Fast direct solvers for elliptic PDEs

Gunnar Martinsson
The University of Colorado at Boulder

PhD Students:

Tracy Babb

Adrianna Gillman (now at Dartmouth)

Nathan Halko (now at Spot Influence, LLC)

Sijia Hao

Patrick Young (now at GeoEye Inc.)

Postdoc:

Sergey Voronin (PACM graduate!)

Collaborators:

James Bremer (UC-Davis)

Alex Barnett (Dartmouth)

Eduardo Corona (NYU)

Leslie Greengard (NYU)

Terry Haut (LLNL)

Eric Michielssen (Michigan)

Vladimir Rokhlin (Yale)

Mark Tygert (NYU)

Denis Zorin (NYU)

Slides: On my webpage — google “Gunnar Martinsson”

The talk will describe "fast direct" techniques for solving the linear systems arising from the discretization of linear boundary value problems (BVPs) of the form

(BVP)
$$\begin{cases} A\,u(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ B\,u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where Ω is a domain in $\mathbb{R}^2$ or $\mathbb{R}^3$ with boundary Γ, and where A is an elliptic differential operator (constant coefficient, or not). Examples include:

- The equations of linear elasticity.
- Stokes' equation.
- Helmholtz' equation (at least at low and intermediate frequencies).
- Time-harmonic Maxwell (at least at low and intermediate frequencies).

Example: Poisson equation with Dirichlet boundary data:

$$\begin{cases} -\Delta\,u(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ \quad u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma. \end{cases}$$

Discretization of linear Boundary Value Problems

↙

Direct discretization of the differential operator via Finite Elements, Finite Differences, spectral composite methods, ...

↓

$N \times N$ discrete linear system.
Very large, sparse, ill-conditioned.

↓

Fast solvers:
iterative (multigrid), $O(N)$,
direct (nested dissection), $O(N^{3/2})$.

↘

Conversion of the BVP to a Boundary Integral Equation (BIE).

↓

Discretization of (BIE) using Nyström, collocation, BEM, ...

↓

$N \times N$ discrete linear system.
Moderate size, dense,
(often) well-conditioned.

↓

Iterative solver accelerated by fast matrix-vector multiplier, $O(N)$.

Discretization of linear Boundary Value Problems

↙

Direct discretization of the differential operator via Finite Elements, Finite Differences, spectral composite methods, ...

↓

$N \times N$ discrete linear system.
Very large, sparse, ill-conditioned.

↓

Fast solvers:
iterative (multigrid), $O(N)$,
direct (nested dissection), $O(N^{3/2})$.
$O(N)$ direct solvers.

↘

Conversion of the BVP to a Boundary Integral Equation (BIE).

↓

Discretization of (BIE) using Nyström, collocation, BEM, ...

↓

$N \times N$ discrete linear system.
Moderate size, dense,
(often) well-conditioned.

↓

Iterative solver accelerated by fast matrix-vector multiplier, $O(N)$.
$O(N)$ direct solvers.

What does a "direct" solver mean in this context?

Basically, it is a solver that is not "iterative" ...

Given a computational tolerance ε, and a linear system

$$\mathbf{A}\,\mathbf{u} = \mathbf{b}, \tag{2}$$

(where the system matrix $\mathbf{A}$ is often defined implicitly), a *direct solver* constructs an operator $\mathbf{S}$ such that

$$||\mathbf{A}^{-1} - \mathbf{S}|| \leq \varepsilon.$$

Then an approximate solution to (2) is obtained by simply evaluating

$$\mathbf{u}_{\text{approx}} = \mathbf{S}\,\mathbf{b}.$$

The matrix $\mathbf{S}$ is typically constructed in a *data-sparse format* (e.g. $\mathcal{H}$-matrix, HSS, etc) that allows the matrix-vector product $\mathbf{S}\,\mathbf{b}$ to be evaluated rapidly.

Variation: Find factors $\mathbf{B}$ and $\mathbf{C}$ such that $||\mathbf{A} - \mathbf{B}\,\mathbf{C}|| \leq \varepsilon$, and linear solves involving the matrices $\mathbf{B}$ and $\mathbf{C}$ are fast. (LU-decomposition, Cholesky, *etc*.)

"Iterative" versus "direct" solvers

Two classes of methods for solving an $N \times N$ linear algebraic system

$$\mathbf{A}\,\mathbf{u} = \mathbf{b}.$$

Iterative methods:

Examples: GMRES, conjugate gradients, Gauss-Seidel, *etc*.

Construct a sequence of vectors $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \ldots$ that (hopefully!) converge to the exact solution.

Many iterative methods access **A** only via its action on vectors.

Often require problem specific preconditioners.

High performance when they work well. $O(N)$ solvers.

Direct methods:

Examples: Gaussian elimination, LU factorizations, matrix inversion, *etc*.

Always give an answer. Deterministic.

Robust. No convergence analysis.

Great for multiple right hand sides.

Have often been considered too slow for high performance computing.

(Directly access elements or blocks of **A**.)

(Exact except for rounding errors.)

Advantages of direct solvers over iterative solvers:

1. Applications that require a very large number of solves for a fixed operator:
   - Molecular dynamics.
   - Scattering problems.
   - Optimal design. (Local updates to the system matrix are cheap.)

   A couple of orders of magnitude speed-up is often possible.

2. Solving problems intractable to iterative methods (singular values do not "cluster"):
   - Scattering problems near resonant frequencies.
   - Ill-conditioning due to geometry (elongated domains, percolation, etc).
   - Ill-conditioning due to lazy handling of corners, cusps, etc.
   - Finite element and finite difference discretizations.

   Scattering problems intractable to existing methods can (sometimes) be solved.

3. Direct solvers can be adapted to construct spectral decompositions:
   - Analysis of vibrating structures. Acoustics.
   - Buckling of mechanical structures.
   - Wave guides, bandgap materials, etc.

   Work in progress . . .

Advantages of direct solvers over iterative solvers, continued:

Perhaps most important: **Engineering considerations.**

Direct methods tend to be more robust than iterative ones.

This makes them more suitable for “black-box” implementations.

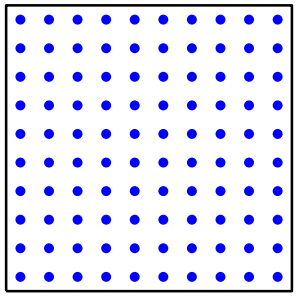
Commercial software developers appear to avoid implementing iterative solvers whenever possible. (Sometimes for good reasons.)

The effort to develop direct solvers aims to help in the development of general purpose software packages solving the basic linear boundary value problems of mathematical physics.

Outline of direct solver

All direct solvers to be described are based on hierarchical domain decomposition.

Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.



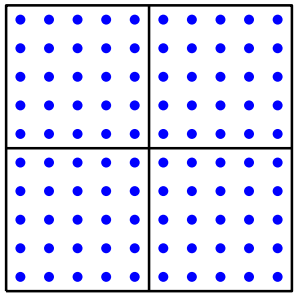
The original grid.

Outline of direct solver

All direct solvers to be described are based on hierarchical domain decomposition.

Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

Split the domain into "small" patches we call "leaves" (they will be organized in a tree).



The original grid.

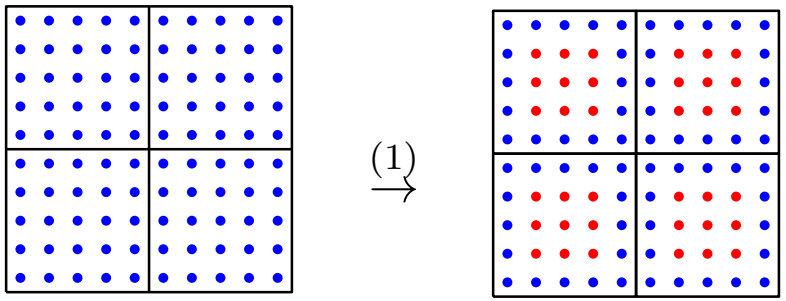
Outline of direct solver

All direct solvers to be described are based on hierarchical domain decomposition.

Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

Split the domain into "small" patches we call "leaves" (they will be organized in a tree).

On each leaf, compute by "brute force" a local solution operator (e.g. a DtN operator). This eliminates "internal" grid points from the computation. ("Static condensation.")



The original grid.

$\xrightarrow{(1)}$

Leaves reduced.

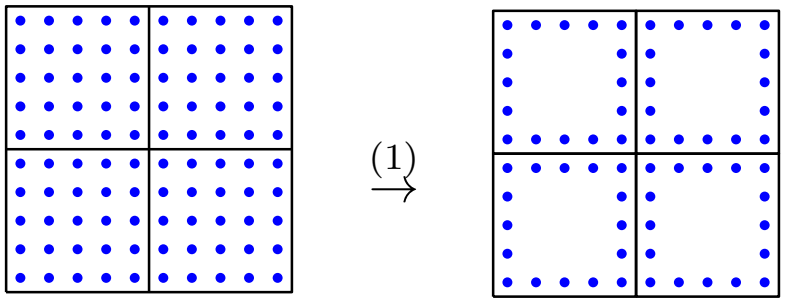
Outline of direct solver

All direct solvers to be described are based on hierarchical domain decomposition.

Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

Split the domain into "small" patches we call "leaves" (they will be organized in a tree).

On each leaf, compute by "brute force" a local solution operator (e.g. a DtN operator). This eliminates "internal" grid points from the computation. ("Static condensation.")



The original grid.

$\xrightarrow{(1)}$

Leaves reduced.

Outline of direct solver

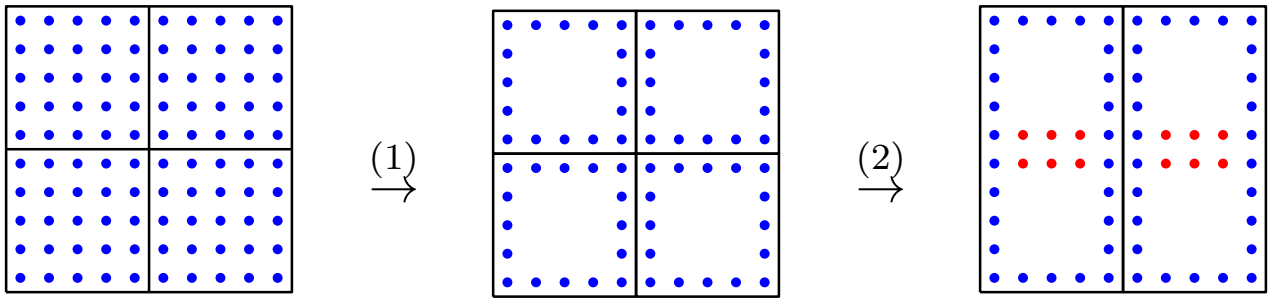
All direct solvers to be described are based on hierarchical domain decomposition.

Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

Split the domain into "small" patches we call "leaves" (they will be organized in a tree).

On each leaf, compute by "brute force" a local solution operator (e.g. a DtN operator). This eliminates "internal" grid points from the computation. ("Static condensation.")

Merge the leaves in pairs of two.



The original grid. *Leaves reduced.* *After merge.*

Outline of direct solver

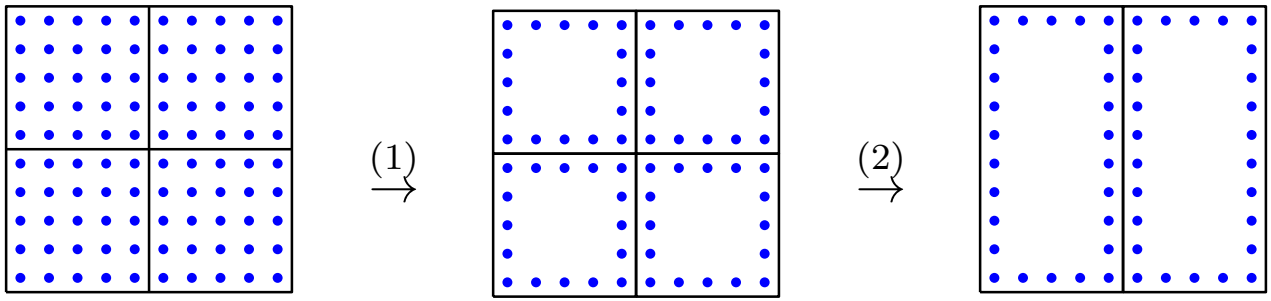
All direct solvers to be described are based on hierarchical domain decomposition.

Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

Split the domain into "small" patches we call "leaves" (they will be organized in a tree).

On each leaf, compute by "brute force" a local solution operator (e.g. a DtN operator). This eliminates "internal" grid points from the computation. ("Static condensation.")

Merge the leaves in pairs of two. For each pair, compute a local solution operator by combining the solution operators of the two leaves.



The original grid. *Leaves reduced.* *After merge.*

Outline of direct solver

All direct solvers to be described are based on hierarchical domain decomposition.

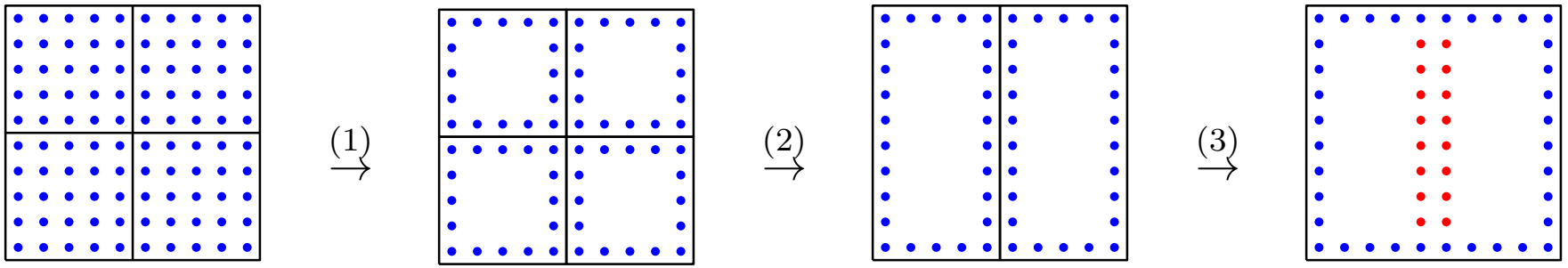
Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

Split the domain into "small" patches we call "leaves" (they will be organized in a tree).

On each leaf, compute by "brute force" a local solution operator (e.g. a DtN operator). This eliminates "internal" grid points from the computation. ("Static condensation.")

Merge the leaves in pairs of two. For each pair, compute a local solution operator by combining the solution operators of the two leaves.

Continue merging by pairs, organizing the domain in a tree of patches.



The original grid. *Leaves reduced.* *After merge.* *After merge.*

Outline of direct solver

All direct solvers to be described are based on hierarchical domain decomposition.

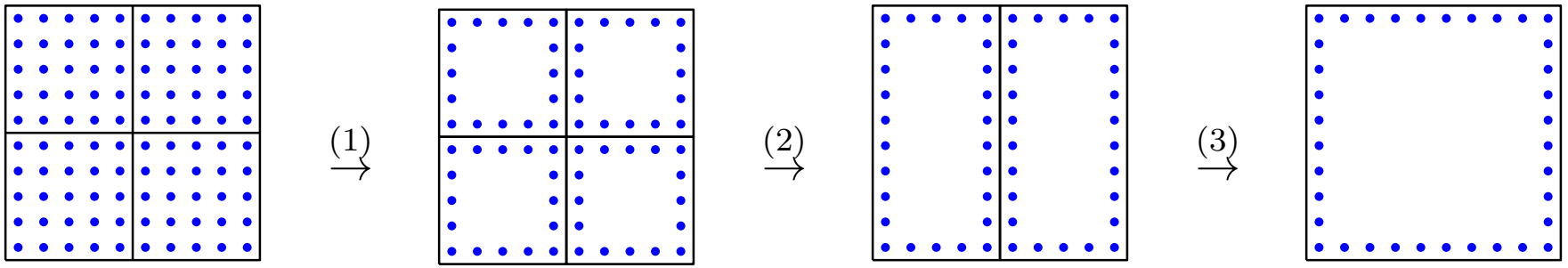
Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

Split the domain into "small" patches we call "leaves" (they will be organized in a tree).

On each leaf, compute by "brute force" a local solution operator (e.g. a DtN operator). This eliminates "internal" grid points from the computation. ("Static condensation.")

Merge the leaves in pairs of two. For each pair, compute a local solution operator by combining the solution operators of the two leaves.

Continue merging by pairs, organizing the domain in a tree of patches.



The original grid. *Leaves reduced.* *After merge.* *After merge.*

Outline of direct solver

All direct solvers to be described are based on hierarchical domain decomposition.

Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

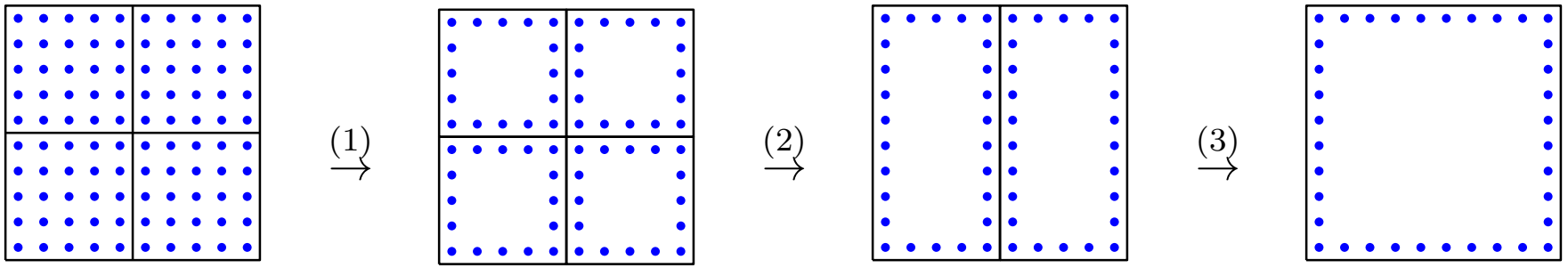
Split the domain into "small" patches we call "leaves" (they will be organized in a tree).

On each leaf, compute by "brute force" a local solution operator (e.g. a DtN operator). This eliminates "internal" grid points from the computation. ("Static condensation.")

Merge the leaves in pairs of two. For each pair, compute a local solution operator by combining the solution operators of the two leaves.

Continue merging by pairs, organizing the domain in a tree of patches.

When you reach the top level, perform a solve on the reduced problem by brute force.



The original grid. *Leaves reduced.* *After merge.* *After merge.*

Outline of direct solver

All direct solvers to be described are based on hierarchical domain decomposition.

Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

Split the domain into "small" patches we call "leaves" (they will be organized in a tree).

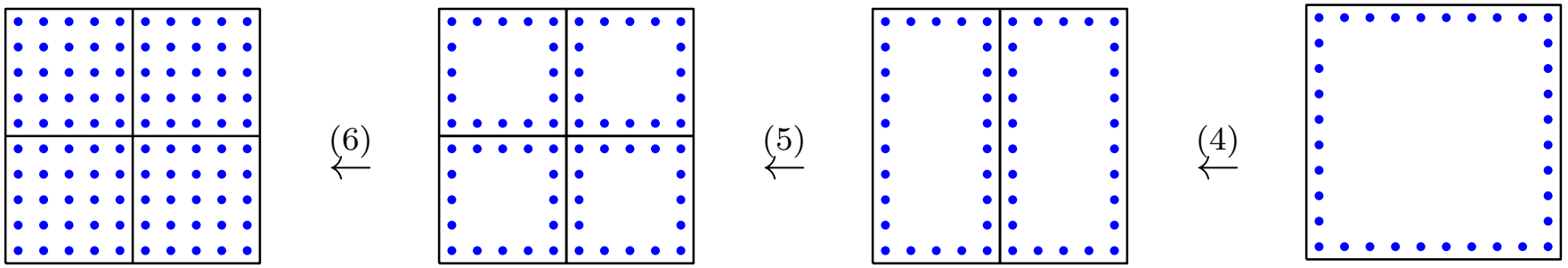
On each leaf, compute by "brute force" a local solution operator (e.g. a DtN operator). This eliminates "internal" grid points from the computation. ("Static condensation.")

Merge the leaves in pairs of two. For each pair, compute a local solution operator by combining the solution operators of the two leaves.

Continue merging by pairs, organizing the domain in a tree of patches.

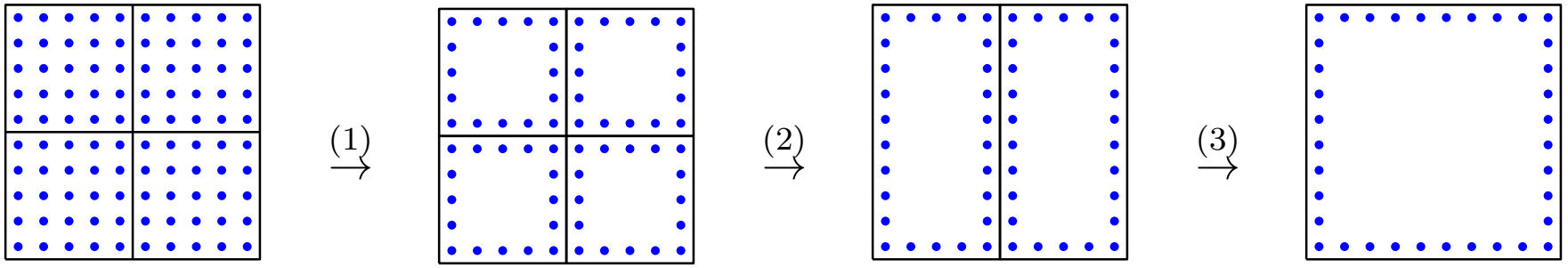
When you reach the top level, perform a solve on the reduced problem by brute force.

Then reconstruct the solution at all internal points via a downwards pass.



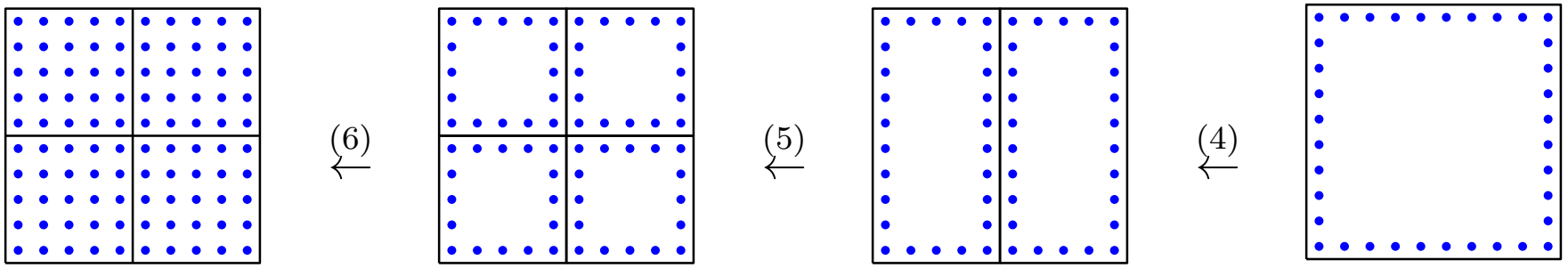
Full solution. *Solve.* *Solve.* *Top level solve.*

Upwards pass — build all solution operators:



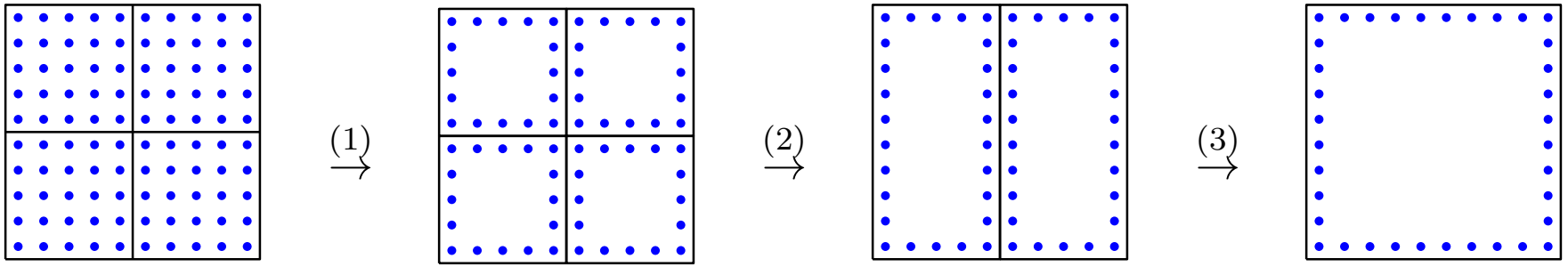
The original grid. *Leaves reduced.* *After merge.* *After merge.*

Downwards pass — solve for a particular data function (*very* fast!):



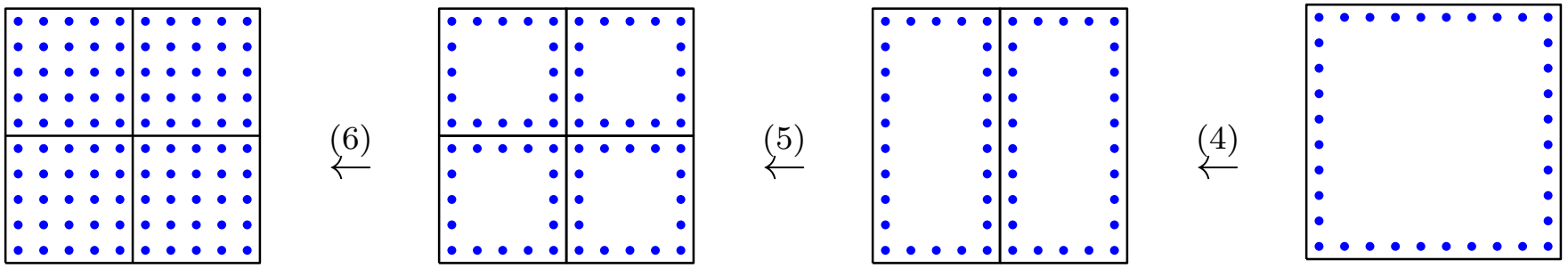
Full solution. *Solve.* *Solve.* *Top level solve.*

Upwards pass — build all solution operators:



The original grid. *Leaves reduced.* *After merge.* *After merge.*

Downwards pass — solve for a particular data function (*very* fast!):

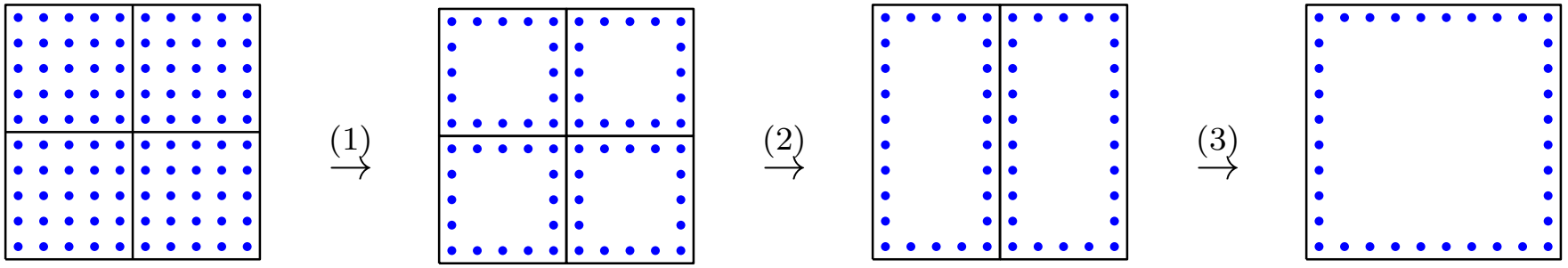


Full solution. *Solve.* *Solve.* *Top level solve.*

Note: The computational template outlined is the same as the classical multifrontal / nested dissection method (George 1973, later Duff, Davis, etc). Typical complexities:

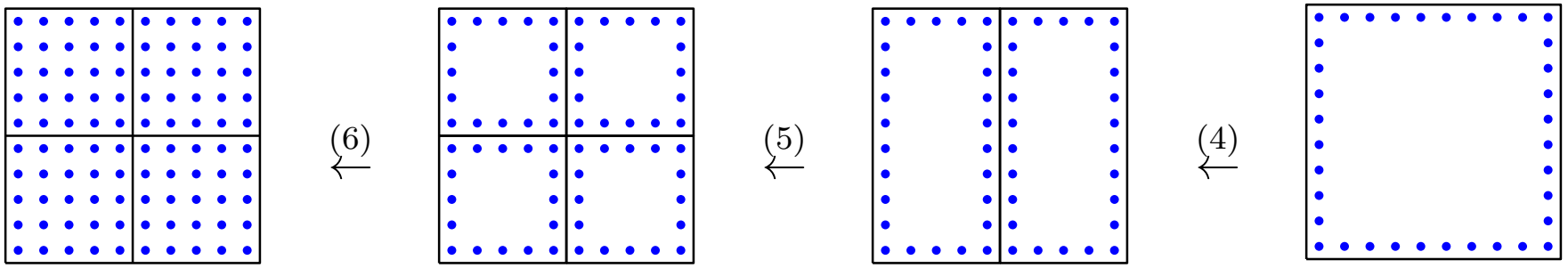
	Build stage	*Solve stage*
2D	$N^{3/2}$	$N \log N$
3D	N^2	$N^{4/3}$

Upwards pass — build all solution operators:



The original grid. *Leaves reduced.* *After merge.* *After merge.*

Downwards pass — solve for a particular data function (*very* fast!):



Full solution. *Solve.* *Solve.* *Top level solve.*

Note: The computational template outlined is the same as the classical multifrontal / nested dissection method (George 1973, later Duff, Davis, etc). Typical complexities:

	Build stage		*Solve stage*	
2D	$N^{3/2}$	$\to N$	$N \log N$	$\to N$
3D	N^2	$\to N$	$N^{4/3}$	$\to N$

Novelty: High-order discretizations, integral equations, $O(N)$ complexity for all stages.

Outline of talk

Part 1: Introduction (done)

Part 2: Case study — we will show in detail a direct solver designed for variable coefficient problems with smooth solutions. A multi-domain spectral collocation method will be used.

Part 3: Superficial description of direct solvers for integral equations (mostly numerical examples).

Part 2 — case study: Variable coefficient problems with smooth solutions

Consider an elliptic Boundary Value Problem (BVP) of the form

$$\begin{cases} -\Delta\, u(\boldsymbol{x}) + b(\boldsymbol{x})\; u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where $\Omega = [0,1]^2$ and $\Gamma = \partial\Omega$. The function b can be positive or negative (and large!).

We will describe a solver for (BVP) with the following characteristics:

- ***Direct solver:*** Excellent for problems for which iterative methods struggle (e.g. Helmholtz). Extremely fast for problems with multiple right hand sides.
- ***High order:*** Based on composite spectral discretization with local 21×21 point Chebyshev tensor product grids. Capable of solving (BVP) to 10 correct digits or more.
- ***Linear complexity:*** $O(N)$ or $O(N \log N)$ asymptotic complexity for all stages (for non-oscillatory problems). High practical efficiency.
- ***Drawbacks:*** Memory hog, in particular in 3D.
  Works best for problems with smooth solutions.

Part 2 — case study: Variable coefficient problems with smooth solutions

Consider an elliptic Boundary Value Problem (BVP) of the form

$$\begin{cases} -\Delta\, u(\boldsymbol{x}) + b(\boldsymbol{x})\; u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where $\Omega = [0,1]^2$ and $\Gamma = \partial\Omega$. The function b can be positive or negative (and large!).

Several generalizations are possible:

- It is straight-forward to modify the scheme to more *general elliptic operators*

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

  For instance, we can handle (heterogeneous) Laplace, Helmholtz, linear elasticity, etc.

- *More general domains* (L-shaped, curved domains, ...).
- *Free space problems* (acoustic scattering) with $\Omega = \mathbb{R}^2$.
- *Local mesh refinement.*
- *Non-smooth boundary data.* (At least localized non-smoothness.)
- *Body loads* can be included.
- Problems in *3D*.

Prior work

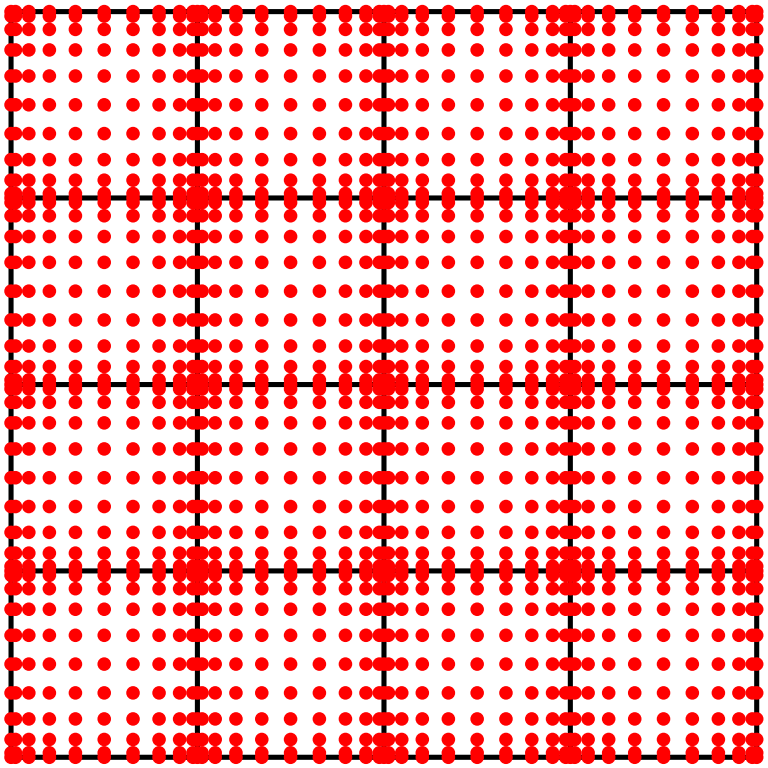
The *discretization scheme* is similar to existing composite (or "multi-domain") spectral collocation methods by Hesthaven and others. In particular, quite similar to work of *Pfeiffer, Kidder, Scheel, Teukolsky, (2003)*.

The general idea of *efficient direct solvers* for block-sparse systems goes back to the nested dissection and multifrontal methods by George (1973), Duff, and others. These have complexity $O(N^{1.5})$ for the "build stage" and $O(N \log N)$ for the "solve stage."

Nested dissection with $O(N)$ complexity was in the last several years described by *Xia, Chandrasekaran, Gu, Li (2009)*, *Le Borne, Grasedyck, Kriemann (2007)*, *Schmitz and Ying (2012)*, *Gillman and Martinsson (2011)*. Our scheme is heavily inspired by this work.

$O(N)$ direct solvers for integral equations were developed by *Martinsson, Rokhlin (2005)*, *Greengard, Gueyffier, Martinsson, Rokhlin (2009)*, *Gillman, Young, Martinsson (2012)*, *Ho, Greengard (2012)*. Our direct solver is also related to an $O(N^{1.5})$ complexity direct solver for Lippman-Schwinger, see *Chen (2002) & (2013)*. Also related to direct solvers for integral equations based on $\mathcal{H}$ and $\mathcal{H}^2$ matrix arithemetic by Hackbusch (1998 and forwards), Börm, Bebendorf, etc.

The solver relies on a composite spectral grid:



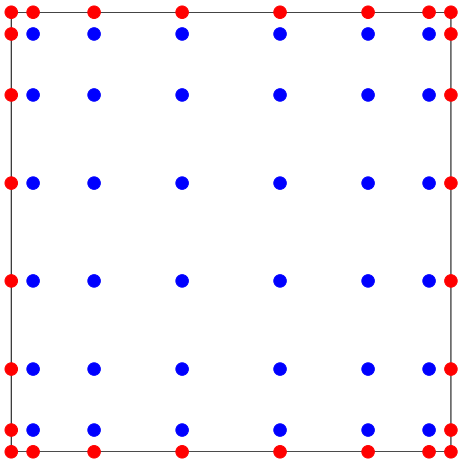
We will perform a local brute-force solve on each small square to build a local “solution operator” in the form of a “Dirichlet-to-Neumann” operator.

The global “solution operator” will be built via a hierarchical merge process.

Classical spectral collocation — notation

Recall: Our model problem is $-\Delta\, u(\boldsymbol{x}) + b(\boldsymbol{x})\; u(\boldsymbol{x}) = 0$ with Dirichlet BC on $\Omega = [0,1]^2$.

Pick an integer p and place a Cartesian mesh of $p \times p$ Chebyshev nodes on Ω.



Let $\{\boldsymbol{x}_j\}_{j=1}^{p^2}$ denote an enumeration of the nodes in the grid.

Partition the index vector $I = \{1,\, 2,\, 3,\, \ldots,\, p^2\} = I_{\rm i} \cup I_{\rm e}$ as follows:

- $I_{\rm i}$ holds the $(p-2)^2$ *interior* nodes (blue dots).
- $I_{\rm e}$ holds the $4p-4$ *exterior* nodes (red dots).

Let $\mathbf{D}^{(1)}$, $\mathbf{D}^{(2)}$, and $\mathbf{L}$, denote the $p^2 \times p^2$ matrices approximating $\partial/\partial x_1$, $\partial/\partial x_2$, and $-\Delta$, in the spectral sense (i.e. they are exact for tensor products of polynomials of degree $\leq p-1$.)

Classical spectral collocation — solving a Dirichlet problem

Recall: Our model problem is $-\Delta u(\boldsymbol{x}) + b(\boldsymbol{x})\, u(\boldsymbol{x}) = 0$ with Dirichlet BC on a square $\Omega = [0,1]^2$. The domain is discretized using a $p \times p$ tensor product grid of Chebyshev nodes split into I_{e} *exterior nodes* and I_{i} *interior nodes*.

Let **B** denote the diagonal matrix with entries $[b(\boldsymbol{x}j)]_{j=1}^{p^2}$ and let **L** be the spectral Laplacian. Then $\mathbf{A} = \mathbf{L} + \mathbf{B}$ is our spectral approximation of the differential operator.

Let $\mathbf{u} \in \mathbb{R}^{p^2}$ denote a vector of approximate values of the solution u, $\mathbf{u}(j) \approx u(\mathbf{x}_j)$.

For the ***exterior nodes***, simply set $\mathbf{u}$ equal to the Dirichlet data: $\mathbf{u}(j) = f(\mathbf{x}_j)$ for $j \in I_{\mathrm{e}}$.

For ***interior nodes***, enforce the PDE via collocation: $\mathbf{A}(j,:)\mathbf{u} = 0$ for $j \in I_{\mathrm{i}}$.

Set $\mathbf{u}_{\mathrm{i}} = \mathbf{u}(I_{\mathrm{i}})$, $\mathbf{u}_{\mathrm{e}} = \mathbf{u}(I_{\mathrm{e}})$, $\mathbf{A}_{\mathrm{i,i}} = \mathbf{A}(I_{\mathrm{i}}, I_{\mathrm{i}})$, and $\mathbf{A}_{\mathrm{i,e}} = \mathbf{A}(I_{\mathrm{i}}, I_{\mathrm{e}})$.
Then the collocation condition can be written

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

Solving for $\mathbf{u}_{\mathrm{i}}$ we find the solution process:

$$\mathbf{u}_{\mathrm{e}} = \mathbf{f}_{\mathrm{e}} = [f(\boldsymbol{x}_j)]_{j \in I_{\mathrm{e}}}$$

$$\mathbf{u}_{\mathrm{i}} = -\mathbf{A}_{\mathrm{i,i}}^{-1}\, \mathbf{A}_{\mathrm{i,e}}\, \mathbf{f}_{\mathrm{e}}.$$

Classical spectral collocation — build the Dirichlet-to-Neumman (DtN) map

Recall: Our model problem is $-\Delta\, u(\boldsymbol{x}) + b(\boldsymbol{x})\; u(\boldsymbol{x}) = 0$ with Dirichlet BC on a square $\Omega = [0,1]^2$. The domain is discretized using a $p \times p$ tensor product grid of Chebyshev nodes split into I_{e} *exterior nodes* and I_{i} *interior nodes*.

At this point, we have constructed a linear map from Dirichlet data $\mathbf{f}_{\mathrm{e}}$ to the full solution vector $\mathbf{u}$ via:

1. For exterior nodes, set the potential to equal the given Dirichlet data

$$\mathbf{u}_{\mathrm{e}} = [f(\mathbf{x}_j)]_{j \in I_{\mathrm{e}}} = \mathbf{f}_{\mathrm{e}}.$$

2. For the interior nodes, enforce the PDE via spectral collocation,

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

   Solving for $\mathbf{u}_{\mathrm{i}}$, we find $\mathbf{u}_{\mathrm{i}} = -\mathbf{A}_{\mathrm{i,i}}^{-1}\,\mathbf{A}_{\mathrm{i,e}}\,\mathbf{f}_{\mathrm{e}}$.

New objective: We seek to build the *Dirichlet-to-Neumann (DtN) map* that maps given Dirichlet data to the corresponding boundary fluxes. This map acts as a local solution operator that encodes all information about the box that we need to solve the global problem.

Classical spectral collocation — build the Dirichlet-to-Neumman (DtN) map

Recall: Our model problem is $-\Delta\, u(\boldsymbol{x}) + b(\boldsymbol{x})\; u(\boldsymbol{x}) = 0$ with Dirichlet BC on a square $\Omega = [0,1]^2$. The domain is discretized using a $p \times p$ tensor product grid of Chebyshev nodes split into I_{e} *exterior nodes* and I_{i} *interior nodes*.

At this point, we have constructed a linear map from Dirichlet data $\mathbf{f}_{\mathrm{e}}$ to the full solution vector $\mathbf{u}$ via:

1. For exterior nodes, set the potential to equal the given Dirichlet data

$$\mathbf{u}_{\mathrm{e}} = [f(\mathbf{x}_j)]_{j \in I_{\mathrm{e}}} = \mathbf{f}_{\mathrm{e}}.$$

2. For the interior nodes, enforce the PDE via spectral collocation,

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

   Solving for $\mathbf{u}_{\mathrm{i}}$, we find $\mathbf{u}_{\mathrm{i}} = -\mathbf{A}_{\mathrm{i,i}}^{-1}\,\mathbf{A}_{\mathrm{i,e}}\,\mathbf{f}_{\mathrm{e}}$.

New objective: We seek to build the *Dirichlet-to-Neumann (DtN) map* that maps given Dirichlet data to the corresponding boundary fluxes. This map acts as a local solution operator that encodes all information about the box that we need to solve the global problem.

Solution: Simply apply spectral differentiation to the constructed solution $\mathbf{u} = [\mathbf{u}_{\mathrm{i}},\, \mathbf{u}_{\mathrm{e}}]$.

Classical spectral collocation — build the Dirichlet-to-Neumman (DtN) map

Recall: Our model problem is $-\Delta\, u(\boldsymbol{x}) + b(\boldsymbol{x})\; u(\boldsymbol{x}) = 0$ with Dirichlet BC on a square $\Omega = [0,1]^2$. The domain is discretized using a $p \times p$ tensor product grid of Chebyshev nodes split into I_{e} *exterior nodes* and I_{i} *interior nodes*.

At this point, we have constructed a linear map from Dirichlet data $\mathbf{f}_{\mathrm{e}}$ to the full solution vector $\mathbf{u}$ via:

1. For exterior nodes, set the potential to equal the given Dirichlet data

$$\mathbf{u}_{\mathrm{e}} = [f(\mathbf{x}_j)]_{j \in I_{\mathrm{e}}} = \mathbf{f}_{\mathrm{e}}.$$

2. For the interior nodes, enforce the PDE via spectral collocation,

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

   Solving for $\mathbf{u}_{\mathrm{i}}$, we find $\mathbf{u}_{\mathrm{i}} = -\mathbf{A}_{\mathrm{i,i}}^{-1}\,\mathbf{A}_{\mathrm{i,e}}\,\mathbf{f}_{\mathrm{e}}$.

3. Now that $\mathbf{u}$ is known at *all* nodes, apply the spectral differentiation matrices $\mathbf{D}^{(1)}$ or $\mathbf{D}^{(2)}$ to compute the boundary fluxes. (Corners get special treatment.)

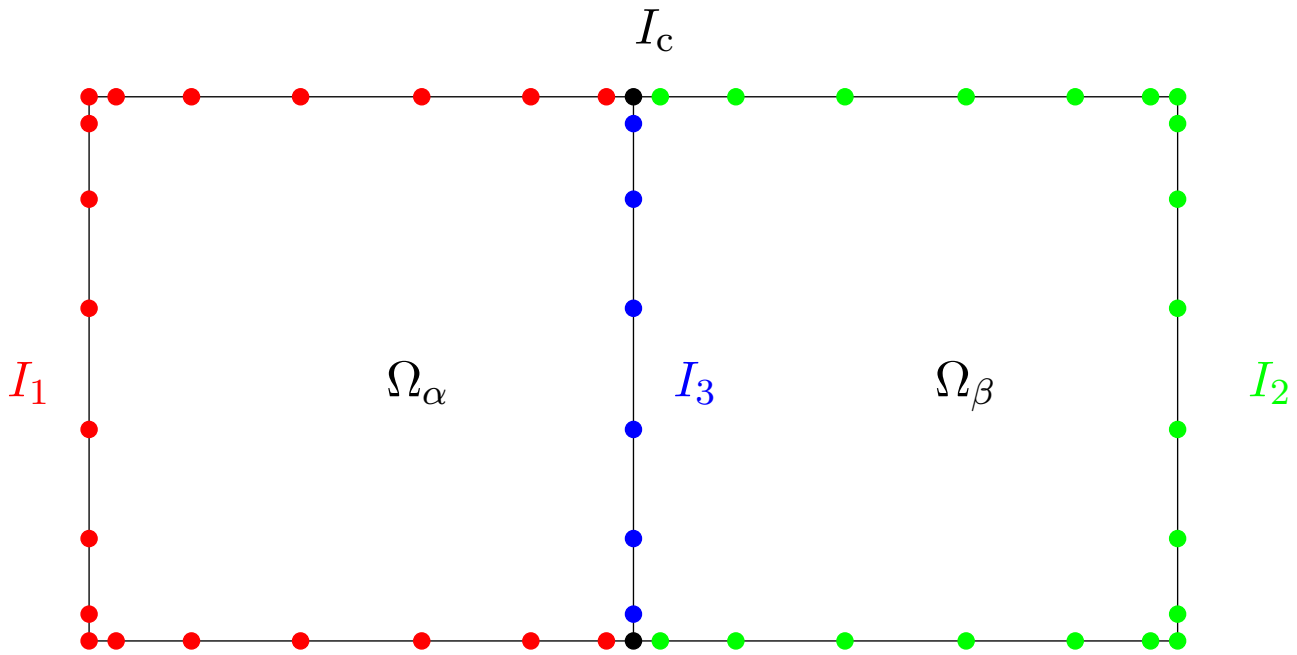
Merging two DtN operators

Question: Given the DtN matrices of two boxes, how form the DtN matrix of the union box?

Merging two DtN operators

Question: Given the DtN matrices of two boxes, how form the DtN matrix of the union box?

Answer: Let the rectangular domain Ω be formed by two squares Ω_α and Ω_β. The sets I_1, I_2, and I_3 form the exterior nodes, while I_4 consists of the interior nodes.



Let $\mathbf{v}_j$ denote the boundary fluxes on side j, and let $\mathbf{u}_j$ denote the potential. Then from the left and the right DtN maps we get the equilibrium equations

$$\begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{T}^{\alpha}_{1,1} & \mathbf{T}^{\alpha}_{1,3} \\ \mathbf{T}^{\alpha}_{3,1} & \mathbf{T}^{\alpha}_{3,3} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_3 \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{T}^{\beta}_{2,2} & \mathbf{T}^{\beta}_{2,3} \\ \mathbf{T}^{\beta}_{3,2} & \mathbf{T}^{\beta}_{3,3} \end{bmatrix} \begin{bmatrix} \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix}.$$

Collating the two equilibrium equations (by eliminating $\mathbf{v}_3$) we get

$$\left[\begin{array}{c|c|c} \mathbf{T}^{\alpha}_{1,1} & \mathbf{0} & \mathbf{T}^{\alpha}_{1,3} \\ \hline \mathbf{0} & \mathbf{T}^{\beta}_{2,2} & \mathbf{T}^{\beta}_{2,3} \\ \hline \mathbf{T}^{\alpha}_{3,1} & -\mathbf{T}^{\beta}_{3,2} & \mathbf{T}^{\alpha}_{3,3} - \mathbf{T}^{\beta}_{3,3} \end{array}\right] \left[\begin{array}{c} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{array}\right] = \left[\begin{array}{c} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{0} \end{array}\right].$$

Eliminate $\mathbf{u}_3$ to find the desired map

$$\left[\begin{array}{c} \mathbf{v}_1 \\ \mathbf{v}_2 \end{array}\right] = \mathbf{T}^{\tau} \left[\begin{array}{c} \mathbf{u}_1 \\ \mathbf{u}_2 \end{array}\right]$$

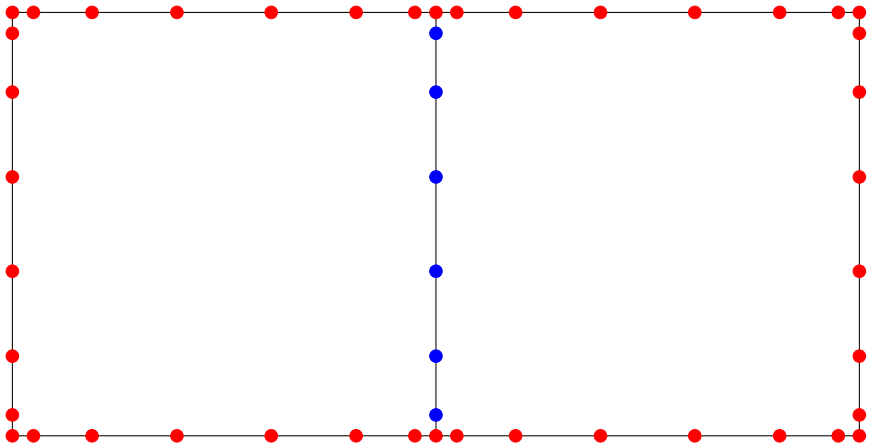
where

$$\mathbf{T}^{\tau} = \left[\begin{array}{c|c} \mathbf{T}^{\alpha}_{1,1} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{T}^{\beta}_{2,2} \end{array}\right] - \left[\begin{array}{c} \mathbf{T}^{\alpha}_{1,3} \\ \mathbf{T}^{\beta}_{2,3} \end{array}\right] \left(\mathbf{T}^{\alpha}_{3,3} - \mathbf{T}^{\beta}_{3,3}\right)^{-1} \left[\begin{array}{c|c} \frac{1}{2}\mathbf{T}^{\alpha}_{3,1} & -\frac{1}{2}\mathbf{T}^{\beta}_{3,2} \end{array}\right].$$

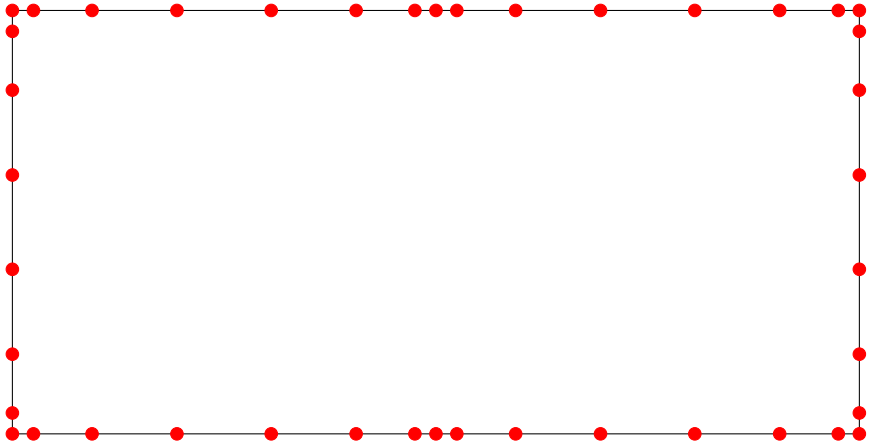
(We skipped a step — the corner nodes are eliminated by re-interpolating to *Gaussian* nodes on the boundaries.)

Illustration of the merge operation

Before elimination of interior (blue) nodes:



After elimination of interior nodes:

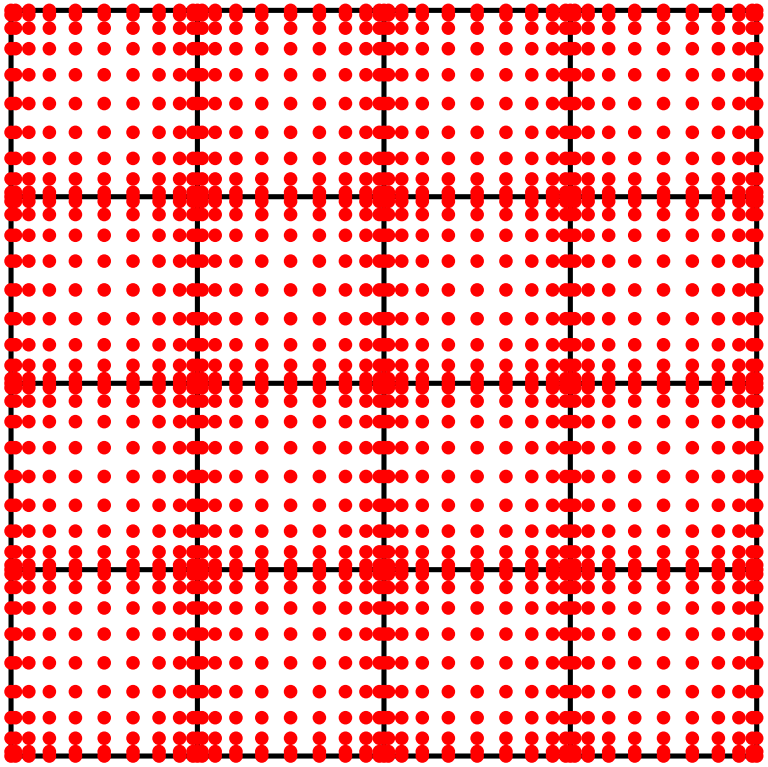


Model problem: Given f and b, find u such that

$$\begin{cases} -\Delta u(\boldsymbol{x}) - b(\boldsymbol{x})\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Pre-process: Put down a spectral composite grid on Ω:



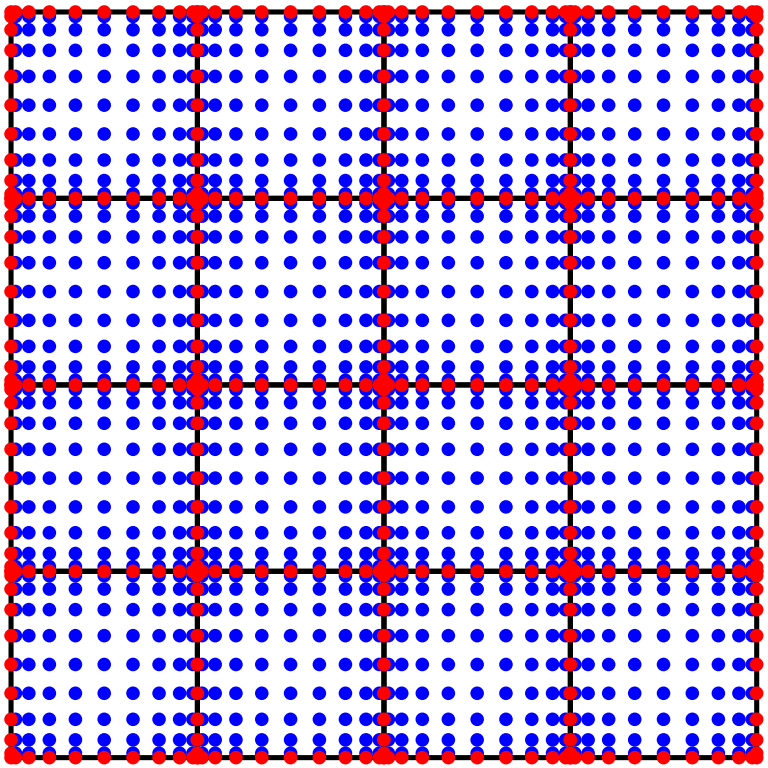
Model problem: Given f and b, find u such that

$$\begin{cases} -\Delta u(\boldsymbol{x}) - b(\boldsymbol{x})\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Process leaves: Eliminate the interior (blue) nodes.

Technically, we compute the Dirichlet-to-Neumann operator via a local spectral computation.



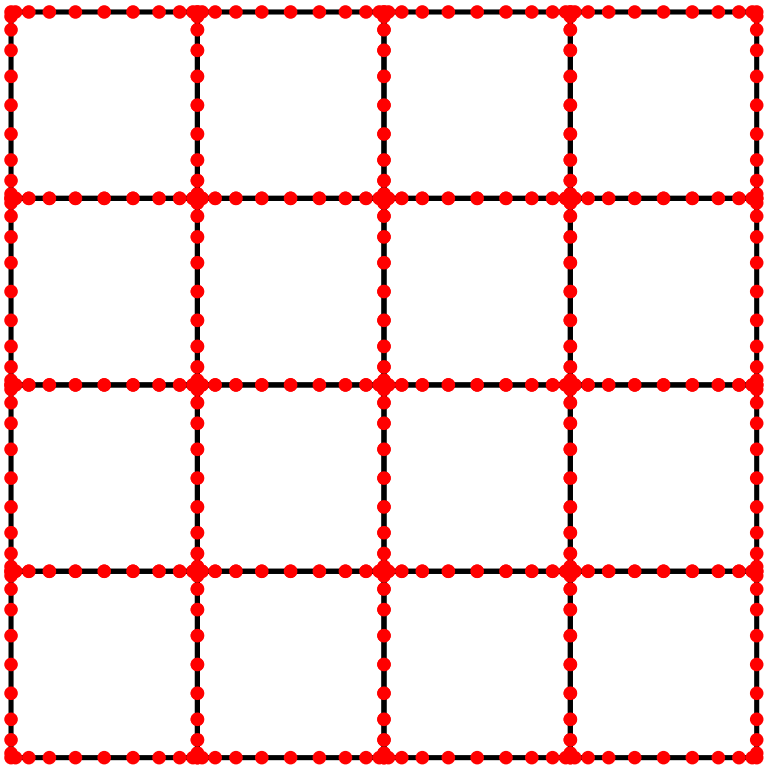
Model problem: Given f and b, find u such that

$$\begin{cases} -\Delta u(\boldsymbol{x}) - b(\boldsymbol{x})\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Process leaves: Eliminate the interior (blue) nodes.

Technically, we compute the Dirichlet-to-Neumann operator via a local spectral computation.

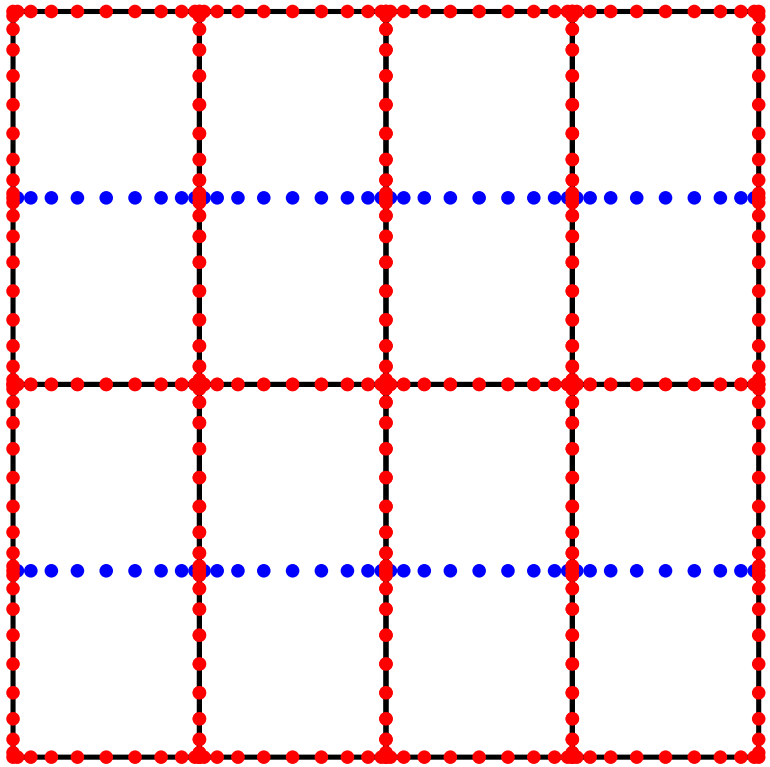


Model problem: Given f and b, find u such that

$$\begin{cases} -\Delta u(\boldsymbol{x}) - b(\boldsymbol{x})\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Upwards sweep: Merge boxes by pairs and eliminate the interior (blue) nodes.
To do this, use the computed DtN operators to enforce continuity of u and du/dn across interior boundaries. Compute the DtN operator for the larger box.

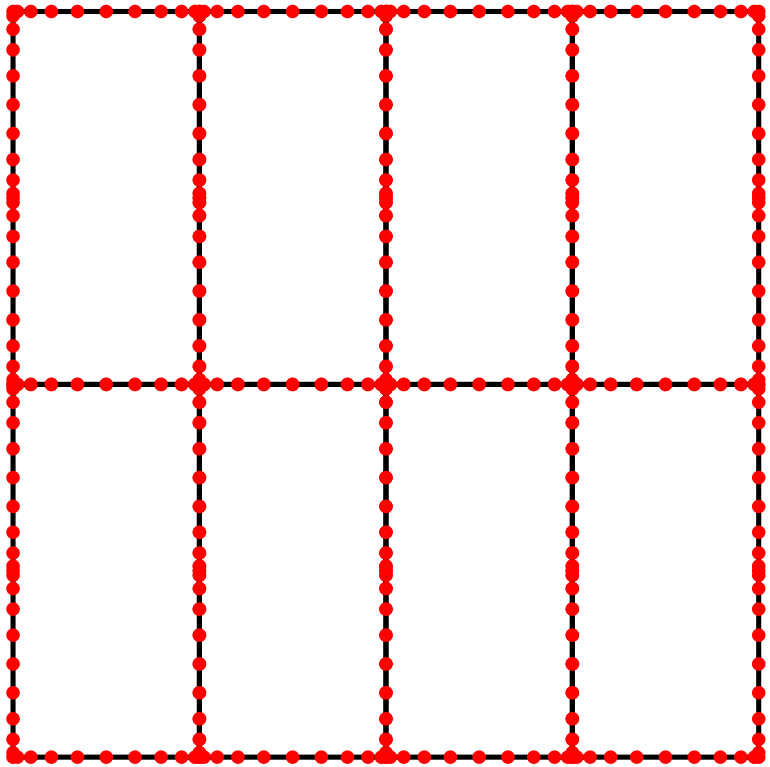


Model problem: Given f and b, find u such that

$$\begin{cases} -\Delta u(\boldsymbol{x}) - b(\boldsymbol{x})\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Upwards sweep: Merge boxes by pairs and eliminate the interior (blue) nodes.
To do this, use the computed DtN operators to enforce continuity of u and du/dn across interior boundaries. Compute the DtN operator for the larger box.

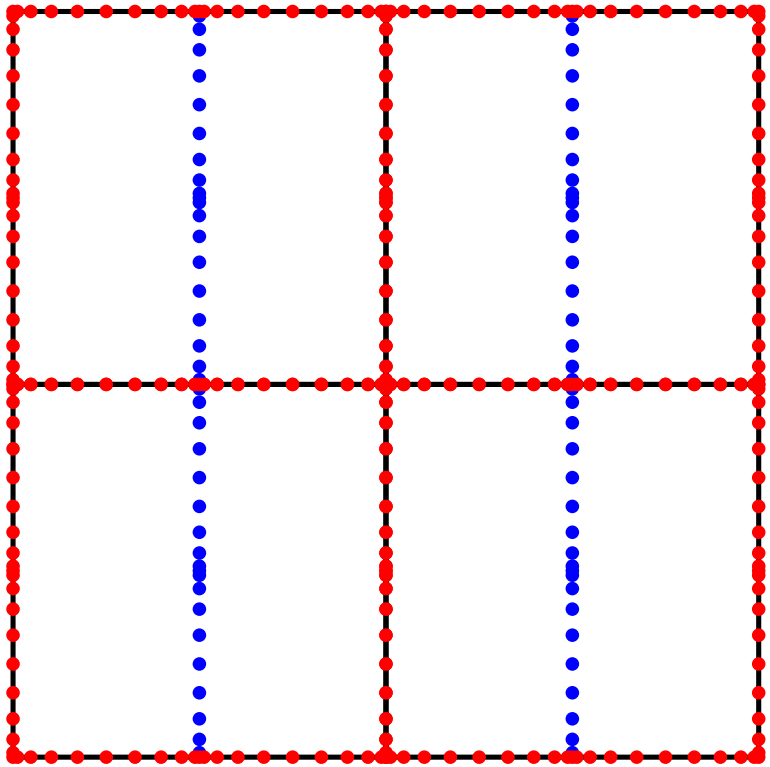


Model problem: Given f and b, find u such that

$$\begin{cases} -\Delta u(\boldsymbol{x}) - b(\boldsymbol{x})\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Upwards sweep: Merge boxes by pairs and eliminate the interior (blue) nodes.
To do this, use the computed DtN operators to enforce continuity of u and du/dn across interior boundaries. Compute the DtN operator for the larger box.

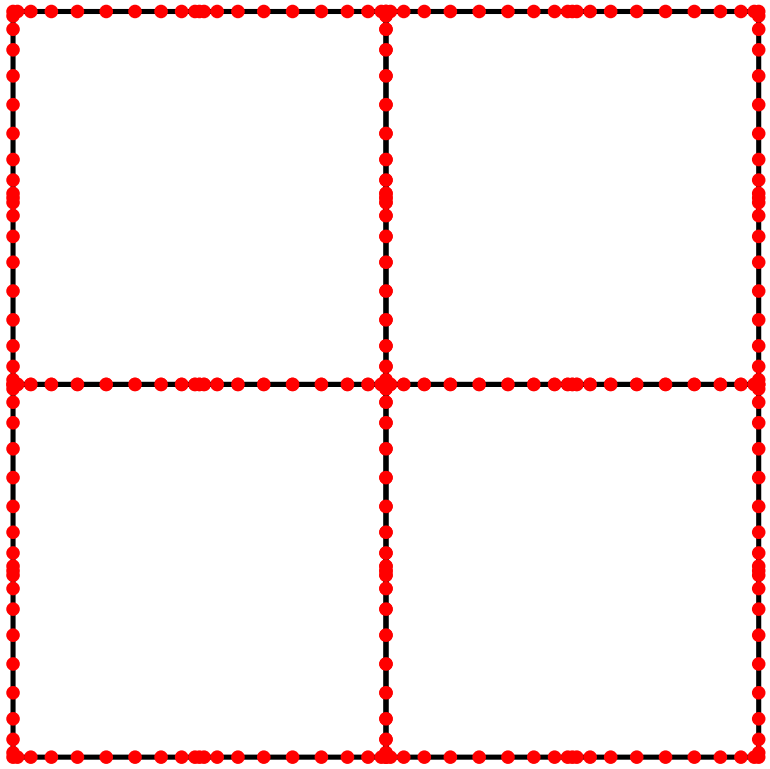


Model problem: Given f and b, find u such that

$$\begin{cases} -\Delta u(\boldsymbol{x}) - b(\boldsymbol{x})\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Upwards sweep: Merge boxes by pairs and eliminate the interior (blue) nodes.
To do this, use the computed DtN operators to enforce continuity of u and du/dn across interior boundaries. Compute the DtN operator for the larger box.

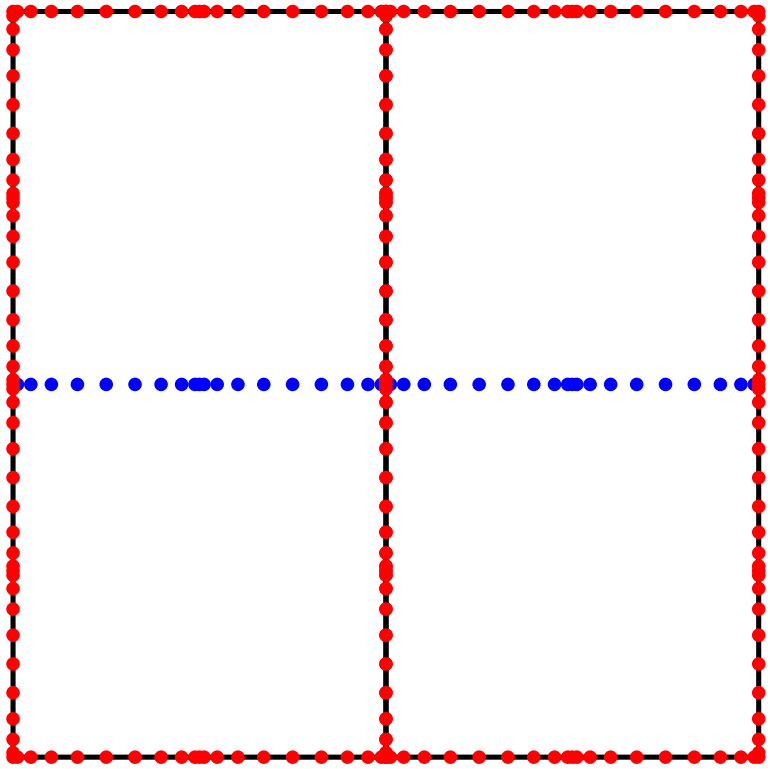


Model problem: Given f and b, find u such that

$$\begin{cases} -\Delta u(\boldsymbol{x}) - b(\boldsymbol{x})\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Upwards sweep: Merge boxes by pairs and eliminate the interior (blue) nodes.
To do this, use the computed DtN operators to enforce continuity of u and du/dn across interior boundaries. Compute the DtN operator for the larger box.

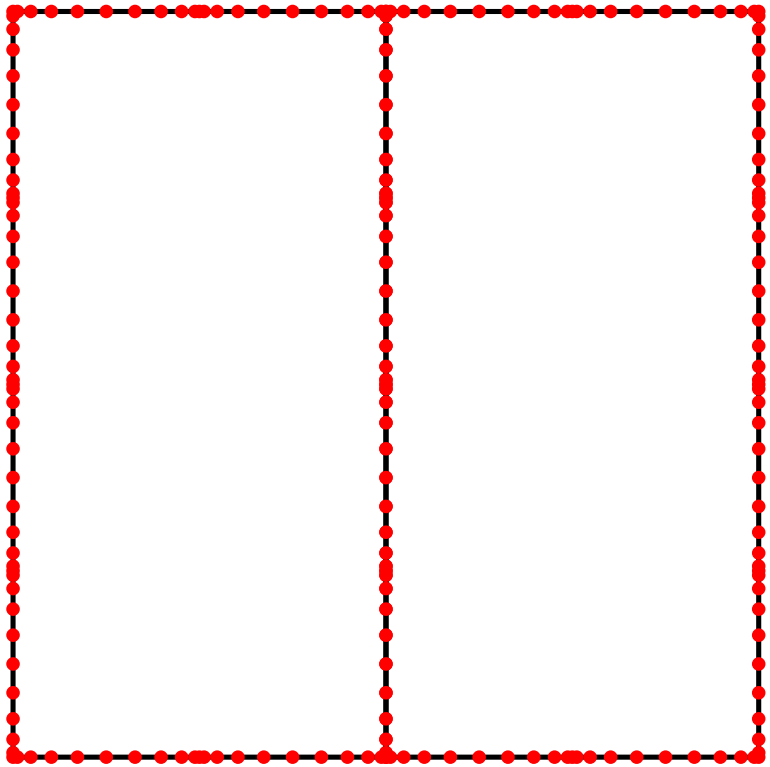


Model problem: Given f and b, find u such that

$$\begin{cases} -\Delta u(\boldsymbol{x}) - b(\boldsymbol{x})\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Upwards sweep: Merge boxes by pairs and eliminate the interior (blue) nodes.
To do this, use the computed DtN operators to enforce continuity of u and du/dn across interior boundaries. Compute the DtN operator for the larger box.

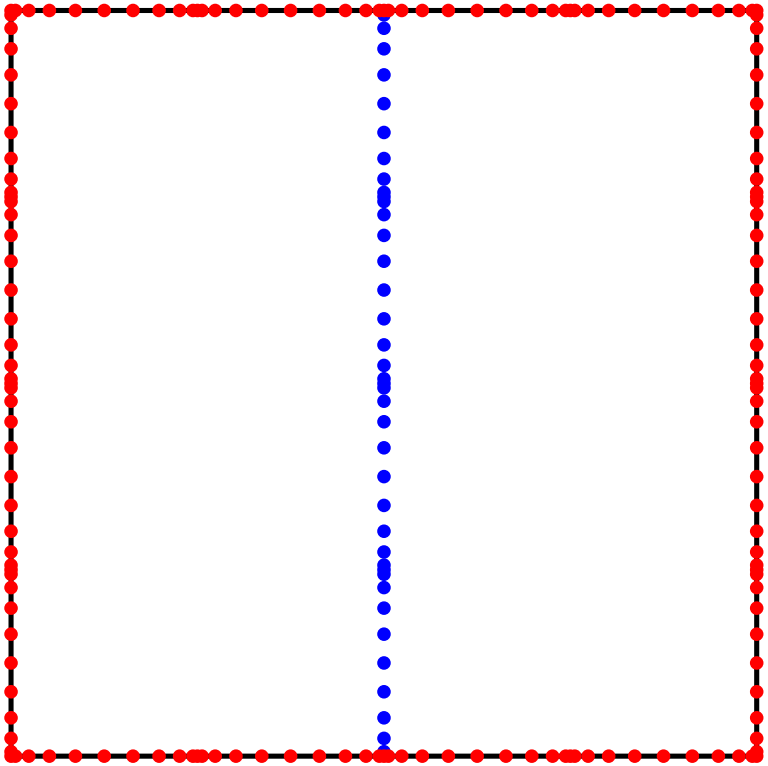


Model problem: Given f and b, find u such that

$$\begin{cases} -\Delta u(\boldsymbol{x}) - b(\boldsymbol{x})\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Upwards sweep: Merge boxes by pairs and eliminate the interior (blue) nodes.
To do this, use the computed DtN operators to enforce continuity of u and du/dn across interior boundaries. Compute the DtN operator for the larger box.

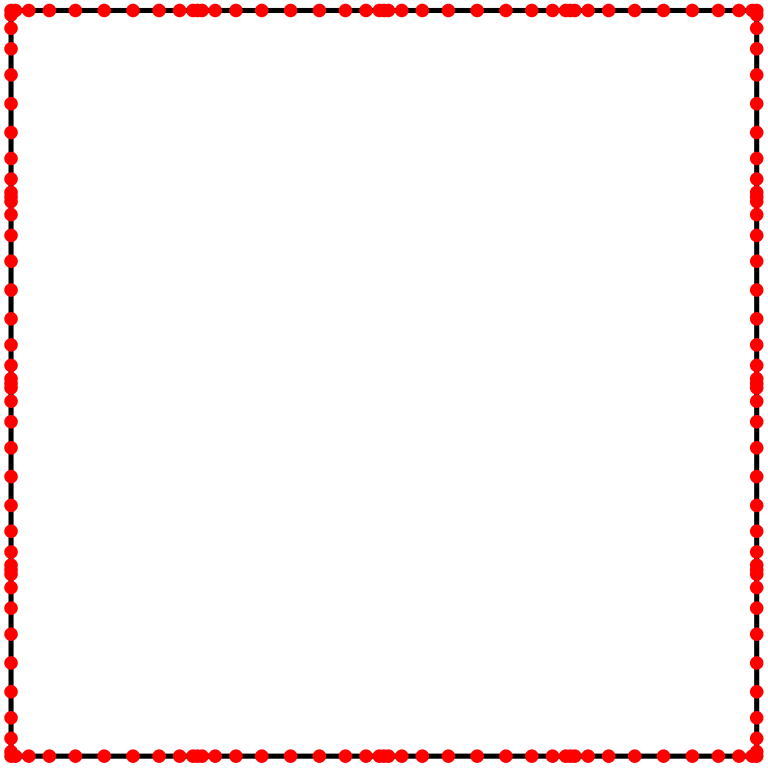


Model problem: Given f and b, find u such that

$$\begin{cases} -\Delta u(\boldsymbol{x}) - b(\boldsymbol{x})\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Upwards sweep: Merge boxes by pairs and eliminate the interior (blue) nodes.
To do this, use the computed DtN operators to enforce continuity of u and du/dn across interior boundaries. Compute the DtN operator for the larger box.

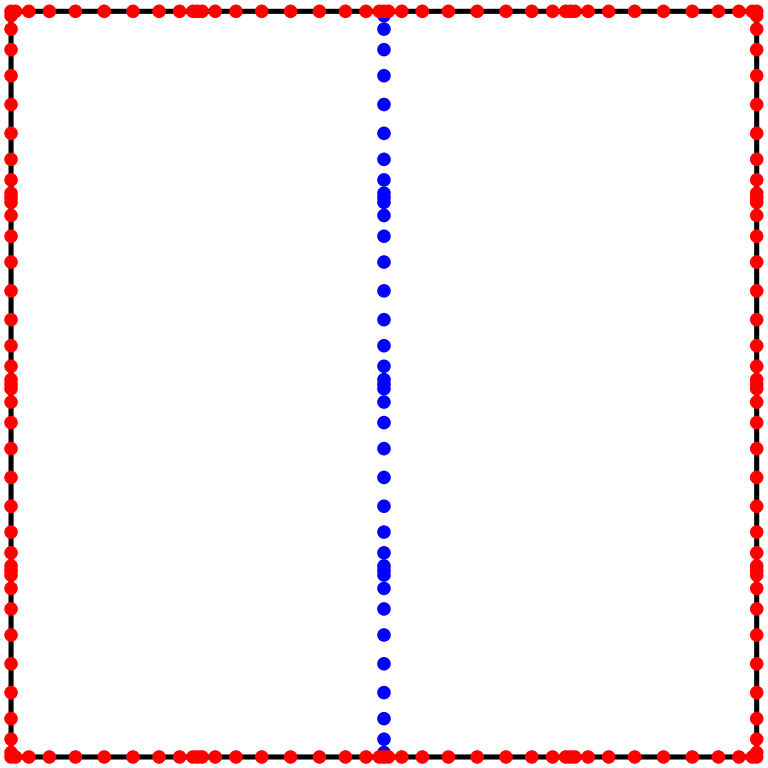


Model problem: Given f and b, find u such that

$$\begin{cases} -\Delta u(\boldsymbol{x}) - b(\boldsymbol{x})\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Downwards sweep: We know u on the red nodes. We can use the computed DtN operators to reconstruct u on the blue nodes.

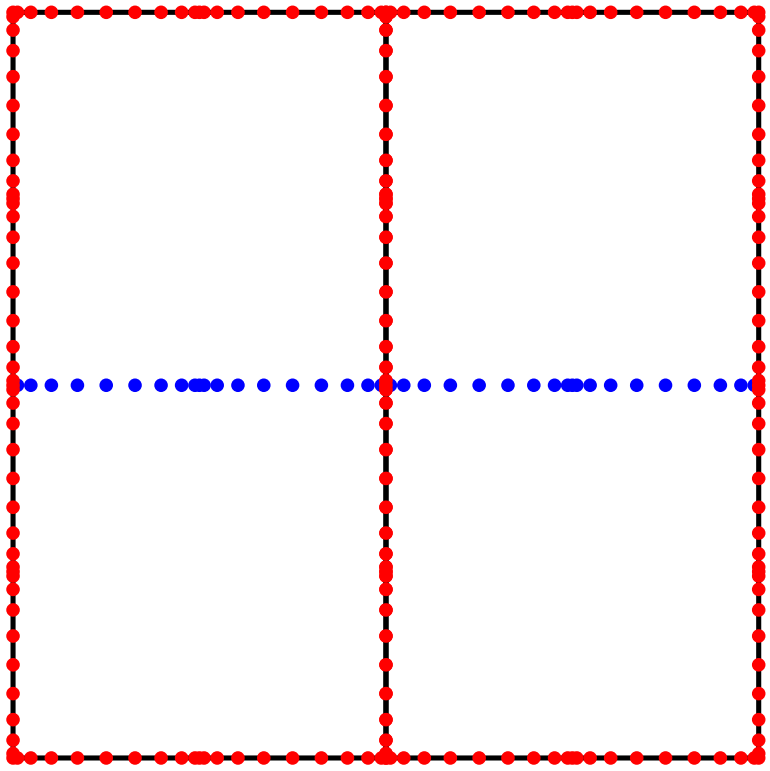


Model problem: Given f and b, find u such that

$$\begin{cases} -\Delta u(\boldsymbol{x}) - b(\boldsymbol{x})\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Downwards sweep: We know u on the red nodes. We can use the computed DtN operators to reconstruct u on the blue nodes.

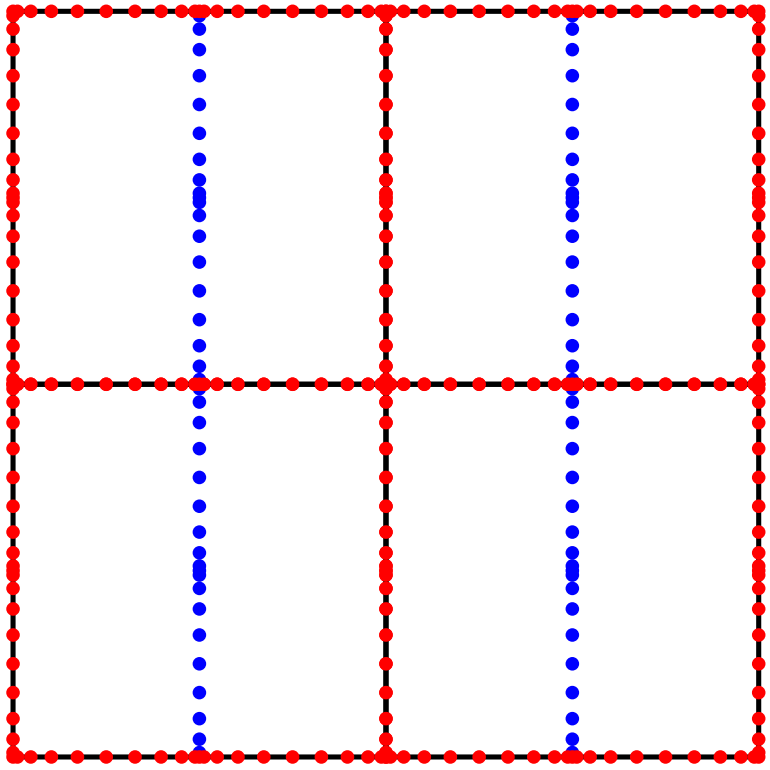


Model problem: Given f and b, find u such that

$$\begin{cases} -\Delta u(\boldsymbol{x}) - b(\boldsymbol{x})\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Downwards sweep: We know u on the red nodes. We can use the computed DtN operators to reconstruct u on the blue nodes.

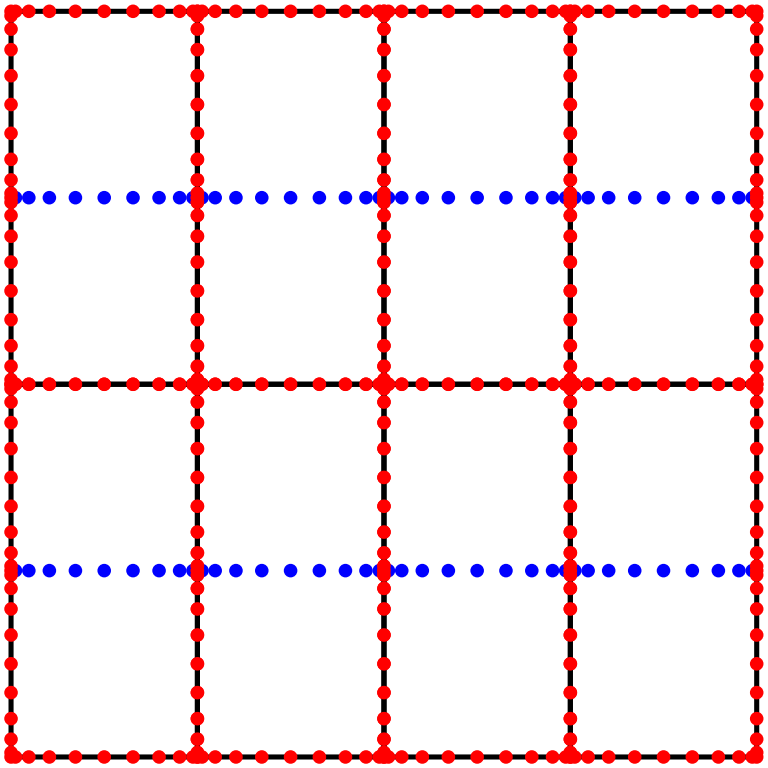


Model problem: Given f and b, find u such that

$$\begin{cases} -\Delta u(\boldsymbol{x}) - b(\boldsymbol{x})\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Downwards sweep: We know u on the red nodes. We can use the computed DtN operators to reconstruct u on the blue nodes.

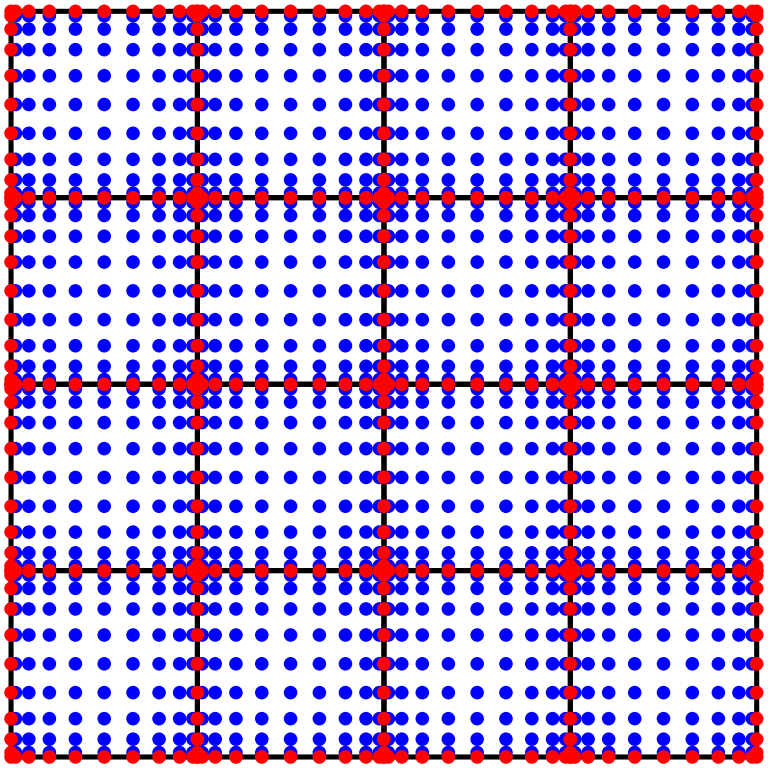


Model problem: Given f and b, find u such that

$$\begin{cases} -\Delta u(\boldsymbol{x}) - b(\boldsymbol{x})\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Downwards sweep: We know u on the red nodes. We can use the computed DtN operators to reconstruct u on the blue nodes.



Summary of the hierarchical scheme:

1. *Construct a quad-tree:* Partition the grid into a hierarchy of boxes.

2. *Process the leaves:* For each leaf box in the tree, construct its DtN (Dirichlet-to-Neumann) operator.

3. *Hierarchical merge:* Loop over all levels of the tree, from finer to smaller. For each box on a level, compute its DtN operator by merging the (already computed) DtN operators of its children.

4. *Process the root of the tree:* After completing Step 3, the DtN operator for the entire domain is available. Invert (or factor) it to construct the solution operator.

The first solve costs $O(N^{1.5})$ operations.

Subsequent solves cost $O(N \log N)$ operations.

Remark: For simplicity, the algorithm is described in a level-by-level manner (process all leaves first, then proceed one level at a time in going upwards). In fact, there is flexibility to travel through the tree in any order that ensures that no node is processed before its children. Since all Schur complements can be discarded once their information has been passed on to a parent, smarter orderings can greatly reduce the memory requirements.

Note: There are no corner singularities!

The exact mathematical object we are approximating is the Dirichlet-to-Neumann operator for a rectangle. This object exhibits complicated (singular) behavior near the corners.

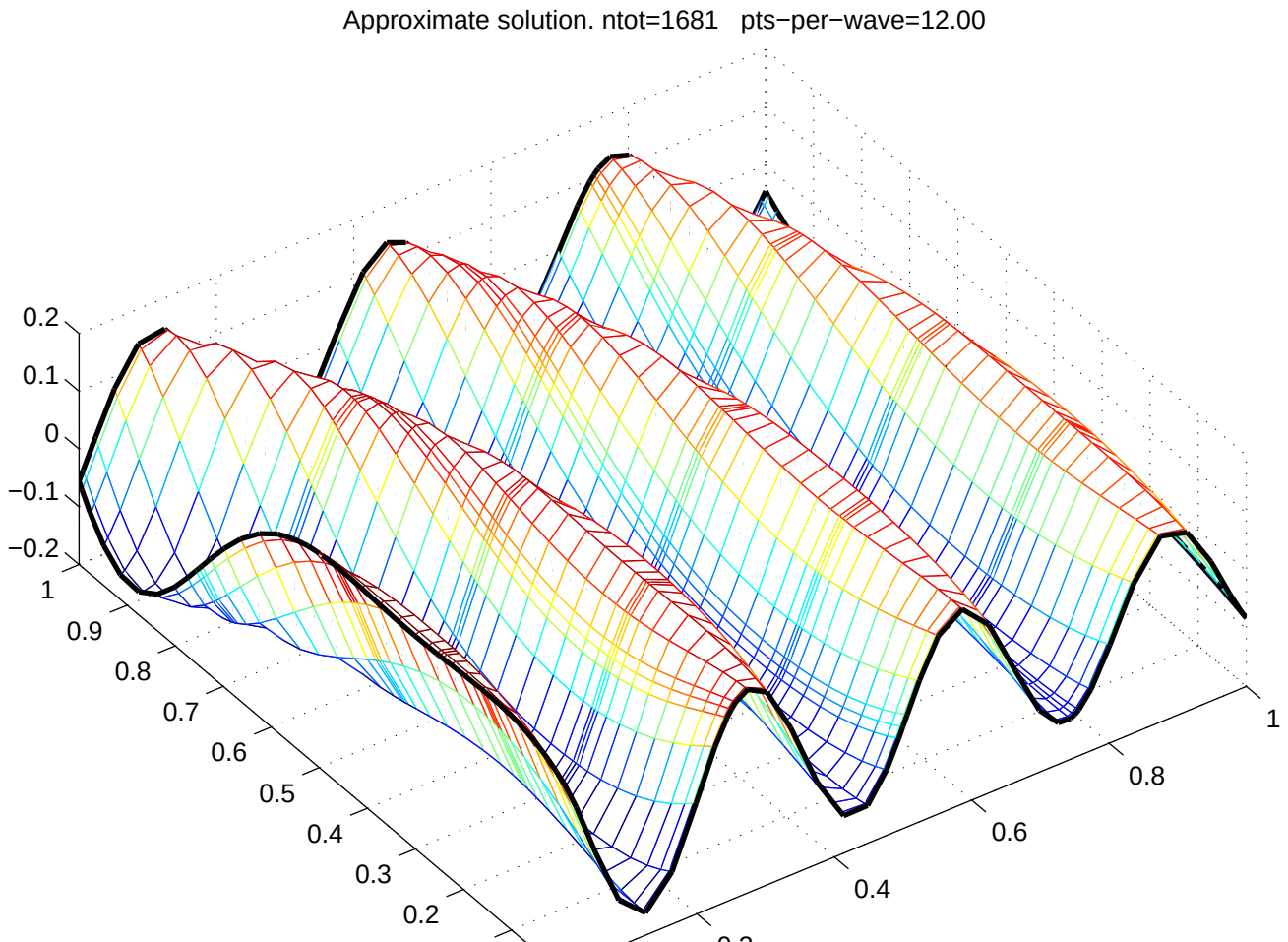
However, *we are only concerned with the projection of the exact operator onto a space of smooth functions*. In particular, we need high accuracy only for functions that are restrictions of smooth global solutions.

Spectral composite method: numerical results

Set $\Omega = [0,1]^2$ and $\Gamma = \partial\Omega$. Consider the problem

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

We pick f as the restriction of a wave from a point source, $\boldsymbol{x} \mapsto Y_0(\kappa|\boldsymbol{x} - \hat{\boldsymbol{x}}|)$.
We then know the exact solution, $u_{\text{exact}}(\boldsymbol{x}) = Y_0(\kappa|\boldsymbol{x} - \hat{\boldsymbol{x}}|)$.



Spectral composite method: numerical results

Set $\Omega = [0,1]^2$ and $\Gamma = \partial\Omega$. Consider the problem

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

We pick f as the restriction of a wave from a point source, $\boldsymbol{x} \mapsto Y_0(\kappa|\boldsymbol{x} - \hat{\boldsymbol{x}}|)$.
We then know the exact solution, $u_{\text{exact}}(\boldsymbol{x}) = Y_0(\kappa|\boldsymbol{x} - \hat{\boldsymbol{x}}|)$.
The spectral computation on a leaf involves 21×21 points.
κ is chosen so that there are 12 points per wave-length.

p	N	N_{wave}	t_{factor} (sec)	t_{solve} (sec)	E_{pot}	E_{grad}	M (MB)	M/N (reals/DOF)
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

Error is measured in sup-norm: $e = \max_{\boldsymbol{x}\in\Omega} |u(\boldsymbol{x}) - u_{\text{exact}}(\boldsymbol{x})|$.

Note 1: Translation invariance is *not* exploited.

Note 2: The times refer to a simple Matlab implementation executed on a $1k laptop.

Spectral composite method: numerical results

Set $\Omega = [0,1]^2$ and $\Gamma = \partial\Omega$. Consider the problem

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

We pick f as the restriction of a wave from a point source, $\boldsymbol{x} \mapsto Y_0(\kappa|\boldsymbol{x} - \hat{\boldsymbol{x}}|)$.
We then know the exact solution, $u_{\text{exact}}(\boldsymbol{x}) = Y_0(\kappa|\boldsymbol{x} - \hat{\boldsymbol{x}}|)$.
The spectral computation on a leaf involves 41×41 points.
κ is chosen so that there are 12 points per wave-length.

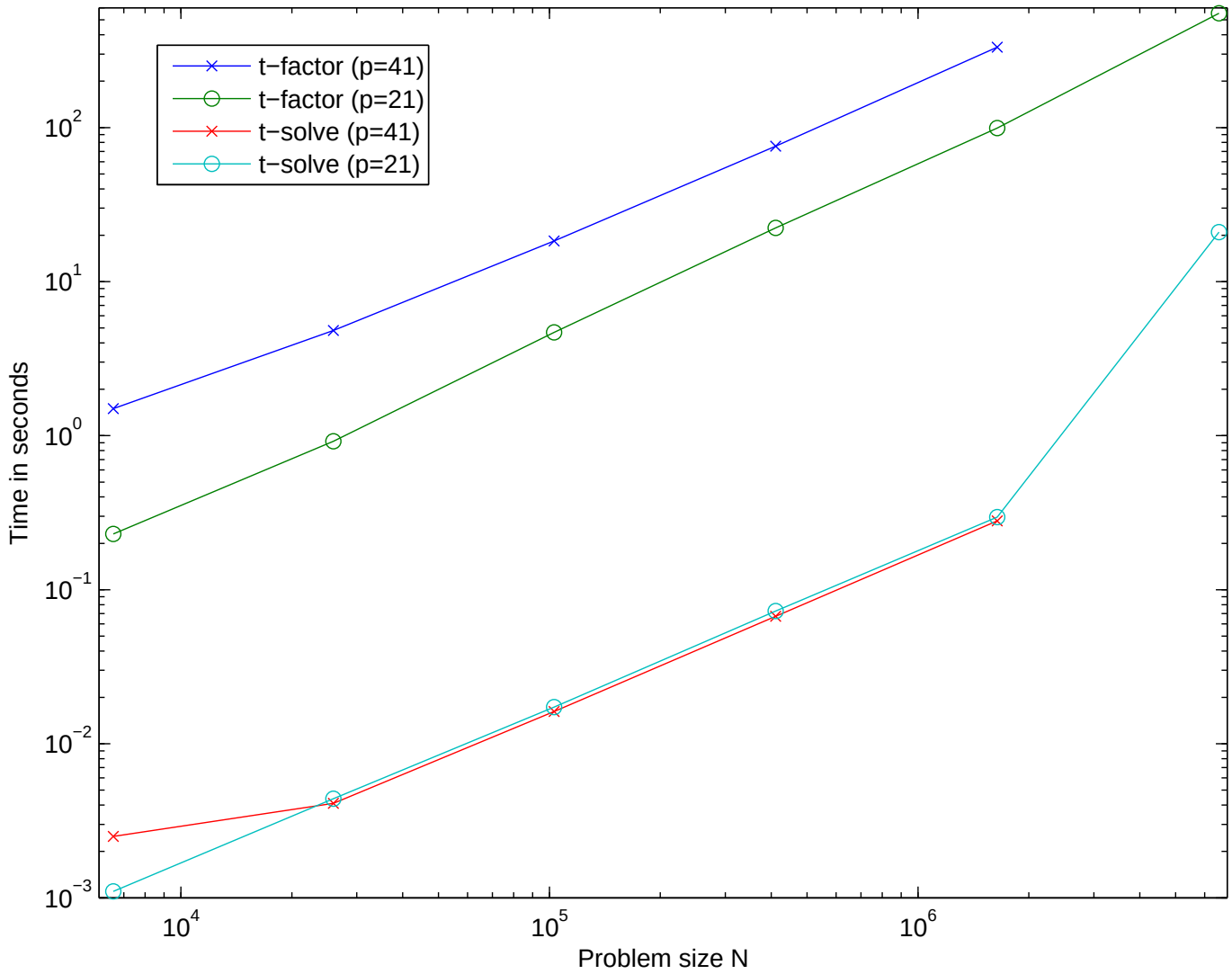
p	N	N_{wave}	t_{factor} (sec)	t_{solve} (sec)	E_{pot}	E_{grad}	M (MB)	M/N (reals/DOF)
41	6561	6.7	1.50	0.0025	9.88931e-14	3.46762e-12	7.9	157.5
41	25921	13.3	4.81	0.0041	1.58873e-13	1.12883e-11	32.9	166.4
41	103041	26.7	18.34	0.0162	3.95531e-13	5.51141e-11	137.1	174.4
41	410881	53.3	75.78	0.0672	3.89079e-13	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

Error is measured in sup-norm: $e = \max_{\boldsymbol{x}\in\Omega} |u(\boldsymbol{x}) - u_{\text{exact}}(\boldsymbol{x})|$.

Note 1: Translation invariance is *not* exploited.

Note 2: The times refer to a simple Matlab implementation executed on a $1k laptop.

Spectral composite method: numerical results



The line t_{solve} scales perfectly linearly (until memory problems kick in), as expected.

Interesting: The line t_{factor} also scales almost linearly. (Unexpectedly?) What happens is that t_{factor} is dominated by the leaf computation; we have not hit the $O(N^{1.5})$ asymptotic yet.

Spectral composite method: numerical results — variable coefficients

Now consider the variable coefficient problem

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

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 $f(\boldsymbol{x}) = \cos(8x_1)\left(1 - 2x_2\right)$.
The error estimator $E_N^{\text{int}} = u_N(\hat{\boldsymbol{x}}) - u_{4N}(\hat{\boldsymbol{x}})$ where $\hat{\boldsymbol{x}} = (0.75,\, 0.25)$ is reported below:

p	N	pts per wave	$u_N(\hat{\boldsymbol{x}})$	E_N^{int}	$w_N(\hat{\boldsymbol{y}})$	E_N^{bnd}
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	

Example: Free space scattering

$$\begin{cases} -\Delta u_{\text{out}}(\boldsymbol{x}) - k^2\,(1 - b(\boldsymbol{x}))\,u_{\text{out}}(\boldsymbol{x}) = -k^2\,b(\boldsymbol{x})\,u_{\text{in}}(\boldsymbol{x}) \\ \lim_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\big(\partial_{|\boldsymbol{x}|} u_{\text{out}}(\boldsymbol{x}) - ik\,u_{\text{out}}(\boldsymbol{x})\big) = 0 \end{cases}$$

Suppose that b is a smooth scattering potential such that for some rectangle Ω, we have

$$\text{support}(b) \subset \Omega.$$

We also suppose that u_{in} satisfies

$$-\Delta u_{\text{in}}(\boldsymbol{x}) - k^2\,u_{\text{in}}(\boldsymbol{x}) = 0, \qquad \boldsymbol{x} \in \Omega.$$

Solution strategy:

(1) Use the composite spectral method to construct the DtN map for the variable coefficient problem in Ω.

(2) Use boundary integral equation techniques to find the DtN map for the constant coefficient problem on Ω^{c}.

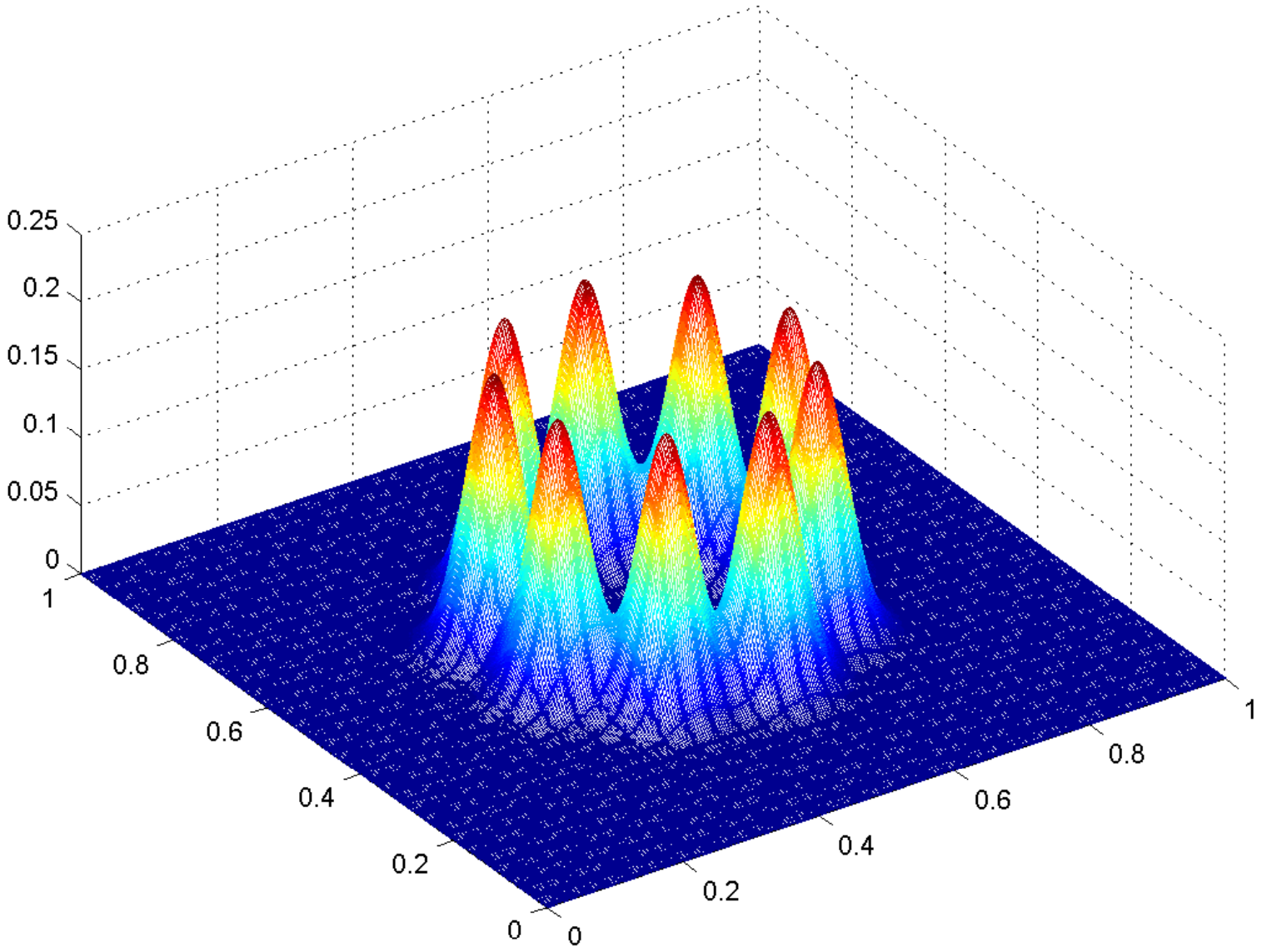
(3) Glue the two DtN maps together and you're all set!

Joint work with Adrianna Gillman and Alex Barnett.

Example: Free space scattering

$$\begin{cases} -\Delta u_{\text{out}}(\boldsymbol{x}) - k^2\,(1 - b(\boldsymbol{x}))\,u_{\text{out}}(\boldsymbol{x}) = -k^2\,b(\boldsymbol{x})\,u_{\text{in}}(\boldsymbol{x}) \\ \lim_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\big(\partial_{|\boldsymbol{x}|}u_{\text{out}}(\boldsymbol{x}) - ik\,u_{\text{out}}(\boldsymbol{x})\big) = 0 \end{cases}$$

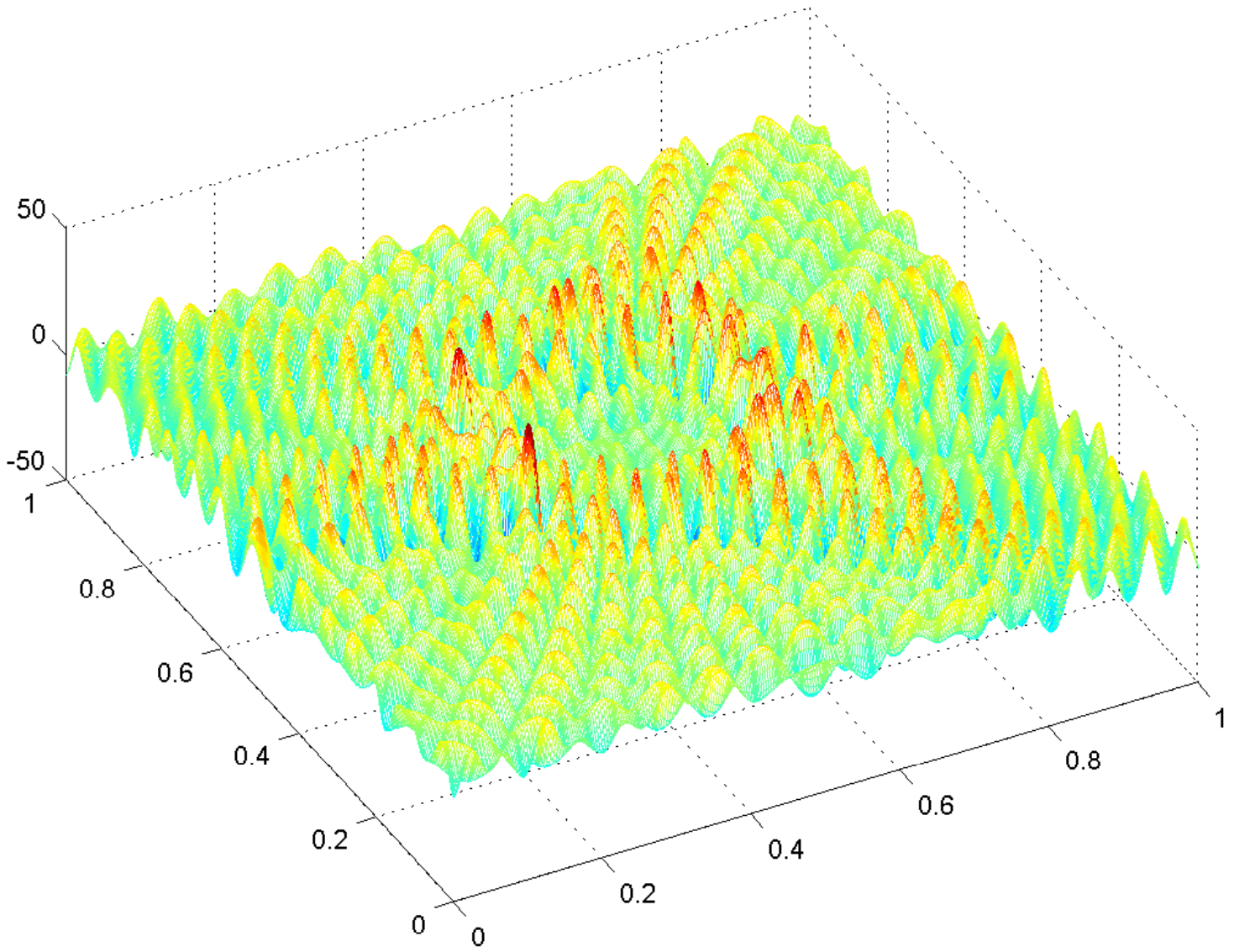
The scattering potential b



Example: Free space scattering

$$\begin{cases} -\Delta u_{\text{out}}(\boldsymbol{x}) - k^2\,(1 - b(\boldsymbol{x}))\,u_{\text{out}}(\boldsymbol{x}) = -k^2\,b(\boldsymbol{x})\,u_{\text{in}}(\boldsymbol{x}) \\ \lim_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\big(\partial_{|\boldsymbol{x}|}u_{\text{out}}(\boldsymbol{x}) - ik\,u_{\text{out}}(\boldsymbol{x})\big) = 0 \end{cases}$$

The outgoing field u_{out} (resulting from an incoming plane wave $u_{\text{in}}(x) = \cos(k\,x_1)$)

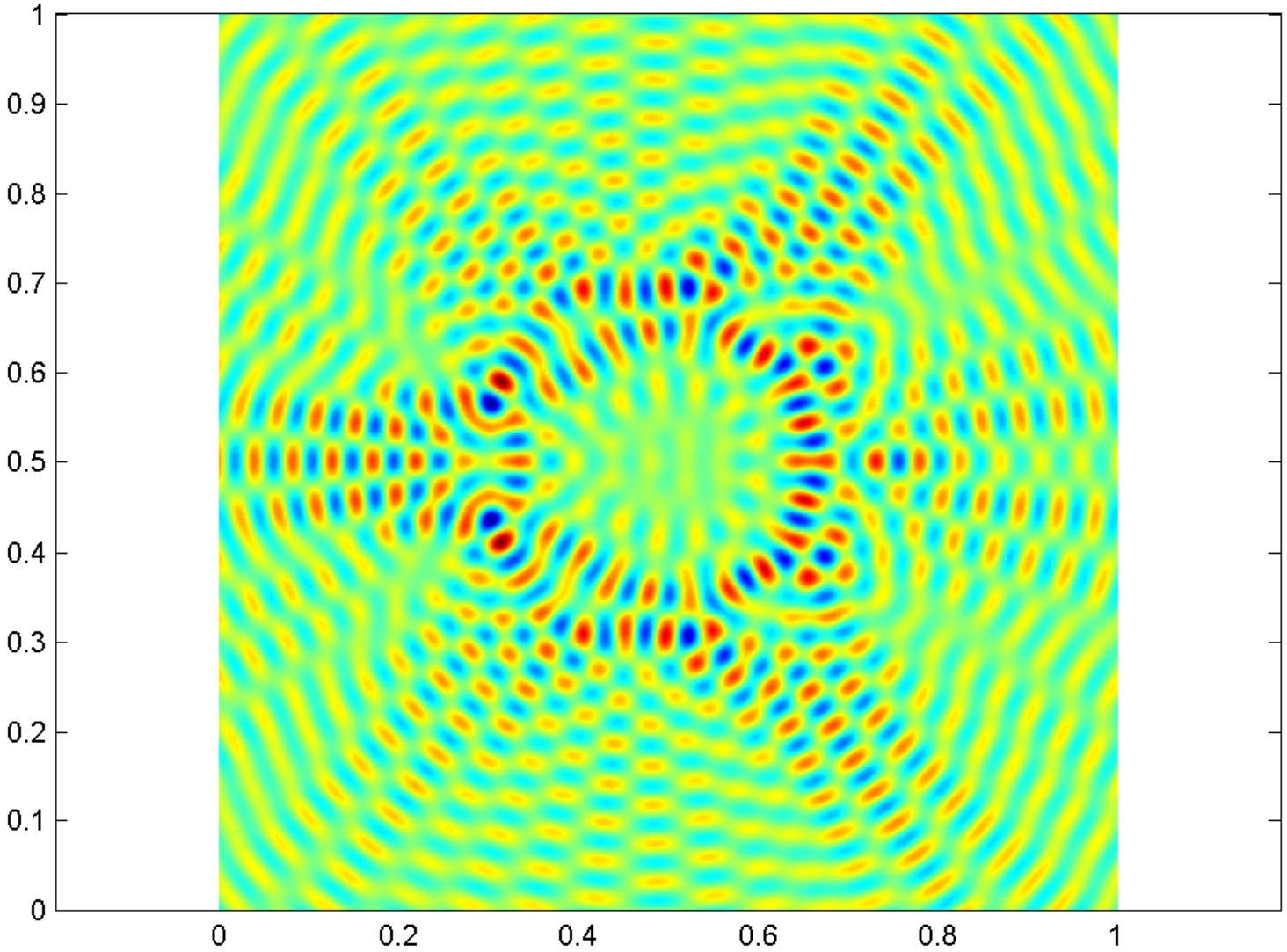


$N = 231\,361$ $T_{\text{build}} = 7.2$ sec $T_{\text{solve}} = 0.06$ sec $E \approx 10^{-7}$ (estimated)

Example: Free space scattering

$$\begin{cases} -\Delta u_{\text{out}}(\boldsymbol{x}) - k^2\,(1 - b(\boldsymbol{x}))\,u_{\text{out}}(\boldsymbol{x}) = -k^2\,b(\boldsymbol{x})\,u_{\text{in}}(\boldsymbol{x}) \\ \lim_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\big(\partial_{|\boldsymbol{x}|}u_{\text{out}}(\boldsymbol{x}) - ik\,u_{\text{out}}(\boldsymbol{x})\big) = 0 \end{cases}$$

The outgoing field u_{out} (resulting from an incoming plane wave $u_{\text{in}}(x) = \cos(k\,x_1)$)

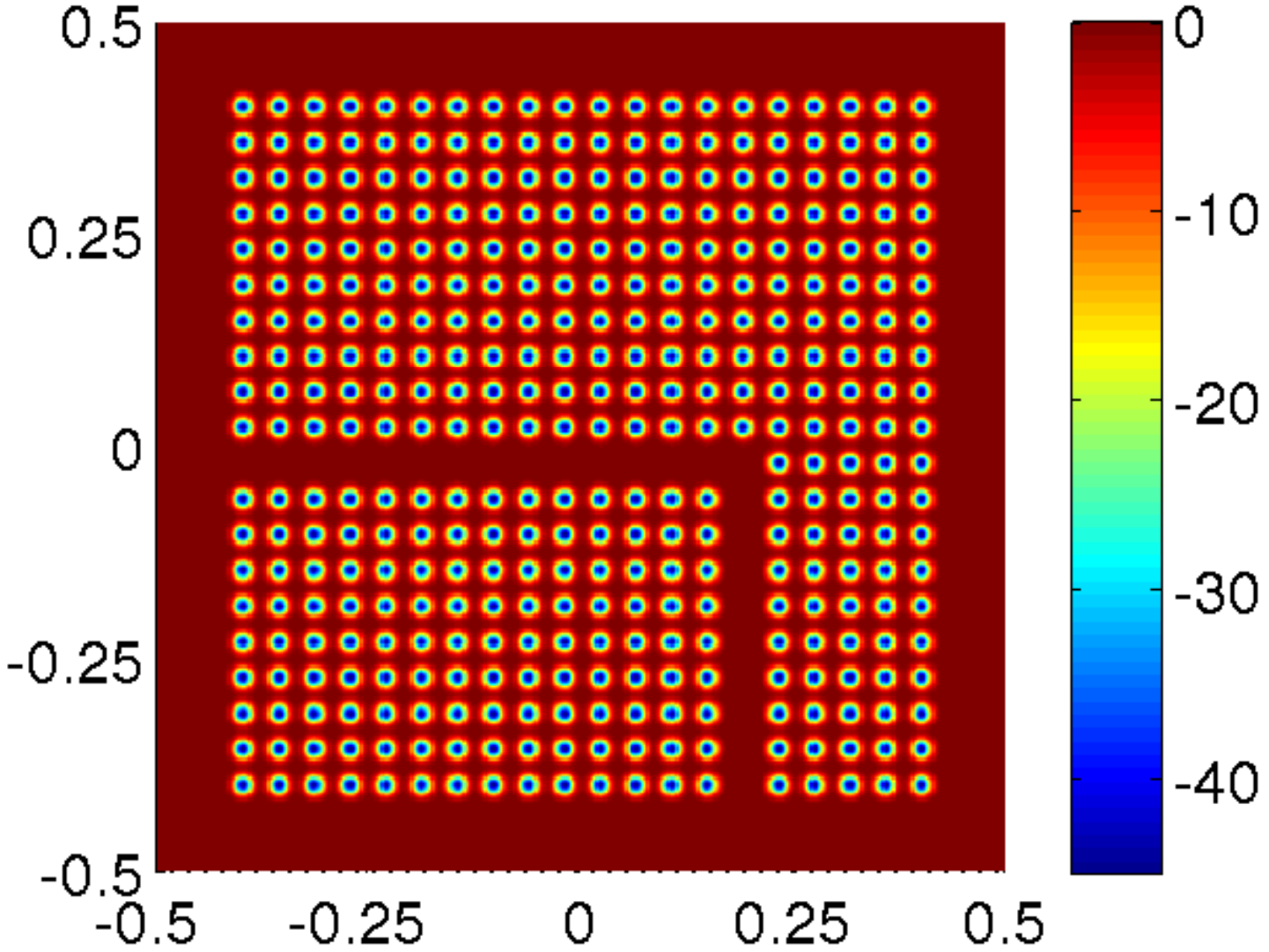


$N = 231\,361$ $T_{\text{build}} = 7.2$ sec $T_{\text{solve}} = 0.06$ sec $E \approx 10^{-7}$ (estimated)

Example: Free space scattering

$$\begin{cases} -\Delta u_{\text{out}}(\boldsymbol{x}) - k^2\,(1 - b(\boldsymbol{x}))\,u_{\text{out}}(\boldsymbol{x}) = -k^2\,b(\boldsymbol{x})\,u_{\text{in}}(\boldsymbol{x}) \\ \lim_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\big(\partial_{|\boldsymbol{x}|}u_{\text{out}}(\boldsymbol{x}) - ik\,u_{\text{out}}(\boldsymbol{x})\big) = 0 \end{cases}$$

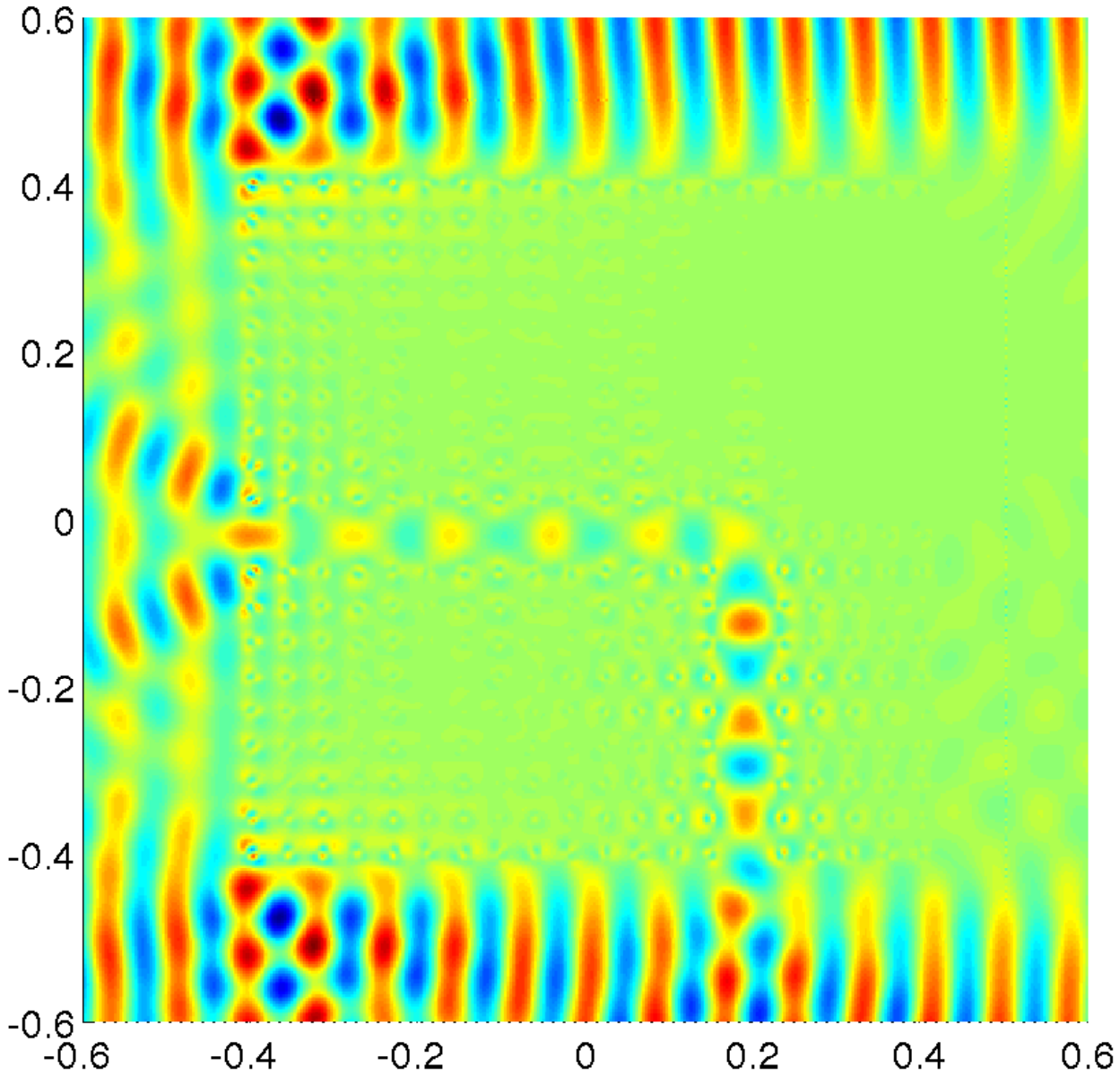
The scattering potential b — now a photonic crystal with a wave guide.



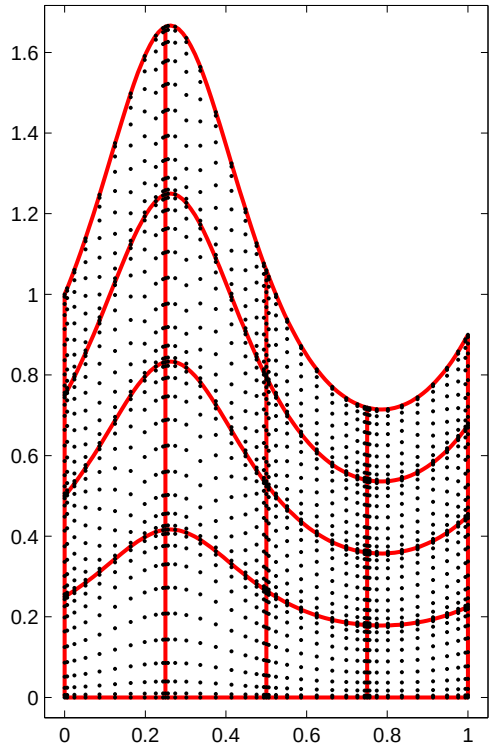
Example: Free space scattering

$$\begin{cases} -\Delta u_{\text{out}}(\boldsymbol{x}) - k^2\,(1 - b(\boldsymbol{x}))\,u_{\text{out}}(\boldsymbol{x}) = -k^2\,b(\boldsymbol{x})\,u_{\text{in}}(\boldsymbol{x}) \\ \lim_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\big(\partial_{|\boldsymbol{x}|} u_{\text{out}}(\boldsymbol{x}) - ik\,u_{\text{out}}(\boldsymbol{x})\big) = 0 \end{cases}$$

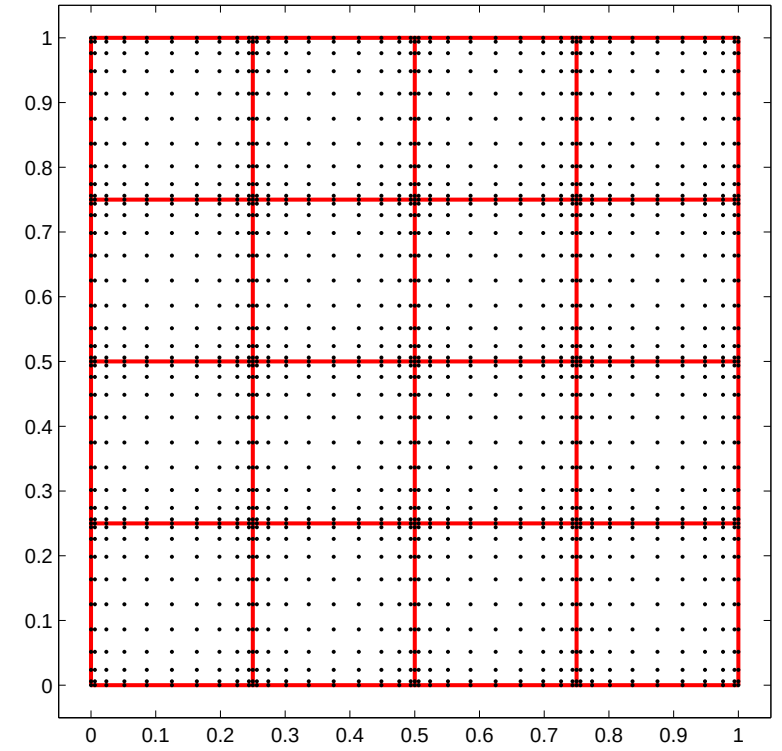
The total field $u = u_{\text{in}} + u_{\text{out}}$ *(resulting from an incoming plane wave* $u_{\text{in}}(x) = \cos(k\,x_1)$*).*



A curved domain



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



$$\Omega = [0,1]^2$$

Consider a *curved domain* Ψ as shown above and the equation

$$\text{(3)} \qquad \begin{cases} -\Delta u(\boldsymbol{y}) - \kappa^2\, u(\boldsymbol{y}) = 0 & \boldsymbol{y} \in \Psi, \\ u(\boldsymbol{y}) = f(\boldsymbol{y}) & \boldsymbol{y} \in \partial\Psi. \end{cases}$$

The reparameterization is $y_1 = x_1$ and $y_2 = \psi(y_1)\, y_2$, and so the Helmholtz equation (3) takes the form

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

Numerical results for curved domain

The equation is (constant coefficient) Helmholtz on a domain of size 35×50 wave lengths.

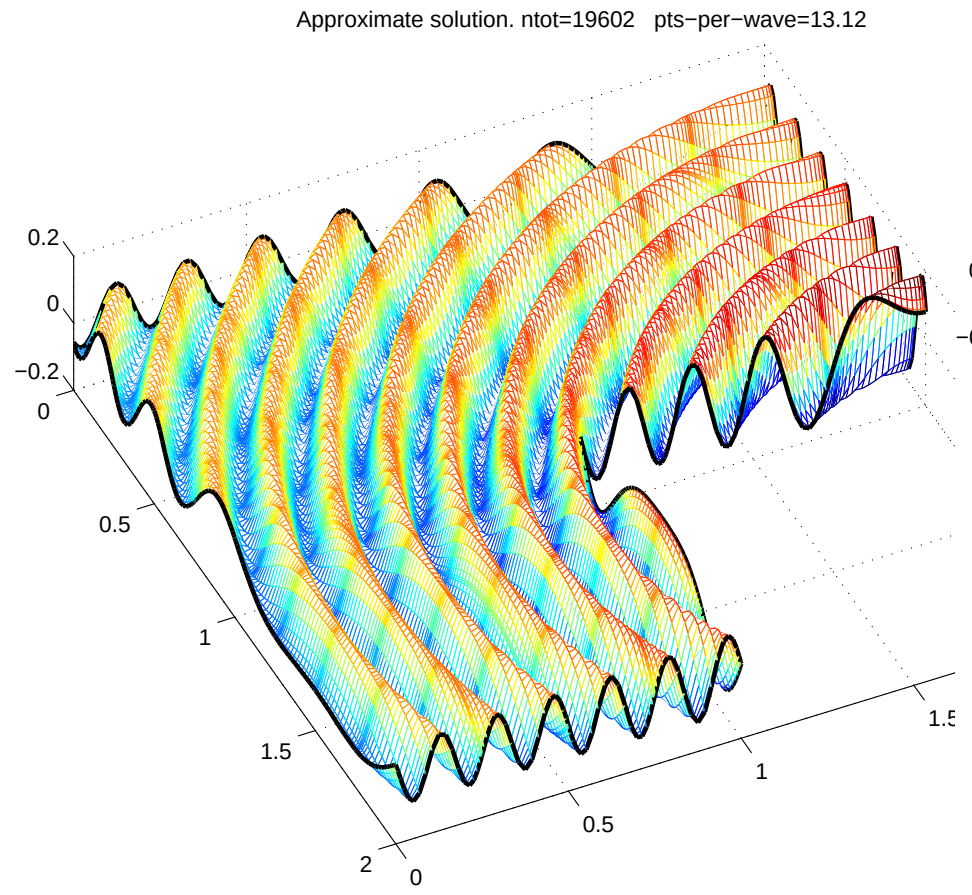
	Exact solution known		*Dirichlet data* $f = 1$		
N	E_{pot}	E_{grad}	$E_N^{(1)}$	$E_N^{(2)}$	$E_N^{(3)}$
25921	2.12685e+02	3.55772e+04	2.24618e-01	4.99854e-01	6.69023e-01
103041	3.29130e-01	5.89976e+01	1.10143e-02	5.28238e-03	6.14890e-02
410881	1.40813e-05	1.98907e-03	4.57900e-06	2.18438e-06	1.13415e-05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

Spectral composite method: numerical results — L-shaped domain

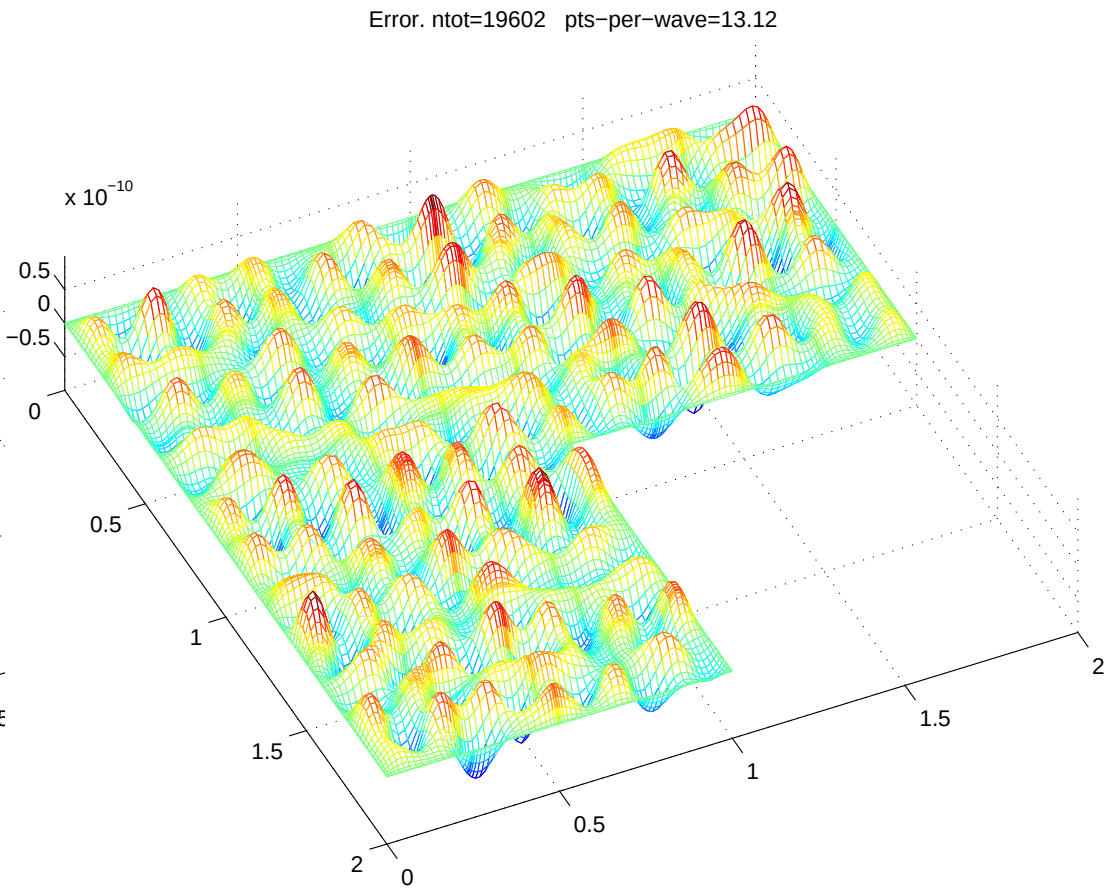
Set $\Omega = [0,2]^2 \backslash [1,2]^2$ and $\Gamma = \partial\Omega$. Consider the problem

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

We pick f as the restriction of a wave from a point source, $\boldsymbol{x} \mapsto Y_0(\kappa|\boldsymbol{x} - \hat{\boldsymbol{x}}|)$.
We then know the exact solution.



The approximate solution



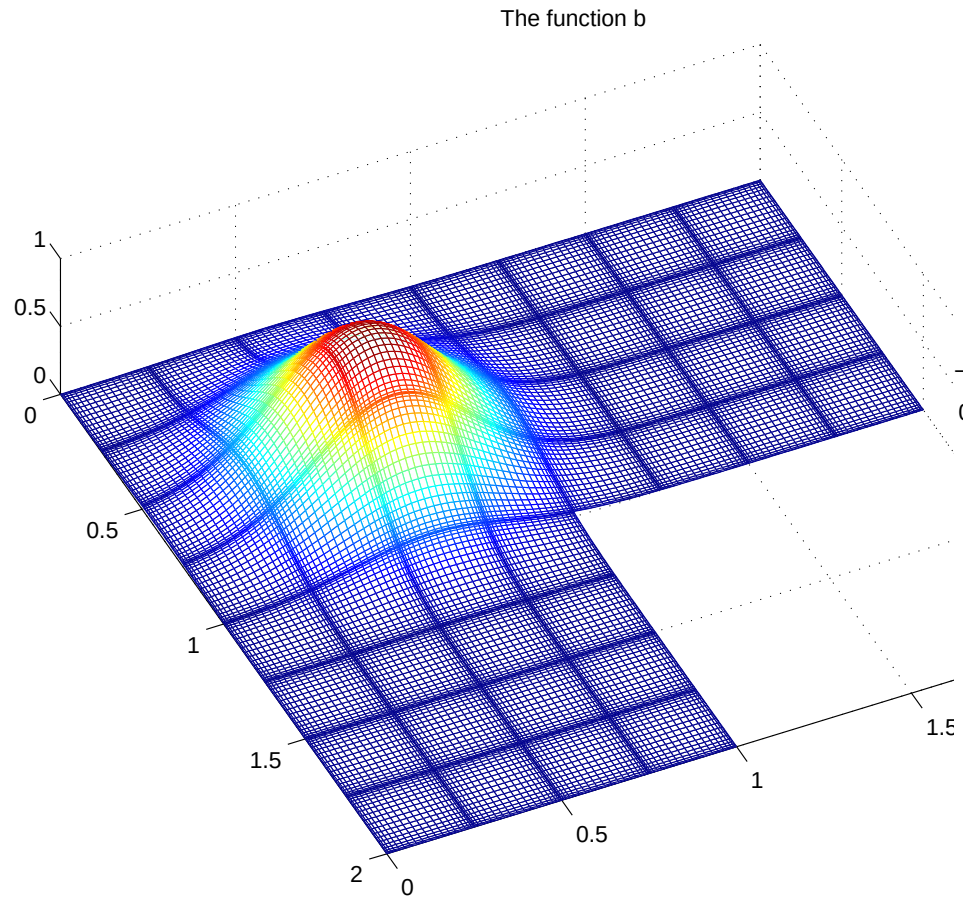
The error

Spectral composite method: numerical results — L-shaped domain

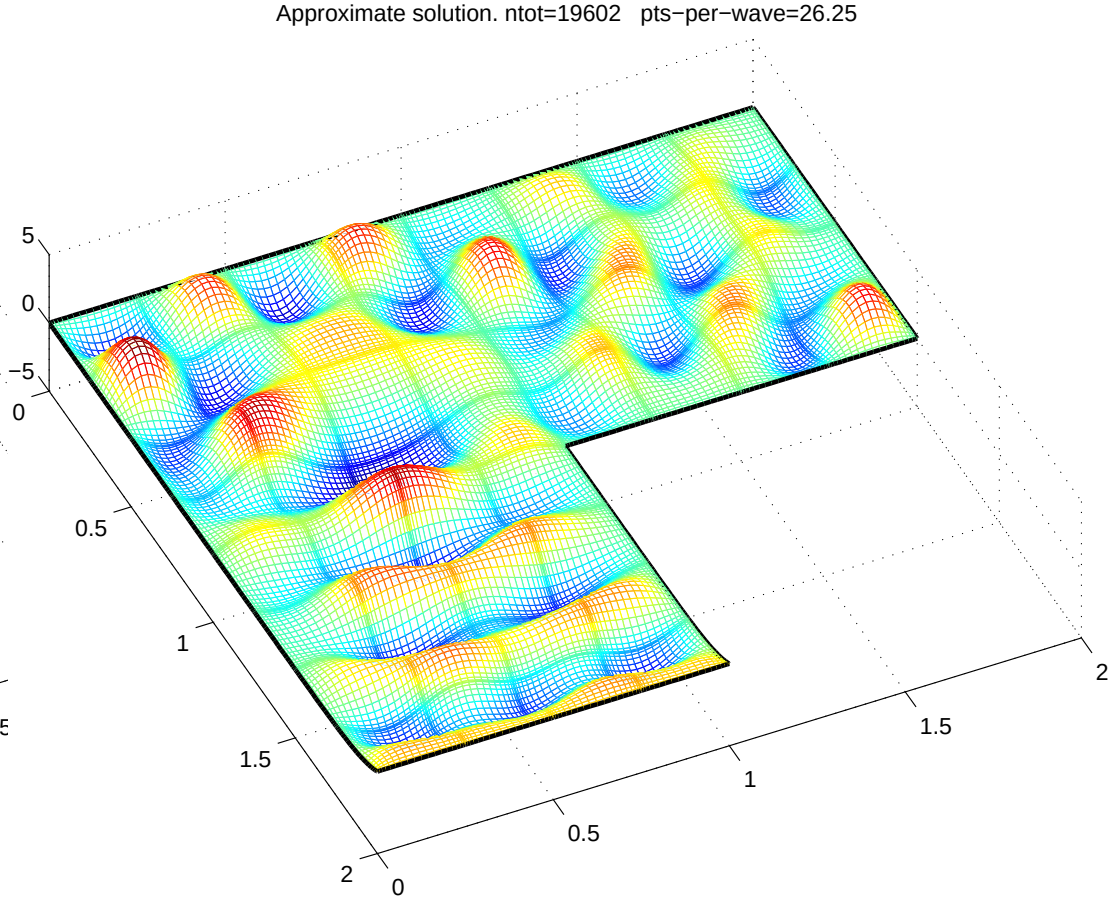
Set $\Omega = [0,2]^2 \backslash [1,2]^2$ and $\Gamma = \partial\Omega$. Consider the problem

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

We pick f that generates a sort of plane wave at the line $\{(2,t) : \ 0 \le t \le 1\}$.



The function b

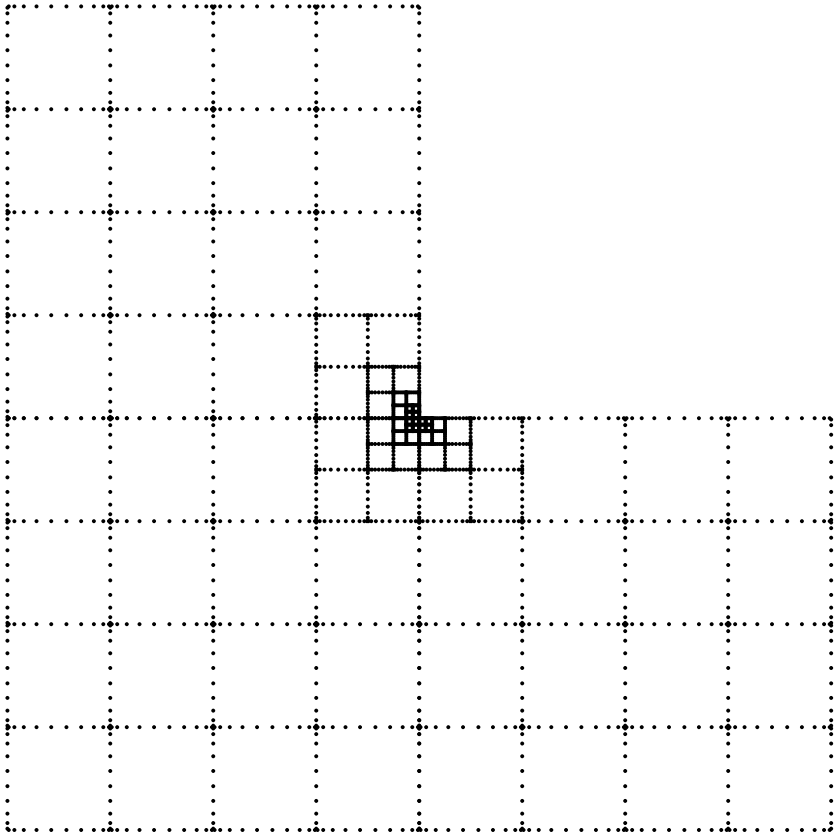


The approximate solution

Solutions for the L-shaped domain are in general singular near at the re-entrant corner.

Ignoring this problem leads to solutions with “only” 5 or 6 correct digits.

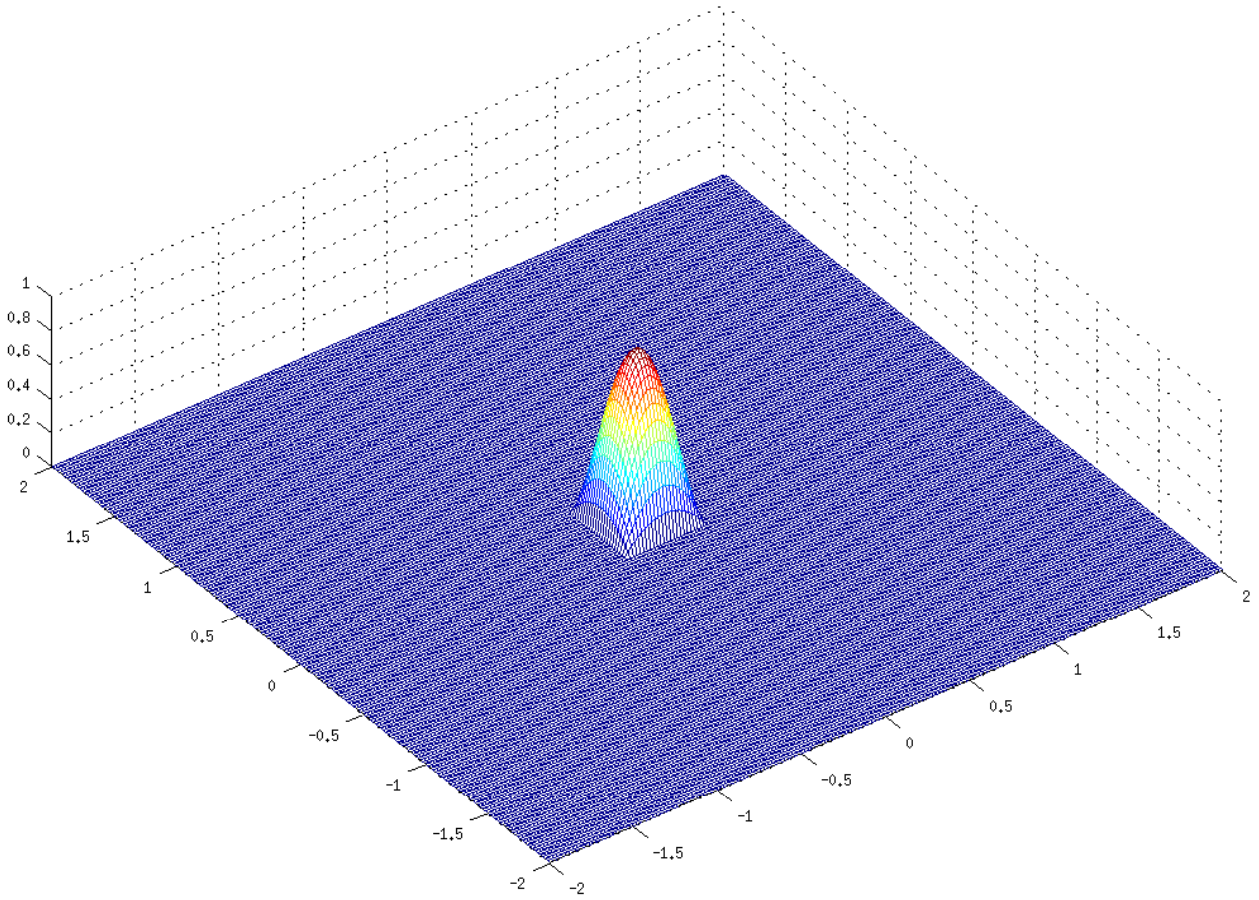
To achieve ten correct digits, *local grid refinement* is required.



New problem: Tabulation points on two boxes involved in a “merge” may not line up.
Solution: Down-sample via interpolation (joint work with S. Hao and A. Gillman).

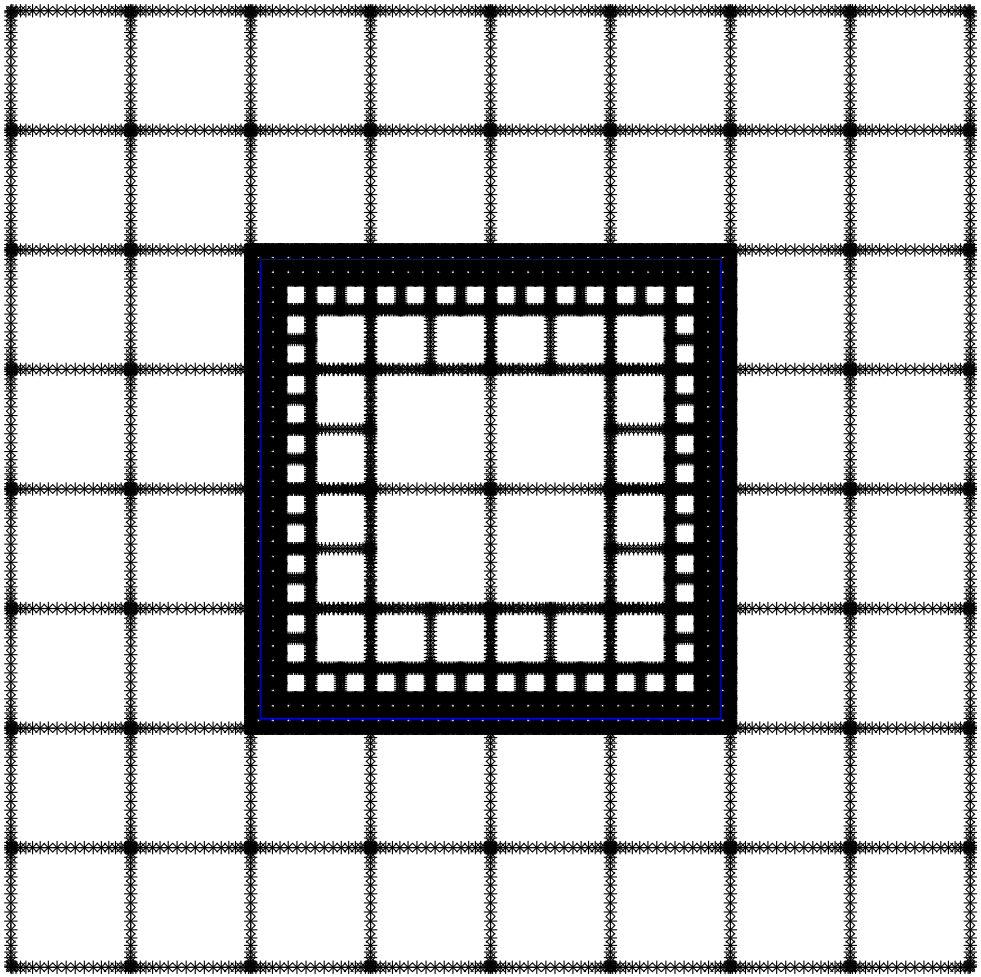
Example: Free space scattering $\begin{cases} -\Delta u_{\text{out}}(\boldsymbol{x}) - k^2\,(1 - b(\boldsymbol{x}))\,u_{\text{out}}(\boldsymbol{x}) = -k^2\,b(\boldsymbol{x})\,u_{\text{in}}(\boldsymbol{x}) \\ \lim\limits_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\big(\partial_{|\boldsymbol{x}|}u_{\text{out}}(\boldsymbol{x}) - ik\,u_{\text{out}}(\boldsymbol{x})\big) = 0 \end{cases}$

Now consider a *non-smooth* scattering potential b:



Example: Free space scattering

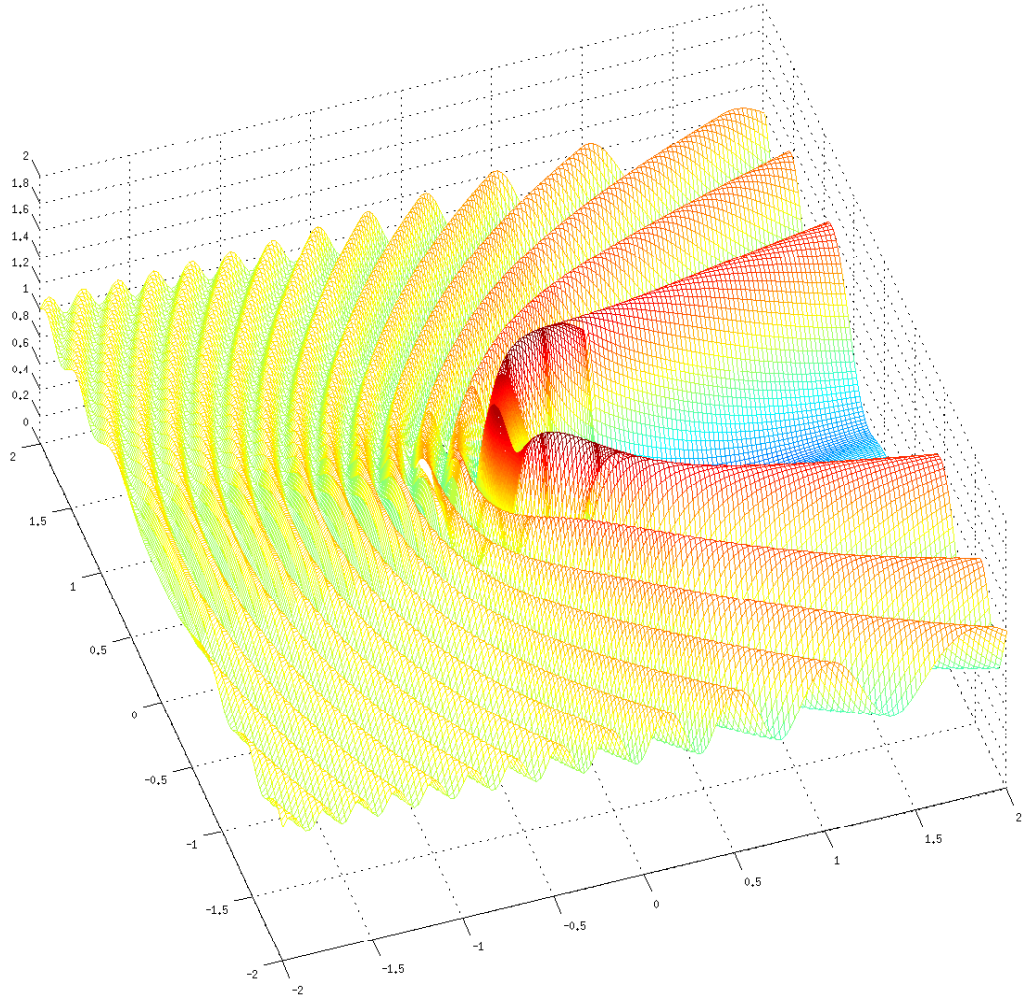
$$\begin{cases} -\Delta u_{\text{out}}(\boldsymbol{x}) - k^2\,(1 - b(\boldsymbol{x}))\,u_{\text{out}}(\boldsymbol{x}) = -k^2\,b(\boldsymbol{x}) f\,u_{\text{in}}(\boldsymbol{x}) \\ \lim_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\big(\partial_{|\boldsymbol{x}|} u_{\text{out}}(\boldsymbol{x}) - ik\,u_{\text{out}}(\boldsymbol{x})\big) = 0 \end{cases}$$



The computational grid. The support of b is the blue box.

Example: Free space scattering

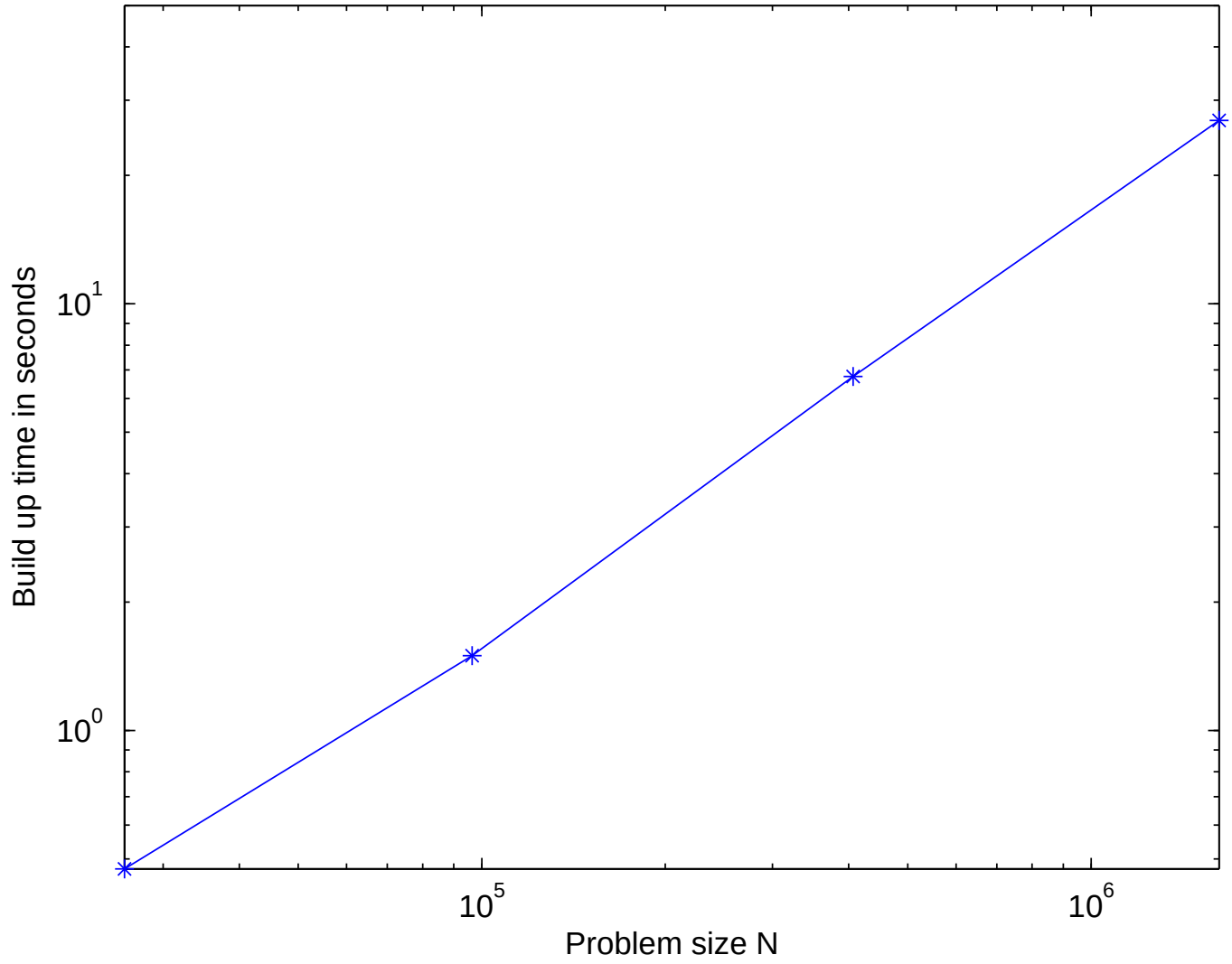
$$\begin{cases} -\Delta u_{\text{out}}(\boldsymbol{x}) - k^2\,(1 - b(\boldsymbol{x}))\,u_{\text{out}}(\boldsymbol{x}) = -k^2\,b(\boldsymbol{x})\,u_{\text{in}}(\boldsymbol{x}) \\ \lim_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\big(\partial_{|\boldsymbol{x}|}u_{\text{out}}(\boldsymbol{x}) - ik\,u_{\text{out}}(\boldsymbol{x})\big) = 0 \end{cases}$$



The magnitude of the total field resulting from an incident field $u^{\text{in}}(x) = e^{i\kappa x_1}$.

Example: Free space scattering

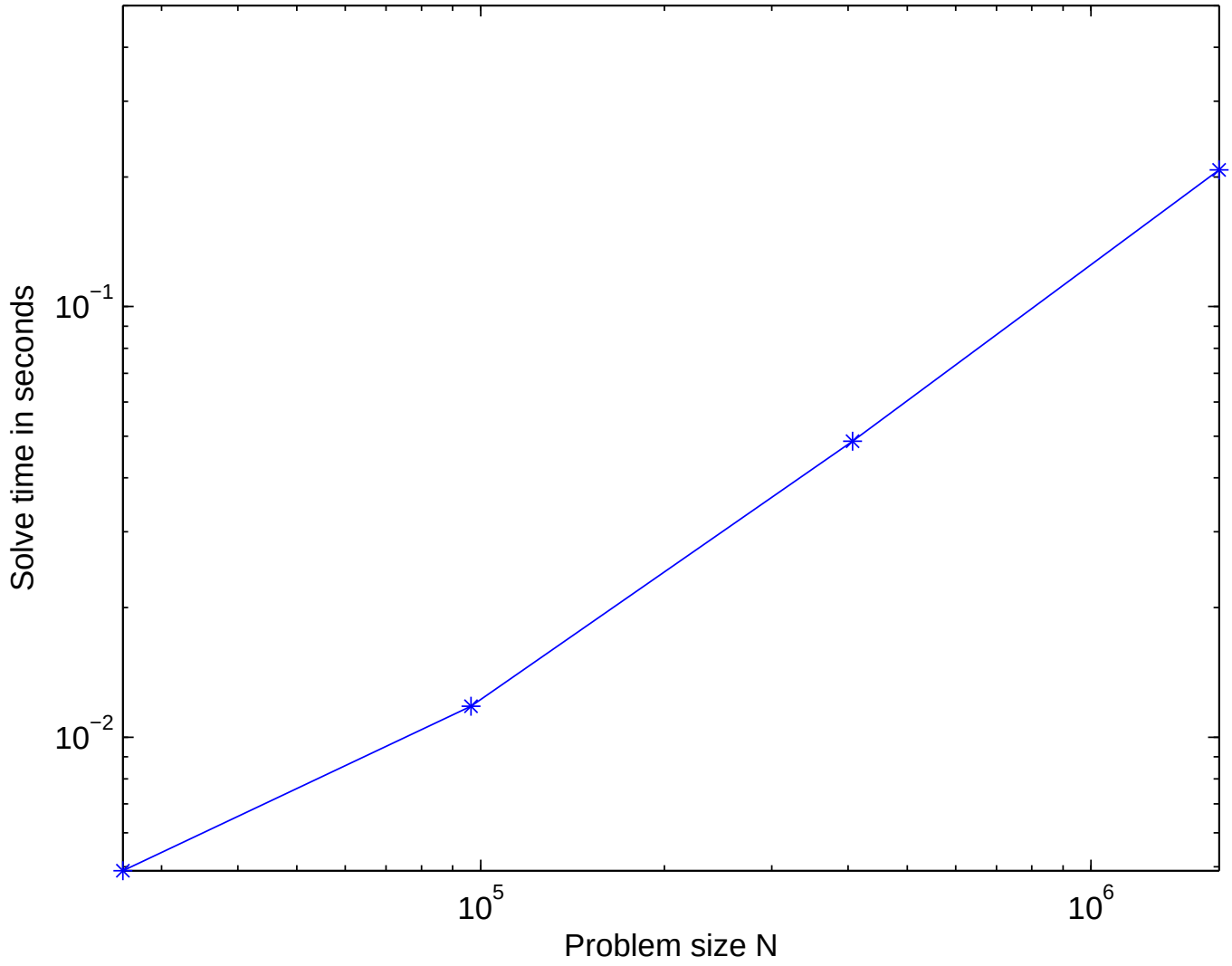
$$\begin{cases} -\Delta u_{\text{out}}(\boldsymbol{x}) - k^2\,(1 - b(\boldsymbol{x}))\,u_{\text{out}}(\boldsymbol{x}) = -k^2\,b(\boldsymbol{x})\,u_{\text{in}}(\boldsymbol{x}) \\ \lim_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\big(\partial_{|\boldsymbol{x}|}\,u_{\text{out}}(\boldsymbol{x}) - ik\,u_{\text{out}}(\boldsymbol{x})\big) = 0 \end{cases}$$



Time (in seconds) required for building the scattering operator.

Example: Free space scattering

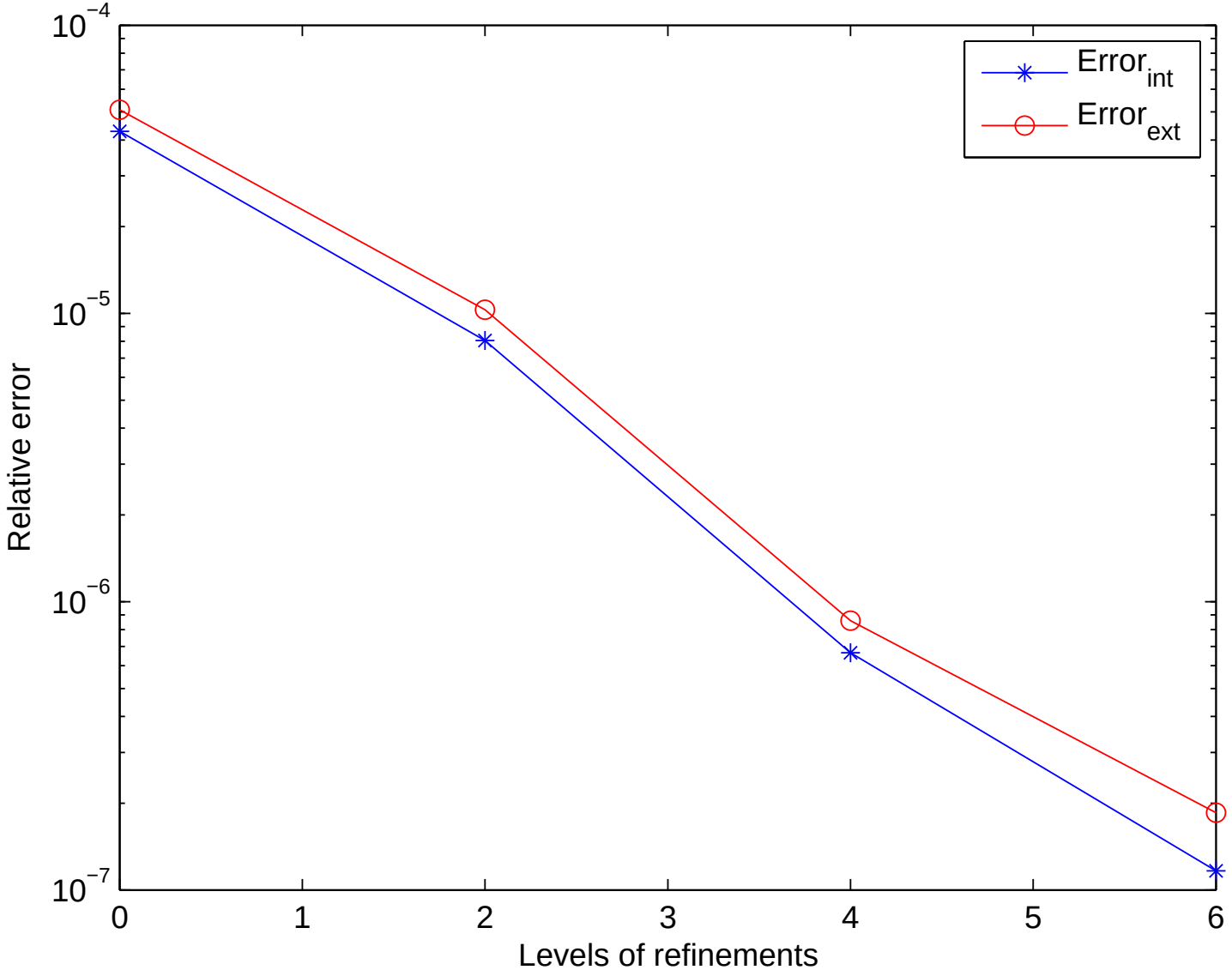
$$\begin{cases} -\Delta u_{\text{out}}(\boldsymbol{x}) - k^2\,(1 - b(\boldsymbol{x}))\,u_{\text{out}}(\boldsymbol{x}) = -k^2\,b(\boldsymbol{x})\,u_{\text{in}}(\boldsymbol{x}) \\ \lim_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\big(\partial_{|\boldsymbol{x}|}u_{\text{out}}(\boldsymbol{x}) - ik\,u_{\text{out}}(\boldsymbol{x})\big) = 0 \end{cases}$$



Time (in seconds) required for doing a full solve given an incident field.

Example: Free space scattering

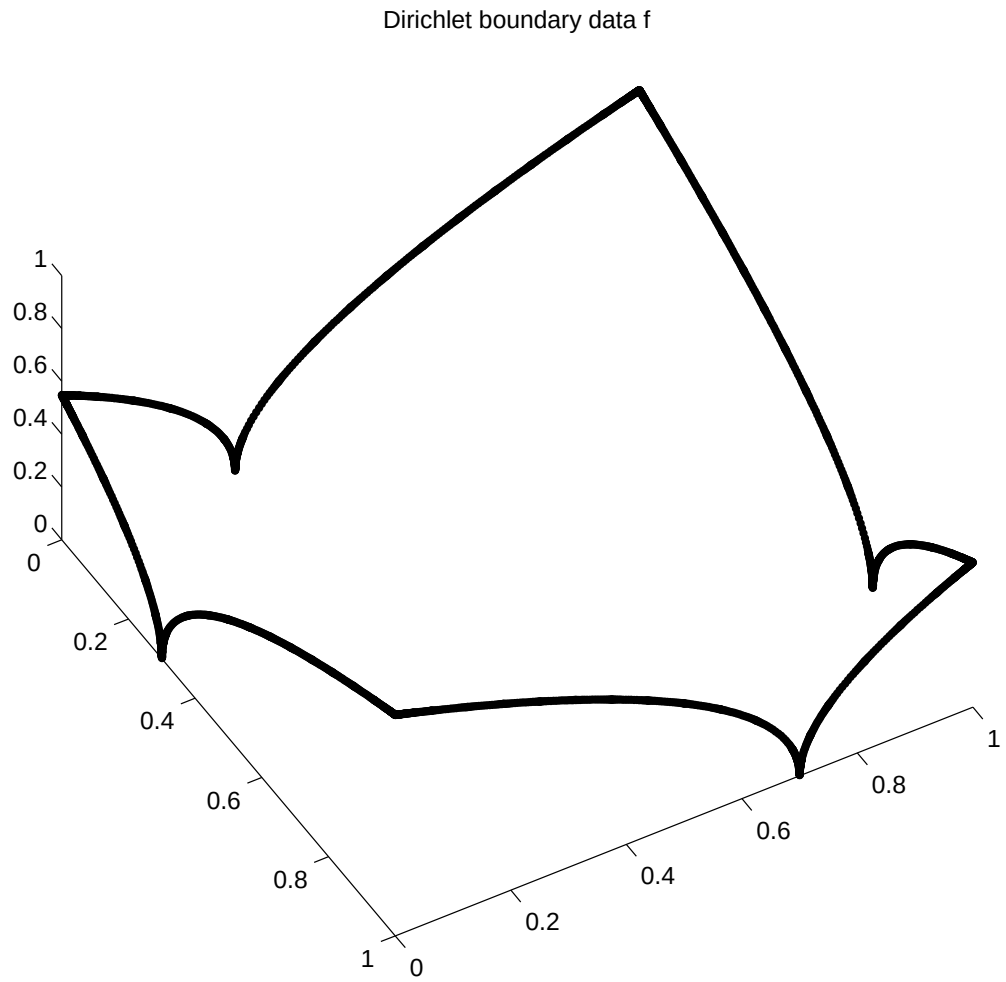
$$\begin{cases} -\Delta u_{\text{out}}(\boldsymbol{x}) - k^2\,(1 - b(\boldsymbol{x}))\,u_{\text{out}}(\boldsymbol{x}) = -k^2\,b(\boldsymbol{x})\,u_{\text{in}}(\boldsymbol{x}) \\ \lim_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\big(\partial_{|\boldsymbol{x}|}u_{\text{out}}(\boldsymbol{x}) - ik\,u_{\text{out}}(\boldsymbol{x})\big) = 0 \end{cases}$$



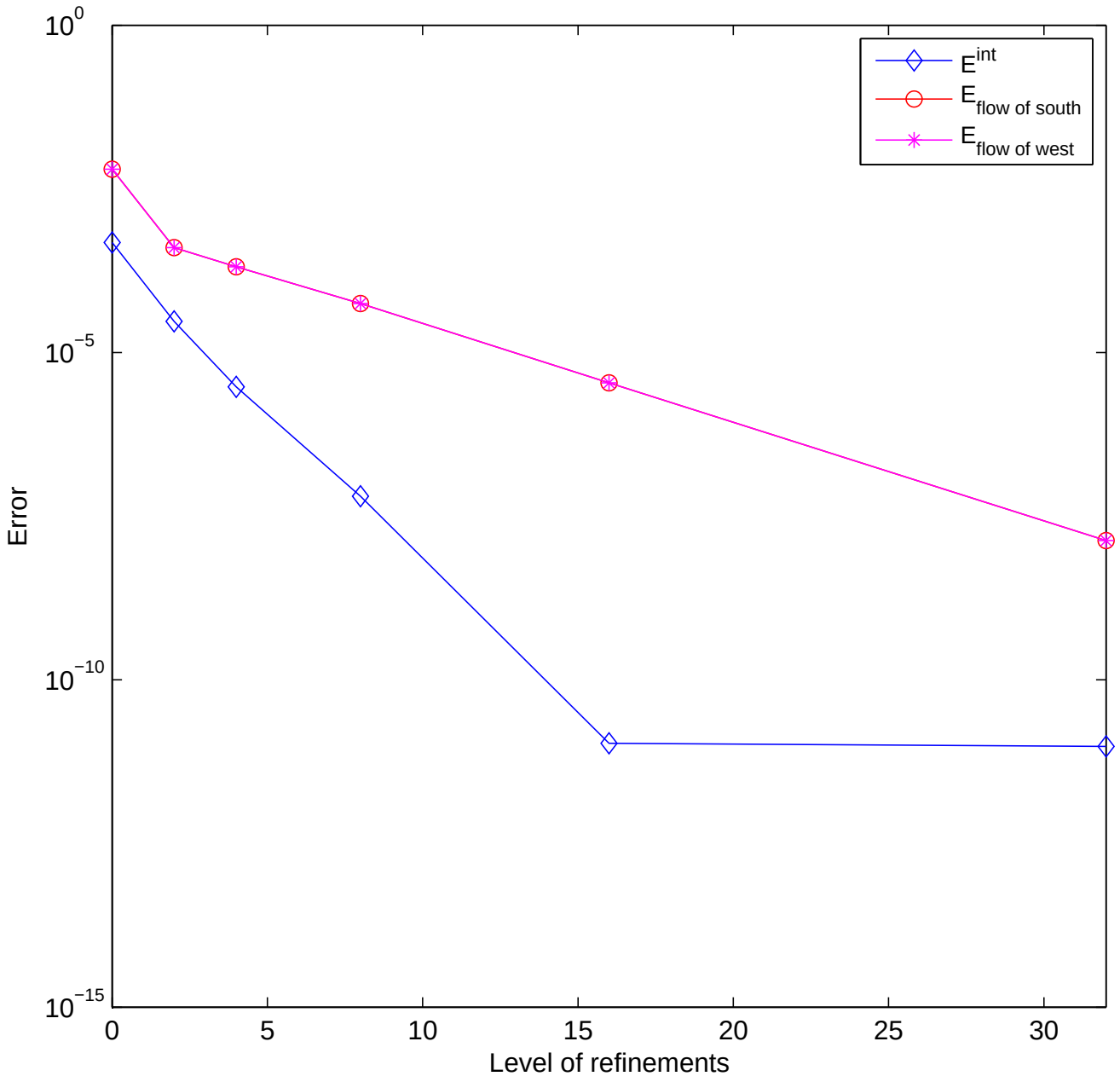
The grid refinement strategy can also be used to handle piece-wise smooth boundary data. Consider for instance a Helmholtz problem

$$\begin{cases} -\Delta\, u(\boldsymbol{x}) - \kappa^2 u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases} \tag{4}$$

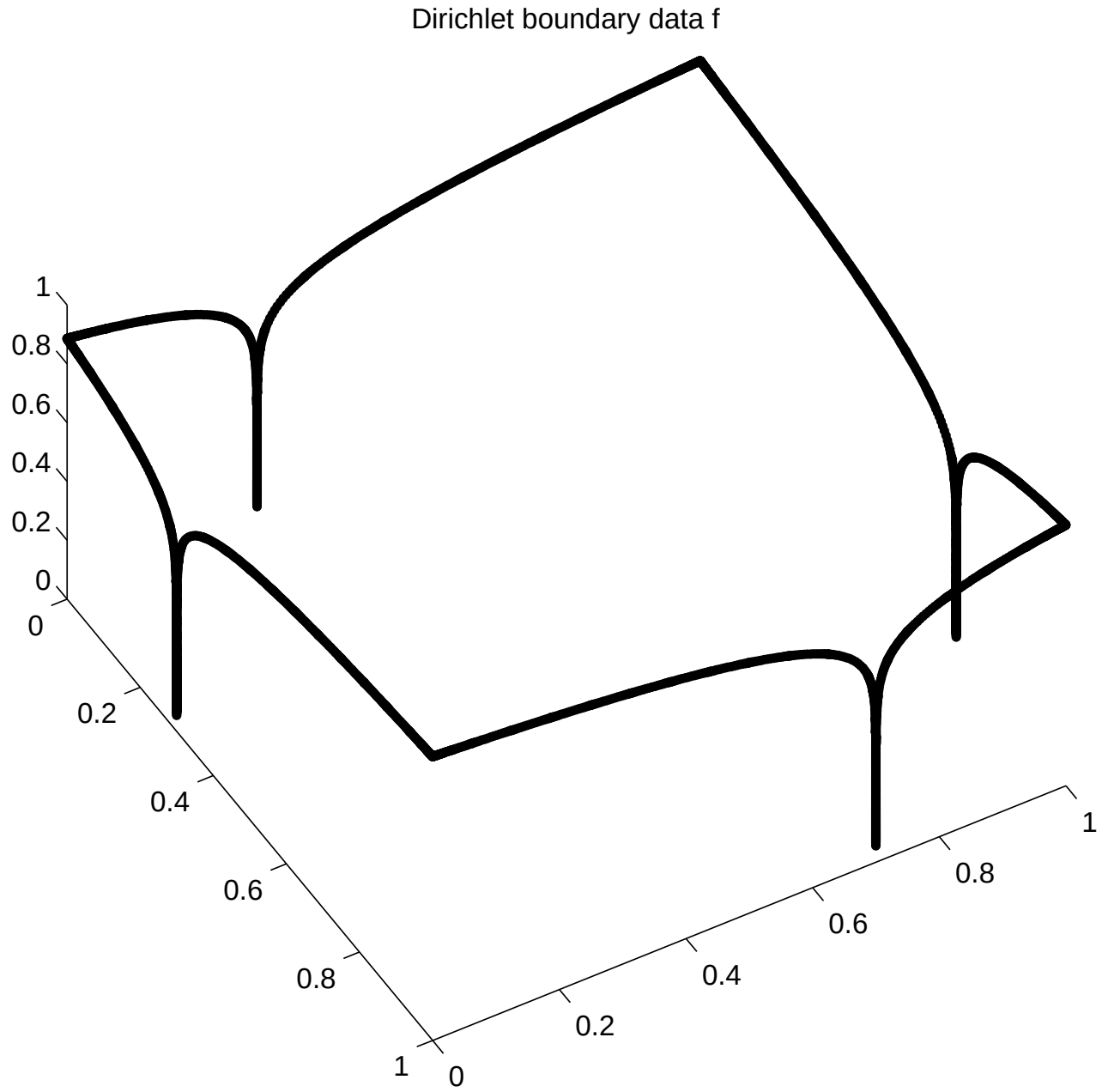
on a domain of size 8×8 wave-lengths, and where f is the function with singularities of strength $f(t) \sim \sqrt{|t - t_0|}$.



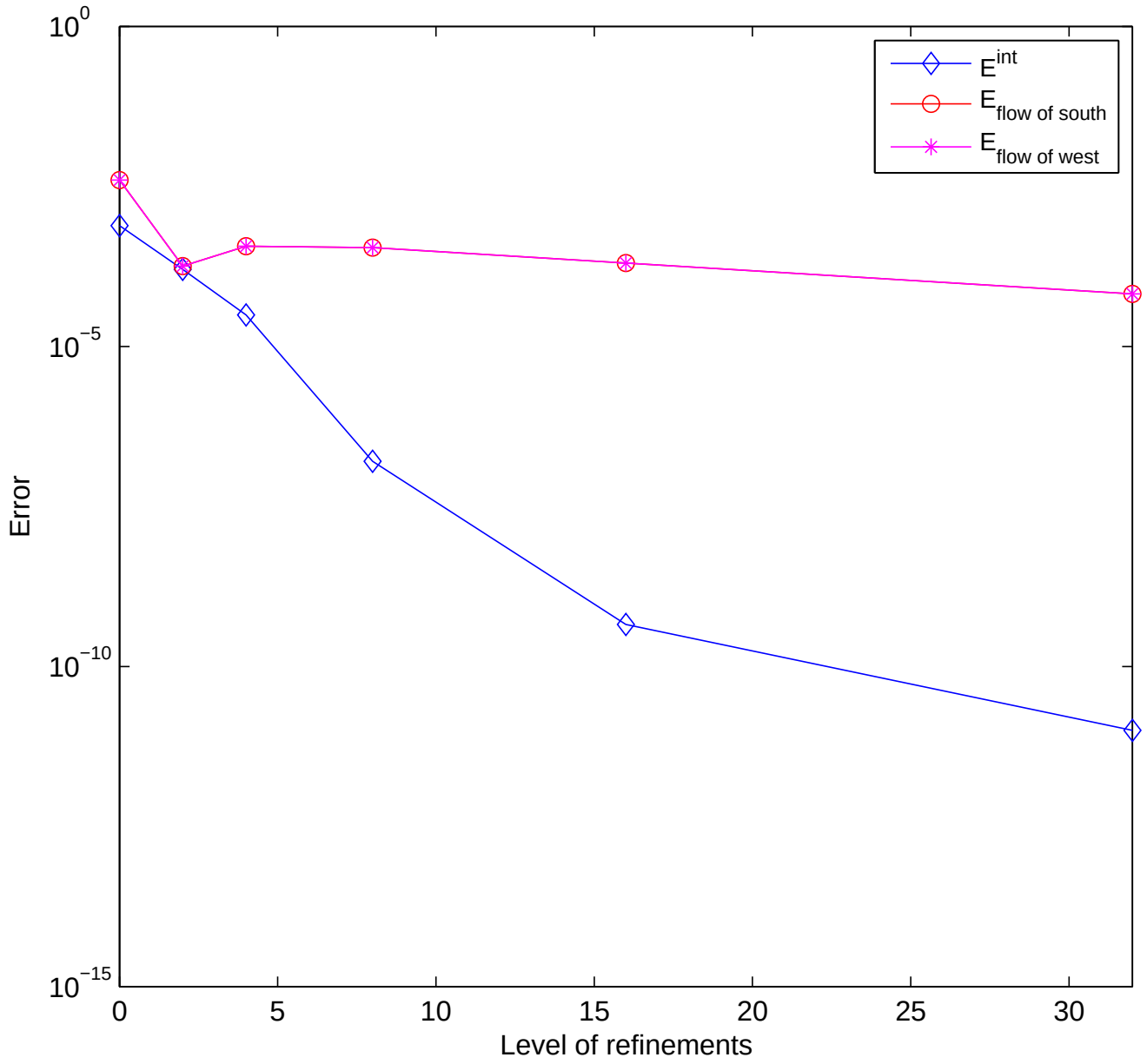
The errors in the potential and the boundary flux (estimated):



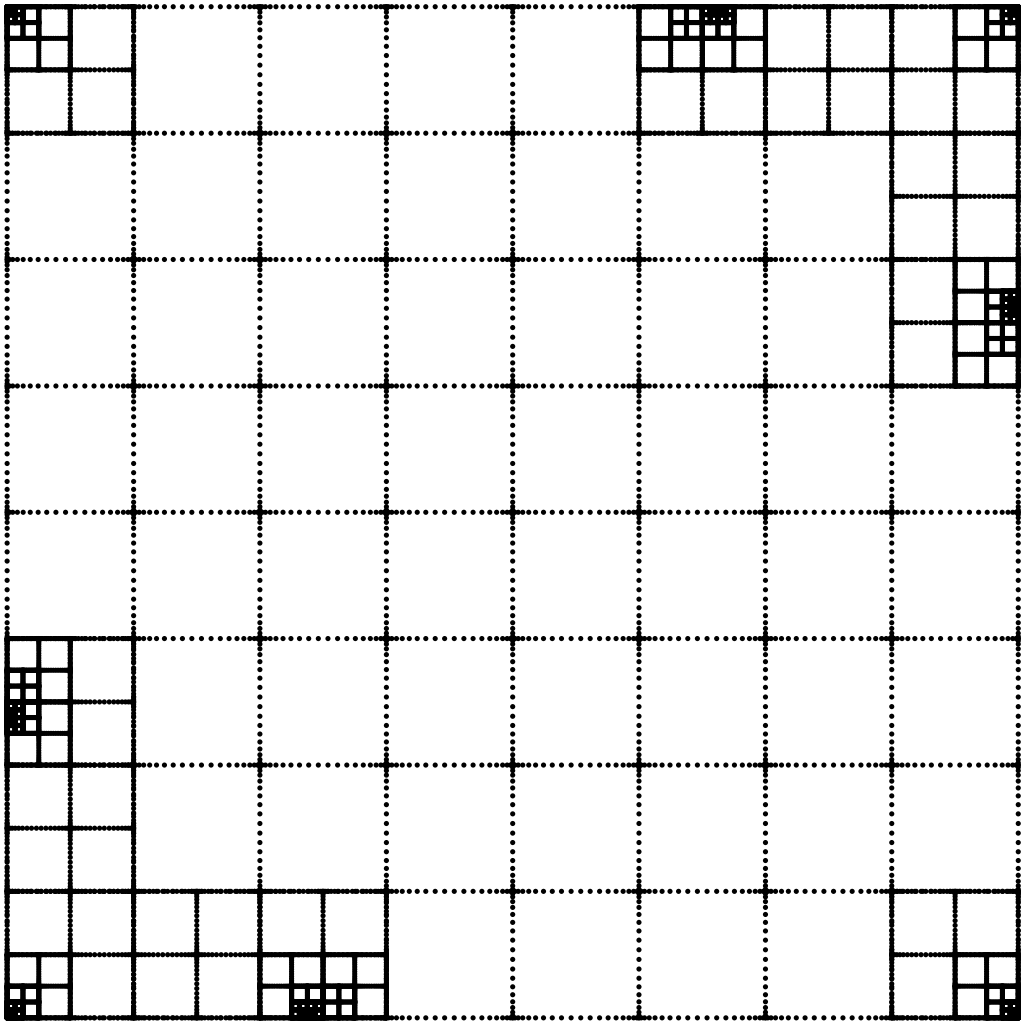
We can try something a little nastier still — now $f(t) \sim |t - t_0|^{0.1}$.



... and obtain the errors:



The locally refined mesh



Recall: The method as presented relies on a hierarchical construction of Dirichlet-to-Neumann operators for every box in a hierarchical tree.

Problem! The interior Helmholtz equation may encounter *resonances* — even for zero Dirichlet data, there may be non-trivial solutions.

Conceptual problem : The DtN operator does not always exist.

Practical problem: The DtN operator can be very ill-conditioned.

Solution: Rather than the *Dirichlet-to-Neumann map*

$$T: \quad u|_\Gamma \quad \mapsto \quad \left.\frac{\partial u}{\partial n}\right|_\Gamma$$

consider the *impedance map*

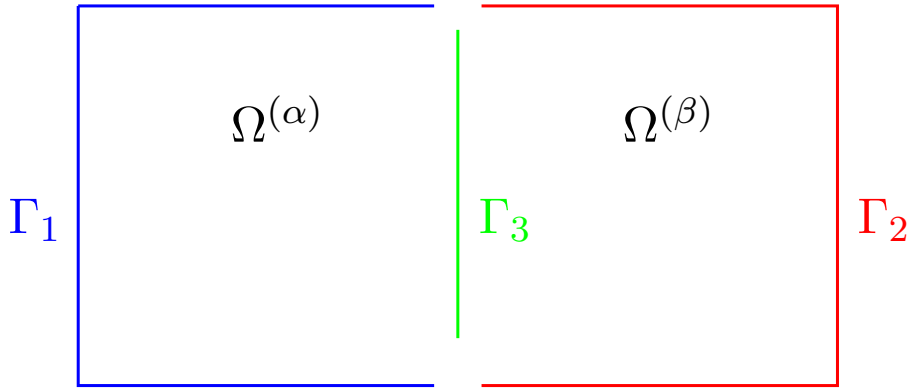
$$E: \quad u|_\Gamma + \mathrm{i}\left.\frac{\partial u}{\partial n}\right|_\Gamma \quad \mapsto \quad u|_\Gamma - \mathrm{i}\left.\frac{\partial u}{\partial n}\right|_\Gamma$$

The impedance map exists for every wave-number, and is a unitary map.

Joint work with Alexander Barnett and Adrianna Gillman of Dartmouth college.

The build stage can be accelerated to optimal $O(N)$ complexity!

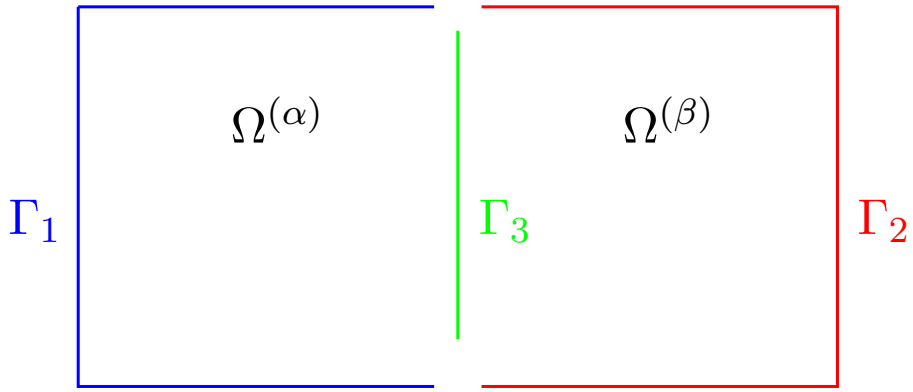
Consider the merge of two patches $\Omega^{(\alpha)}$ and $\Omega^{(\beta)}$ with boundaries Γ_1, Γ_2, Γ_3:



In the composite spectral method we have

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_{1,1}^{(\alpha)} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{2,2}^{(\beta)} \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{T}_{1,3}^{(\alpha)} \\ \mathbf{T}_{2,3}^{(\beta)} \end{bmatrix} \left(\mathbf{T}_{3,3}^{(\alpha)} - \mathbf{T}_{3,3}^{(\beta)}\right)^{-1} \left[-\mathbf{T}_{3,1}^{(\alpha)} \mid \mathbf{T}_{3,2}^{(\beta)}\right]}_{\text{low rank update!}}.$$

Consider the merge of two patches $\Omega^{(\alpha)}$ and $\Omega^{(\beta)}$ with boundaries Γ_1, Γ_2, Γ_3:

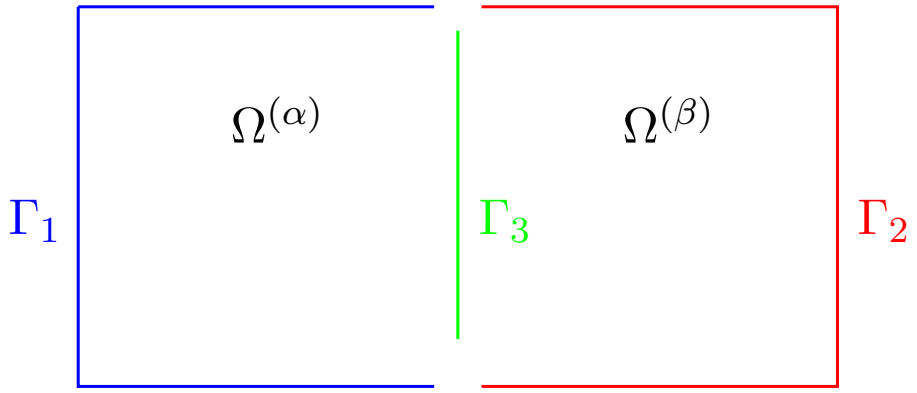


In the composite spectral method we have

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_{1,1}^{(\alpha)} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{2,2}^{(\beta)} \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{T}_{1,3}^{(\alpha)} \\ \mathbf{T}_{2,3}^{(\beta)} \end{bmatrix} \left(\mathbf{T}_{3,3}^{(\alpha)} - \mathbf{T}_{3,3}^{(\beta)}\right)^{-1} \left[-\mathbf{T}_{3,1}^{(\alpha)} \mid \mathbf{T}_{3,2}^{(\beta)}\right]}_{\text{low rank update!}}.$$

There is more structure!

Consider the merge of two patches $\Omega^{(\alpha)}$ and $\Omega^{(\beta)}$ with boundaries Γ_1, Γ_2, Γ_3:



In the composite spectral method we have

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}^{(\alpha)}_{1,1} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}^{(\beta)}_{2,2} \end{bmatrix} + \begin{bmatrix} \mathbf{T}^{(\alpha)}_{1,3} \\ \mathbf{T}^{(\beta)}_{2,3} \end{bmatrix} \left(\mathbf{T}^{(\alpha)}_{3,3} - \mathbf{T}^{(\beta)}_{3,3}\right)^{-1} \left[-\mathbf{T}^{(\alpha)}_{3,1} \mid \mathbf{T}^{(\beta)}_{3,2}\right].$$

There is more structure:

- The blue terms are of low numerical rank (say rank 40 to precision 10^{-10}).
- The red terms are "hierarchically block separable" matrices.
  (Their off-diagonal blocks have low rank, cf. $\mathcal{H}$-matrices, etc).

The bottom line is that *the solution operators can be built in optimal $O(N)$ time.*
(Not true when N is scaled to the wave-length for Helmholtz-type problems.)

Joint work with Adrianna Gillman.

Problem	N	T_{build}	T_{solve}	MB
Laplace	1.7e6	91.68	0.34	1611.19
	6.9e6	371.15	1.803	6557.27
	2.8e7	1661.97	6.97	26503.29
	1.1e8	6894.31	30.67	106731.61
Helmholtz I	1.7e6	62.07	0.202	1611.41
	6.9e6	363.19	1.755	6557.12
	2.8e7	1677.92	6.92	26503.41
	1.1e8	7584.65	31.85	106738.85
Helmholtz II	1.7e6	93.96	0.29	1827.72
	6.9e6	525.92	2.13	7151.60
	2.8e7	2033.91	8.59	27985.41
Helmholtz III	1.7e6	105.58	0.44	1712.11
	6.9e6	510.37	2.085	7157.47
	2.8e7	2714.86	10.63	29632.89

(About six accurate digits in solution.)

Thanks to A. Barnett for use of a work-station!

Problem	$\epsilon = 10^{-7}$		$\epsilon = 10^{-10}$		$\epsilon = 10^{-12}$	
	E_{pot}	E_{grad}	E_{pot}	E_{grad}	E_{pot}	E_{grad}
Laplace	6.54e-05	1.07e-03	2.91e-08	5.52e-07	1.36e-10	8.07e-09
Helmholtz I	7.45e-06	6.56e-04	5.06e-09	4.89e-07	1.38e-10	8.21e-09
Helmholtz II	6.68e-07	3.27e-04	1.42e-09	8.01e-07	8.59e-11	4.12e-08
Helmholtz III	7.40e-07	4.16e-04	2.92e-07	5.36e-06	1.66e-09	8.02e-08

$$\text{(5)} \qquad \begin{cases} -\Delta u(\boldsymbol{x}) - c_1(\boldsymbol{x})\,\partial_1 u(\boldsymbol{x}) - c_2(\boldsymbol{x})\,\partial_2 u(\boldsymbol{x}) - c(\boldsymbol{x})\,u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

Laplace Let $c_1(\boldsymbol{x}) = c_2(\boldsymbol{x}) = c(\boldsymbol{x}) = 0$ in (5).

Helmholtz I Let $c_1(\boldsymbol{x}) = c_2(\boldsymbol{x}) = 0$, and $c(\boldsymbol{x}) = \kappa^2$ where $\kappa = 80$ in (5). This represents a vibration problem on a domain Ω of size roughly 12×12 wave-lengths. (Recall that the wave-length is given by $\lambda = \frac{2\pi}{\kappa}$.)

Helmholtz II Let $c_1(\boldsymbol{x}) = c_2(\boldsymbol{x}) = 0$, and $c(\boldsymbol{x}) = \kappa^2$ where $\kappa = 640$ in (5). This corresponds to a domain of size roughly 102×102 wave-lengths.

Helmholtz III We again set $c_1(\boldsymbol{x}) = c_2(\boldsymbol{x}) = 0$, and $c(\boldsymbol{x}) = \kappa^2$ in (5), but now we let κ grow as the number of discretization points grows to maintain a constant 12 points per wavelength.

For details on $O(N)$ methods, see:

> A. Gillman and P.G. Martinsson
> *A direct solver with O(N) complexity for variable coefficient elliptic PDEs discretized via a high-order composite spectral collocation method*
> In review. arxiv.org report #1307.2665.

Our work on linear complexity solvers is related to earlier work on linear complexity nested dissection:

- *Xia, Chandrasekaran, Gu, Li (2009)*
- *Le Borne, Grasedyck, Kriemann (2007)*
- *Schmitz and Ying (2012)*
- *Gillman and Martinsson (2011)*

There may be short-cuts to finding the inverses ...

Recent work indicates that randomized sampling could be used to very rapidly find a data-sparse representation of a matrix (in $\mathcal{H}$ / $\mathcal{H}^2$ / HSS / ...format).

The idea is to extract information by applying the operator to be compressed to a sequence of random vectors. In the present context, "applying the inverse" of course corresponds simply to a linear solve.

- P.G. Martinsson, "A fast randomized algorithm for computing a Hierarchically Semi-Separable representation of a matrix". *SIAM J. on Matrix Analysis and Appl.*, **32**(4), pp. 1251–1274, 2011. (Preprint version 2008.)

- "Fast construction of hierarchical matrix representation from matrix-vector multiplication", L. Lin, J. Lu, L. Ying., *J. of Computational Physics*, **230**(10), 2011.

- "Randomized Sparse Direct Solvers", Jianlin Xia, SIMAX, **34**(1), 2013.

For more on randomized sampling in numerical linear algebra, see:
N. Halko, P.G. Martinsson, J. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions."
SIAM Review, **53**(2), 2011. pp. 217–288.

Part 3 — Direct solvers for integral equations

Recall that many boundary value problems can advantageously be recast as *boundary integral equations.* For instance, consider (sound-soft) acoustic scattering from a finite body:

$$\text{(6)} \qquad \begin{cases} -\Delta u(\boldsymbol{x}) - k^2\, u(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \mathbb{R}^3 \backslash \overline{\Omega} \\ u(\boldsymbol{x}) = f(\boldsymbol{x}) & \boldsymbol{x} \in \partial\Omega \\ \lim\limits_{|\boldsymbol{x}|\to\infty} |\boldsymbol{x}| \big(\partial_{|\boldsymbol{x}|} u_{\text{out}}(\boldsymbol{x}) - ik\, u_{\text{out}}(\boldsymbol{x})\big) = 0. \end{cases}$$

The BVP (6) is in many ways equivalent to the BIE

$$\text{(7)} \qquad -\pi i \sigma(\boldsymbol{x}) + \int_\Gamma \left(\big(\partial_{\boldsymbol{n}(\boldsymbol{y})} + ik\big) \frac{e^{ik|\boldsymbol{x}-\boldsymbol{y}|}}{|\boldsymbol{x}-\boldsymbol{y}|} \right) \sigma(\boldsymbol{y})\, dS(\boldsymbol{y}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \partial\Omega.$$

The integral equation (7) has several advantages over the PDE (6), including:

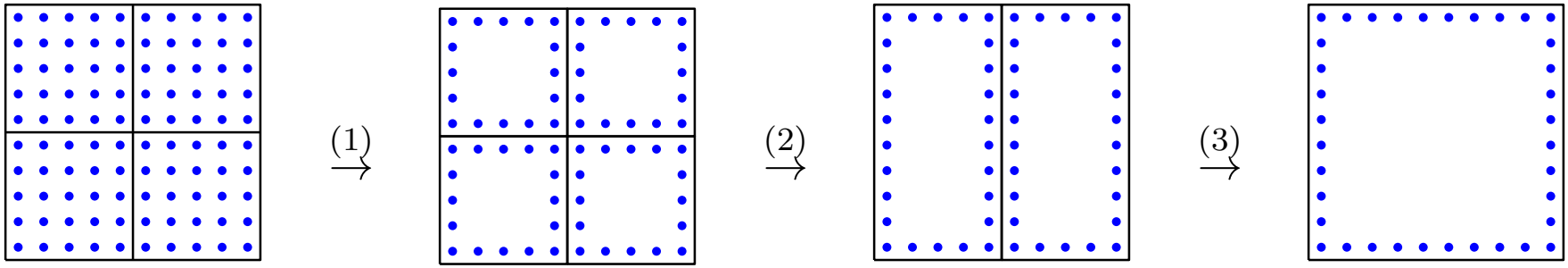
- The domain of computation $\partial\Omega$ is finite.
- The domain of computation $\partial\Omega$ is 2D instead of 3D.
- The equation (7) is inherently well-conditioned (it is a “second kind Fredholm equation”).

A serious drawback of integral equations is that they lead to *dense coefficient matrices.* Since we are interested in constructing inverses anyway, this is not a serious problem.

Part 3 — Direct solvers for integral equations, continued

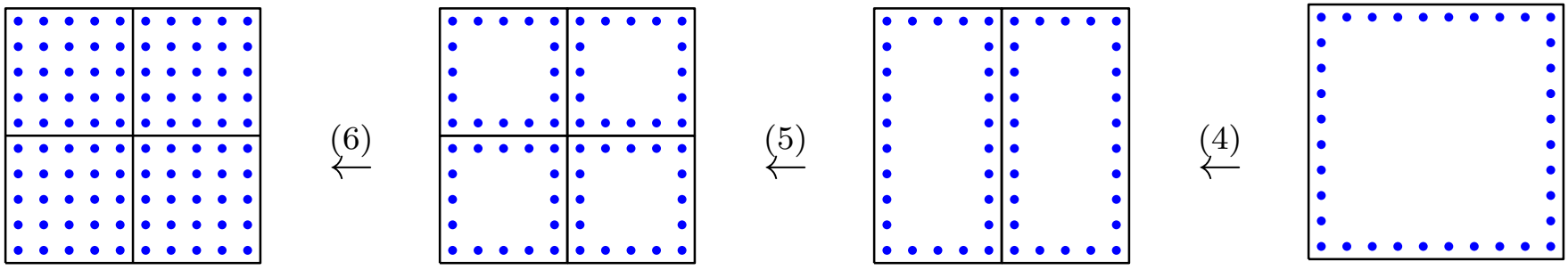
It is possible to construct direct solvers that follow the same template as before.

Upwards pass — build all solution operators:



The original grid. *Leaves reduced.* *After merge.* *After merge.*

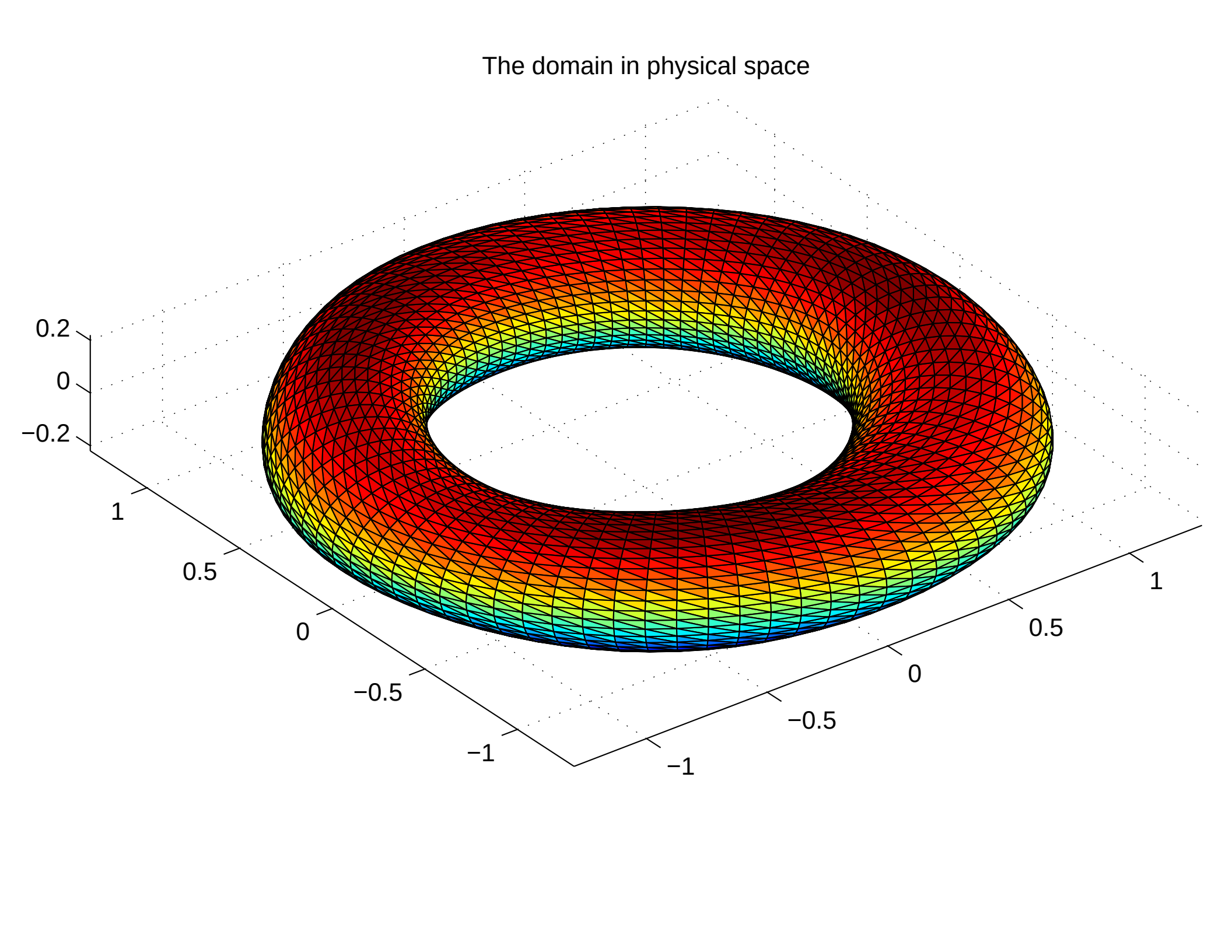
Downwards pass — solve for a particular data function (*very* fast!):

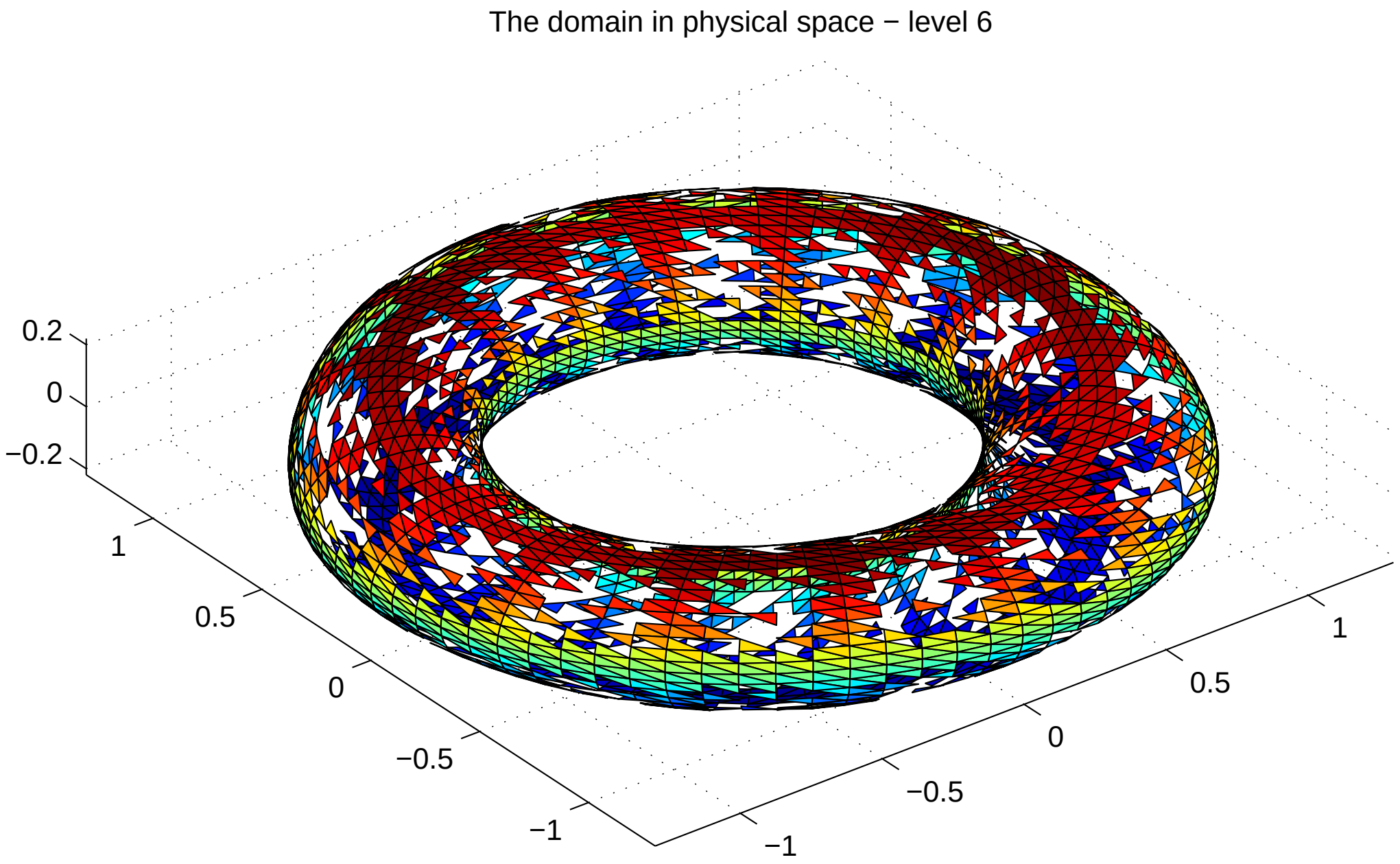


Full solution. *Solve.* *Solve.* *Top level solve.*

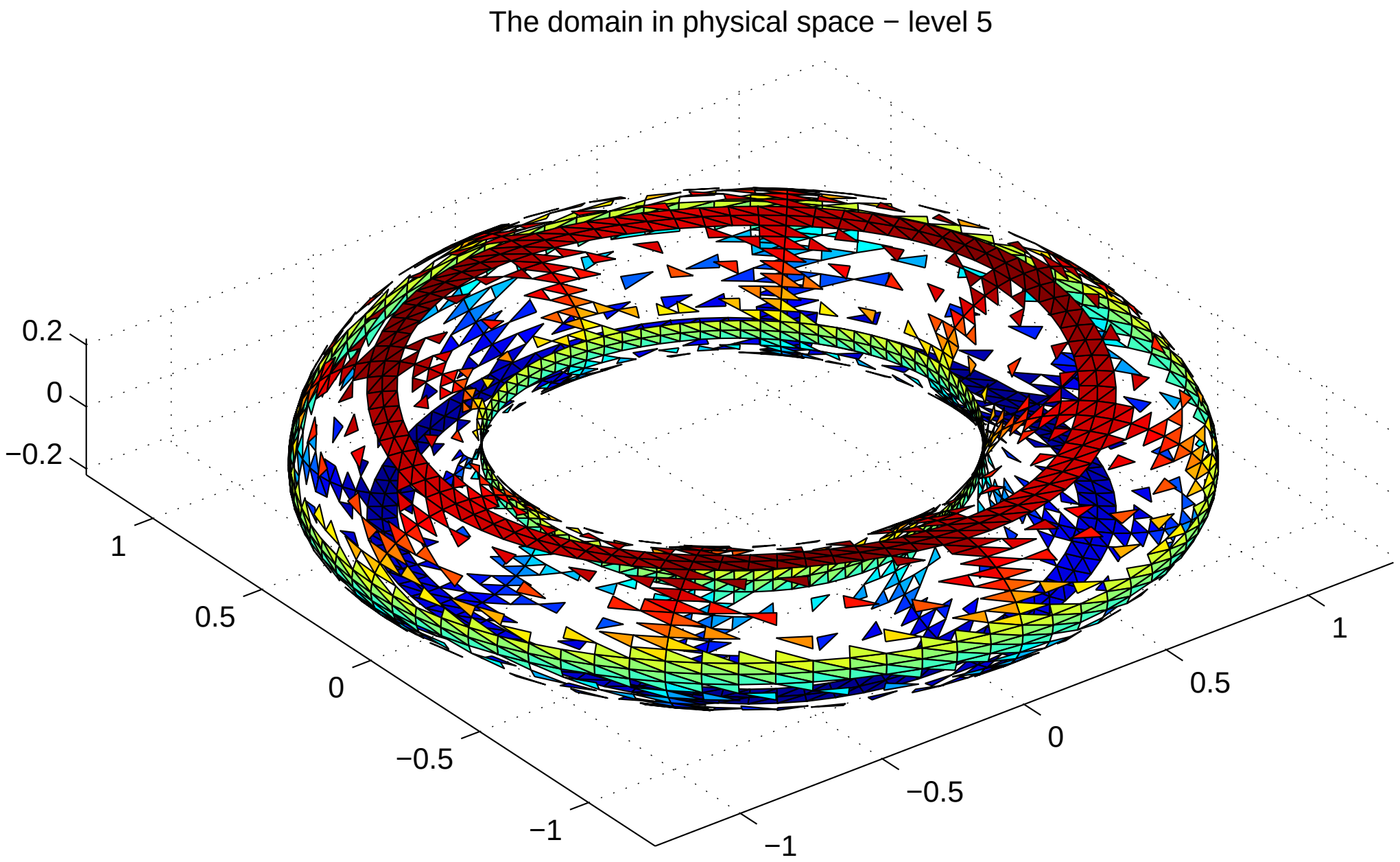
Our "solution operators" will be (conceptually) *scattering matrices* instead of DtN operators.

The operators will no longer be pure boundary operators.

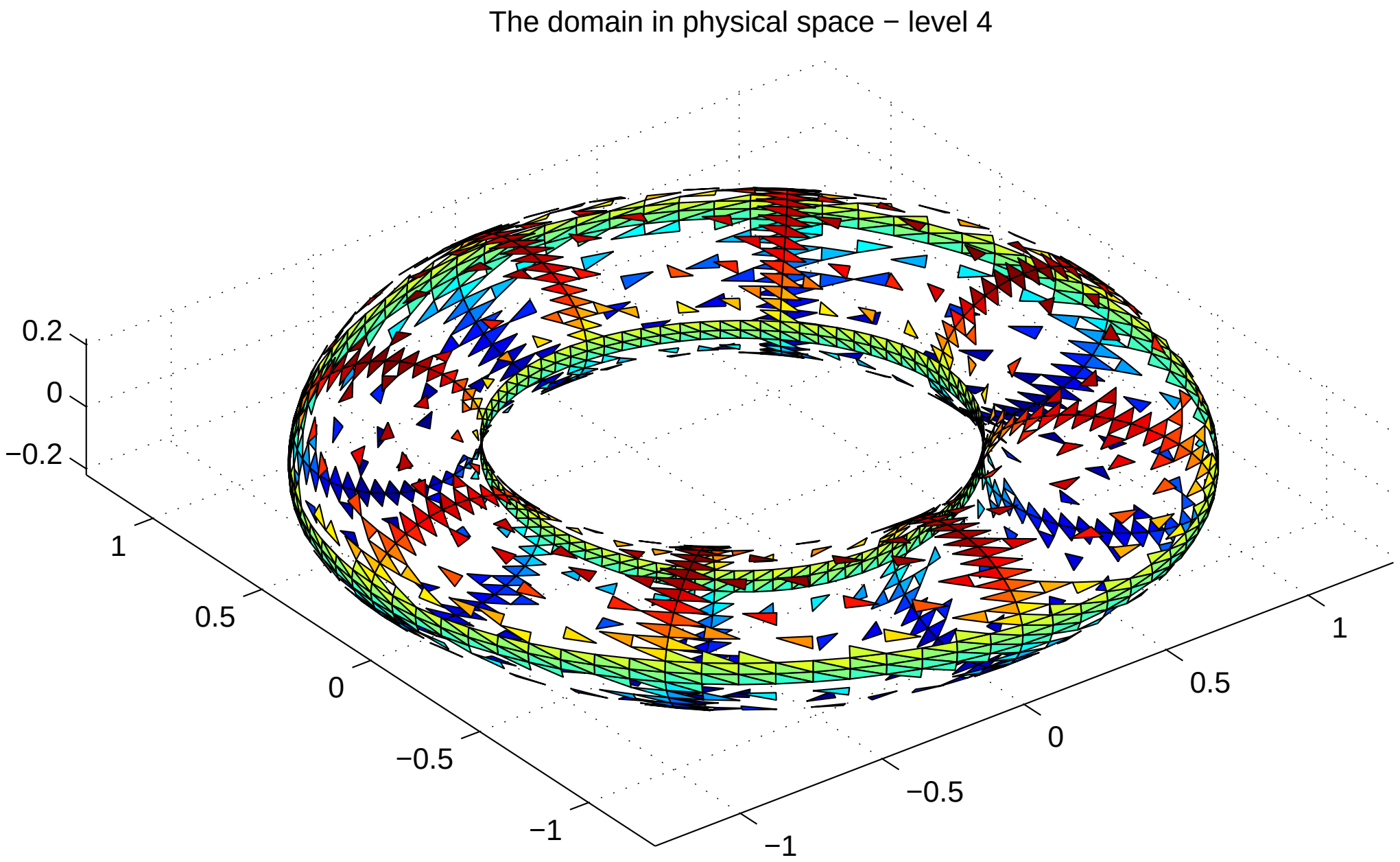

The domain in physical space
0.2
0
−0.2
1
0.5
0
−0.5
−1
−1
−0.5
0
0.5
1



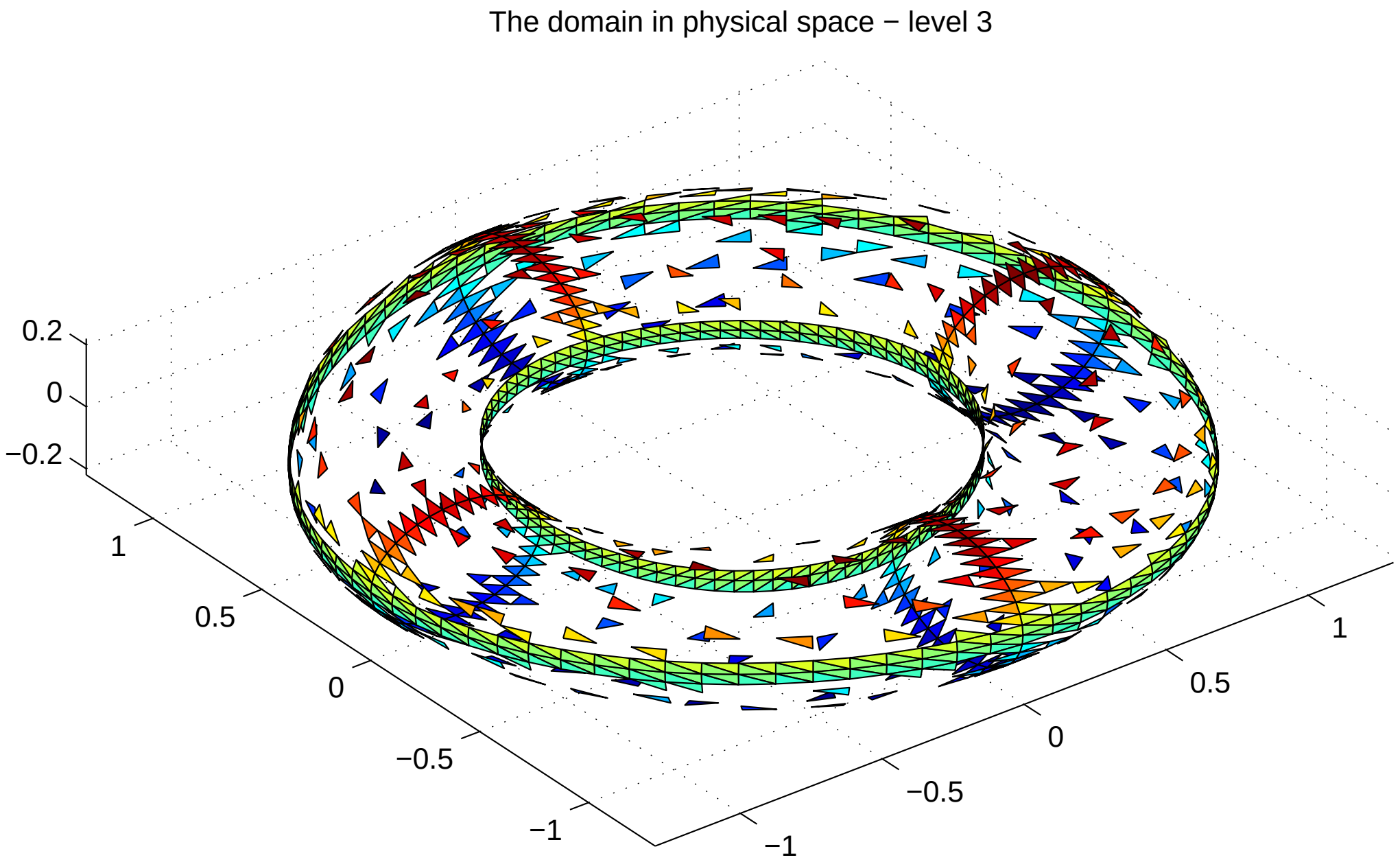
The reduced matrix represents a Nyström discretization supported on the panels shown.



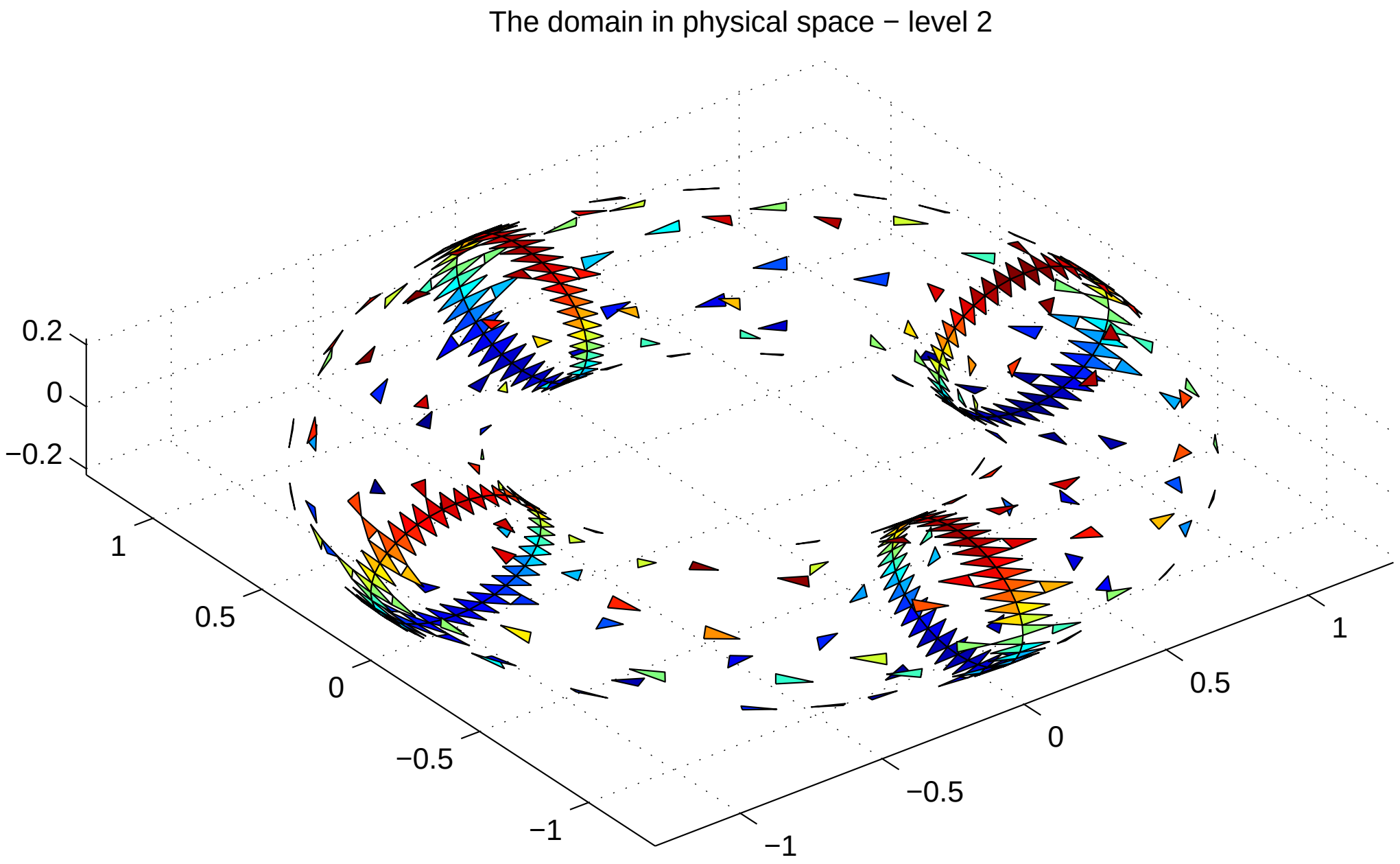
The reduced matrix represents a Nyström discretization supported on the panels shown.



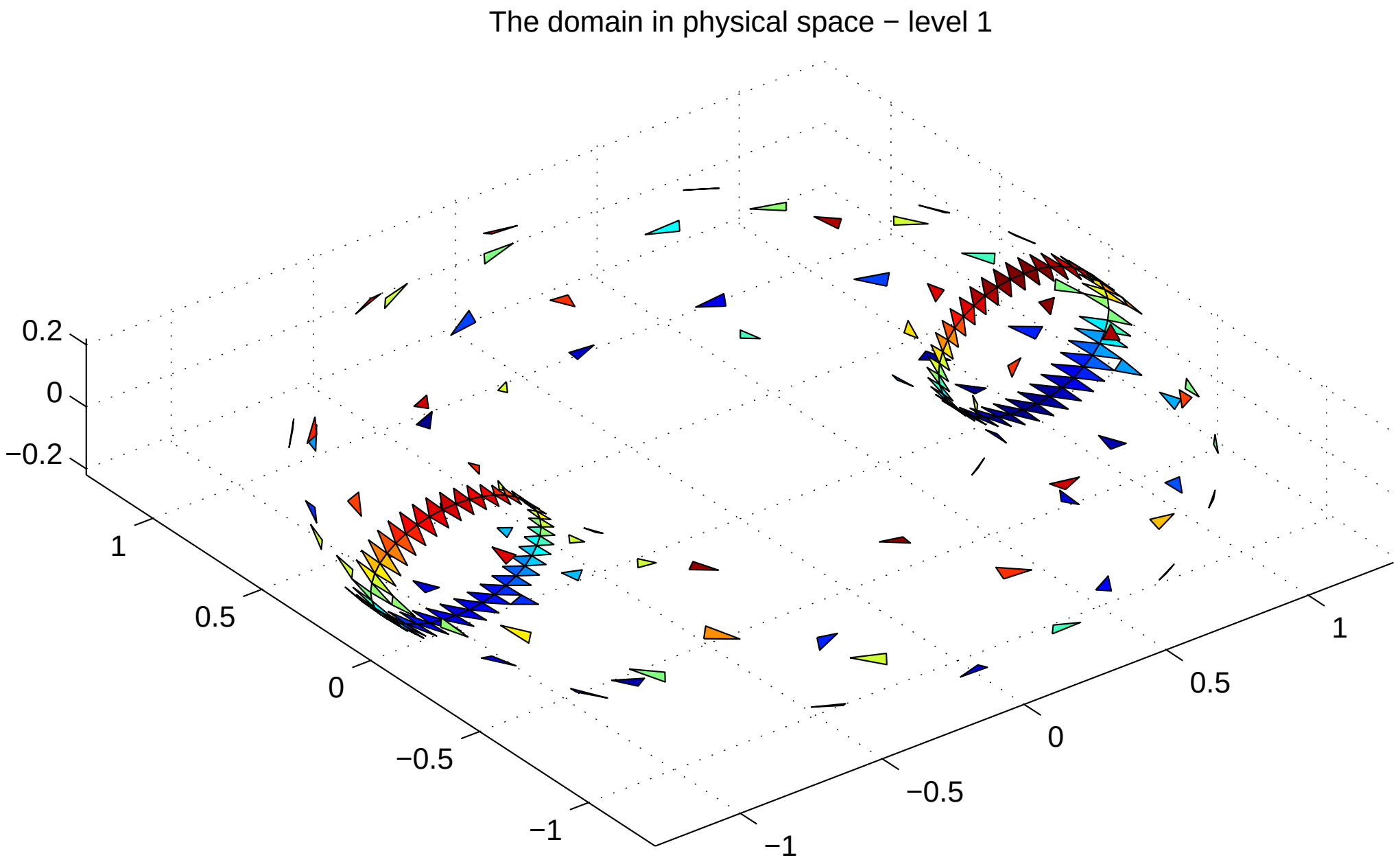
The reduced matrix represents a Nyström discretization supported on the panels shown.



The reduced matrix represents a Nyström discretization supported on the panels shown.

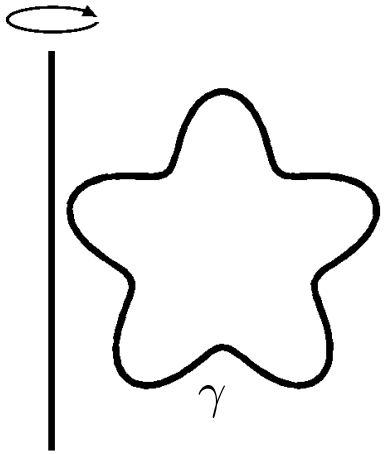


The reduced matrix represents a Nyström discretization supported on the panels shown.

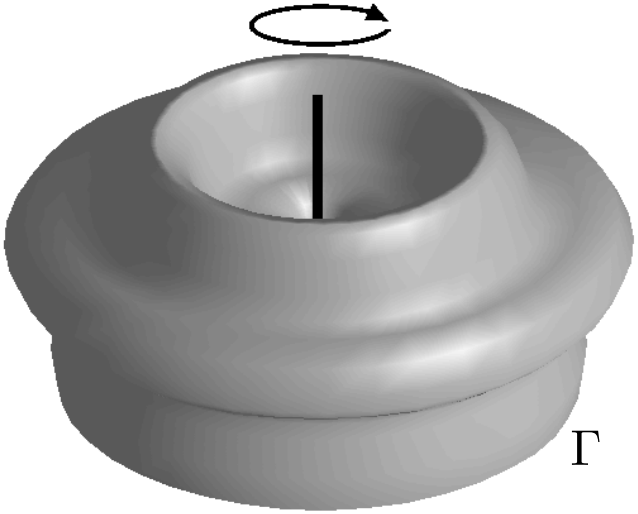


The reduced matrix represents a Nyström discretization supported on the panels shown.

Example — BIEs on rotationally symmetric bodies (2011, with S. Hao and P. Young)



Generating curve



Surface

Let Γ be a surface of rotation generated by a curve γ, and consider a BIE associated with Laplace's equation:

$$\text{(8)} \qquad \frac{1}{2}\sigma(\boldsymbol{x}) + \int_\Gamma \frac{\boldsymbol{n}(\boldsymbol{y})\cdot(\boldsymbol{x}-\boldsymbol{y})}{4\pi|\boldsymbol{x}-\boldsymbol{y}|^3}\,\sigma(\boldsymbol{y})\,dA(\boldsymbol{y}) = f(\boldsymbol{x}). \quad \boldsymbol{x}\in\Gamma$$

To (8), we apply the Fourier transform in the azimuthal angle (executed computationally via the FFT) and get

$$\frac{1}{2}\sigma_n(\boldsymbol{x}) + \int_\gamma k_n(\boldsymbol{x},\boldsymbol{y})\,\sigma_n(\boldsymbol{y})\,dl(\boldsymbol{y}) = f_n(\boldsymbol{x}), \quad \boldsymbol{x}\in\gamma, \qquad n\in\mathbb{Z}.$$

Then discretize the sequence of equations on γ using the direct solvers described (with special quadratures, *etc*).

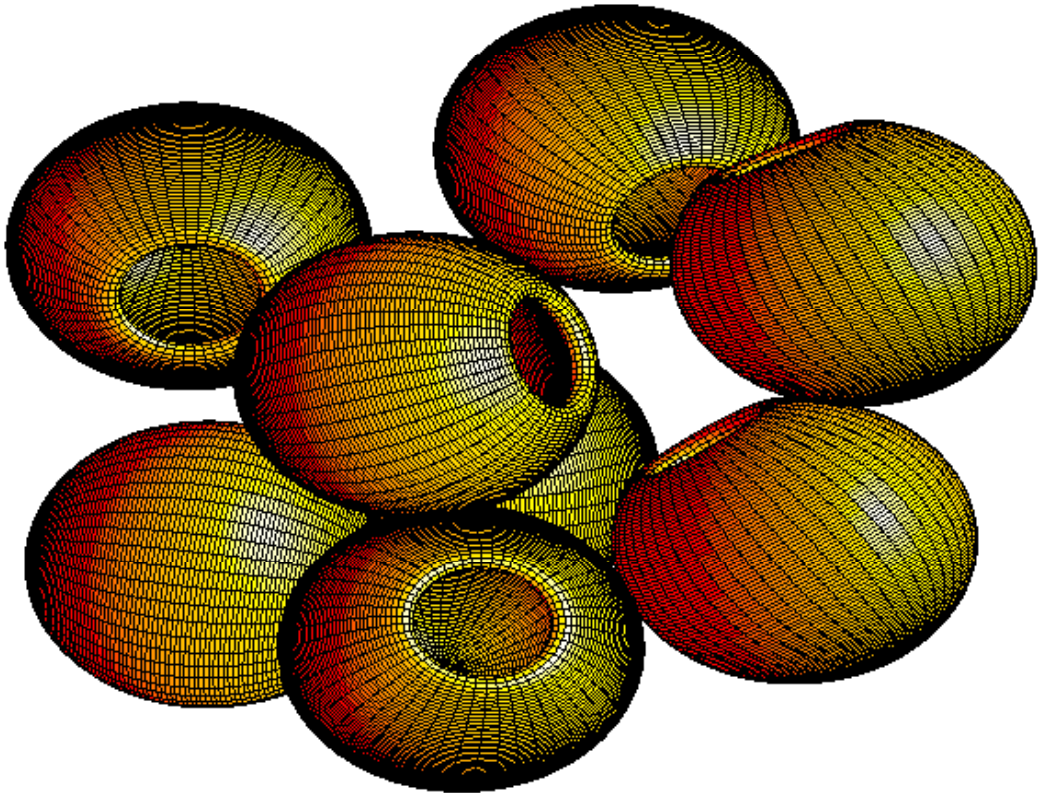
We discretized the surface using 400 Fourier modes, and 800 points on γ for a total problem size of

$$N = 320\,000.$$

For typical loads, the relative error was less than 10^{-10} and the CPU times were

$$T_{\text{invert}} = 2\text{min} \qquad T_{\text{solve}} = 0.3\text{sec}.$$

Example — scattering on domain with cavities (with S. Hao and P. Young)



Acoustic scattering on the exterior domain.

Each bowl is about 5λ.

A hybrid direct/iterative solver is used (a highly accurate scattering matrix is computed for each body).

On an office desktop, we achieved an accuracy of 10^{-5}, in about 6h (all the time is spent in applying the inter-body interactions via the Fast Multipole Method). Accuracy 10^{-7} took 27h.

Example — BIEs on rotationally symmetric bodies (2012, with S. Hao and P. Young)

N	N_{body}	T_{fmm}	I_{GMRES} (precond /no precond)	T_{total} (precond /no precond)	E_{∞}^{rel}
10000	50× 25	1.23e+00	21 /358	2.70e+01 /4.49e+02	4.414e-04
20000	100×25	3.90e+00	21 /331	8.57e+01 /1.25e+03	4.917e-04
40000	200×25	6.81e+00	21 /197	1.62e+02 /1.18e+03	4.885e-04
80000	400×25	1.36e+01	21 / 78	3.51e+02 /1.06e+03	4.943e-04
20400	50×51	4.08e+00	21 /473	8.67e+01 /1.99e+03	1.033e-04
40800	100×51	7.20e+00	21 /442	1.56e+02 /3.17e+03	3.212e-05
81600	200×51	1.35e+01	21 /198	2.99e+02 /2.59e+03	9.460e-06
163200	400×51	2.50e+01	21 /102	5.85e+02 /2.62e+03	1.011e-05
40400	50×101	7.21e+00	21 /483	1.53e+02 /3.52e+03	1.100e-04
80800	100×101	1.34e+01	22 /452	2.99e+02 /6.31e+03	3.972e-05
161600	200×101	2.55e+01	22 /199	5.80e+02 /5.12e+03	2.330e-06
323200	400×101	5.36e+01	22 /112	1.25e+03 /5.84e+03	3.035e-06

Exterior Laplace problem solved on the multibody bowl domain with and without preconditioner.

Example — BIEs on rotationally symmetric bodies (2012, with S. Hao and P. Young)

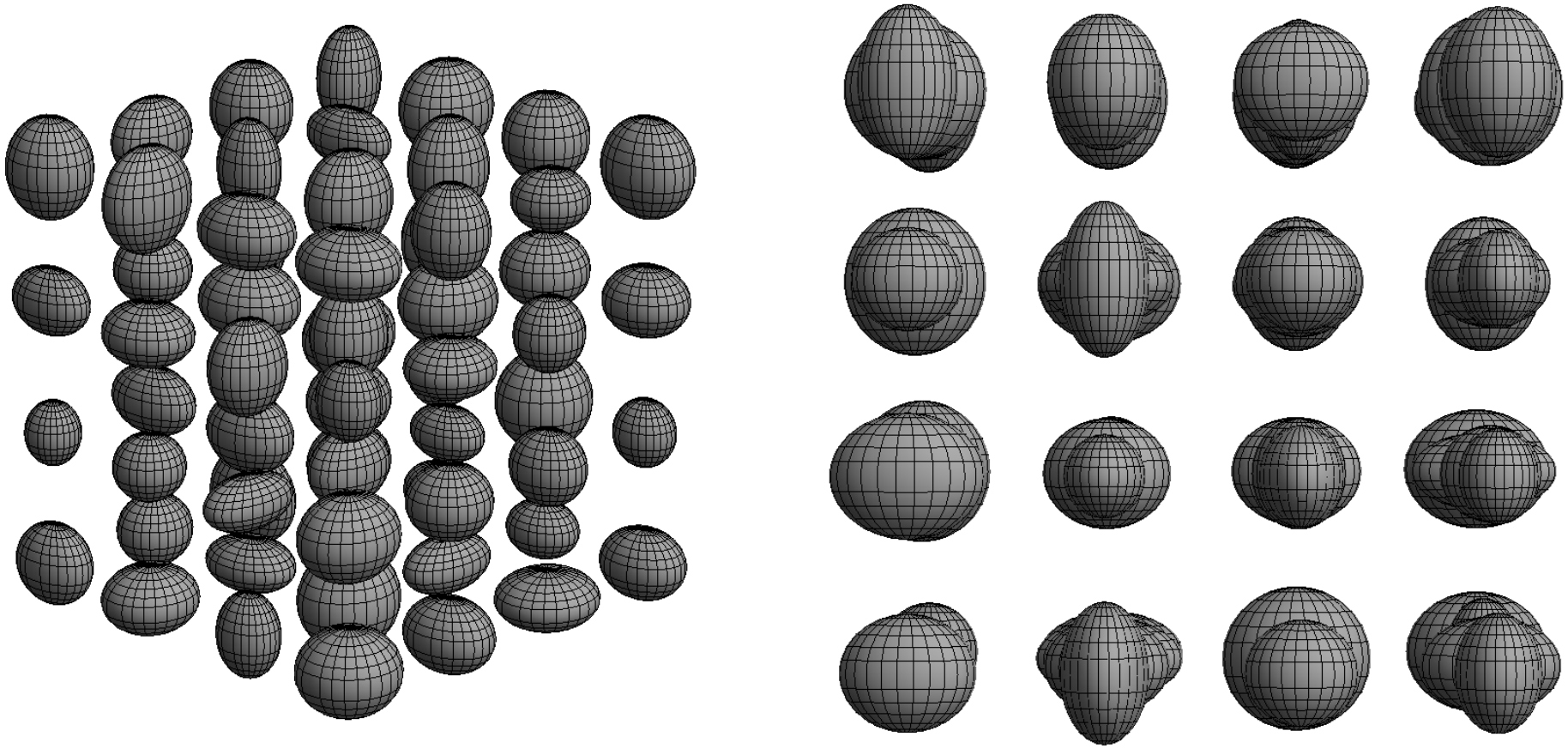
N	N_{body}	$T_{\text{precompute}}$	I_{GMRES}	T_{solve}	E_{∞}^{rel}
80800	100×101	6.54e-01	62	5.17e+03	1.555e-03
161600	200×101	1.82e+00	63	9.88e+03	1.518e-04
323200	400×101	6.46e+00	64	2.19e+04	3.813e-04
160800	100×201	1.09e+00	63	9.95e+03	1.861e-03
321600	200×201	3.00e+00	64	2.19e+04	2.235e-05
643200	400×201	1.09e+01	64	4.11e+04	8.145e-06
641600	200×401	5.02e+00	64	4.07e+04	2.485e-05
1283200	400×401	1.98e+01	65	9.75e+04	6.884e-07

Exterior Helmholtz problem solved on multibody bowl domain.
Each bowl is 5 wavelength in diameter.

We do not give timings for standard iterative methods since in this example, they typically did not converge at all (even though the BIE is a 2nd kind Fredholm equation).

Numerical example — BIE on surfaces in 3D (2013, with J. Bremer and A. Gillman)

Consider sound-soft scattering from a multi-body scatterer of size 4 wave-lengths:



The ellipsoids are not rotationally symmetric.

The global scattering matrix is computed using the HBS direct solver described.

Numerical example — BIE on surfaces in 3D (2013, with J. Bremer and A. Gillman)

The local truncation error is set to 10^{-3}.

Grid dimensions	N	$N_{\text{out}} \times N_{\text{in}}$	T	E	Ratio	Predicted
$2 \times 2 \times 2$	$12\,288$	410×401	$1.02 \times 10^{+1}$	3.37×10^{-04}	-	-
$3 \times 3 \times 3$	$41\,472$	464×475	$3.43 \times 10^{+1}$	4.81×10^{-04}	3.4	6.2
$4 \times 4 \times 4$	$98\,304$	483×532	$7.92 \times 10^{+1}$	1.57×10^{-04}	2.3	3.7
$6 \times 6 \times 6$	$331\,776$	504×707	$2.96 \times 10^{+2}$	7.03×10^{-04}	3.7	6.2
$8 \times 8 \times 8$	$786\,432$	513×1014	$6.70 \times 10^{+2}$	4.70×10^{-04}	2.3	3.7
$10 \times 10 \times 10$	$1\,536\,000$	518×1502	$2.46 \times 10^{+3}$	3.53×10^{-04}	3.7	2.7

Numerical example — BIE on surfaces in 3D (2013, with J. Bremer and A. Gillman)

The local truncation error is set to 10^{-3}.

Grid dimensions	N	$N_{\text{out}} \times N_{\text{in}}$	T	E	Ratio	Predicted
$2 \times 2 \times 2$	$12\,288$	410×401	$1.02 \times 10^{+1}$	3.37×10^{-04}	-	-
$3 \times 3 \times 3$	$41\,472$	464×475	$3.43 \times 10^{+1}$	4.81×10^{-04}	3.4	6.2
$4 \times 4 \times 4$	$98\,304$	483×532	$7.92 \times 10^{+1}$	1.57×10^{-04}	2.3	3.7
$6 \times 6 \times 6$	$331\,776$	504×707	$2.96 \times 10^{+2}$	7.03×10^{-04}	3.7	6.2
$8 \times 8 \times 8$	$786\,432$	513×1014	$6.70 \times 10^{+2}$	4.70×10^{-04}	2.3	3.7
$10 \times 10 \times 10$	$1\,536\,000$	518×1502	$2.46 \times 10^{+3}$	3.53×10^{-04}	3.7	2.7

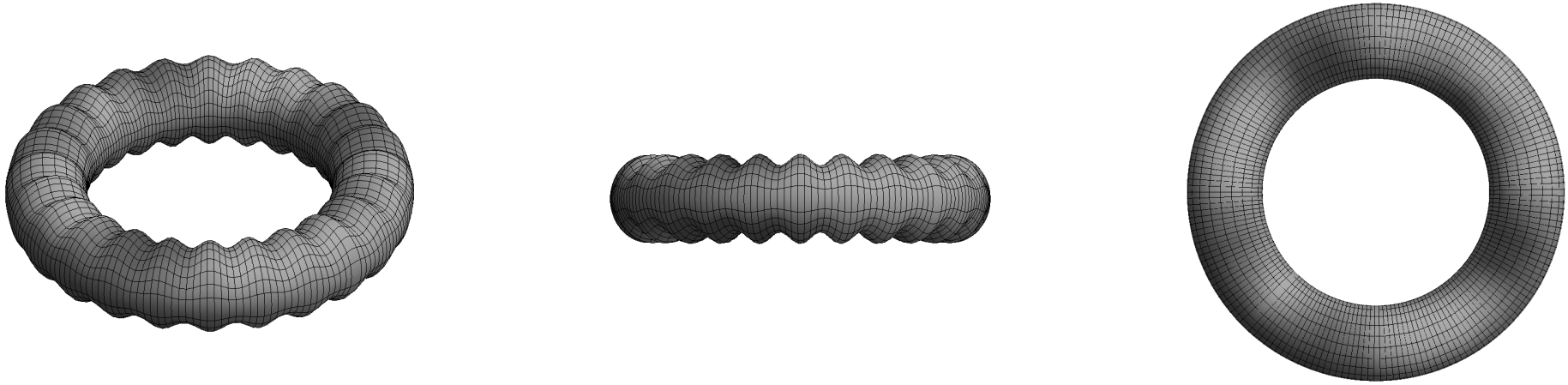
Increasing the accuracy is possible, but comes at a cost.

Now the local truncation error is set to 10^{-6}.

Grid dimensions	N	$N_{\text{out}} \times N_{\text{in}}$	T	E	Ratio	Predicted
$2 \times 2 \times 2$	$49\,152$	601×584	$1.61 \times 10^{+2}$	1.22×10^{-07}	-	-
$3 \times 3 \times 3$	$165\,888$	676×677	$6.87 \times 10^{+2}$	4.92×10^{-07}	4.3	6.2
$4 \times 4 \times 4$	$393\,216$	703×747	$1.68 \times 10^{+3}$	5.31×10^{-07}	2.4	3.6
$6 \times 6 \times 6$	$1\,327\,104$	728×925	$6.66 \times 10^{+3}$	4.60×10^{-06}	4.0	6.2
$8 \times 8 \times 8$	$3\,145\,728$	742×1237	$1.59 \times 10^{+4}$	2.30×10^{-07}	2.4	3.6

Numerical example — BIE on surfaces in 3D (2013, with J. Bremer and A. Gillman)

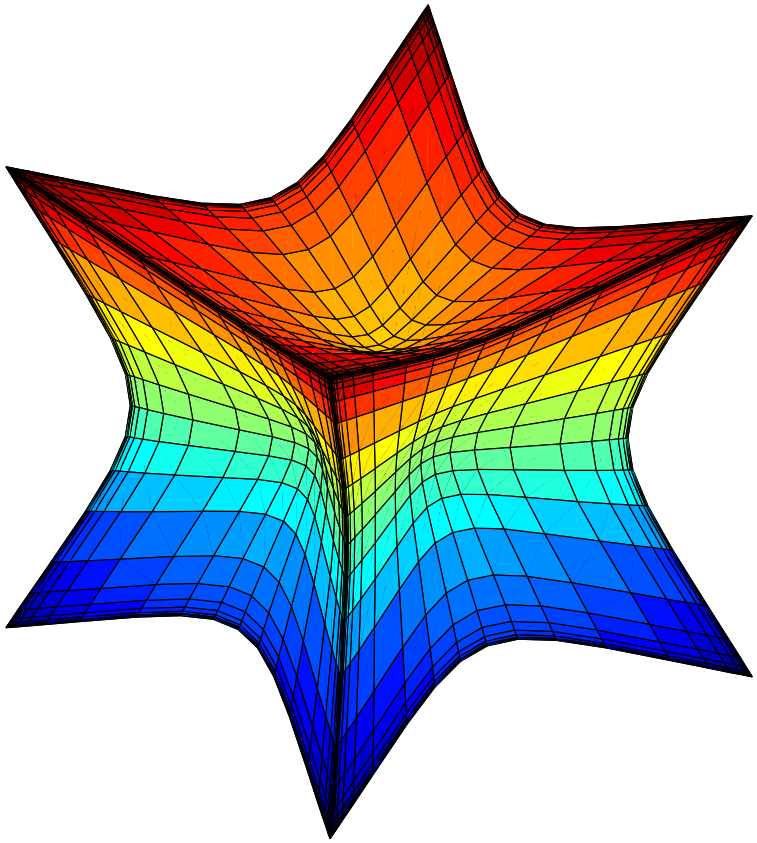
Scattering from a deformed torus of size $4\lambda \times 4\lambda \times 1.5\lambda$:



$4th$ order discretization quadrature

N_{tris}	N	T	E	$N_{\text{out}} \times N_{\text{in}}$	Ratio
128	2304	$9.12 \times 10^{+00}$	$4.78 \times 10^{+02}$	469×589	-
512	9216	$7.09 \times 10^{+01}$	2.59×10^{-02}	477×607	7.8
2048	36864	$5.29 \times 10^{+02}$	1.68×10^{-02}	475×607	7.5
8192	147456	$3.96 \times 10^{+03}$	2.50×10^{-05}	476×611	7.5
32768	589824	$2.06 \times 10^{+04}$	5.07×10^{-05}	477×614	5.2

Numerical example — BIE on "edgy" surface (2013, with J. Bremer and A. Gillman)



N_{tris}	N	E	T	$N_{\text{out}} \times N_{\text{in}}$
192	$21\,504$	2.60×10^{-08}	$6.11 \times 10^{+02}$	617×712
432	$48\,384$	2.13×10^{-09}	$1.65 \times 10^{+03}$	620×694
768	$86\,016$	3.13×10^{-10}	$3.58 \times 10^{+03}$	612×685

Results from a Helmholtz problem (acoustic scattering) on the domain exterior to the "edgy" cube.
The domain is about 3.5 wave-lengths in diameter.
Mesh refinement near edges & corners. The direct solver eliminates "extra" DOFs.

Numerical example — Volume int. eq. in 2D (2013, with E. Corona and D. Zorin)

Consider a volume integral equation in the plane:

$$q(x) + \int_\Omega b(x) \log|x-y|\, q(y)\, dy = f(x), \qquad x \in \Omega,$$

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

$$b(x) = 1 + 0.5e^{-(x_1-0.3)^2-(x_2-0.6)^2}.$$

The domain is discretized on a uniform grid, with simplistic quadrature.

By exploiting internal structure (HBS structure) in the scattering matrices, we have built a direct solver with optimal $O(N)$ complexity for every step.

References:

- E. Corona, P.G. Martinsson, D. Zorin *"An $O(N)$ Direct Solver for Integral Equations in the Plane"* In review. (arXiv.org report 1303.5466).
- K. Ho and L. Ying, *"Hierarchical interpolative factorization for elliptic operators: differential equations."* In review. (arXiv.org report 1307.2895).
- K. Ho and L. Ying, *"Hierarchical interpolative factorization for elliptic operators: integral equations."* In review. (arXiv.org report 1307.2666).

Numerical example — Volume int. eq. in 2D (2013, with E. Corona and D. Zorin)

N	T_{build} (non-accelerated) $O(N^{1.5})$	T_{build} (accelerated) $O(N)$	Memory (non-acccelerated) $O(N \log N)$	Memory (accelerated) $O(N)$
784	0.11 s	0.17 s	4.68 MB	4.48 MB
3,136	0.67 s	1.70 s	29.09 MB	25.24 MB
12,544	4.50 s	8.32 s	159.59 MB	123.07 MB
50,176	31.45 s	40.43 s	819.58 MB	538.51 MB
200,704	3.79 m	3.23 m	3.72 GB	2.23 GB
802,816	28.35 m	13.66 m	17.27 GB	9.23 GB
3,211,264	3.58 hr	54.795 m	70.99 GB	34.09 GB

Execution times in Matlab, on an Intel Xeon X5650 (6 core) 2.67 GHz.

Numerical example — Volume int. eq. in 2D (2013, with E. Corona and D. Zorin)

N	T_{solve} (non-accelerated)	T_{solve} (accelerated)	Error
784	0.0014 sec	0.0018 sec	1.6e-14
3,136	0.0064 sec	0.0090 sec	1.8e-14
12,544	0.0292 sec	0.0362 sec	8.6e-11
50,176	0.1320 sec	0.1546 sec	1.6e-10
200,704	0.5993 sec	0.6772 sec	2.3e-10
802,816	2.6611 sec	2.8193 sec	4.0e-10
3,211,264	11.816 sec	11.737 sec	5.1e-9

Execution times in Matlab, on an Intel Xeon X5650 (6 core) 2.67 GHz.

For a computed approximate inverse $\mathbf{B} \approx \mathbf{A}^{-1}$, the error reported is

$$\text{Error} = \max_i \frac{||\mathbf{v}^{(i)} - \mathbf{AB}\mathbf{v}^{(i)}||}{||\mathbf{v}^{(i)}||}$$

where $\{\mathbf{v}^{(i)}\}_i$ is a collection of random vectors. (How many?)

Themes:

- *Multi-resolution representations of operators.*
  Nested-dissection-type hierarchical domain decomposition.
  Excellent localization $\rightarrow$ less communication, easier to parallelize.

- *Low rank approximations to compact operators.*
  Many operators in potential theory have exponentially decaying singular values.
  Typically, this applies only to off-diagonal blocks.
  Data-sparse matrix formats, $\mathcal{H}$-matrix algebra, HSS-matrices, etc.

- *Randomized compression algorithms.*
  Constructing a low-rank approximations can be done using *randomized methods.*
  Highly efficient. Low communication costs.

- *High-order discretizations.*
  Many of the physical problems modeled are inherently well-conditioned, and in this case, well-conditioned mathematical formulations and numerical methods should be used.
  Integral equation formulations are often highly preferable to PDE formulations.
  Relative accuracy of 10^{-10} or better is often perfectly achievable.
  When the physics is ill-conditioned (e.g. scattering), high accuracy is *crucial.*

CBMS conference

(i.e. summer school)

"Fast direct solvers for elliptic PDEs"

Dartmouth College, June 23 - 27, 2014

Lecturer:	Per-Gunnar Martinsson
Speakers:	Alex Barnett
	Adrianna Gillman
	Leslie Greengard
	Vladimir Rokhlin
Travel support available:	Google "CBMS Fast Direct Solvers"

Assertions:

- Fast direct solvers excel for many integral equations; they should become default. (Integral eq$^{\text{ns}}$ on $\mathbb{R}$ and $\mathbb{R}^2$, BIEs on curves and surfaces.)
- $O(N)$ direct solvers for variable coefficient problems in $\mathbb{R}^2$ exist and work very well. In $\mathbb{R}^3$, they can be game-changing in specialized environments.
- Direct solvers are perfect for managing local refinement near *corners and edges.* They "squeeze out" the superfluous local degrees of freedom.
- Multidomain spectral collocation methods perform extremely well with direct solvers.

Predictions with (some...) empirical support:

- For BIEs associated with non-oscillatory problems on surfaces in $\mathbb{R}^3$, the complexity will be reduced from $O(N(\log N)^p)$ to $O(N)$, with a modest scaling constant.
- *Randomized methods* will prove enormously helpful.
- Direct solvers will provide a fantastic tool for *numerical homogenization*, and for *scattering problems* (despite lack of $O(N)$ methods).

Open questions:

- Can direct solvers for variable coefficient problems in 3D compete with multigrid and related methods? (My guess: *No* for "over-resolved" Laplace, and a single solve. *Yes* for certain scattering problems, and for multiple RHS problems.)
- Are $O(N)$ direct solvers for highly oscillatory problems possible?

References: P.G. Martinsson, “A direct solver for variable coefficient elliptic PDEs discretized via a composite spectral collocation method.” *J. of Computational Physics*, **242**(1), pp. 460–479, 2013.

A. Gillman and P.G. Martinsson “A direct solver with O(N) complexity for variable coefficient elliptic PDEs discretized via a high-order composite spectral collocation method” arxiv.org #1307.2665.

A. Gillman, A. Barnett, and P.G. Martinsson “A spectrally accurate direct solution technique for frequency-domain scattering problems with variable media” arxiv.org #1308.5998.