.3	4.28e-13	1.59e-01	1.16e+00
101	39.9	0.1151	1465.8	1.32e-12	2.00e-08	2.71e-01
121	112.1	0.2205	3059.9	1.84e-12	4.89e-13	4.96e-01

TABLE 3. Accuracies resulting from solving the constant coefficient Helmholtz equation (6.1) using a $p \times p$ tensor product grid of Chebyshev nodes. The accuracy is the sup norm error E_{pot} as defined by (6.3), given for three different domain sizes.

		$n = 2$	$n = 4$	$n = 8$	$n = 16$	$n = 32$	$n = 64$	$n = 128$	$n = 256$
$N_{\text{wave}} = 5.1$	$p = 6$	1.1e+00	2.8e-01	2.8e-02	3.0e-01	5.7e-03	3.6e-04	2.3e-05	1.4e-06
	$p = 11$	1.3e-01	5.7e-03	4.0e-05	6.5e-08	8.9e-11	6.2e-11	3.1e-10	—
	$p = 21$	1.4e-07	4.3e-12	1.2e-12	1.2e-11	6.0e-11	1.6e-10	—	—
	$p = 41$	6.8e-13	4.7e-12	1.7e-11	5.8e-11	—	—	—	—
$N_{\text{wave}} = 25.1$	$p = 6$	1.3e-01	1.2e-01	1.8e-01	1.3e+00	5.8e-01	3.0e-02	3.2e-03	2.3e-04
	$p = 11$	1.4e-01	2.3e-01	3.4e-01	2.7e-02	4.9e-05	1.1e-07	1.4e-10	1.4e-11
	$p = 21$	1.7e+00	1.4e+00	1.3e-05	2.8e-10	8.0e-14	2.8e-12	1.1e-12	—
	$p = 41$	9.8e-01	3.6e-10	2.9e-13	9.4e-13	5.4e-12	—	—	—
$N_{\text{wave}} = 100.1$	$p = 6$	6.2e-02	6.0e-02	6.9e-02	6.7e-02	1.0e-01	6.0e+00	6.2e-01	7.8e-02
	$p = 11$	6.8e-02	6.4e-02	8.1e-02	1.3e-01	5.2e-01	7.8e-02	4.1e-04	1.0e-06
	$p = 21$	6.7e-02	1.5e-01	3.5e+00	1.2e+00	8.5e-05	2.5e-09	2.1e-12	—
	$p = 41$	8.4e-01	9.2e-01	5.2e-01	4.4e-09	1.7e-12	—	—	—

TABLE 4. Accuracies resulting from solving the constant coefficient Helmholtz equation (6.1) on a domain of size $N_{\text{wave}} \times N_{\text{wave}}$. The discretization used $n \times n$ panels and a $p \times p$ tensor product Chebyshev grid on each panel (so that the panel size h satisfies $h = 1/n$). The accuracy given is the sup norm error E_{pot} as defined by (6.3).

where $\Omega = [0, 1]^2$, where $\Gamma = \partial\Omega$, and where

$$b(\boldsymbol{x}) = \left(\sin(4\pi x_1)\, \sin(4\pi x_2)\right)^2 .$$

The Helmholtz parameter was kept fixed at $\kappa = 80$, corresponding to a domain size of 12.7×12.7 wave lengths. The boundary data was given by

$$f(\boldsymbol{x}) = \cos(8x_1)\, \big(1 - 2x_2\big).$$

The equation (6.5) was discretized and solved as described in Section 6.1. The speed and memory requirements for this computation are exactly the same as for the example in Section 6.1 (they do not depend on what equation is being solved), so we now focus on the accuracy of the method. We do not know of an exact solution, and therefore report pointwise convergence. Letting u_N denote the value of u computed using N degrees of freedom, we used

$$E_N^{\text{int}} = u_N(\hat{\boldsymbol{x}}) - u_{4N}(\hat{\boldsymbol{x}})$$

as an estimate for the pointwise error in u at the point $\hat{\boldsymbol{x}} = (0.75,\, 0.25)$. We analogously estimated convergence of the normal derivative at the point $\hat{\boldsymbol{y}} = (0.75,\, 0.00)$ by measuring

$$E_N^{\text{bnd}} = w_N(\hat{\boldsymbol{y}}) - w_{4N}(\hat{\boldsymbol{y}}).$$

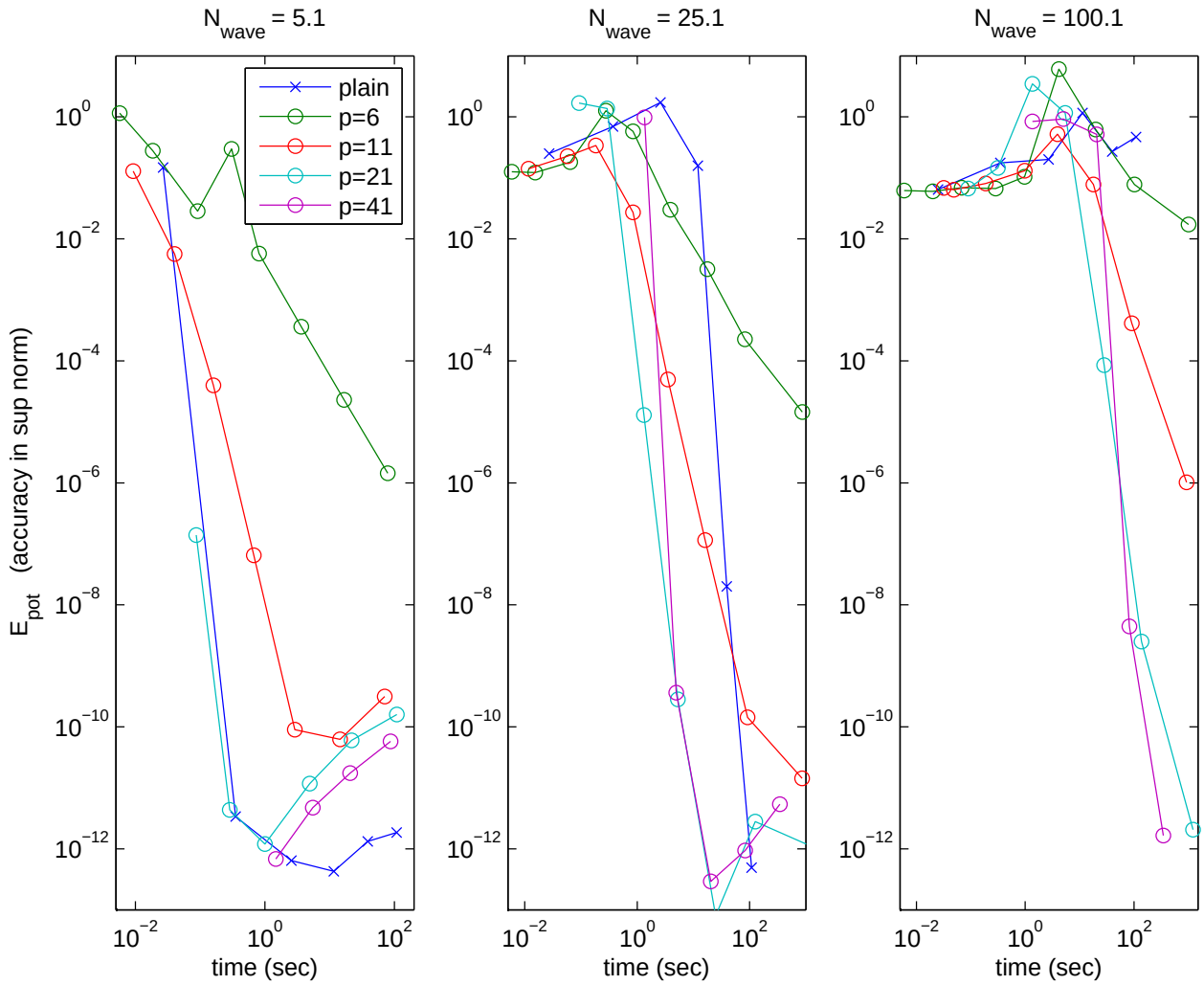


FIGURE 8. Computational accuracy is plotted against computational time (t_{build}). The constant coefficient Helmholtz equation (6.1) was solved on a domain of size $N_{\text{wave}} \times N_{\text{wave}}$ wave-lengths for three different values of N_{wave}. The lines with crosses correspond to a single-domain spectral method, while the lines with circles correspond to the composite scheme with $p \times p$ local grids for $p = 6$, 11, 21, 41.

The results are reported in Table 5. Table 6 reports the results from an analogous experiment, but now for a domain of size 102×102 wave-lengths.

We observe that accuracy is almost as good as for the constant coefficient case. Ten digits of accuracy is easily attained, but getting more than that seems challenging; increasing N further leads to no improvement in accuracy. The method appears to be stable in the sense that nothing bad happens when N is either too large or too small.

6.3. **L-shaped domain.** We solved the equation

$$\begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2\, u(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}) & \boldsymbol{x} \in \Gamma, \end{cases} \tag{6.6}$$

where Ω is the L-shaped domain $\Omega = [0,2]^2 \backslash [1,2]^2$ shown in Figure 9(a), and where the Helmholtz parameter κ is held fixed at $\kappa = 40$, making the domain 12.7×12.7 wave-lengths large. The pointwise

p	N	pts per wave	$u_N(\hat{\boldsymbol{x}})$	E_N^{int}	$w_N(\hat{\boldsymbol{y}})$	E_N^{bnd}
6	6561	6.28	-2.505791196753718	-2.457e-01	-661.0588680825	-8.588e+03
6	25921	12.57	-2.260084219562163	1.676e-01	7926.8096554095	8.141e+04
6	103041	25.13	-2.427668162910011	1.779e-02	-73484.9989261573	-3.894e+04
6	410881	50.27	-2.445455646843485	1.233e-03	-34547.6403539568	-1.235e+03
6	1640961	100.53	-2.446688310709834	7.891e-05	-33313.0000081604	-7.627e+01
6	6558721	201.06	-2.446767218259172		-33236.7252190062	
11	6561	6.28	-2.500353149793093	-5.375e-02	-27023.0713474340	6.524e+03
11	25921	12.57	-2.446599788642489	1.728e-04	-33547.3621639994	-3.153e+02
11	103041	25.13	-2.446772604281610	-9.465e-08	-33232.0940315585	-4.754e-01
11	410881	50.27	-2.446772509631734	3.631e-10	-33231.6186528531	-7.331e-04
11	1640961	100.53	-2.446772509994819		-33231.6179197169	
21	6561	6.28	-2.448236804078803	-1.464e-03	-32991.4583727724	2.402e+02
21	25921	12.57	-2.446772430608166	7.976e-08	-33231.6118304666	5.984e-03
21	103041	25.13	-2.446772510369452	5.893e-11	-33231.6178142514	-5.463e-06
21	410881	50.27	-2.446772510428384	2.957e-10	-33231.6178087887	-2.792e-05
21	1640961	100.53	-2.446772510724068		-33231.6177808723	
41	6561	6.28	-2.446803898373796	-3.139e-05	-33233.0037457220	-1.386e+00
41	25921	12.57	-2.446772510320572	1.234e-10	-33231.6179029824	-8.940e-05
41	103041	25.13	-2.446772510443995	2.888e-11	-33231.6178135860	-1.273e-05
41	410881	50.27	-2.446772510472872	7.731e-11	-33231.6178008533	-4.668e-05
41	1640961	100.53	-2.446772510550181		-33231.6177541722	

TABLE 5. Results from a variable coefficient Helmholtz problem on a domain of size 12.7×12.7 wave-lengths.

p	N	pts per wave	$u_N(\hat{\boldsymbol{x}})$	E_N^{int}	$w_N(\hat{\boldsymbol{y}})$	E_N^{bnd}
21	6561	0.79	0.007680026148649	4.085e-03	3828.84075823538	6.659e+03
21	25921	1.57	0.003595286353011	1.615e+00	-2829.88055527014	-1.791e+02
21	103041	3.14	-1.611350573683137	1.452e+00	-2650.80640712917	-5.951e+03
21	410881	6.28	-3.063762877533994	4.557e-03	3299.72573600854	-7.772e+00
21	1640961	12.57	-3.068320356836451	-7.074e-08	3307.49786015114	9.592e-05
21	6558721	25.13	-3.068320286093162		3307.49776422768	
41	6561	0.79	-0.000213617359480	-1.608e-01	-833.919575889393	-1.006e+03
41	25921	1.57	0.160581838547352	-7.415e-01	171.937456004515	-1.797e+03
41	103041	3.14	0.902057033817060	3.970e+00	1969.187023322940	-1.338e+03
41	410881	6.28	-3.068320045777766	2.405e-07	3307.497852217234	8.792e-05
41	1640961	12.57	-3.068320286282830	(-1.897e-10)	3307.497764294187	(6.651e-8)

TABLE 6. Results from a variable coefficient Helmholtz problem on a domain of size 102×102 wave-lengths.

errors were estimated at the points

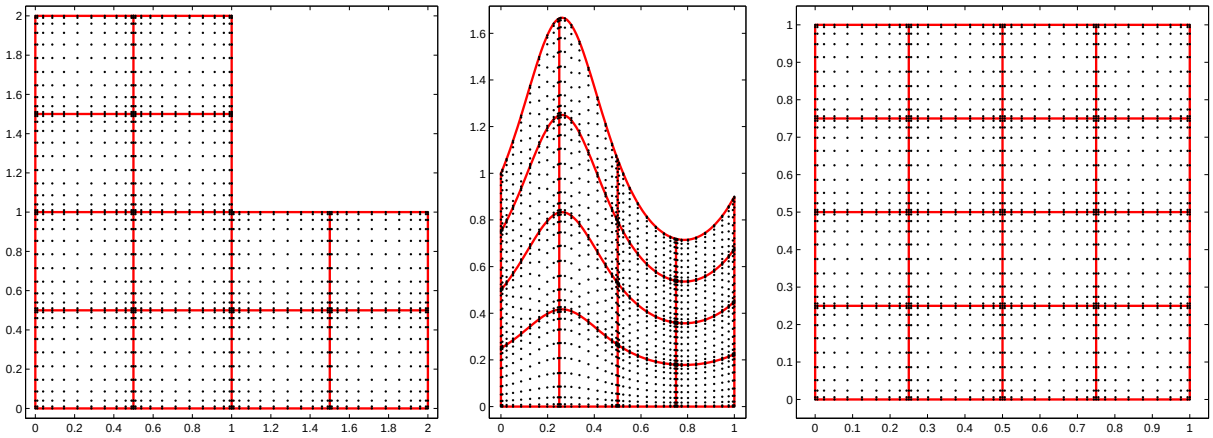
$$\hat{\boldsymbol{x}} = (0.75,\, 0.75), \qquad \text{and} \qquad \hat{\boldsymbol{y}} = (1.25,\, 1.00),$$

via

$$E_N^{\text{int}} = u_N(\hat{\boldsymbol{x}}) - u_{4N}(\hat{\boldsymbol{x}}), \qquad \text{and} \qquad E_N^{\text{bnd}} = w_N(\hat{\boldsymbol{y}}) - w_{4N}(\hat{\boldsymbol{y}}).$$

The results are given in Table 7.

We observe that the errors are in this case significantly larger than they were for square domains. This is presumably due to the fact that the solution u is singular near the re-entrant corner in the L-shaped domain. (When boundary conditions corresponding to an exact solution like (6.2) are



(a) (b) (c)

FIGURE 9. (a) The domain Ω in Section 6.3. (b) The domain Ψ in Section 6.4. (c) The domain Ω in Section 6.4.

p	N	pts per wave	$u_N(\hat{\boldsymbol{x}})$	E_N^{int}	$w_N(\hat{\boldsymbol{y}})$	E_N^{bnd}
6	19602	12.57	8.969213152495405	2.258e+00	226.603823940515	8.748e+01
6	77602	25.13	6.711091204119065	1.317e-01	139.118986915759	4.949e+00
6	308802	50.27	6.579341284597024	8.652e-03	134.169908546083	3.261e-01
6	1232002	100.53	6.570688999911585		133.843774958376	
11	19602	12.57	6.571117172871830	9.613e-04	133.865552472382	3.851e-02
11	77602	25.13	6.570155895761215	5.154e-05	133.827043929015	5.207e-03
11	308802	50.27	6.570104356719250	1.987e-05	133.821836691967	2.052e-03
11	1232002	100.53	6.570084491282650		133.819785089497	
21	19602	12.57	6.570152809642857	4.905e-05	133.898328735897	7.663e-02
21	77602	25.13	6.570103763348836	1.951e-05	133.821703687416	1.943e-03
21	308802	50.27	6.570084254517955	7.743e-06	133.819760759394	7.996e-04
21	1232002	100.53	6.570076511737839		133.818961147570	

TABLE 7. Results from a constant coefficient Helmholtz problem on an L-shaped domain of size 12.7×12.7 wave-lengths.

imposed, the method is just as accurate as it is for the square domain.) Nevertheless, the method easily attains solutions with between four and five correct digits.

6.4. **A curved domain.** A key feature of a composite, or "multi-domain," spectral method such as the one described here is its ability to accommodate a range of different domains. We illustrate this capability by considering a simple curved domain Ψ given by an analytic parameterization over the unit square $\Omega = [0,1]^2$. We solve the constant coefficient Helmholtz equation

$$\begin{cases} -\Delta u(\boldsymbol{y}) - \kappa^2\, u(\boldsymbol{y}) = 0 & \boldsymbol{y} \in \Psi, \\ u(\boldsymbol{y}) = f(\boldsymbol{y}) & \boldsymbol{y} \in \partial\Psi, \end{cases} \tag{6.7}$$

by mapping the equation to a variable coefficient equation on our reference domain Ω. We consider the simple domain Ω given in Figure 9(b) which is parameterized as

$$\Psi = \left\{ (y_1,\, y_2) :\, 0 \le y_1 \le 1 \quad 0 \le y_2 \le \frac{1}{\psi(y_1)} \right\} = \left\{ \left(x_1,\, \frac{x_2}{\psi(x_1)} \right) \,:\, (x_1, x_2) \in \Omega \right\},$$

	Exact solution known		*Dirichlet data* $f = 1$		
N	E_{pot}	E_{grad}	$E_N^{(1)}$	$E_N^{(2)}$	$E_N^{(3)}$
25921	2.12685e+02	3.55772e+04	2.24618e-01	4.99854e-01	6.69023e-01
103041	3.29130e-01	5.89976e+01	1.10143e-02	5.28238e-03	6.14890e-02
410881	1.40813e-05	1.98907e-03	4.57900e-06	2.18438e-06	1.13415e-05
1640961	7.22959e-10	1.17852e-07	5.12914e-07	1.67971e-06	4.97764e-06
3690241	1.63144e-09	2.26204e-07	—	—	—

TABLE 8. Errors in solving the Helmholtz problem (6.7) on a curved domain of size 35×50 wavelengths (see Figure 9(b)). Local Chebyshev grids with 21×21 points were used. The errors reported are defined by (6.3), (6.4), and (6.9).

where, in our computational examples, $\psi(y_1) = 1 - 0.3\,\sin(6\,y_1)$. A simple application of the chain rules (see Appendix B) shows that the Helmholtz equation (6.7) takes the form

$$\frac{\partial^2 u}{\partial x_1^2} + \frac{2\psi'(x_1)\,x_2}{\psi(x_1)}\frac{\partial^2 u}{\partial x_1 \partial x_2} + \left(\frac{x_2^2\psi'(x_1)^2}{\psi(x_1)^2} + \psi(x_1)^2\right)\frac{\partial^2 u}{\partial x_2^2} + \frac{x_2\psi''(x_1)}{\psi(x_1)}\frac{\partial u}{\partial x_2} + k^2 u = 0, \qquad \boldsymbol{x} \in \Omega. \tag{6.8}$$

We solved equation (6.8) for $k = 220$ which corresponds to a domain Ψ of size roughly 35×50 wave-lengths. We used local Chebyshev grids with 21×21 points, and ran two different experiments: In the first experiment, we used Dirichlet data corresponding to an analytically known solution $u_{\text{exact}}(\boldsymbol{y}) = \cos(0.6\,\kappa\,y_1 + 0.8\,\kappa\,y_2)$. For this case, we computed the errors E_{pot} in the potential and E_{grad} in the boundary gradients, as defined by (6.3) and (6.4). In the second experiment, we solved (6.8) for constant Dirichlet data $f = 1$. In this case, we used a point-wise error estimate

$$E_N^{(i)} = (1/M)\,|u_{\text{computed}}(\boldsymbol{x}^{(i)}) - u_{\text{ref}}(\boldsymbol{x}^{(i)})|, \tag{6.9}$$

where $M = 87.1$ is an estimate for $||u||_\infty$, where $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}\}$ are three fixed observation points, and where u_{ref} is the solution resulting from the finest mesh. We used:

$\boldsymbol{x}^{(1)} = (0.125, 0.875)$	$\boldsymbol{x}^{(2)} = (0.75, 0.25)$	$\boldsymbol{x}^{(3)} = (0.875, 0.875)$
$u_{\text{ref}}(\boldsymbol{x}^{(1)}) = -14.10206115033$	$u_{\text{ref}}(\boldsymbol{x}^{(2)}) = -12.09692774593$	$u_{\text{ref}}(\boldsymbol{x}^{(3)}) = 53.784389296457$

The results are given in Table 8. We observe that the method handles this situation very well, but that we lose a couple of digits worth of accuracy compared to a square domain.

6.5. **Convection diffusion.** We solved the equation

$$\begin{cases} -\Delta u(\boldsymbol{x}) - 1000\,[\partial_2 u](\boldsymbol{x}) = 0 & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}) & \boldsymbol{x} \in \Gamma, \end{cases} \tag{6.10}$$

where $\Omega = [0,1]^2$, where $\Gamma = \partial\Omega$, and where the boundary data was given by $f(\boldsymbol{x}) = \cos(x_1)\,e^{x_2}$. The equation (6.5) was discretized and solved as described in Section 6.1.

The pointwise errors were estimated at the points

$$\hat{\boldsymbol{x}} = (0.75,\, 0.25), \qquad \text{and} \qquad \hat{\boldsymbol{y}} = (0.75,\, 0.00),$$

via

$$E_N^{\text{int}} = u_N(\hat{\boldsymbol{x}}) - u_{4N}(\hat{\boldsymbol{x}}), \qquad \text{and} \qquad E_N^{\text{bnd}} = w_N(\hat{\boldsymbol{y}}) - w_{4N}(\hat{\boldsymbol{y}}).$$

The results are given in Table 9. Table 10 reports results from an analogous experiment, but now with the strength of the convection term further increased by a factor of 10.

We observe that the method has no difficulties resolving steep gradients, and that moderate order methods ($p = 11$) perform very well here.

p	N	$u_N(\hat{\boldsymbol{x}})$	E_N^{int}	$w_N(\hat{\boldsymbol{y}})$	E_N^{bnd}
11	25921	1.987126286905920	-3.191e-04	1255.25512379751	-7.191e-03
11	103041	1.987445414657945	3.979e-13	1255.26231503666	-6.529e-04
11	410881	1.987445414657547	2.455e-12	1255.26296795281	-1.889e-05
11	1640961	1.987445414655092		1255.26298684450	
21	25921	1.987076984861468	-3.684e-04	1255.26075989546	-2.186e-03
21	103041	1.987445414658047	-3.009e-13	1255.26294637880	-4.054e-05
21	410881	1.987445414658348	-2.600e-13	1255.26298691798	-7.881e-08
21	1640961	1.987445414658608		1255.26298699680	
41	25921	1.988004762686629	5.593e-04	1255.26290210213	-8.478e-05
41	103041	1.987445414657579	-9.706e-13	1255.26298687891	-1.178e-07
41	410881	1.987445414658550	-1.237e-12	1255.26298699669	-1.636e-09
41	1640961	1.987445414659787		1255.26298699832	

TABLE 9. Errors for the convection diffusion problem (6.10).

p	N	$u_N(\hat{\boldsymbol{x}})$	E_N^{int}	$w_N(\hat{\boldsymbol{y}})$	E_N^{bnd}
21	25921	1.476688750775769	-4.700e-01	13002.9937044202	4.325e+02
21	103041	1.946729131937971	-4.206e-02	12570.4750256324	-7.862e-03
21	410881	1.988785675941193	-1.716e-06	12570.4828877374	-4.900e-03
21	1640961	1.988787391699051	(6.719e-13)	12570.4877875310	(-4.411e-04)
41	25921	2.587008191566030	6.407e-01	13002.1084152522	4.316e+02
41	103041	1.946284950165041	-4.250e-02	12570.4835546978	-2.618e-03
41	410881	1.988785277235741	-2.114e-06	12570.4861729647	-2.127e-03
41	1640961	1.988787391699218		12570.4882994934	

TABLE 10. Errors for a convection diffusion problem similar to (6.10), but now for the even more convection dominated operator $A = -\Delta - 10\,000\,\partial_2$.

7. EXTENSIONS

7.1. **Linear complexity algorithms.** In discussing the asymptotic complexity of the scheme in Section 5.1 we distinguish between the case where N is increased for a fixed wave-number κ, and the case where $\kappa \sim N^{0.5}$ to keep the number of degrees of freedom per wavelength constant.

Let us first discuss the case where the wave-number κ is kept fixed as N is increased. In this situation, the matrices $\mathbf{U}^\tau$ will for the parent nodes become highly rank deficient (to finite precision). By factoring these matrices in the build stage, the solve stage can be reduced to $O(N)$ complexity. Moreover, the off-diagonal blocks of the matrices $\mathbf{V}^\tau$ and $\mathbf{W}^\tau$ will also be of low numerical rank; technically, they can efficiently be represented in data sparse formats such as the $\mathcal{H}$-matrix format [11], or the *Hierarchically Block-Separable*-format of [9, 35, 16]. Exploiting the internal structure in these matrices, the complexity of the build stage can be reduced from $O(N^{1.5})$ to $O(N)$. This acceleration is analogous to what was done for classical nested dissection for finite-element and finite-difference matrices in [8, 20, 29, 34]. Note that while the acceleration of the solve phase is trivial, it takes some work to exploit the more complicated structure in $\mathbf{V}^\tau$ and $\mathbf{W}^\tau$.

Now consider the case where $\kappa \sim N^{0.5}$ as N increases. The numerical ranks of the matrices $\mathbf{U}^\tau$ and the off-diagonal blocks of $\mathbf{V}^\tau$ and $\mathbf{W}^\tau$ will in this situation grow in such a way that no reduction in asymptotic complexity can be expected. However, the matrices are in practice far from full rank, and exploiting this internal structure is likely to lead to substantial practical accelerations.

7.2. **Free space scattering problems in the plane.** Once you know the DtN operator for the inhomogeneous square, you can rapidly solve free space scattering problem. Consider the equation

$$-\Delta u(\boldsymbol{x}) - \kappa^2 \left(1 - b(\boldsymbol{x})\right) u(\boldsymbol{x}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \mathbb{R}^2.$$

Appropriate radiation conditions at infinity are imposed on u. We assume that there is a rectangle Ω such that b is compactly supported inside Ω, and that f is compactly supported outside Ω. The standard technique is to look for a solution u of the form $u = v + w$, where v is an *incoming field* and w is the *outgoing field*. The incoming field is defined by $v(\boldsymbol{x}) = [\phi_\kappa * f](\boldsymbol{x}) = \int_{\mathbb{R}^2} \phi_\kappa(\boldsymbol{x} - \boldsymbol{y})\, f(\boldsymbol{y})\, d\boldsymbol{y}$, where ϕ_κ is the fundamental solution to the free space Helmholtz problem. Then $-\Delta v - \kappa^2 v = f$, and since $b(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \in \Omega^{\rm c}$, we find that the outgoing potential w must satisfy

$$-\Delta w(\boldsymbol{x}) - \kappa^2 w(\boldsymbol{x}) = 0, \qquad \boldsymbol{x} \in \Omega^{\rm c}. \tag{7.1}$$

Now use the method in Section 5.1 to construct the DtN map T for the problem

$$-\Delta u(\boldsymbol{x}) - \kappa^2 \left(1 - b(\boldsymbol{x})\right) u(\boldsymbol{x}) = 0, \qquad \boldsymbol{x} \in \Omega.$$

Then we know that

$$v_n|_\Gamma + w_n|_\Gamma = T\left(v|_\Gamma + w|_\Gamma\right), \tag{7.2}$$

where v_n and w_n are normal derivatives of v and w, respectively. Now use boundary integral equation methods to construct the DtN map S for the problem (7.1) on the exterior domain $\Omega^{\rm c}$. Then

$$w_n|_\Gamma = S\, w|_\Gamma. \tag{7.3}$$

Combining (7.2) and (7.3) we find $v_n|_\Gamma + S\, w|_\Gamma = T\, v|_\Gamma + T\, w|_\Gamma$. In other words,

$$(S - T)\, w|_\Gamma = T\, v|_\Gamma - v_n|_\Gamma. \tag{7.4}$$

Observing that both $v|_\Gamma$ and $v_n|_\Gamma$ can be obtained directly from the explicit formula for v, and that S and T are now available, we see that $w|_\Gamma$ can be determined by solving (7.4).

7.3. **Problems in 3D.** The method described can after trivial modifications be applied to problems in $\mathbb{R}^3$. However, since the fraction of points located on interfaces will increase, the complexity of the build and the solve stages will be $O(N^2)$ and $O(N^{4/3})$, respectively. To attain a truly efficient 3D code, the use of the acceleration techniques described in Section 7.1 will likely become crucial.

8. Conclusions

The paper describes a composite spectral scheme for solving variable coefficient elliptic PDEs with smooth coefficients. The method is described for the case of simple domains such as squares and rectangles, but can be extended to other geometries. The method involves a *direct* solver and can in a single sweep solve problems for which state-of-the-art iterative methods require thousands of iterations. High order spectral approximations are used. As a result, potential fields can be computed to a relative precision of about 10^{-10} using twelve points per wave-length or less.

Numerical experiments indicate that the method is very fast. For a problem involving 1.6M degrees of freedom discretizing a Helmholtz problem on a domain of size 100×100 wavelengths, the build stage of the direct solver took less than 2 minutes on a laptop. Once the solution operator had been computed, the solve stage that given a vector of Dirichlet data on the boundary constructs the solution at all 1.6M tabulation points required only 0.3 seconds. The computed solution had a relative accuracy of 10^{-9}.

The asymptotic complexity of the method presented is $O(N^{1.5})$ for the construction of the solution operator, and $O(N \log N)$ for a solve once the solution operator has been created. For a situation where N is increased while the wave-number is kept fixed, it is possible to improve the asymptotic complexity to $O(N)$ for both the build and the solve stages (see Section 7.1); this work will be reported in a subsequent paper.

Acknowledgments: The author benefitted greatly from conversations with Vladimir Rokhlin of Yale university. Valuable suggestions were also made by the anonymous referees, and by Yu Chen (NYU), Adrianna Gillman (Dartmouth), Leslie Greengard (NYU), and Mark Tygert (NYU). The work was supported by the NSF under contracts 0748488 and 0941476, and by the Wenner-gren foundation. Most of the work was conducted during a sabbatical spent at the Department of Mathematics at Chalmers University and at the Courant Institute at NYU. The support from these institutions is gratefully acknowledged.

References

[1] M. Ainsworth and H.A. Wajid, *Dispersive and dissipative behavior of the spectral element method*, SIAM Journal on Numerical Analysis **47** (2009), no. 5, 3910–3937.

[2] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz, *Minimizing communication in numerical linear algebra*, SIAM Journal on Matrix Analysis and Applications **32** (2011), no. 3, 866–901.

[3] Yu Chen, *A fast, direct algorithm for the Lippmann-Schwinger integral equation in two dimensions*, Adv. Comput. Math. **16** (2002), no. 2-3, 175–190, Modeling and computation in optics and electromagnetics.

[4] Ran Duan and Vladimir Rokhlin, *High-order quadratures for the solution of scattering problems in two dimensions*, J. Comput. Physics **228** (2009), no. 6, 2152–2174.

[5] I.S. Duff, A.M. Erisman, and J.K. Reid, *Direct methods for sparse matrices*, Oxford, 1989.

[6] F. Ethridge and L. Greengard, *A new fast-multipole accelerated poisson solver in two dimensions*, SIAM Journal on Scientific Computing **23** (2001), no. 3, 741–760.

[7] A. George, *Nested dissection of a regular finite element mesh*, SIAM J. on Numerical Analysis **10** (1973), 345–363.

[8] A. Gillman, *Fast direct solvers for elliptic partial differential equations*, Ph.D. thesis, Applied Mathematics, University of Colorado at Boulder, 2011.

[9] Adrianna Gillman, Patrick Young, and Per-Gunnar Martinsson, *A direct solver $o(n)$ complexity for integral equations on one-dimensional domains*, Frontiers of Mathematics in China **7** (2012), 217–247, 10.1007/s11464-012-0188-3.

[10] IG Graham and IH Sloan, *Fully discrete spectral boundary integral methods for helmholtz problems on smooth closed surfaces in \mathbbr^3*, Numerische Mathematik **92** (2002), no. 2, 289–323.

[11] Lars Grasedyck and Wolfgang Hackbusch, *Construction and arithmetics of $\mathcal{H}$-matrices*, Computing **70** (2003), no. 4, 295–334.

[12] Leslie Greengard, Denis Gueyffier, Per-Gunnar Martinsson, and Vladimir Rokhlin, *Fast direct solvers for integral equations in complex three-dimensional domains*, Acta Numer. **18** (2009), 243–275.

[13] P. Haldenwang, G. Labrosse, Abboudi S., and M. Deville, *Chebyshev 3-d spectral and 2-d pseudospectral solvers for the helmholtz equation*, Journal of Computational Physics **55** (1984), no. 1, 115 – 128.

[14] M. Haney, R. Snieder, J.-P. Ampuero, and R. Hofmann, *Spectral element modeling of fault-plane reflections arising from fluid pressure distributions*, Geophys. J. Int. **170** (2007), no. 2, 933–951.

[15] J.S. Hesthaven, P.G. Dinesen, and J.P. Lynov, *Spectral collocation time-domain modeling of diffractive optical elements*, Journal of Computational Physics **155** (1999), no. 2, 287 – 306.

[16] K.L. Ho and L. Greengard, *A fast direct solver for structured linear systems by recursive skeletonization*, SIAM Journal on Scientific Computing **34** (2012), no. 5, 2507–2532.

[17] A. J. Hoffman, M. S. Martin, and D. J. Rose, *Complexity bounds for regular finite difference and finite element grids*, SIAM J. Numer. Anal. **10** (1973), 364–369.

[18] D.A. Kopriva, *A staggered-grid multidomain spectral method for the compressible navierstokes equations*, Journal of Computational Physics **143** (1998), no. 1, 125 – 158.

[19] Y.Y. Kwan and J. Shen, *An efficient direct parallel spectral-element solver for separable elliptic problems*, Journal of Computational Physics **225** (2007), no. 2, 1721–1735.

[20] Sabine Le Borne, Lars Grasedyck, and Ronald Kriemann, *Domain-decomposition based $\mathcal{H}$-LU preconditioners*, Domain decomposition methods in science and engineering XVI, Lect. Notes Comput. Sci. Eng., vol. 55, Springer, Berlin, 2007, pp. 667–674. MR 2334161

[21] P.G. Martinsson, *A high-order accurate discretization scheme for variable coefficient elliptic pdes in the plane with smooth solutions*, Jan. 2011, arXiv:1101.3383.

[22] P.G. Martinsson and V. Rokhlin, *A fast direct solver for boundary integral equations in two dimensions*, J. Comp. Phys. **205** (2005), no. 1, 1–23.

[23] O.Z. Mehdizadeh and M. Paraschivoiu, *Investigation of a two-dimensional spectral element method for helmholtz equation*, Journal of Computational Physics **189** (2003), no. 1, 111 – 129.

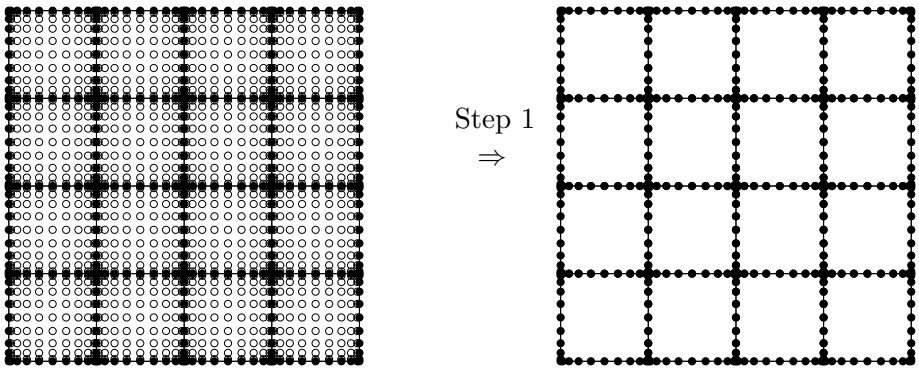
[24] J.M. Melenk and S. Sauter, *Convergence analysis for finite element discretizations of the helmholtz equation with dirichlet-to-neumann boundary conditions*, Math. Comp **79** (2010), no. 272, 1871–1914.

[25] A.T. Patera, *A spectral element method for fluid dynamics: Laminar flow in a channel expansion*, Journal of Computational Physics **54** (1984), no. 3, 468 – 488.
[26] H.P. Pfeiffer, L.E. Kidder, M.A. Scheel, and S.A. Teukolsky, *A multidomain spectral method for solving elliptic equations*, Computer physics communications **152** (2003), no. 3, 253–273.
[27] J. Poulson, B. Marker, J.R. Hammond, N.A. Romero, and R. van de Geijn, *Elemental: A new framework for distributed memory dense matrix computations*, ACM Trans. Math. Software, Note: to appear (2010).
[28] C. Pozrikides, *Finite and spectral element methods*, Taylor & Francis, 2005.
[29] P.G. Schmitz and L. Ying, *A fast direct solver for elliptic problems on general meshes in 2d*, Journal of Computational Physics **231** (2012), no. 4, 1314 – 1338.
[30] J. Shen and L.L. Wang, *Spectral approximation of the helmholtz equation with high wave numbers*, SIAM journal on numerical analysis **43** (2005), no. 2, 623–644.
[31] ———, *Analysis of a spectral-galerkin approximation to the helmholtz equation in exterior domains*, SIAM Journal on Numerical Analysis **45** (2007), no. 5, 1954–1978.
[32] Page Starr and Vladimir Rokhlin, *On the numerical solution of two-point boundary value problems. II*, Comm. Pure Appl. Math. **47** (1994), no. 8, 1117–1159.
[33] L.N. Trefethen, *Spectral methods in matlab*, SIAM, Philadelphia, 2000.
[34] Jianlin Xia, Shivkumar Chandrasekaran, Ming Gu, and Xiaoye S. Li, *Superfast multifrontal method for large structured linear systems of equations*, SIAM J. Matrix Anal. Appl. **31** (2009), no. 3, 1382–1411. MR 2587783 (2011c:65072)
[35] ———, *Fast algorithms for hierarchically semiseparable matrices*, Numerical Linear Algebra with Applications **17** (2010), no. 6, 953–976.
[36] B. Yang and J.S. Hesthaven, *Multidomain pseudospectral computation of maxwell's equations in 3-d general curvilinear coordinates*, Applied Numerical Mathematics **33** (2000), no. 14, 281 – 289.
[37] C. Zhu, G. Qin, and J. Zhang, *Implicit chebyshev spectral element method for acoustics wave equations*, Finite Elements in Analysis and Design **47** (2011), no. 2, 184 – 194.

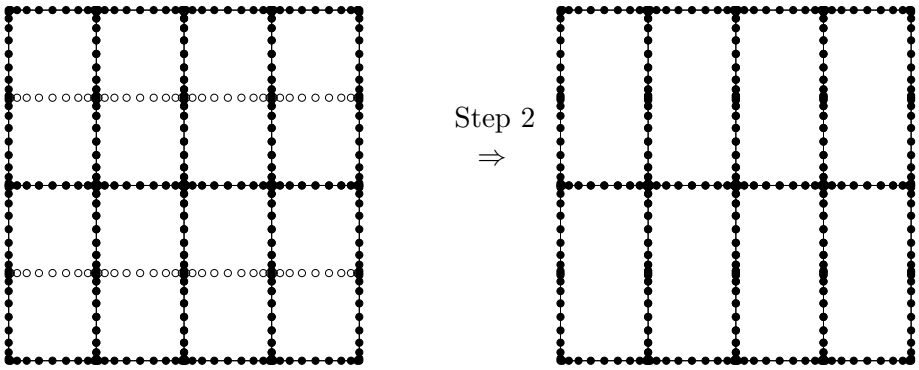
Appendix A. Graphical illustration of the hierarchical merge process

This section provides an illustrated overview of the hierarchical merge process described in detail in Section 5.1 and in Figure 6. The figures illustrate a situation in which a square domain $\Omega = [0,1]^2$ is split into 4×4 leaf boxes on the finest level, and an 8×8 spectral grid is used in each leaf.

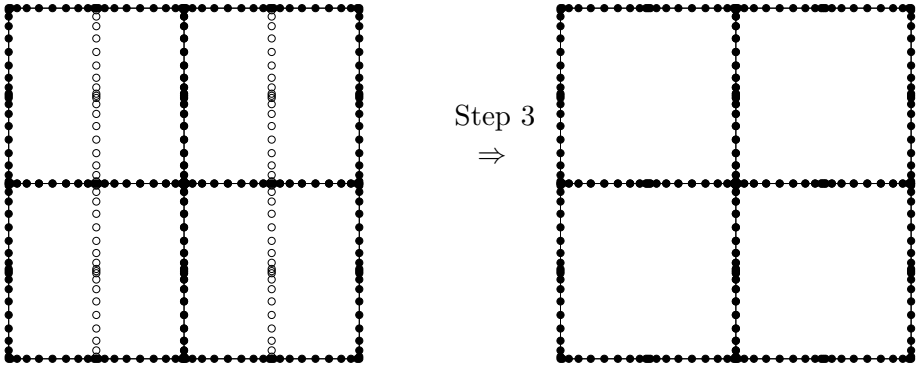
Step 1: Partition the box Ω into 16 small boxes that each holds an 8×8 mesh of Chebyshev nodes. For each box τ, identify the internal nodes (marked in white) and eliminate them as described in Section 3. Construct the solution operator $\mathbf{U}^\tau$, and the DtN operators $\mathbf{V}^\tau$ and $\mathbf{W}^\tau$.



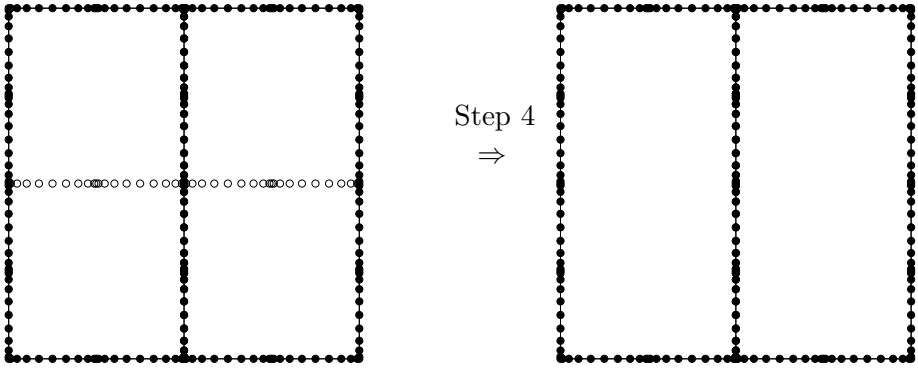
Step 2: Merge the small boxes by pairs as described in Section 4. The equilibrium equation for each rectangle is formed using the DtN operators of the two small squares it is made up of. The result is to eliminate the interior nodes (marked in white) of the newly formed larger boxes. Construct the solution operator $\mathbf{U}$ and the DtN matrices $\mathbf{V}$ and $\mathbf{W}$ for the new boxes.



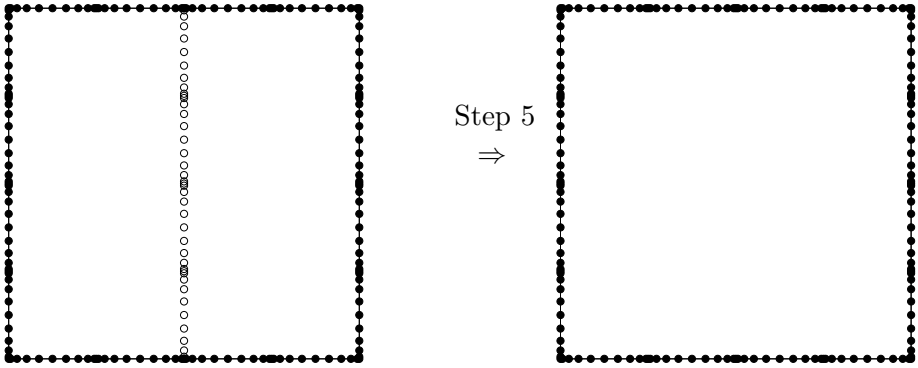
Step 3: Merge the boxes created in Step 2 in pairs, again via the process described in Section 4.



Step 4: Repeat the merge process once more.



Step 5: Merge one final time to obtain the DtN operator for the boundary of the whole domain.



Appendix B. Derivation of the change of variables formula

In this section, we provide the derivation of equation (6.8) of the Helmholtz equation in a curved domain after a change of variables. Recall that the starting point is the equation

$$\begin{cases} -\Delta u(\boldsymbol{y}) - \kappa^2\, u(\boldsymbol{y}) = 0 & \boldsymbol{y} \in \Psi, \\ u(\boldsymbol{y}) = f(\boldsymbol{y}) & \boldsymbol{y} \in \partial\Psi, \end{cases} \tag{B.1}$$

and that we seek to effect the change of variables in (B.1) from $\boldsymbol{y}$ to $\boldsymbol{x}$, where

$$\begin{aligned} x_1 &= y_1 \\ x_2 &= \psi(y_1)\, y_2. \end{aligned}$$

Now

$$\frac{\partial u}{\partial y_1} = \frac{\partial x_1}{\partial y_1}\frac{\partial u}{\partial x_1} + \frac{\partial x_2}{\partial y_1}\frac{\partial u}{\partial x_2} = \frac{\partial u}{\partial x_1} + \psi'(y_1) y_2 \frac{\partial u}{\partial x_2}. \tag{B.2}$$

Next

$$\begin{aligned} \frac{\partial^2 u}{\partial y_1^2} &= \frac{\partial x_1}{\partial y_1}\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial x_2}{\partial y_1}\frac{\partial^2 u}{\partial x_1 \partial x_2} + \psi''(y_1) y_2 \frac{\partial u}{\partial x_2} + \psi'(y_1) y_2 \frac{\partial x_1}{\partial y_1}\frac{\partial u}{\partial x_1 \partial x_2} + \psi'(y_1) y_2 \frac{\partial x_2}{\partial y_1}\frac{\partial u}{\partial x_2^2} \\ &= \frac{\partial^2 u}{\partial x_1^2} + 2\psi'(y_1) y_2 \frac{\partial u}{\partial x_1 \partial x_2} + \big(\psi'(y_1) y_2\big)^2 \frac{\partial u}{\partial x_2^2} + \psi''(y_1) y_2 \frac{\partial u}{\partial x_2}. \end{aligned} \tag{B.3}$$

Similarly

$$\frac{\partial u}{\partial y_2} = \frac{\partial x_1}{\partial y_2}\frac{\partial u}{\partial x_1} + \frac{\partial x_2}{\partial y_2}\frac{\partial u}{\partial x_2} = \psi(y_1)\frac{\partial u}{\partial x_2}. \tag{B.4}$$

And

$$\frac{\partial^2 u}{\partial y_2^2} = \psi(y_1)\frac{\partial x_1}{\partial y_2}\frac{\partial^2 u}{\partial x_1 \partial x_2} + \psi(y_1)\frac{\partial x_2}{\partial y_2}\frac{\partial^2 u}{\partial x_2^2} = \big(\psi(y_1)\big)^2 \frac{\partial^2 u}{\partial x_2^2}. \tag{B.5}$$

Combining, we find that the Helmholtz equation (6.7) takes the form

$$\frac{\partial^2 u}{\partial x_1^2} + 2\psi'(y_1)\, y_2 \frac{\partial^2 u}{\partial x_1 \partial x_2} + \big(y_2^2 \psi'(y_1)^2 + \psi(y_1)^2\big)\frac{\partial^2 u}{\partial x_2^2} + y_2 \psi''(y_1)\frac{\partial u}{\partial x_2} + k^2 u = 0, \qquad \boldsymbol{x} \in \Omega. \tag{B.6}$$

To obtain an equation with no $\boldsymbol{y}$'s at all, substitute $y_1 = x_1$ and $y_2 = x_2/\psi(x_1)$:

$$\frac{\partial^2 u}{\partial x_1^2} + \frac{2\psi'(x_1)\, x_2}{\psi(x_1)}\frac{\partial^2 u}{\partial x_1 \partial x_2} + \left(\frac{x_2^2 \psi'(x_1)^2}{\psi(x_1)^2} + \psi(x_1)^2\right)\frac{\partial^2 u}{\partial x_2^2} + \frac{x_2 \psi''(x_1)}{\psi(x_1)}\frac{\partial u}{\partial x_2} + k^2 u = 0, \qquad \boldsymbol{x} \in \Omega. \tag{B.7}$$

We recognize (B.7) as (6.8).

Remark B.1. Observe that formulas (B.2) and (B.4) can be used to compute the normal and tangential derivatives at a point at the boundary of Ψ, provided $\partial u/\partial x_1$ and $\partial u/\partial x_2$ are known.