Introduction to research seminar:

Fast methods for solving elliptic PDEs

P.G. Martinsson
Department of Applied Math
University of Colorado at Boulder

Consider for a moment one of the most classical elliptic PDE, the Poisson equation with Dirichlet boundary data

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

Question: How would you solve this numerically?

Standard answers:

- Finite differences ...
- Finite elements ...

We will start this talk by revisiting some classical techniques from mathematical physics. These by themselves are only of limited value, but they can with minor twists be turned in to very powerful tools for numerical work.

Example 1: The Poisson equation in free space

$$-\Delta u(\boldsymbol{x}) = f(\boldsymbol{x}) \qquad \boldsymbol{x} \in \mathbb{R}^2.$$

The classical analytic solution is

$$u(\boldsymbol{x}) = \int_{\mathbb{R}^2} \phi(\boldsymbol{x}-\boldsymbol{y})\, f(\boldsymbol{x})\, d\boldsymbol{x}, \qquad \boldsymbol{x} \in \mathbb{R}^2, \tag{1}$$

where ϕ is the *fundamental solution*

$$\phi(\boldsymbol{x}-\boldsymbol{y}) = -\frac{1}{2\pi} \log|\boldsymbol{x}-\boldsymbol{y}|.$$

How would you evaluate (1) numerically?
You need a *quadrature rule* and a *fast summation scheme* (e.g. the Fast Multipole Method, or the FFT).

Example 2: The Laplace equation with Dirichlet data on a circle. Set

$$\Omega = \{\boldsymbol{x} \in \mathbb{R}^2 : \ |\boldsymbol{x}| \leq R\},$$

and consider the equation

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

It is convenient to work with polar coordinates (r, t) so that $x_1 = r\cos t$ and $x_2 = r\sin t$. Then g is a periodic function of t. We write its Fourier expansion as

$$g(t) = \sum_{n=-\infty}^{\infty} a_n\, e^{int}$$

where

$$a_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-ins}\, g(s)\, ds.$$

Now consider the function

$$u(r, t) = \sum_{n=-\infty}^{\infty} a_n \left(\frac{r}{R}\right)^{|n|} e^{int}.$$

Then u obviously satisfies the boundary condition, and it is easily verified that $-\Delta u = 0$. Due to uniqueness, we must have found the correct solution.

Recall that the solution to the Dirichlet problem can be written

$$u(r,t) = \sum_{n=-\infty}^{\infty} a_n \left(\frac{r}{R}\right)^{|n|} e^{int}.$$

where

$$a_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-ins}\, g(s)\, ds.$$

By combining the two expressions, we find that

$$u(\boldsymbol{x}) = \int_{-\pi}^{\pi} G(r,t,s)\, g(s)\, ds$$

where G is the *Green's function* of the problem

$$G(r,t,s) = \frac{1}{2\pi} \sum_{n=-\infty}^{\infty} \left(\frac{r}{R}\right)^{|n|} e^{in(t-s)}.$$

More generally, for any domain Ω (that is not too nasty), there exists a Green's function $G = G(\boldsymbol{x}, \boldsymbol{y})$ such that the function

$$u(\boldsymbol{x}) = \int_{\Gamma} G(\boldsymbol{x}, \boldsymbol{y})\, g(\boldsymbol{y})\, ds(\boldsymbol{y})$$

solves the Dirichlet problem

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

Problem: We only know the Green's function G for very simple domains (rectangles, circles, spheres, half-planes, etc).

Question: How do you solve a Dirichlet problem on a general domain?

Solution: Let us look for a solution of the form

$$u(\boldsymbol{x}) = \int_\Gamma \phi(\boldsymbol{x}-\boldsymbol{y})\,\sigma(\boldsymbol{y})\,ds(\boldsymbol{y}). \tag{2}$$

Note that we now use a known kernel function — the free space fundamental solution $\phi(\boldsymbol{x}-\boldsymbol{y}) = -\frac{1}{2\pi}\log|\boldsymbol{x}-\boldsymbol{y}|$. The price we have to pay is that now the function σ is not known. But the boundary condition immediately provides an equation for σ:

$$g(\boldsymbol{x}) = \int_\Gamma \phi(\boldsymbol{x}-\boldsymbol{y})\,\sigma(\boldsymbol{y})\,ds(\boldsymbol{y}), \qquad \boldsymbol{x}\in\Gamma.$$

(Note that a function of the form (2) satisfies $-\Delta u = 0$ for any σ.)

Boundary Integral Equation methods — summary

Recall that the PDE

$$\begin{cases} -\Delta\, u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ \quad u(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma. \end{cases} \tag{3}$$

can be rewritten as the *boundary integral equation* (BIE)

$$\int_\Gamma -\frac{1}{2\pi} \log|\boldsymbol{x}-\boldsymbol{y}|\, \sigma(\boldsymbol{y})\, ds(\boldsymbol{y}) = g(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma. \tag{4}$$

Once (4) has been solved for σ, the solution u to (3) can be recovered via

$$u(x) = \int_\Gamma -\frac{1}{2\pi} \log|\boldsymbol{x}-\boldsymbol{y}|\, \sigma(\boldsymbol{y})\, d\boldsymbol{y}, \qquad \boldsymbol{x} \in \Omega.$$

The equations (3) and (4) are in a strong sense equivalent.

Note: It should perhaps be mentioned that for technical reasons, the BIE (4) is not ideal. A better formulation is

$$\frac{1}{2}\sigma(\boldsymbol{x}) + \int_\Gamma \frac{\boldsymbol{n}(\boldsymbol{y})\cdot(\boldsymbol{x}-\boldsymbol{y})}{2\pi|\boldsymbol{x}-\boldsymbol{y}|^2}\, \sigma(\boldsymbol{y})\, ds(\boldsymbol{y}) = g(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma. \tag{5}$$

Once (5) has been solved for σ, the solution u can be recovered via

$$u(x) = \int_\Gamma \frac{\boldsymbol{n}(\boldsymbol{y})\cdot(\boldsymbol{x}-\boldsymbol{y})}{2\pi|\boldsymbol{x}-\boldsymbol{y}|^2}\, \sigma(\boldsymbol{y})\, d\boldsymbol{y}, \qquad \boldsymbol{x} \in \Omega.$$

More generally, the idea is to convert some elliptic boundary value problem

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

to an "equivalent" boundary integral equation

$$\alpha\sigma(\boldsymbol{x}) + \int_\Gamma K(\boldsymbol{x},\boldsymbol{y})\,\sigma(\boldsymbol{y})\,dS(\boldsymbol{y}) = g(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma.$$

BIE formulations exist for many classical BVPs:

Laplace	$-\Delta u = f,$	
Elasticity	$\frac{1}{2} E_{ijkl} \left(\frac{\partial^2 u_k}{\partial x_l \partial x_j} + \frac{\partial^2 u_l}{\partial x_k \partial x_j} \right) = f_i,$	
Stokes	$\Delta \mathbf{u} = \nabla p, \qquad \nabla \cdot \mathbf{u} = 0,$	
Heat equation	$-\Delta u = -u_t$	(On the surface of $\Omega \times [0,\, T]$.)
Helmholtz	$(-\Delta - k^2)u = f,$	
Schrödinger	$(-\Delta + V)\,\Psi = i\,\Psi_t$	(In the frequency domain.)
Maxwell	$\begin{cases} \nabla \cdot \mathbf{E} = \rho & \nabla \times \mathbf{E} = -\dfrac{\partial \mathbf{B}}{\partial t} \\ \nabla \cdot \mathbf{B} = 0 & \nabla \times \mathbf{B} = \mathbf{J} + \dfrac{\partial \mathbf{E}}{\partial t} \end{cases}$	(In the frequency domain.)

More generally, the idea is to convert some elliptic boundary value problem

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

to an "equivalent" boundary integral equation

$$\alpha\sigma(\boldsymbol{x}) + \int_\Gamma K(\boldsymbol{x},\boldsymbol{y})\,\sigma(\boldsymbol{y})\,dS(\boldsymbol{y}) = g(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma.$$

Some comments:

- The BIE formulation involves a domain of lower dimensionality.
- The BIE formulation is "nicer" mathematically. (For instance, it involves bounded or even compact operators, it is often a "second kind Fredholm equation", etc.)
- Unbounded domains can be handled easily. Useful in, e.g. external scattering problems.
- Body loads can be handled but require extra work.
- The BIE formulation has a drawback in that it upon discretization leads to *dense* linear systems. This can be overcome by using accelerated methods for linear algebra. The Fast Multipole Method allows you to solve a dense $N \times N$ linear system in $O(N)$ time!
- The BIE formulation is a less versatile method — difficulties arise for multiphysics, non-linear equations, equations with non-constant coefficients, etc.

Open research questions pertaining to BIE methods:

- Finding BIE formulations for specific situations.
  Even better: Find a systematic way to *derive* BIE formulations.

- Fast methods for linear algebra (solve $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ in $O(N)$ time for $\mathbf{A}$ dense $N \times N$ matrix).
  Existing methods (fast multipole methods, etc) can be accelerated.
  Parallel implementations.

- Techniques for discretizing equations — tricky since the kernel is singular.

- Techniques for handling singularities near corners and edges.

Transform methods:

Often a linear PDE looks much nicer in "Fourier space". For instance

$$-\Delta u(\boldsymbol{x}) = f(\boldsymbol{x}) \qquad \boldsymbol{x} \in \mathbb{R}^d \tag{6}$$

Set

$$\hat{u}(\boldsymbol{t}) = [\mathcal{F}u](\boldsymbol{t}) = (2\pi)^{-d/2} \int_{\mathbb{R}^d} e^{-i\boldsymbol{x}\cdot\boldsymbol{t}}\, u(\boldsymbol{x})\, d\boldsymbol{x}, \qquad \boldsymbol{t} \in \mathbb{R}^d.$$

Then the Poisson equation (6) can be written in Fourier space as

$$|\boldsymbol{t}|^2\, \hat{u}(\boldsymbol{t}) = \hat{f}(\boldsymbol{t}), \qquad \boldsymbol{t} \in \mathbb{R}^d. \tag{7}$$

Solve (7) analytically to attain $\hat{u}(\boldsymbol{t}) = \frac{1}{|\boldsymbol{t}|^2}\, \hat{f}(\boldsymbol{t})$, and apply inverse transform

$$u(\boldsymbol{x}) = [\mathcal{F}^{-1}\hat{u}](\boldsymbol{x}) = (2\pi)^{-d/2} \int_{\mathbb{R}^d} e^{i\boldsymbol{x}\cdot\boldsymbol{t}}\, \frac{1}{|\boldsymbol{t}|^2}\, \hat{f}(\boldsymbol{t})\, d\boldsymbol{t}, \qquad \boldsymbol{x} \in \mathbb{R}^d.$$

Similar techniques work for "simple" geometries such as rectangles, circles, spheres, half-planes, etc.

When the FFT can be used to evaluate the Fourier transforms, this is *very* fast.

Open research questions pertaining to transform methods:

- Variations of the FFT.
  - Non-uniform points.
  - Expansion in eigenfunctions for elliptic PDEs (e.g. spherical harmonics).
  - Expansion in prolate wave functions (very useful in data processing).
- Quadratures for discretizing various integrals.
- Perturbations: Slightly deformed geometries, "wrong" boundary conditions, slightly different differential operator, etc.

The big picture:

BIEs and transform based PDE solvers share certain characteristics:

- They leverage techniques from classical mathematical physics.
- They are often capable of computing solutions to full double precision accuracy.
  (I.e. the approximate solution you compute has 15 correct digits.)
- Can be *extremely fast* when they work.
- Not as versatile as finite element methods.
  You need to custom build them for each application.

Together, we call such methods *analysis based fast methods.*

They are primarily applicable to the so called "elliptic" PDEs:

- 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).

Fast direct solvers for elliptic PDEs

Now consider the task of 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. 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, ...

↓

$N \times N$ 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, ...

↓

$N \times N$ 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

$$\mathbf{A}\,\boldsymbol{u} = \boldsymbol{b}, \tag{9}$$

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

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

Then an approximate solution to (9) is obtained by simply evaluating

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

The matrix $\mathbf{T}$ is typically constructed in a *compressed format* that allows the matrix-vector product $\mathbf{T}\,\boldsymbol{b}$ 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

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

Iterative methods:

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

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

Many iterative methods access $\mathbf{A}$ 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 $\mathbf{A}$.)

(Exact except for rounding errors.)

Advantages of direct solvers over iterative solvers:

1. Applications that require a very large number of solves:
   - 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. Problems that are relatively ill-conditioned:
   - 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.*

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.

How do you construct direct solvers with less than $O(N^3)$ complexity?

For *sparse* matrices, algorithms such as "nested dissection" achieve $O(N^{1.5})$ or $O(N^2)$ complexity by reordering the matrix.

More recently, methods that exploit *data-sparsity* have achieved linear or close to linear complexity in a broad variety of environments.

In this talk, the *data-sparse* matrices under consideration have off-diagonal blocks that can to high accuracy (say ten of fifteen digits) be approximated by low-rank matrices.

Numerical examples — BIEs in $\mathbb{R}^2$

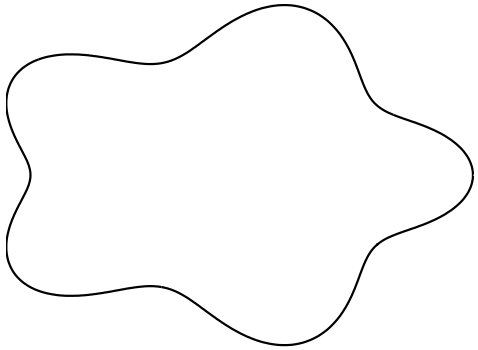
We invert a matrix approximating the operator

$$[A\,u](\boldsymbol{x}) = \frac{1}{2}\,u(\boldsymbol{x}) - \frac{1}{2\,\pi}\int_{\Gamma} D(\boldsymbol{x},\boldsymbol{y})\,u(\boldsymbol{y})\,ds(\boldsymbol{y}), \qquad \boldsymbol{x}\in\Gamma,$$

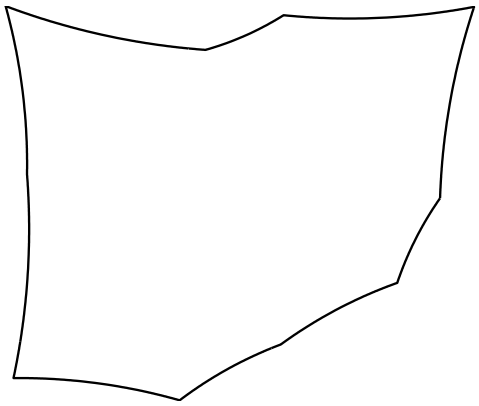
where D is the double layer kernel associated with Laplace's equation,

$$D(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2\pi}\,\frac{\boldsymbol{n}(\boldsymbol{y})\cdot(\boldsymbol{x}-\boldsymbol{y})}{|\boldsymbol{x}-\boldsymbol{y}|^2},$$

and where Γ is either one of the countours:



Smooth star



Star with corners

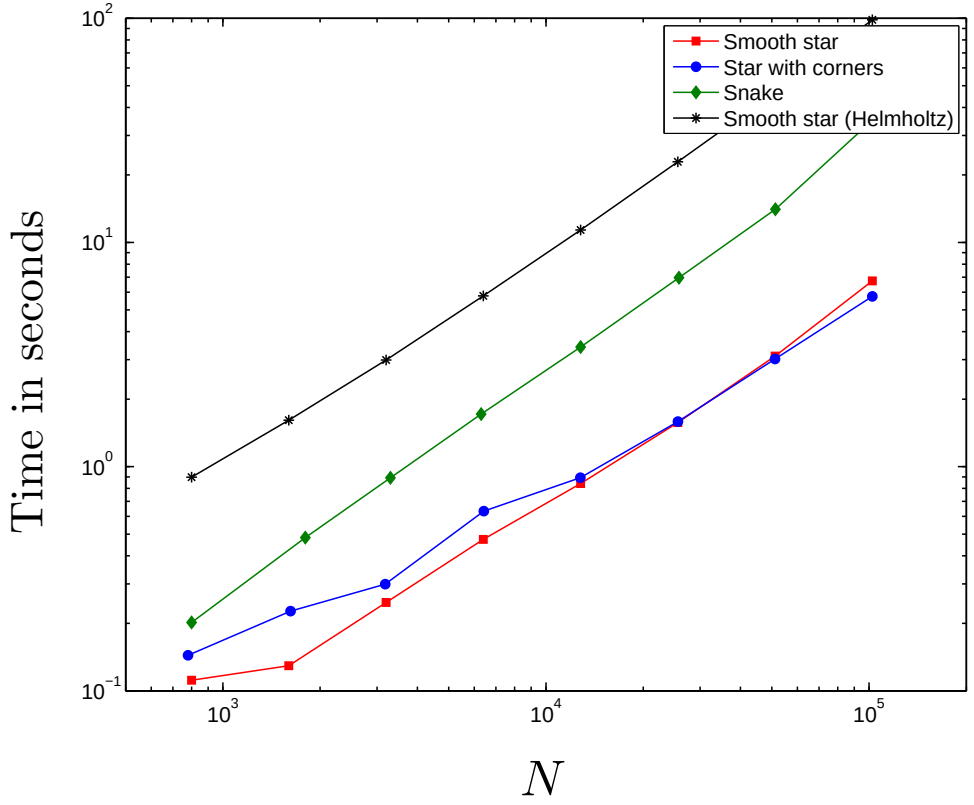
(local refinements at corners)

Snake

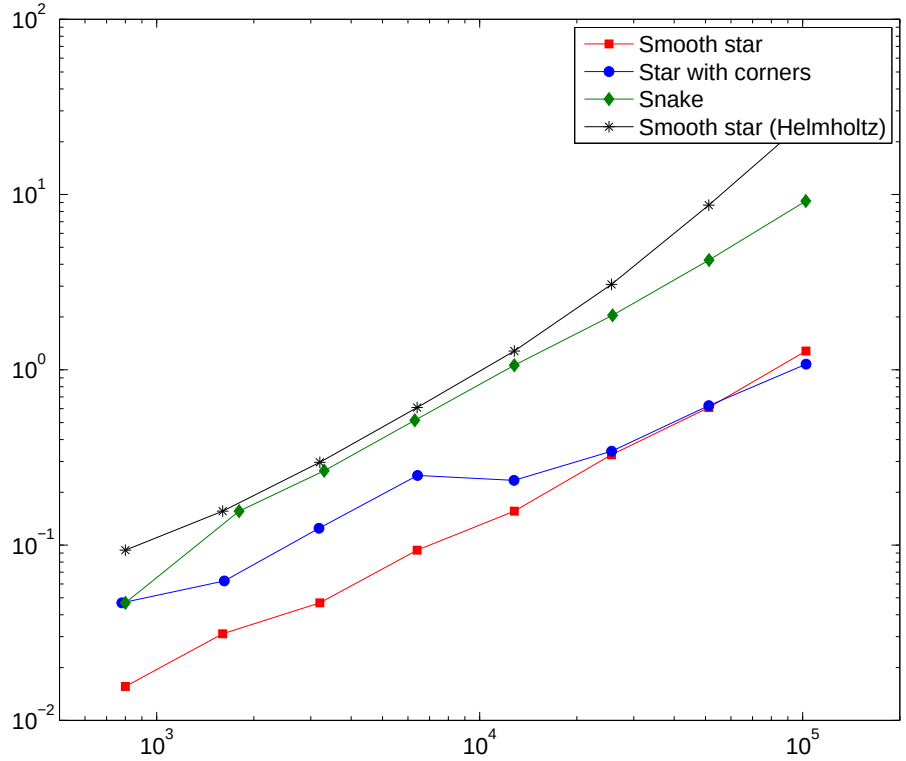
(# oscillations $\sim N$)

Numerical examples — BIEs in $\mathbb{R}^2$

Compression



Inversion



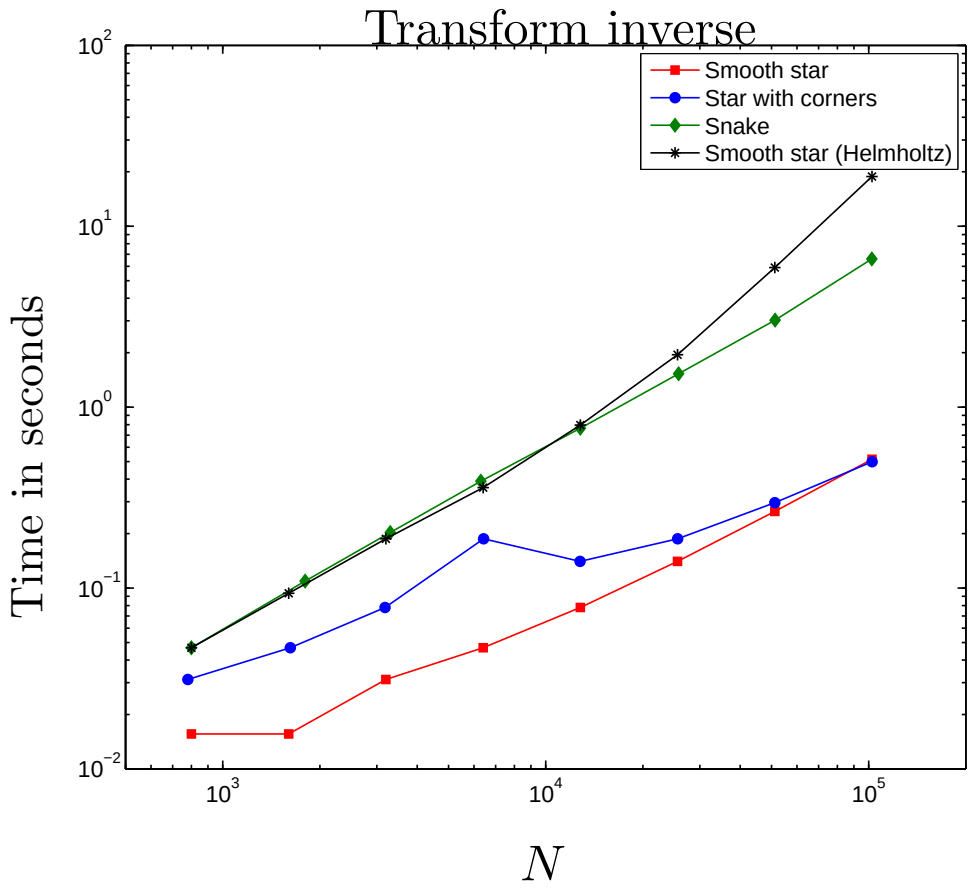
The graphs give the times required for:

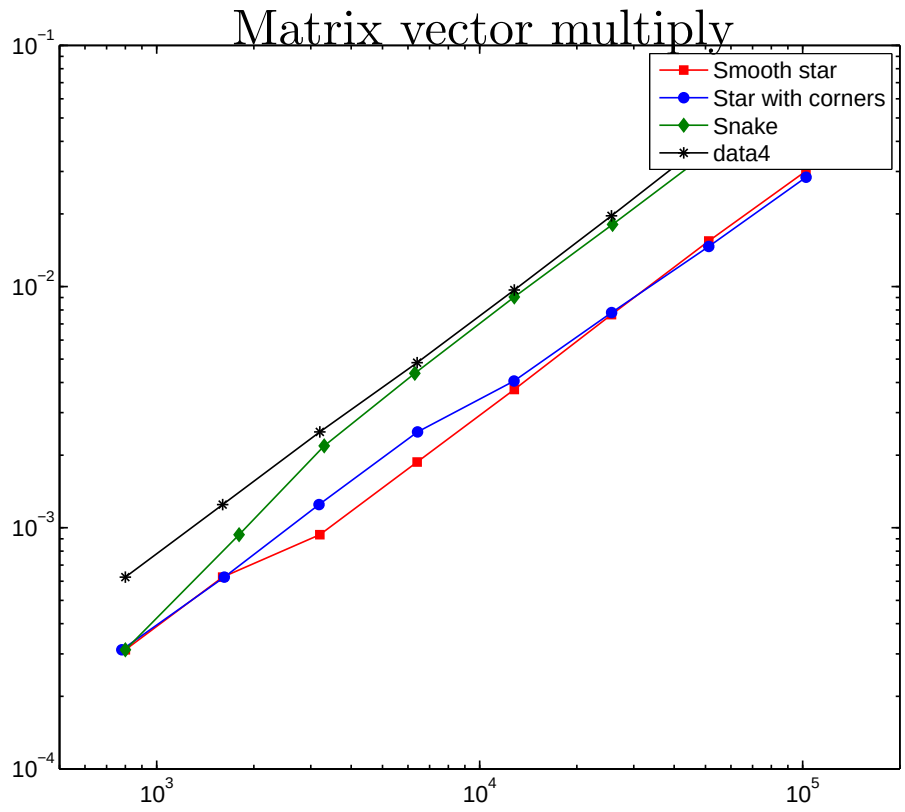
- Computing the HSS representation of the coefficient matrix.
- Inverting the HSS matrix.

Within each graph, the four lines correspond to the four examples considered:

□ Smooth star ○ Star with corners ◇ Snake ∗ Smooth star (Helmholtz)

Numerical examples — BIEs in $\mathbb{R}^2$





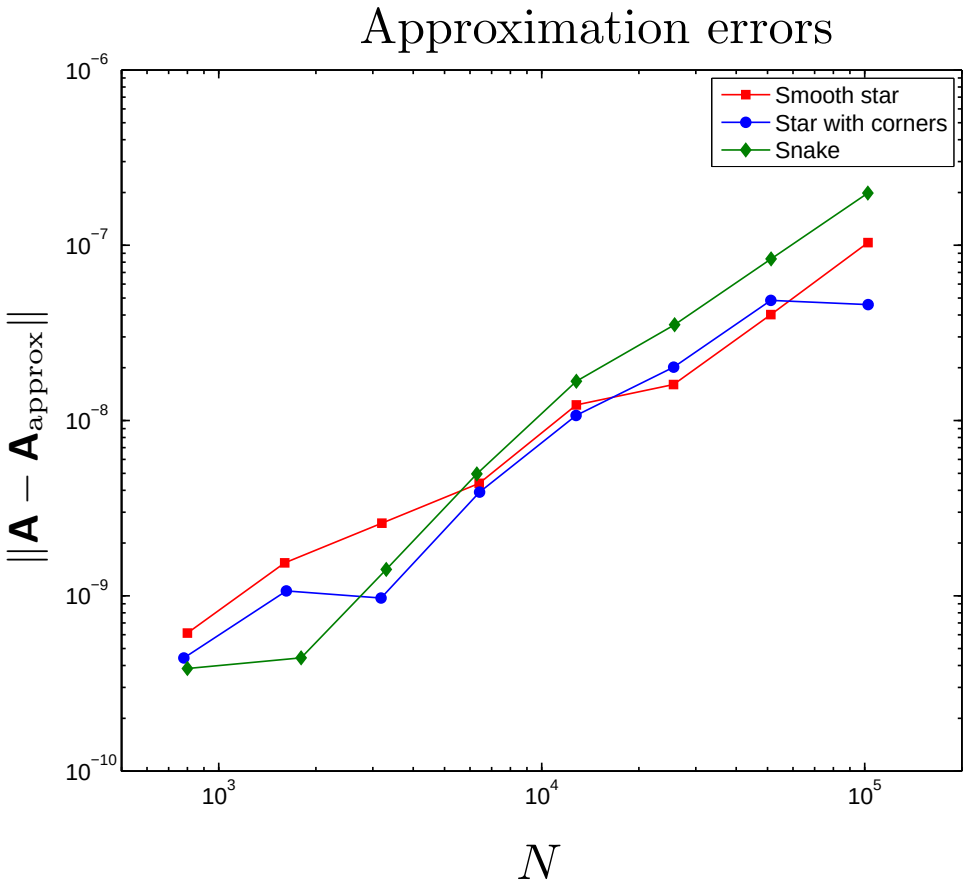
The graphs give the times required for:

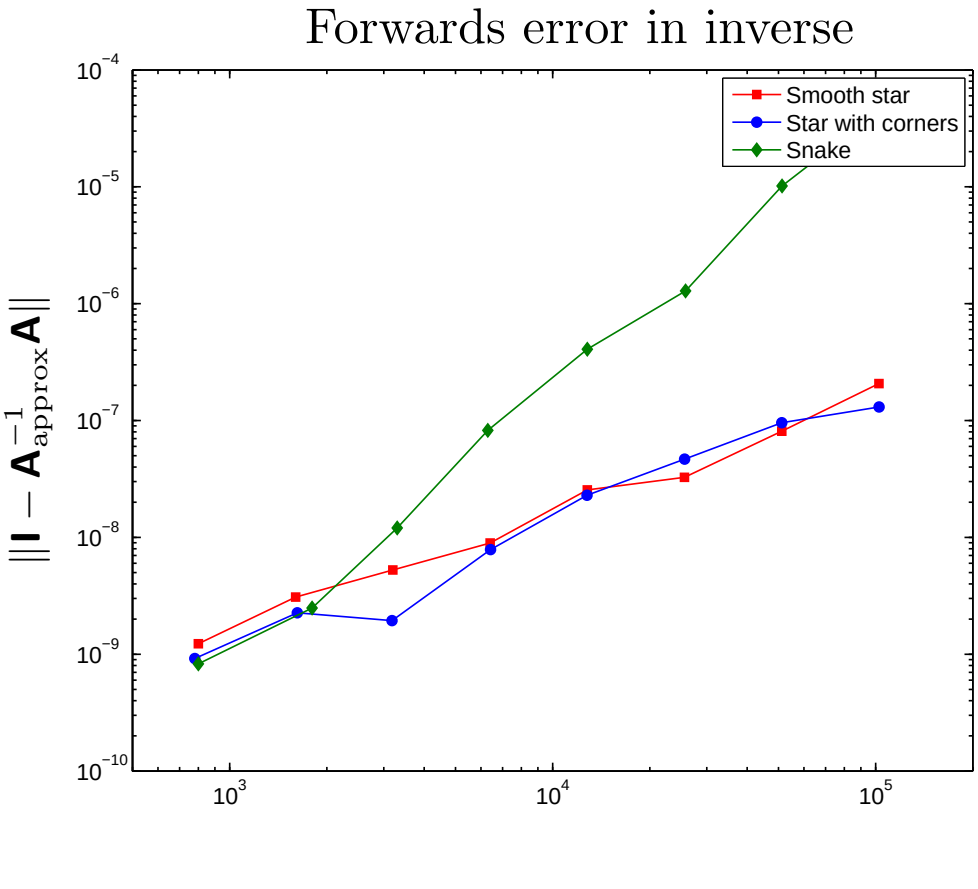
- Transforming the computed inverse to standard HSS format.
- Applying the inverse to a vector (i.e. solving a system).

Within each graph, the four lines correspond to the four examples considered:

□ Smooth star ○ Star with corners ◇ Snake ∗ Smooth star (Helmholtz)

Numerical examples — BIEs in $\mathbb{R}^2$



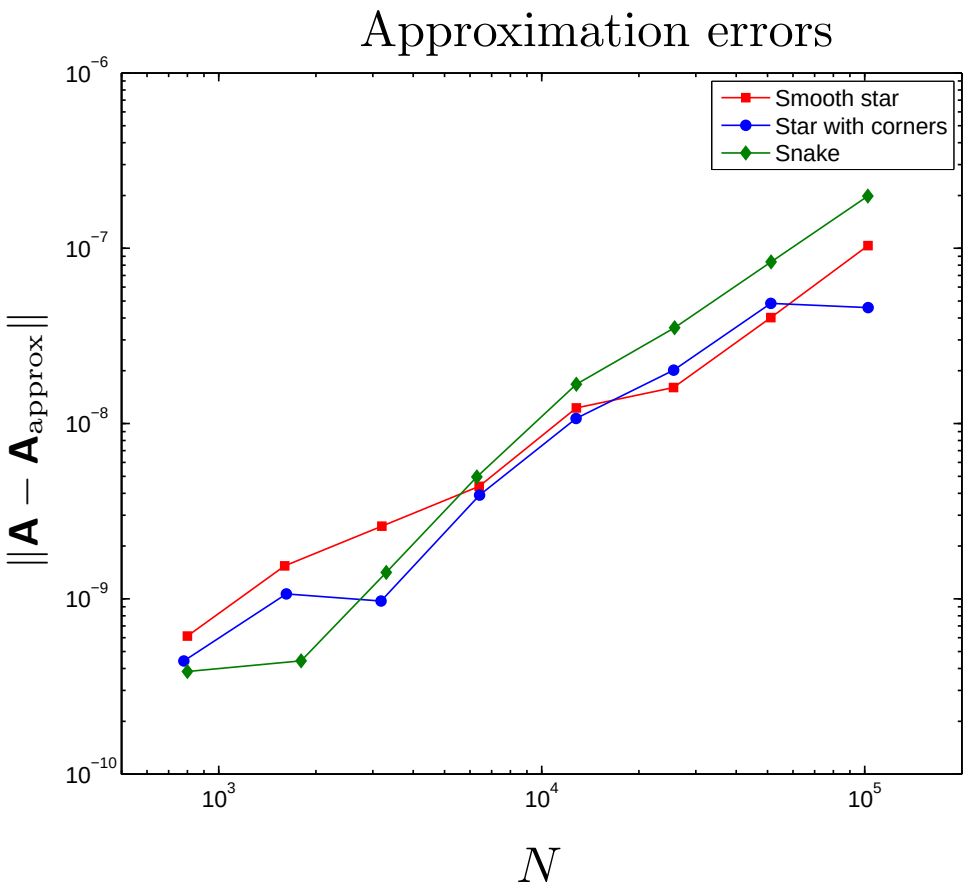


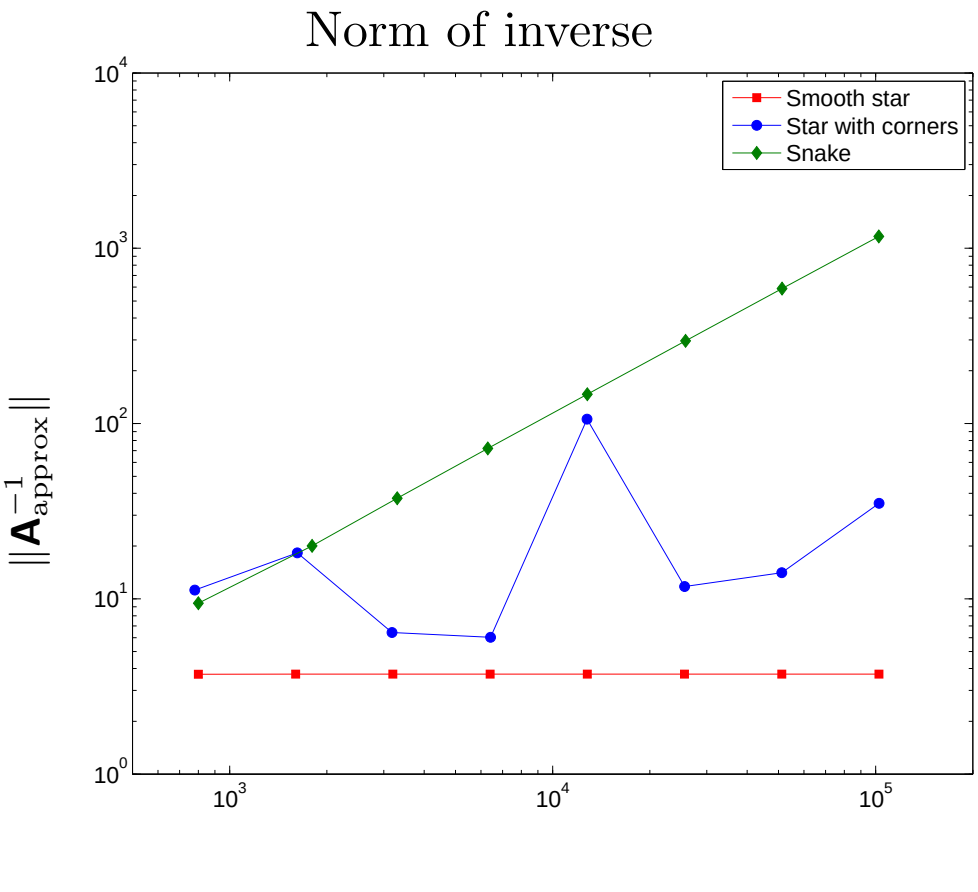
The graphs give the error in the approximation, and the forwards error in the inverse.

Within each graph, the four lines correspond to the four examples considered:

□ Smooth star ○ Star with corners ◇ Snake

Numerical examples — BIEs in $\mathbb{R}^2$





The graphs give the error in the approximation, and the norm of the inverse.

Within each graph, the four lines correspond to the four examples considered:

□ Smooth star ○ Star with corners ◇ Snake

Direct solvers have been implemented for many problems:

- Boundary Integral Equations in $\mathbb{R}^2$ — very effective methods exist.
- Boundary Integral Equations in $\mathbb{R}^3$ — partially done in special environments.
- Finite difference matrices in $\mathbb{R}^2$ — partially done in special environments.
- Finite difference matrices in $\mathbb{R}^3$ — work in progress — hard!

This is work in progress, much work remains.

- Scattering problems at short wavelengths. This is both the hardest environment and the technically most important (direct solvers would be game changing.)
- Implementation on large-scale parallel computers.
- Geometry handling tools — surface representations, etc.
- ...

Lots of opportunities!

Examples of dissertations:

So far, we have talked about two (related) areas of research:

- Analysis based fast methods.
- Fast direct solvers.

What would a possible dissertation project on this stuff look like?
Let us look at three students who have graduated:

- Patrick Young — graduated Spring 2011.
  Now at GeoEye Inc.

- Adrianna Gillman — graduated Summer 2011.
  Now at Dartmouth College.

- Nathan Halko — graduated Spring 2012. (Yesterday!)
  Now at Spot Influence, LLC.

Patrick Young's dissertation:

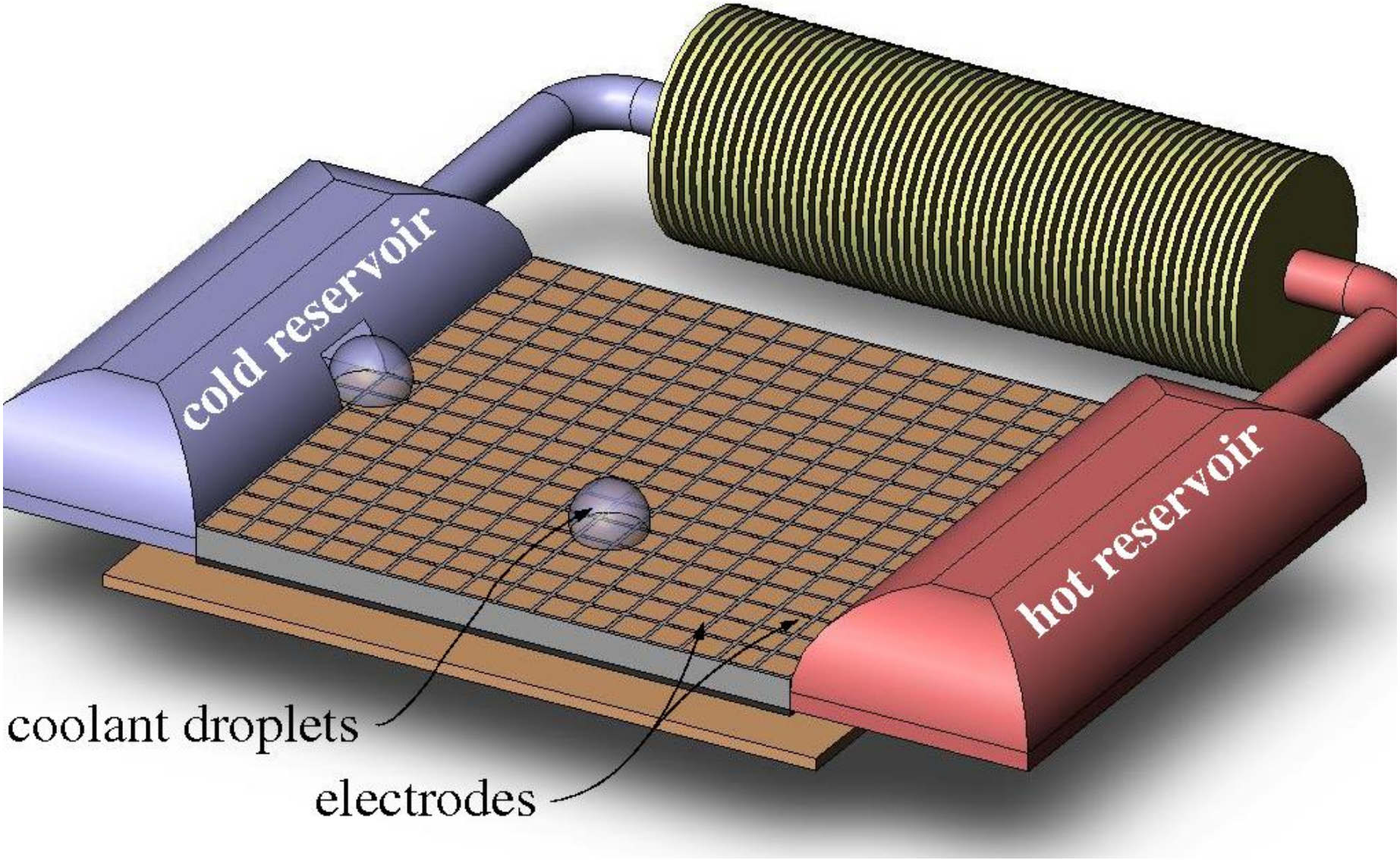
Patrick started by spending two years in the lab of Professor Kamran Mohseni who was then in aerospace at CU. They were working on "electrowetting" and we started working together to develop better methods for handling potential problems with moving boundaries.

Then we developed techniques for modeling problems with micro-structures such as composite materials, lattice problems involving imperfections, etc.

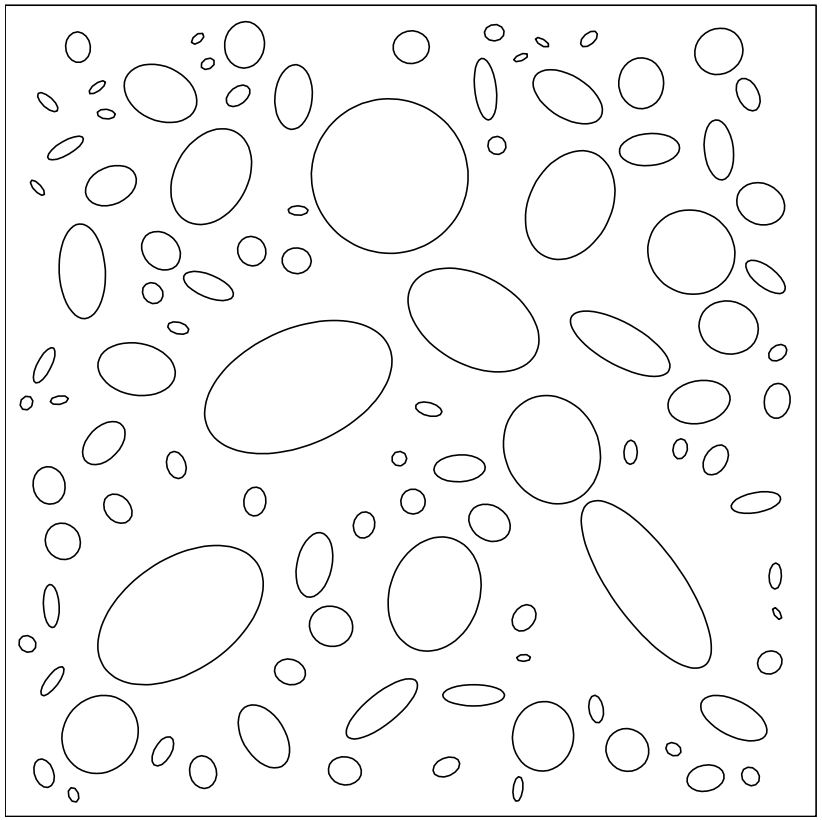
The final two papers concerned a very efficient solver for 3D elliptic problems involving rotationally symmetric surfaces. Several important contributions:

- Development of new methods for evaluating the kernel functions (math).
- Implementation of a new quadrature scheme (numerics).
- Construction of a large code for solving a multibody scattering problems (coding).

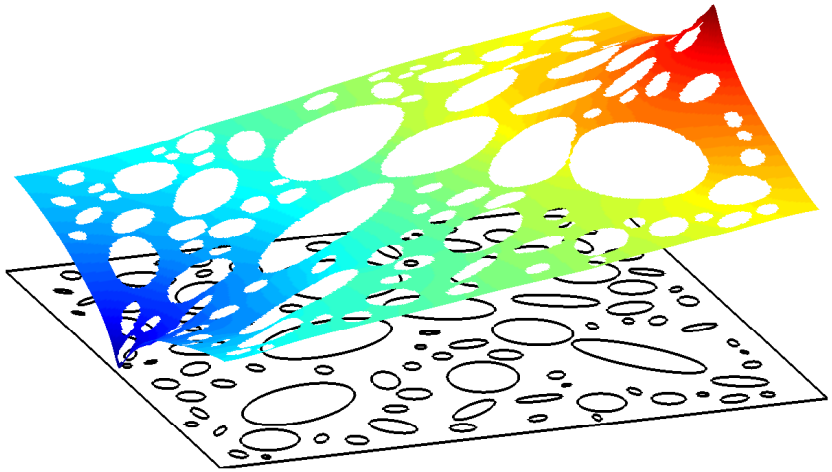
Picture illustrating electrowetting:



Pictures illustrating a conduction problem on a perforated domain:



Geometry



Potential

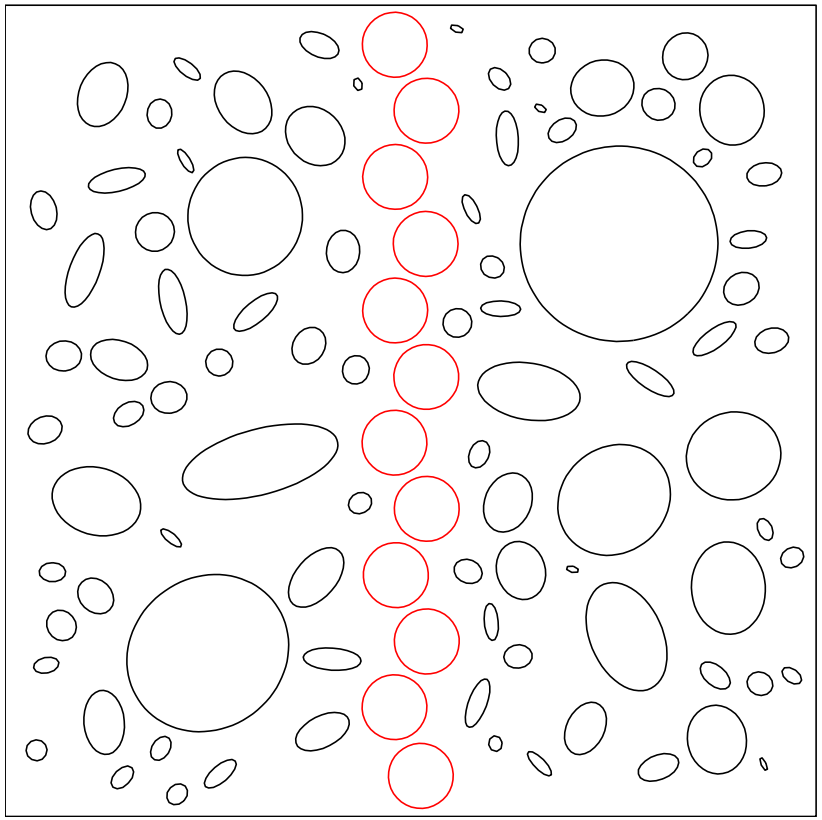
The Neumann-to-Dirichlet operator for the exterior boundary was computed.

The boundary was split into 44 panels, with 26 Gaussian quadrature nodes on each one.

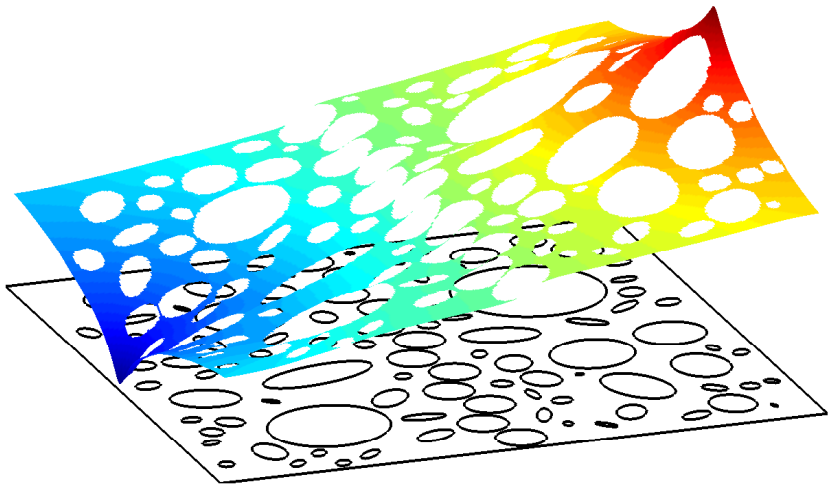
This gives a relative accuracy of 10^{-10} for evaluating fields at points very close to the boundary (up to 0.5% of the side-length removed).

Storing the N2D operator (in a data-sparse format) requires 120 floats per degree of freedom.

Pictures illustrating a conduction problem on a perforated domain — close to "percolation":



Geometry



Potential

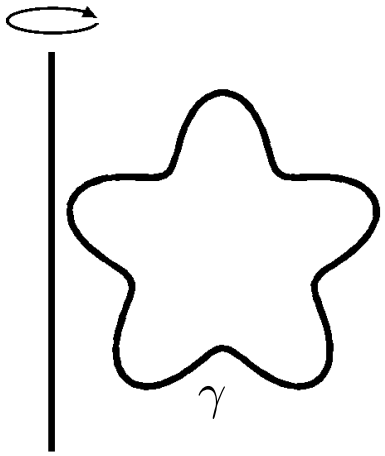
The Neumann-to-Dirichlet operator for the exterior boundary was computed.

The boundary was split into 44 panels, with 26 Gaussian quadrature nodes on each one.

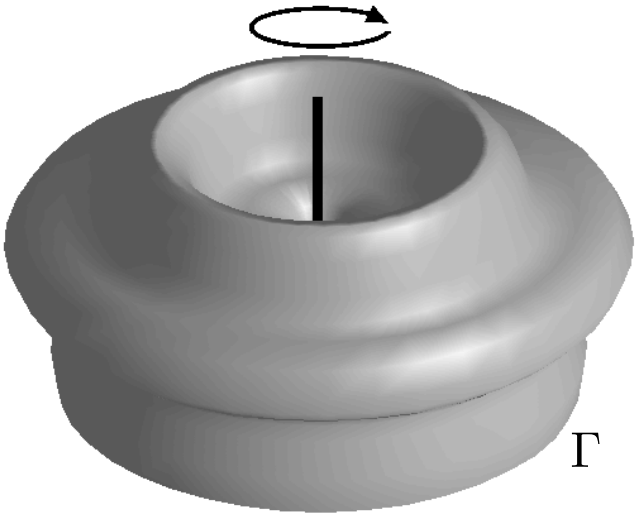
This gives a relative accuracy of 10^{-10} for evaluating fields at points very close to the boundary (up to 0.5% of the side-length removed).

Storing the N2D operator (in a data-sparse format) requires 118 floats per degree of freedom.

Pictures illustrating the set-up for axisymmetric problems:



Generating curve



Surface

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

$$(10) \quad \frac{1}{2}\sigma(\boldsymbol{x}) + \int_\Gamma \frac{\boldsymbol{n}(\boldsymbol{y})\cdot(\boldsymbol{x}-\boldsymbol{y})}{4\pi|\boldsymbol{x}-\boldsymbol{y}|^3}\,\sigma(\boldsymbol{y})\,dA(\boldsymbol{y}) = f(\boldsymbol{x}). \quad \boldsymbol{x}\in\Gamma$$

To (10), 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

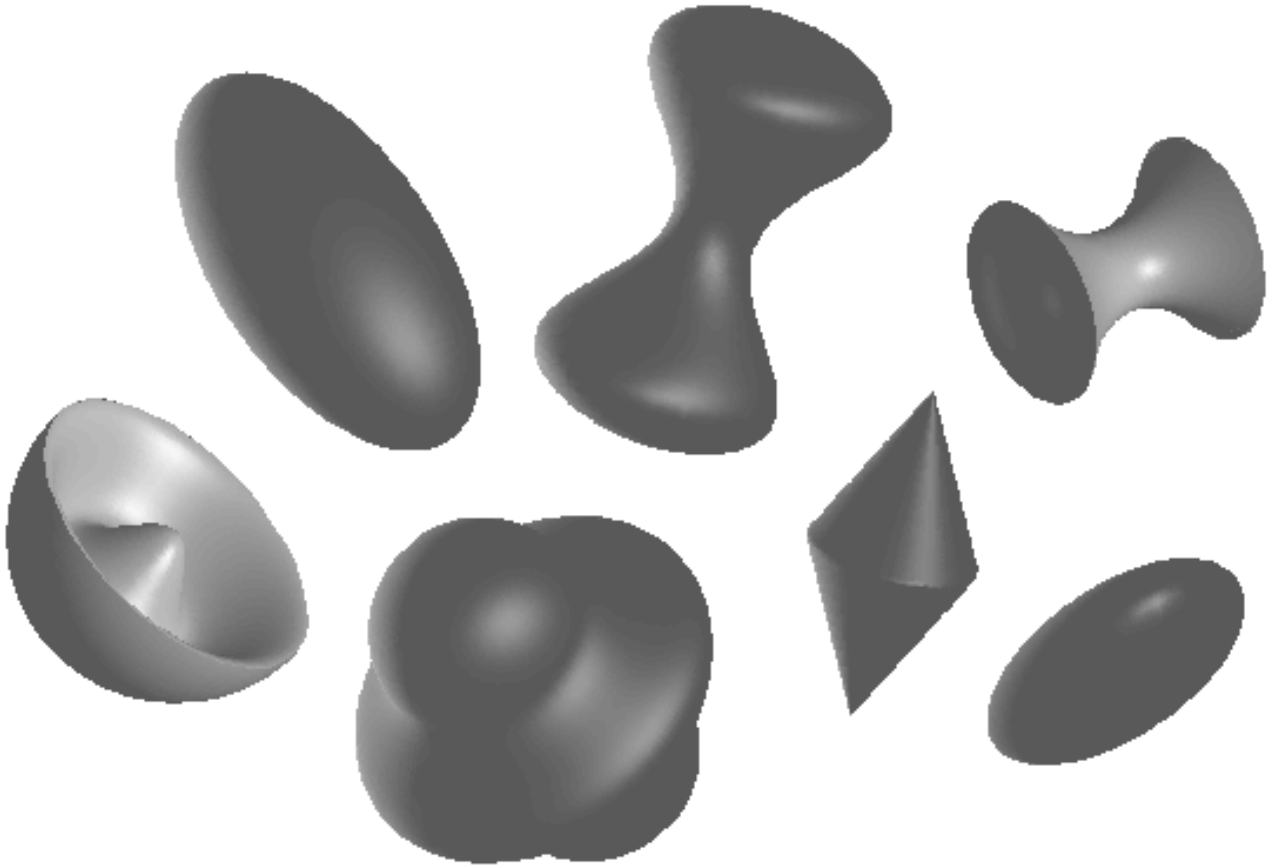
$$T_{\text{invert}} = 2\text{min} \qquad T_{\text{solve}} = 0.3\text{sec}.$$

Pictures illustrating the set-up for axisymmetric problems:

Work in progress (with Sijia Hao) — extension to multibody acoustic scattering.

Individual scattering matrices are constructed via a relatively expensive pre-computation.

Inter-body interactions are handled via the wideband FMM and an iterative solver.

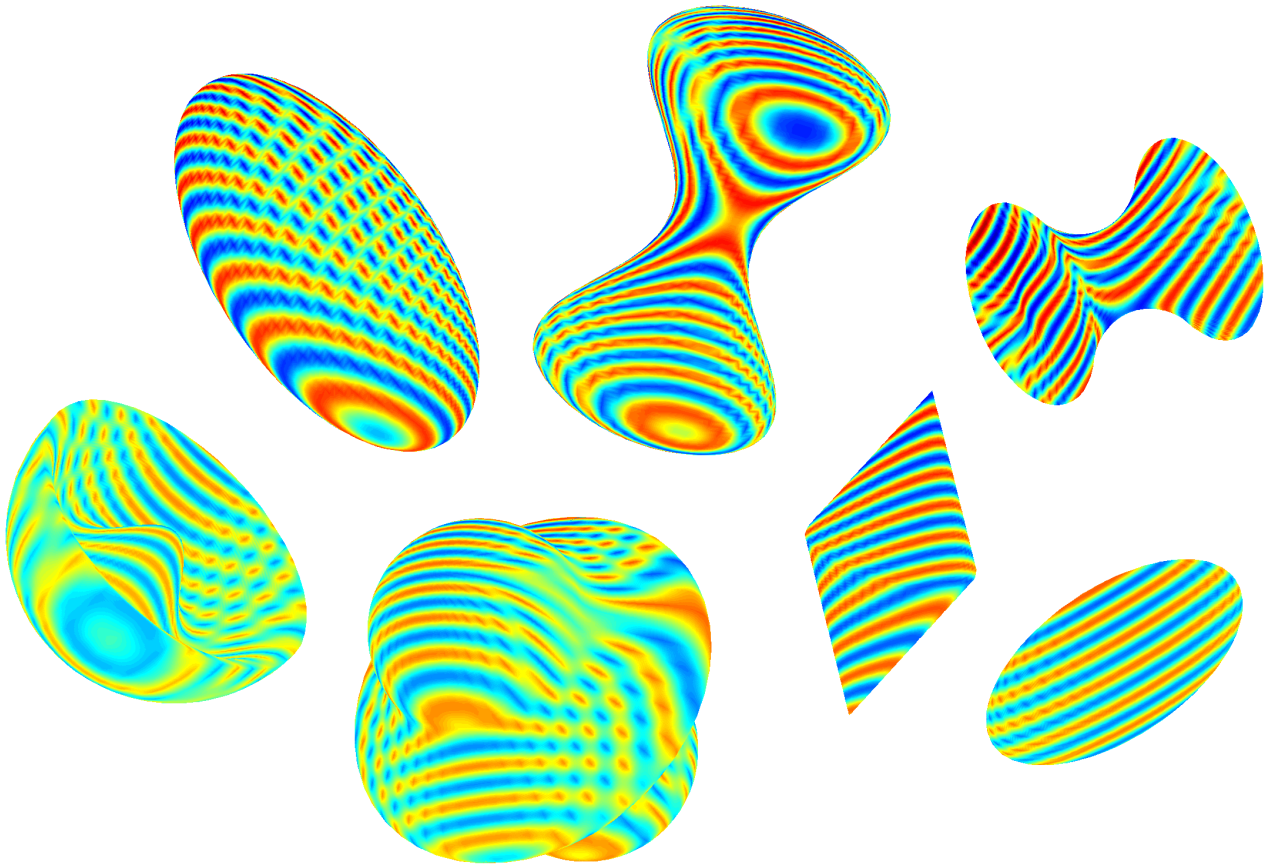


Pictures illustrating the set-up for axisymmetric problems:

Work in progress (with Sijia Hao) — extension to multibody acoustic scattering.

Individual scattering matrices are constructed via a relatively expensive pre-computation.

Inter-body interactions are handled via the wideband FMM and an iterative solver.



Adrianna Gillman's dissertation:

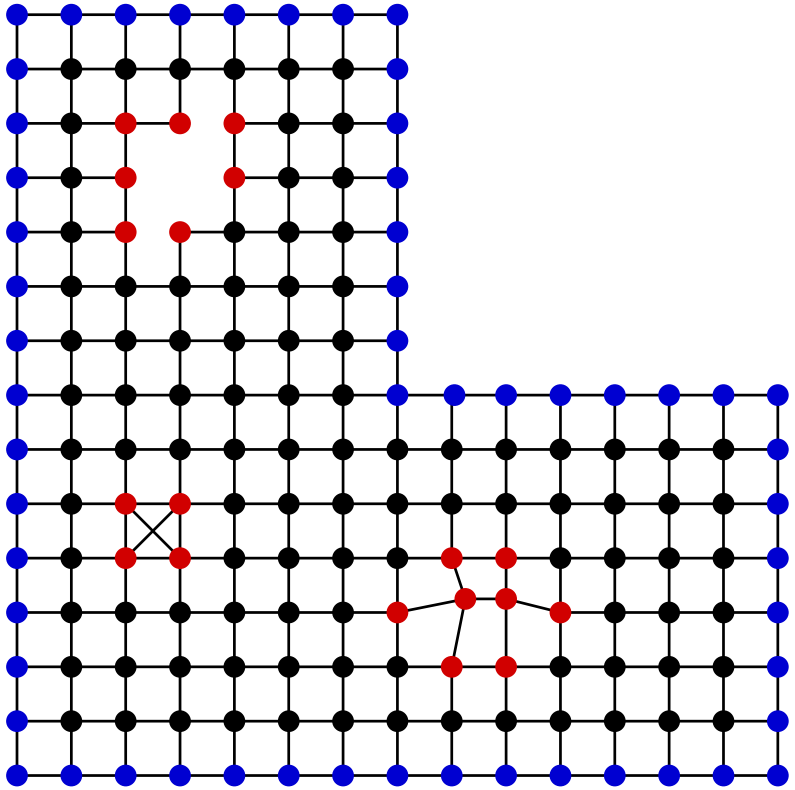
Warm-up project: develop a method for rapidly solving potential problems on large *lattices*. Such problems model large atomic crystals and mechanical truss structures; serve as conceptual models in theoretical physics; and also arise in numerical analysis.

Main project: Develop a direct solver for the linear systems arising from finite difference or finite element discretization of 2D problems with $O(N)$ complexity. This involved developing both theory and a substantial library of functions for performing structured matrix computations.

Fun side-project: Developed a new method (!) for discretizing elliptic PDEs with variable coefficients in the plane.

Future project: Fast solvers for integral equations modeling soft scattering in the plane. These are *volume* integral equations (the so-called Lippman-Schwinger equations).

Picture illustrating the lattice problems we worked on:



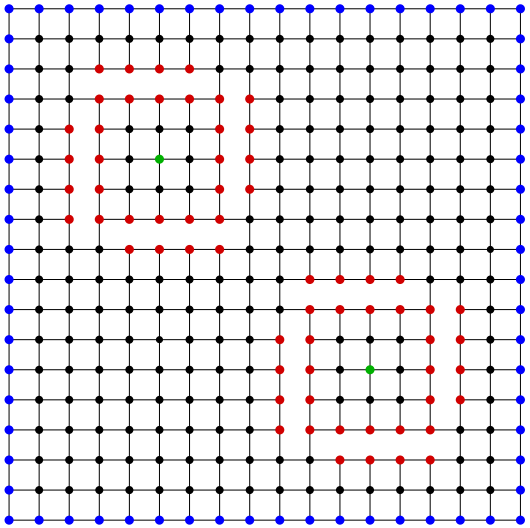
Goal: Solve the following problem:

$$[(\mathbf{A}+\mathbf{B})\boldsymbol{u}](m) = \boldsymbol{f}(m), \qquad m \in \Omega,$$
$$\boldsymbol{u}(m) = \boldsymbol{g}(m), \qquad m \in \Gamma.$$

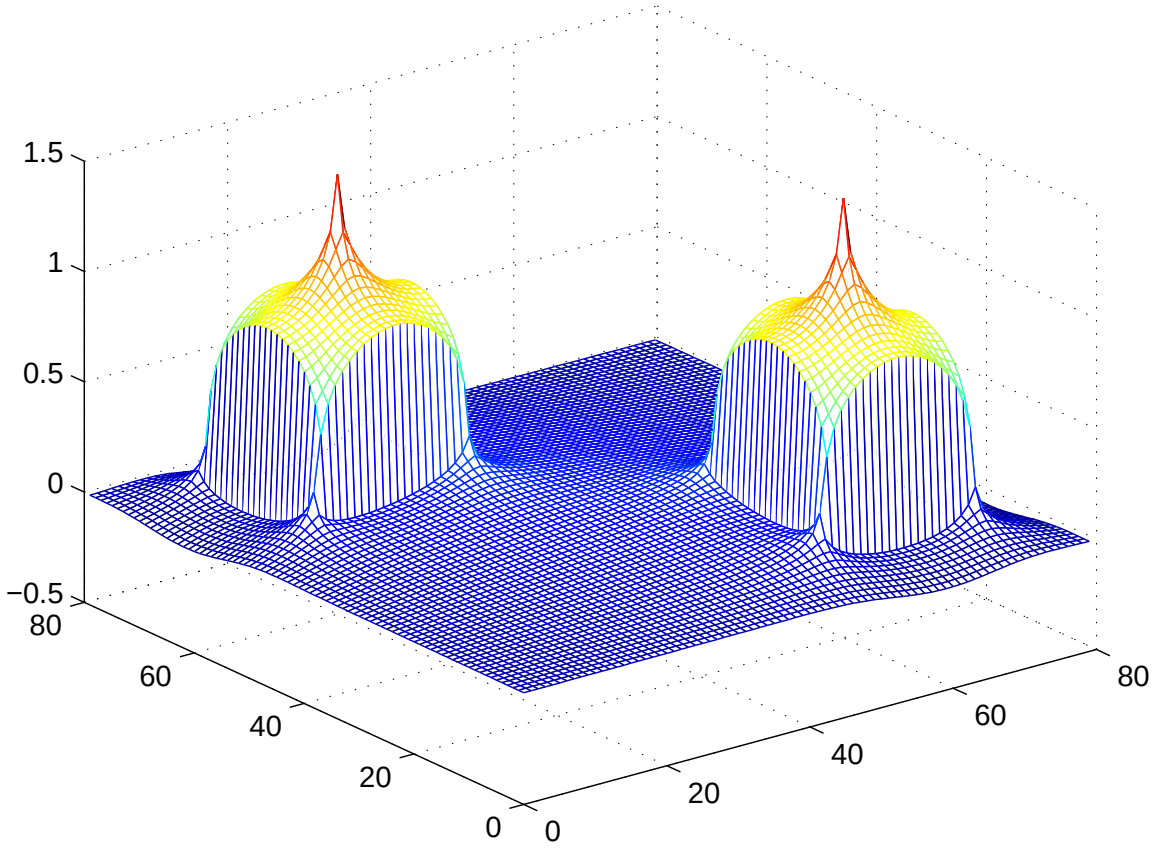
where $\mathbf{B}$ exist on a finite set $\Omega_{\text{imp}} \in \Omega$ such that

- $\mathbf{B}\boldsymbol{u} = 0$ whenever $\text{Support}(u) \bigcap \Omega_{\text{imp}} = \emptyset$
- $\text{Support}(\mathbf{B}\boldsymbol{u}) \subset \Omega_{\text{imp}}$ for any u.

Picture illustrating the lattice problems we worked on — a particularly hard one:



The geometry



A solution

Table illustrating the performance of the fast direct solver for a finite difference matrix.

The matrix is the standard five-point stencil associated with a (non-constant coefficient!) Laplace-type equation. The grid is an $N = n \times n$ square regular one.

N	T_{solve} (sec)	T_{apply} (sec)	M (MB)	e_3	e_4
512^2	7.98	0.007	8.4	$2.7523e-6$	$6.6631e-9$
1024^2	26.49	0.014	18.6	-	-
2048^2	98.46	0.020	33.1	-	-
4096^2	435.8	0.039	65.6	-	-

T_{solve} Time required to compute all Schur complements ("set-up time")

T_{apply} Time required to apply a Dirichlet-to-Neumann op. (of size $4\sqrt{N} \times 4\sqrt{N}$)

M Memory required to store the solution operator

e_3 The l^2-error in the vector $\tilde{\mathbf{A}}_{nn}^{-1} r$ where r is a unit vector of random direction.

e_4 The l^2-error in the first column of $\tilde{\mathbf{A}}_{nn}^{-1}$.

Nathan Halko's dissertation:

Question: How do you efficiently compute a low-rank approximation (e.g. a partial singular value decomposition, a partial QR decomposition, etc) to a given matrix?

(This work has nothing to do with PDE solvers...)

It turns out there is a simple randomized algorithm for doing this:
Let A be a given $m \times n$ matrix and suppose that we seek a rank-k approximation to it:

1. Form a random matrix Ω of size $n \times (k+10)$.
2. Form a sample matrix $Y = A\Omega$.
3. Form an orthonormal basis matrix Q via QR factorization $Y = QR$.
4. Project A onto the space spanned by the columns of Q via $B = Q^* A$.
5. Form the SVD of B so that $B = \hat{U}\Sigma V^*$.
6. Set $U = Q\,\hat{U}$.

The result is a set of matrix U, Σ, and V such that $A \approx U\,\Sigma\,V^*$.

Let A be a given $m \times n$ matrix and suppose that we seek a rank-k approximation to it:

1. Form a random matrix Ω of size $n \times (k+10)$.
2. Form a sample matrix $Y = A\Omega$.
3. Form an orthonormal basis matrix Q via QR factorization $Y = QR$.
4. Project A onto the space spanned by the columns of Q via $B = Q^*A$.
5. Form the SVD of B so that $B = \hat{U}\Sigma V^*$.
6. Set $U = Q\hat{U}$.

The result is a set of matrix U, Σ, and V such that $A \approx U\,\Sigma\,V^*$.

Super-simple to code in Matlab:

1. `Omega = randn(n,k+10)`
2. `Y = A * Omega`
3. `Q = orth(Y)`
4. `B = Q*A`
5. `[hatU,Sigma,V] = svd(B,0)`
6. `U = hatU * Q`

The algorithm described on the previous page is very simple, but it turns out to be highly effective in a modern computing environment where *communication* is the real bottleneck (not flops).

- The randomized method is "pass efficient" — you only need to "look at" the matrix a couple of times. This is very different from classical (non-randomized) algorithms.

- Interesting mathematical theory — new results on random matrices.

- Already part of the "Mahout" package for distributed computing that can run on, e.g., the Amazon cluster.

- Applications in very large scale data mining.

- Cost of computation can be reduced from $O(mnk)$ to $O(mn\log(k))$.
  (This requires choosing a different random matrix.)

- The first paper was published in SIAM Review and was the most down-loaded on the entire SIAM website in July 2011.

The final dissertation consisted of three papers:

- A large paper (72 pages) that outlines the basic techniques.
  New theory and new algorithms, as well as a historic survey.

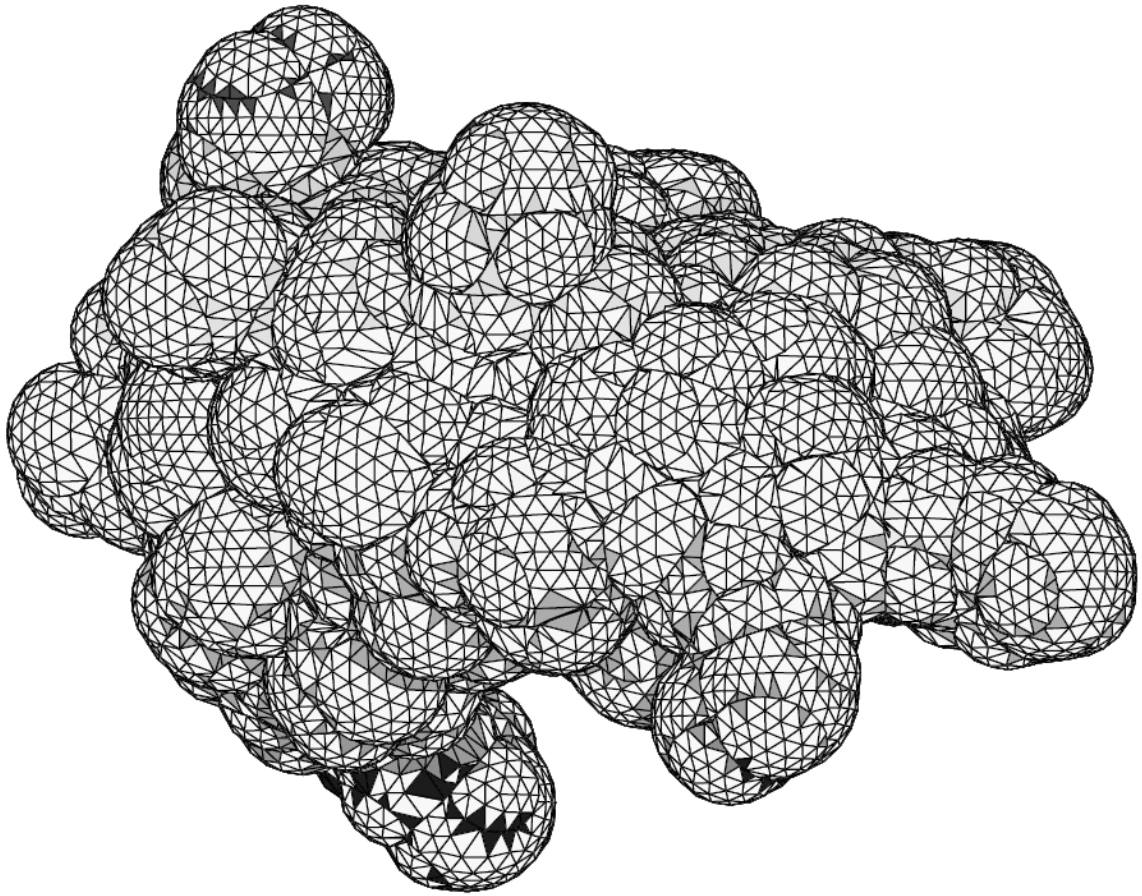
- A paper describing implementation *out-of-core*, meaning that the matrix itself must be stored on a hard drive (it does not fit in RAM memory).

- A paper describing modifications required to execute the algorithms in a fully distributed computing environment. Huge matrices were processed on the Amazon cloud.

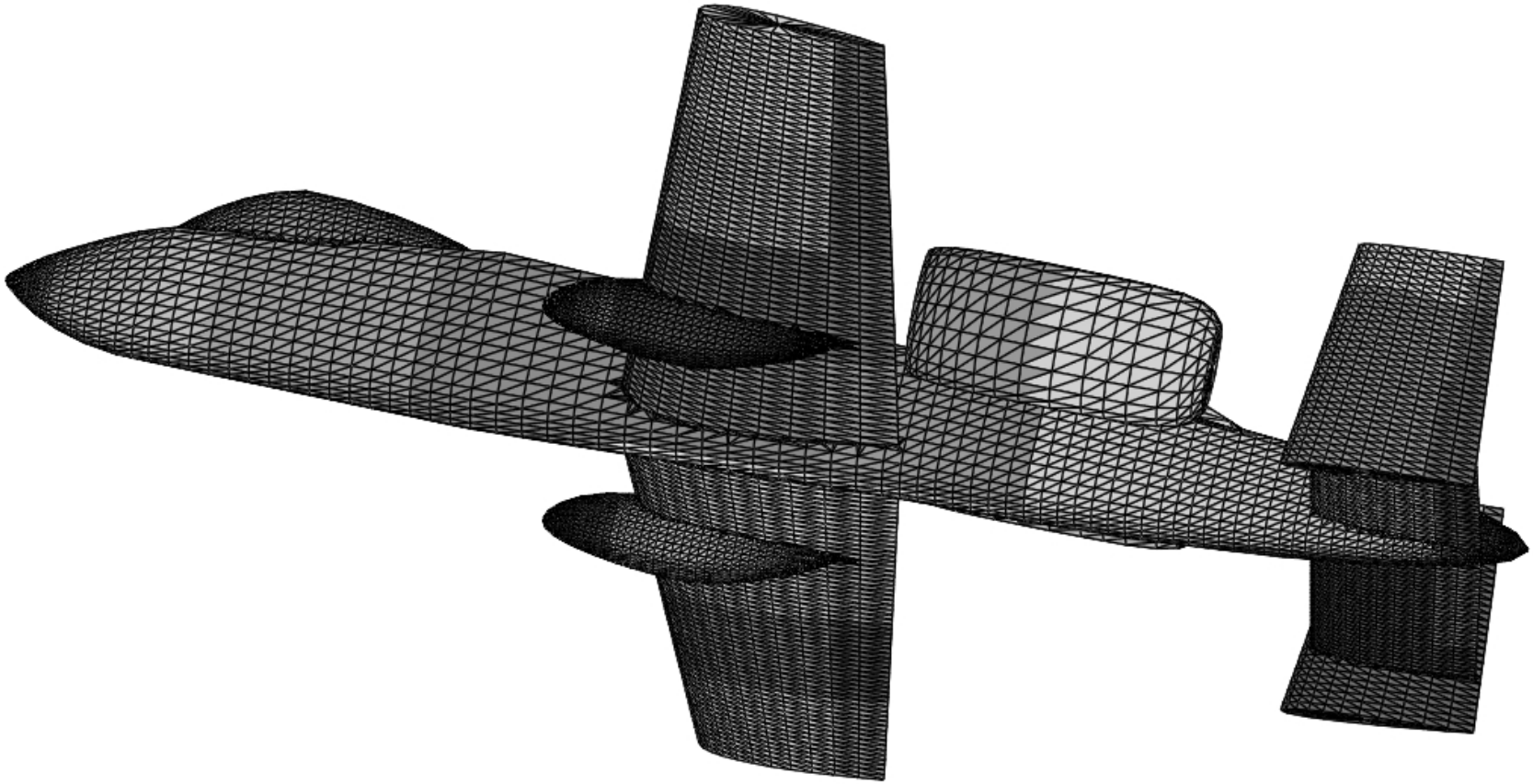
Current projects:

- Continued development of $O(N)$ direct solvers. Extension to 3D.
- Surface representations, quadrature, etc.
- Novel discretization techniques.
- Massive parallel implementations — with Denis Zorin (NYU).
- Applications of direct solvers:
  - Transcranial magnetic stimulation — large 3D examples — with Eric Michielssen (Michigan).
  - Scattering — with Vladimir Rokhlin (Yale).
  - Fluid flows.

Molecular surface — the task is to compute the "solvation energy."



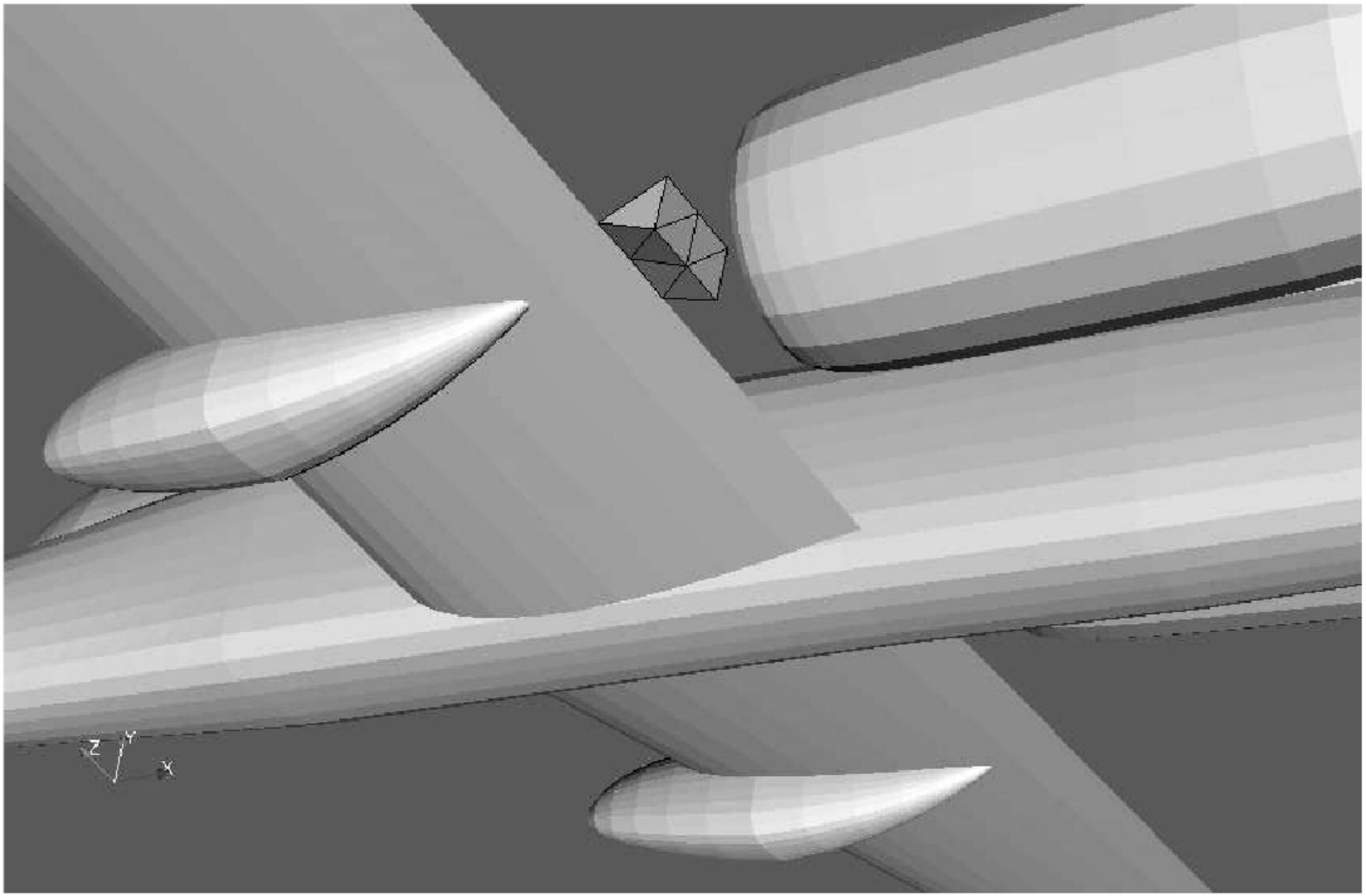
Picture of triangulated aircraft — direct solver works very well here:



Laplace's equation. 28 000 triangles. Standard office desktop.
Cost of very primitive inversion scheme (low accuracy, etc.): 15 min
Cost of applying the inverse: 0.2 sec

From *Fast direct solvers for integral equations in complex three-dimensional domains*, by Greengard, Gueyffier, Martinsson, Rokhlin, Acta Numerica 2009.

Picture of triangulated aircraft — direct solver works very well here:



Observation: Local updates to the geometry are very cheap. Adding a (not so very aerodynamic) flap corresponds to a rank-15 update and can be done in a fraction of a second.

Note: While our codes are very primitive at this point, there exist extensive $\mathcal{H}/\mathcal{H}^2$-matrix based libraries with better asymptotic estimates for inversion. `www.hlib.org`

Transcranial magnetic stimulation project

We have implemented the accelerated nested dissection method to solve the electrostatics problems arising in *transcranial magnetic stimulation (TMS)*. This environment is characterized by:

- There is time to preprocess a given geometry.
- Given a load, the solution should be found "instantaneously."

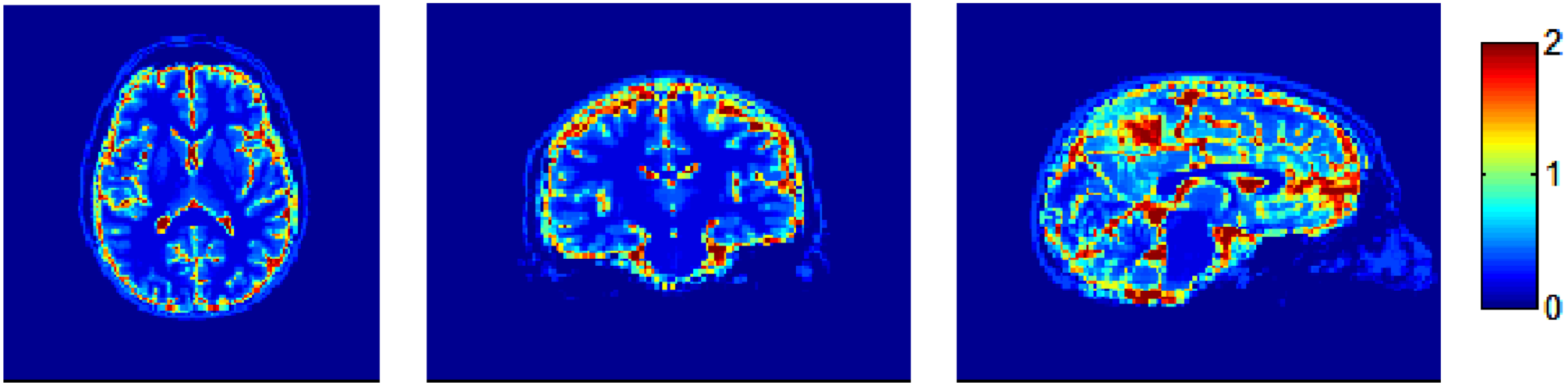


Figure 1: Brain conductivity map. The brain size 77 x 94 x 61. Left to right: Axial, coronal and sagital view.

Joint work with Frantisek Cajko, Luis Gomez, Eric Michielssen, and, Luis Hernandez-Garcia of U. Michigan.

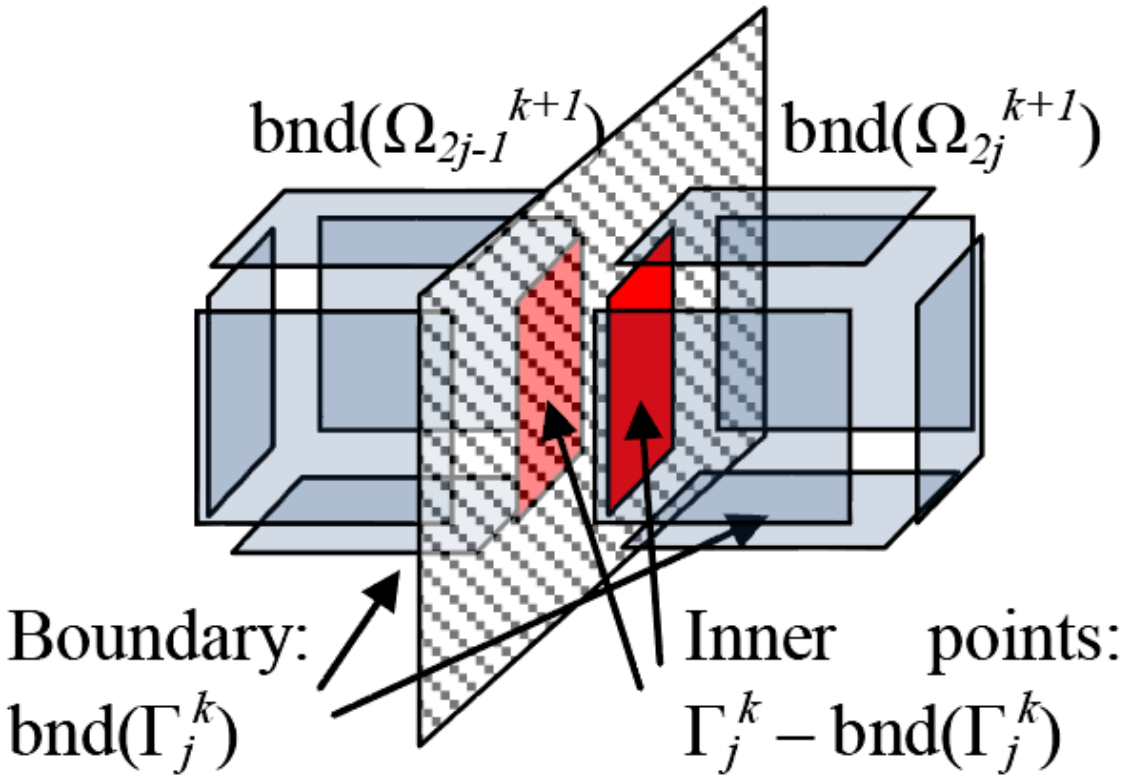


Figure 4: Non-leaf cluster of points and its boundary and inner points. The bisecting plane is shaded.

Transcranial magnetic stimulation project

Numerical results: Single CPU desktop. Local accuracy was 10^{-8}.
Our benchmark was the "Pardiso" package in Intel's MKL.

N	Storage (MB)		Factorization (sec)		Solution (sec)	
	Pardiso	FDS	Pardiso	FDS	Pardiso	FDS
$32^3 = 32768$	157	283	8	53	0.1	0.1
$64^3 = 262144$	2900	3313	549	1683	1.3	1.2
$72^3 = 373248$	4700	4693	1106	3142	2.2	1.9
$81^3 = 531441$	8152	6929	2330	5669	4.1	3.2
$90^3 = 729000$	12500	9550	4630	8658	7.2	4.7

Note: Direct solvers require substantially more memory than, e.g., multigrid.

Note: The gain over Pardiso is modest for small problem sizes, but:

- FDS has better asymptotic scaling as N grows.
- FDS is better suited for large-scale parallelization.
- FDS will be extremely fast for pure "boundary value problems" (this is speculation . . .)
- The FDS methods are still fairly immature; improvements are to be expected.

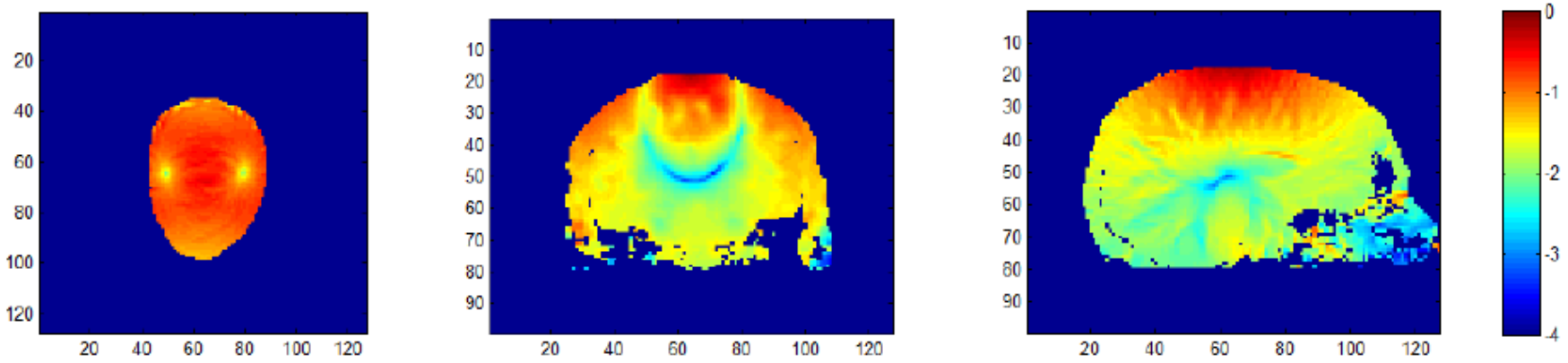


Figure 14: Magnitude of the total electric field inside the head in logarithmic scale. The field is normalized with respect to the highest value in the head. Left to right: Axial (xy-plane, z=24), coronal (xz-palne, y=63) and sagital (yz-plane, x=63) views.

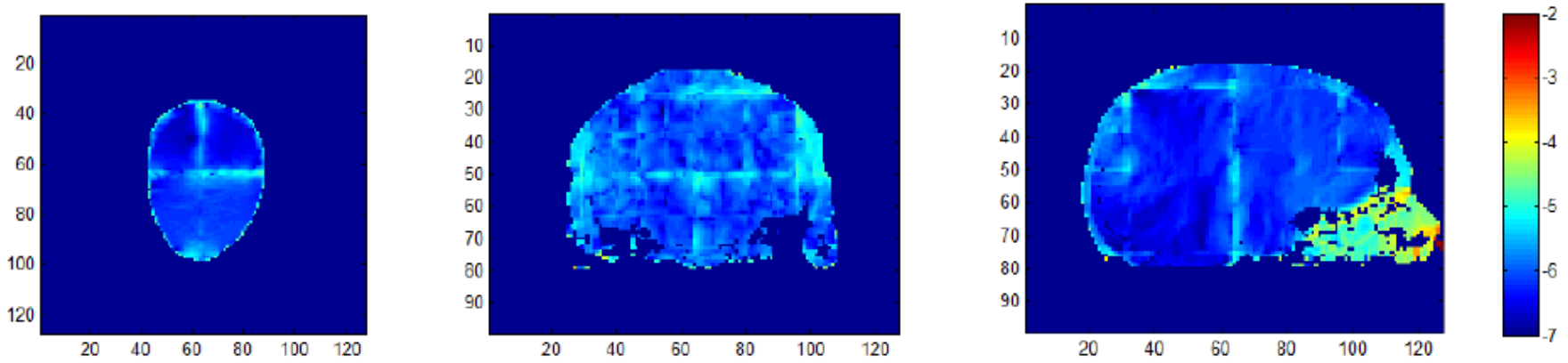


Figure 15: Error magnitude of the total electric field inside the head using a logarithmic scale. The error is the magnitude of the difference between solutions of obtained by the fast direct solver with threshold 1E-8 by the iterative solver with residual 9E-13. The error was normalized to the highest value of the electric field in the head. The largest magnitude of the error is 0.02. Left to right: Axial (z=24), coronal (y=63) and sagital (x=63) views.

Current projects:

- Continued development of $O(N)$ direct solvers. Extension to 3D.
- Surface representations, quadrature, etc.
- Novel discretization techniques.
- Massive parallel implementations — with Denis Zorin (NYU).
- Applications of direct solvers:
  - Transcranial magnetic stimulation — large 3D examples — with Eric Michielssen (Michigan).
  - Scattering — with Vladimir Rokhlin (Yale).
  - Fluid flows.

Nice combination of three main components:

1. Mathematics.
2. Numerical methods.
3. Computing.