Fast direct solvers

P.G. Martinsson, The University of Colorado at Boulder

Acknowledgements: Some of the work presented is joint work with Vladimir Rokhlin and Mark Tygert at Yale University.
In this talk, we will discuss numerical methods for solving the equation

\[
\begin{aligned}
A u(x) &= g(x), & x \in \Omega, \\
B u(x) &= f(x), & x \in \Gamma,
\end{aligned}
\]

where \( A \) is a linear constant-coefficient partial differential operator, and \( B \) is some local linear boundary operator.

Examples include:

- Laplace’s equation.
- Helmholtz’ equation.
- Stokes’ equation.
- The Yukawa equation.
- The equations of linear elasticity.

Specifically, we will be concerned with the fast solution of the system of linear equations obtained upon discretization of (BVP).
There are two standard techniques for obtaining the discretized system:

- **Linear boundary value problem.**

  - Conversion of the BVP to a Boundary Integral Operator (BIE).
  - Discretization of (BIE) using Nyström, collocation, Boundary Element Method, . . .


  - \(N \times N\) system of linear algebraic equations.
1 – Methods based on discretizing PDEs:
Discretize the differential operator directly; instead of

\[
\begin{aligned}
& A u(x) = g(x), \quad x \in \Omega, \\
& u(x) = f(x), \quad x \in \Gamma,
\end{aligned}
\]

solve

\[
A_N u_N = h_N,
\]

where \( u_N \) is a function in an \( N \)-dimensional function space, \( A_N \) is an \( N \times N \) matrix discretizing the operator \( A \) (obtained via Finite Elements / Finite Differences / \ldots), and \( h_N \) is a vector of data derived from \( f \) and \( g \).

Equation (BVP-DISC) is typically very large, and requires fast solvers. Most such solvers are based on iterative methods.

**Fundamental problem:** \( A \) is an unbounded operator \( \Rightarrow \)

The matrix \( A_N \) is ill-conditioned \( \Rightarrow \) The iterative solver converges slowly.
Pre-conditioners can help solving ill-conditioned linear systems.

A pre-conditioner is an operator \( P_N \) such that:

- It is cheap to apply \( P_N \) to a vector.
- The product \( P_N A_N \) is well-conditioned.

Loosely speaking, \( P_N \approx A_N^{-1} \).

The idea is to use an iterative solver to solve

\[
P_N A_N u_N = P_N h_N.
\]

The popular \textbf{multigrid} algorithm is a form of a pre-conditioner.

However, many problems related to ill-conditioning remain.

Would it be possible to directly compute \( A_N^{-1} \)?
2 – Methods based on discretizing integral equations:
*Reformulate the BVP as an Integral Equation.*

The idea is to convert a partial differential equation

\[
\begin{align*}
& A u(x) = g(x), \quad x \in \Omega, \\
& B u(x) = f(x), \quad x \in \Gamma,
\end{align*}
\]

(BVP)

to an “equivalent” integral equation

\[
v(x) + \int_{\Gamma} k(x, y) v(y) \, ds(y) = h(x), \quad x \in \Gamma.
\]

(BIE)

- The kernel \( k \) is derived from the operator \( A \).

- The data function \( h \) is derived from the data of (BVP).

- The conversion from (BVP) to (BIE) sometimes involves the evaluation of certain integrals over \( \Gamma \) and/or \( \Omega \).

- Sometimes the integral equation must be formulated on \( \Omega \).

- ...
**Example:**

Let us consider the equation

(BVP) \[ \begin{cases} -\Delta u(x) = 0, & x \in \Omega, \\ u(x) = f(x), & x \in \Gamma, \end{cases} \]

We make the following Ansatz:

\[ u(x) = \int_{\Gamma} (n(y) \cdot \nabla_y \log |x - y|) v(y) \, ds(y), \quad x \in \Omega, \]

where \( n(y) \) is the outward pointing unit normal of \( \Gamma \) at \( y \). Then the boundary charge distribution \( u \) satisfies the Boundary Integral Equation

(BIE) \[ v(x) + 2 \int_{\Gamma} (n(y) \cdot \nabla_y \log |x - y|) v(y) \, ds(y) = 2f(x), \quad x \in \Gamma. \]

- (BIE) and (BVP) are in a strong sense equivalent.
- (BIE) is appealing mathematically (2\(^{\text{nd}}\) kind Fredholm equation).
When integral equation formulations are available, there are compelling arguments in their favor, these include:

**Conditioning:**
When there exists an IE formulation that is a Fredholm equation of the second kind, the mathematical equation itself is well-conditioned.

**Dimensionality:**
Frequently, an IE can be defined on the boundary of the domain.

**Integral operators are benign objects:**
It is (relatively) easy to implement high order discretizations of integral operators. Relative accuracy of $10^{-10}$ or better is often achieved.
There is a fundamental difficulty with using integral operators in numerics:

Discretization of integral operators typically results in dense matrices.

In the 1950’s when computers made numerical PDE solvers possible, researchers faced a grim choice:

<table>
<thead>
<tr>
<th>PDE-based:</th>
<th>Ill-conditioned, $N$ is too large, low accuracy.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integral Equations:</td>
<td>Dense system.</td>
</tr>
</tbody>
</table>

The integral equations lost and were largely forgotten
— they were simply too expensive.

(Except in some scattering problems where there was no choice.)
The situation changed dramatically in the 1980’s. It was discovered that while $K_N$ (the discretized integral operator) is dense, it is possible to evaluate the matrix-vector product

$$v \mapsto K_N v$$

in $O(N)$ operations – to high accuracy and with a small constant.

A very successful such algorithm is the Fast Multipole Method by Rokhlin and Greengard (circa 1985).

Combining such methods with iterative solvers (GMRES / conjugate gradient / ...) leads to very fast solvers for the integral equations, especially when second kind Fredholm formulations are used.
A prescription for rapidly solving BVPs:

\[
\begin{align*}
\text{(BVP)} & \quad \begin{cases} 
-\Delta v(x) = 0, & x \in \Omega, \\
v(x) = f(x), & x \in \Gamma.
\end{cases}
\end{align*}
\]

Convert (BVP) to a second kind Fredholm equation:

\[
\begin{align*}
\text{(BIE)} & \quad u(x) + \int_{\Gamma} (n(y) \cdot \nabla_y \log |x - y|) u(y) \, ds(y) = f(x), & x \in \Gamma.
\end{align*}
\]

Discretize (BIE) into the discrete equation

\[
\begin{align*}
\text{(DISC)} & \quad (I + K_N)u_N = f_N
\end{align*}
\]

where $K_N$ is a (typically dense) $N \times N$ matrix.

**Fast Multipole Method** — Can multiply $K_N$ by a vector in $O(N)$ time.

**Iterative solver** — Solves (DISC) using $\sqrt{\kappa}$ matrix-vector multiplies, where $\kappa$ is the condition number of $(I + K_N)$.

**Total complexity** — $O(\sqrt{\kappa} N)$. (Recall that $\kappa$ is small. Like 14.)
However, integral equation based methods are quite often not a choice:

*Fundamental limitations:* They require the existence of a fundamental solution to the (dominant part of the) partial difference operator. In practise, this means that the (dominant part of the) operator must be linear and constant-coefficient.

*Practical limitations:* Underdeveloped infra-structure; there does not exist a general framework for discretizing surfaces. Lack of engineering strength codes. Etc.
To summarize:

- There exist $O(N)$ (or $O(N \log^p N)$) algorithms for a wide range of BVPs.

- For some BVPs, the $N$ in $O(N)$ can be the number of degrees of freedom required to discretize the boundary.

- Almost all existing $O(N)$ methods rely on iterative solvers.

- Regardless of how a BVP is discretized, there are complications in solving the resulting linear system.
  - Finite Element Methods: system is sparse but ill-conditioned.
  - Boundary Integral Methods: system is dense (and sometimes ill-conditioned, too).
In some environments, the linear solve presents a serious challenge:

1. Problems whose geometries require a very large number of unknowns:
   - Modeling of heterogeneous materials.
   - Radar scattering off of the ocean surface.

2. Applications that require a very large number of solves:
   - Molecular Dynamics.
   - Optimal design.

3. Problems that are inherently ill-conditioned:
   - Scattering problems at intermediate or high frequencies.
   - Ill-conditioning due to geometry (elongated domains, percolation, etc).

The inadequacy of existing methods in these environments stems from their reliance on iterative solvers. We need direct solvers.
WHAT IS A DIRECT SOLVER?

Recall that many BVPs can be cast in the following form:

\[(\text{BIE}) \quad u(x) + \int_{\Gamma} g(x, y) u(y) \, ds(y) = f(x), \quad x \in \Gamma.\]

Upon discretization, equation (BIE) turns into a discrete equation

\[(\text{DISC}) \quad (I + K_N)u = f\]

where $K_N$ is a (typically dense) $N \times N$ matrix.

A \textit{direct method} computes a compressed representation for $(I + K_N)^{-1}$.

- Cost for pre-computing the inverse.
- Cost for applying the inverse to a vector.

In many environments, both of these costs can be made $O(N)$.

Direct methods are good for (1) ill-conditioned problems, (2) problems with multiple right-hand sides, (3) spectral decompositions, (4) updating, \ldots
Practical 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 should be viewed as a step towards getting a LAPACK-type environment for solving the basic linear boundary value problems of mathematical physics.
Sampling of related work:

1991 Sparse matrix algebra / wavelets, Beylkin, Coifman, Rokhlin,
1996 scattering problems, E. Michielssen, A. Boag and W.C. Chew,
1998 factorization of non-standard forms, G. Beylkin, J. Dunn, D. Gines,
1998 $\mathcal{H}$-matrix methods, W. Hackbusch, et al,
2002 $O(N^{3/2})$ inversion of Lippmann-Schwinger equations, Y. Chen,
2002 inversion of “Hierarchically semi-separable” matrices, M. Gu,
    S. Chandrasekharan, et al,
2004 $O(N)$ inversion of boundary integral equations in 2D, Martinsson, Rokhlin,
2007 $O(N \log N)$ inversion of 2D finite element matrices, Martinsson.
CURRENT STATE OF THE RESEARCH

The fast direct solvers currently being developed exploit the fact that off-diagonal blocks of the matrix to be inverted have low rank.

This restricts the range of application to non-oscillatory, or moderately oscillatory problems. In other words, such methods currently can handle:

- Laplace’s equation, equations of elasticity, Yukawa’s equation, . . .
- Helmholtz’ and Maxwell’s equations for low and intermediate frequencies.
  (In special cases, high frequency problem can also be solved.)

How does the inversion scheme for 2D boundary integral equations work?

Consider the linear system

\[
\begin{bmatrix}
A_{11} & A_{12} & A_{13} & A_{14} \\
A_{21} & A_{22} & A_{23} & A_{24} \\
A_{31} & A_{32} & A_{33} & A_{34} \\
A_{41} & A_{42} & A_{43} & A_{44}
\end{bmatrix}
\begin{bmatrix}
q_1 \\
q_2 \\
q_3 \\
q_4
\end{bmatrix}
= 
\begin{bmatrix}
v_1 \\
v_2 \\
v_3 \\
v_4
\end{bmatrix}.
\]

We suppose that for \( i \neq j \), the blocks \( A_{ij} \) allow the factorization

\[
A_{ij} = U_i \tilde{A}_{ij} U_j^t,
\]

where the ranks \( k_i \) are significantly smaller than the block sizes \( n_i \).

We then let

\[
\tilde{q}_i = U_i^t q_i,
\]

be the variables of the “reduced” system.
Recall: $A_{ij} = U_i \tilde{A}_{ij} U_j^T$ and $\tilde{q}_i = U_i^T q_i$.

The system $\sum_j A_{ij} q_j = v_i$ then takes the form

$$
\begin{bmatrix}
A_{11} & 0 & 0 & 0 \\
0 & A_{22} & 0 & 0 \\
0 & 0 & A_{33} & 0 \\
0 & 0 & 0 & A_{44}
\end{bmatrix}
\begin{bmatrix}
0 \\
U_2 \tilde{A}_{21} \\
U_3 \tilde{A}_{31} \\
U_4 \tilde{A}_{41}
\end{bmatrix}
\begin{bmatrix}
U_1 \tilde{A}_{12} \\
0 \\
U_3 \tilde{A}_{32} \\
U_4 \tilde{A}_{42}
\end{bmatrix}
\begin{bmatrix}
U_1 \tilde{A}_{13} \\
U_2 \tilde{A}_{23} \\
0 \\
U_4 \tilde{A}_{43}
\end{bmatrix}
\begin{bmatrix}
U_1 \tilde{A}_{14} \\
U_2 \tilde{A}_{24} \\
U_3 \tilde{A}_{34} \\
0
\end{bmatrix}
\begin{bmatrix}
q_1 \\
q_2 \\
q_3 \\
q_4
\end{bmatrix}
= 
\begin{bmatrix}
v_1 \\
v_2 \\
v_3 \\
v_4
\end{bmatrix}
.$$ 

Now form the Schur complement to eliminate the $q_j$’s.
After eliminating the “fine-scale” variables $q_i$, we obtain

$$
\begin{bmatrix}
I & U_t^t \tilde{A}_{11}^{-1} U_1 \tilde{A}_{12} & U_t^t \tilde{A}_{11}^{-1} U_1 \tilde{A}_{13} & U_t^t \tilde{A}_{11}^{-1} U_1 \tilde{A}_{14} \\
U_t^t \tilde{A}_{22}^{-1} U_2 \tilde{A}_{21} & I & U_t^t \tilde{A}_{22}^{-1} U_2 \tilde{A}_{23} & U_t^t \tilde{A}_{22}^{-1} U_2 \tilde{A}_{24} \\
U_t^t \tilde{A}_{33}^{-1} U_3 \tilde{A}_{31} & U_t^t \tilde{A}_{33}^{-1} U_3 \tilde{A}_{32} & I & U_t^t \tilde{A}_{33}^{-1} U_3 \tilde{A}_{34} \\
U_t^t \tilde{A}_{44}^{-1} U_4 \tilde{A}_{41} & U_t^t \tilde{A}_{44}^{-1} U_4 \tilde{A}_{42} & U_t^t \tilde{A}_{44}^{-1} U_4 \tilde{A}_{43} & I
\end{bmatrix}
\begin{bmatrix}
\tilde{q}_1 \\
\tilde{q}_2 \\
\tilde{q}_3 \\
\tilde{q}_4
\end{bmatrix}
= 
\begin{bmatrix}
U_t^t \tilde{A}_{11}^{-1} v_1 \\
U_t^t \tilde{A}_{22}^{-1} v_2 \\
U_t^t \tilde{A}_{33}^{-1} v_3 \\
U_t^t \tilde{A}_{44}^{-1} v_4
\end{bmatrix}.
$$

We set

$$
\tilde{A}_{ii} = (U_t^t A_{ii}^{-1} U_i)^{-1},
$$

and multiply line $i$ by $\tilde{A}_{ii}$ to obtain the reduced system

$$
\begin{bmatrix}
\tilde{A}_{11} & \tilde{A}_{12} & \tilde{A}_{13} & \tilde{A}_{14} \\
\tilde{A}_{21} & \tilde{A}_{22} & \tilde{A}_{23} & \tilde{A}_{24} \\
\tilde{A}_{31} & \tilde{A}_{32} & \tilde{A}_{33} & \tilde{A}_{34} \\
\tilde{A}_{41} & \tilde{A}_{42} & \tilde{A}_{43} & \tilde{A}_{44}
\end{bmatrix}
\begin{bmatrix}
\tilde{q}_1 \\
\tilde{q}_2 \\
\tilde{q}_3 \\
\tilde{q}_4
\end{bmatrix}
= 
\begin{bmatrix}
\tilde{v}_1 \\
\tilde{v}_2 \\
\tilde{v}_3 \\
\tilde{v}_4
\end{bmatrix}.
$$

where

$$
\tilde{v}_i = \tilde{A}_{ii} U_t^t A_{ii}^{-1} v_i.
$$

(This derivation was pointed out by Leslie Greengard.)
A globally $O(N)$ algorithm is obtained by hierarchically repeating the process:
The one-level coarse-graining involves the following steps:

- Compute $U_i$ and $\tilde{A}_{ij}$ so that $A_{ij} = U_i \tilde{A}_{ij} U_j^t$.

- Compute the new diagonal matrices

  $$\tilde{A}_{ii} = (U_i A_{ii}^{-1} U_i)^{-1},$$

- Compute the new loads

  $$\tilde{v}_i = \tilde{A}_{ii} U_i^t A_{ii}^{-1} v_i.$$

For the algorithm to be $O(N)$, it has to be able to carry out these steps \textit{locally}.

To achieve this, we use \textit{interpolative} representations.

\textbf{\tilde{A}_{ij} will be a submatrix of $A_{ij}$, so it will not need to be computed.}
The key observation is that $k = \text{rank}(A_{12}) < \min(M, N)$. 

```
(A_{12})
```

```
\text{Sources } \{q_n\}_{n=1}^N \rightarrow \{v_m\}_{m=1}^M
```

"Small" representation
Skeletonization

We can pick $k$ points in $\Omega_S$ with the property that any potential in $\Omega_T$ can be replicated by placing charges on these $k$ points.

- The choice of points does not depend on $\{q_n\}_{n=1}^N$.
- $A_{12}^{\text{ske}}$ is a submatrix of $A_{12}$.
We can “skeletonize” both $\Omega_1$ and $\Omega_2$.

$$\text{Rank} = 19 \text{ at } \varepsilon = 10^{-10}.$$
Skeletonization can be performed for $\Omega_S$ and $\Omega_T$ of various shapes.

Rank = 29 at $\varepsilon = 10^{-10}$. 
Rank = 48 at $\varepsilon = 10^{-10}$. 
Adjacent boxes can be skeletonized.

Rank = 46 at $\varepsilon = 10^{-10}$. 
Benefits:

- The rank is optimal.
- The projection and interpolation are cheap. $U_1$ and $U_2$ contain $k \times k$ identity matrices.
- The projection and interpolation are well-conditioned.
- Finding the $k$ points is cheap.
- The map $\tilde{A}_{12}$ is simply a restriction of the original map $A_{12}$. (We can loosely say that “the physics of the problem is preserved”.)
- Interaction between adjacent boxes can be compressed (no buffering is required).
Similar schemes have been proposed by many researchers:

1993 - C.R. Anderson

1995 - C.L. Berman

1996 - E. Michielssen, A. Boag

1999 - J. Makino

2004 - L. Ying, G. Biros, D. Zorin

A mathematical foundation:

1996 - M. Gu, S. Eisenstat
Let us return to the direct solver environment. Recall:

We convert the system

\[
\begin{bmatrix}
A_{11} & A_{12} & A_{13} & A_{14} \\
A_{21} & A_{22} & A_{23} & A_{24} \\
A_{31} & A_{32} & A_{33} & A_{34} \\
A_{41} & A_{42} & A_{43} & A_{44}
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4
\end{bmatrix}
=
\begin{bmatrix}
f_1 \\
f_2 \\
f_3 \\
f_4
\end{bmatrix}.
\]

to the reduced system

\[
\begin{bmatrix}
\tilde{A}_{11} & \tilde{A}_{12} & \tilde{A}_{13} & \tilde{A}_{14} \\
\tilde{A}_{21} & \tilde{A}_{22} & \tilde{A}_{23} & \tilde{A}_{24} \\
\tilde{A}_{31} & \tilde{A}_{32} & \tilde{A}_{33} & \tilde{A}_{34} \\
\tilde{A}_{41} & \tilde{A}_{42} & \tilde{A}_{43} & \tilde{A}_{44}
\end{bmatrix}
\begin{bmatrix}
\tilde{x}_1 \\
\tilde{x}_2 \\
\tilde{x}_3 \\
\tilde{x}_4
\end{bmatrix}
=
\begin{bmatrix}
\tilde{f}_1 \\
\tilde{f}_2 \\
\tilde{f}_3 \\
\tilde{f}_4
\end{bmatrix}.
\]

We know that \(A_{ij}^{\text{skel}}\) is a submatrix of \(A_{ij}\) when \(i \neq j\).

What is \(\tilde{A}_{ii}\)?
We recall that the new diagonal blocks are defined by
\[
\tilde{A}_{ii} = \left( U^t_i A^{-1}_{ii} U_i \right)^{-1}.
\]

We call these blocks “proxy matrices”.

What are they?

Let \( \Omega_1 \) denote the block marked in red.

Let \( \Omega_2 \) denote the rest of the domain.

Charges on \( \Omega_2 \) \( A_{12} \rightarrow \) Pot. on \( \Omega_1 \) \( A_{11}^{-1} \rightarrow \) Charges on \( \Omega_1 \) \( A_{21} \rightarrow \) Pot. on \( \Omega_2 \)

\( \tilde{A}_{11} \) contains all the information the outside world needs to know about \( \Omega_1 \).
We recall that the new diagonal blocks are defined by
\[
\tilde{A}_{ii} = (U^t_i A_i^{-1} U_i)^{-1}.
\]

We call these blocks “proxy matrices”.

What are they?

Let \( \Omega_1 \) denote the block marked in red.

Let \( \Omega_2 \) denote the rest of the domain.

\( \tilde{A}_{11} \) contains all the information the outside world needs to know about \( \Omega_1 \).
To obtain a globally $O(N)$ scheme, we hierarchically merge proxy matrices.
Numerical examples

In developing direct solvers, the “proof is in the pudding” — recall that from a theoretical point of view, the problem is already solved (by Hackbusch and others).

All computations were performed on standard laptops and desktop computers in the 2.0GHz - 2.8Ghz speed range, and with 512Mb of RAM.
An exterior Helmholtz Dirichlet problem

A smooth contour. Its length is roughly 15 and its horizontal width is 2.
Computational results for an exterior Helmholtz Dirichlet problem discretized with 10\textsuperscript{th} order accurate quadrature. The Helmholtz parameter was chosen to keep the number of discretization points per wavelength constant at roughly 45 points per wavelength (resulting in a quadrature error about $10^{-12}$).

Eventually . . . the complexity is $O(n + k^3)$.
Example 2 - An interior Helmholtz Dirichlet problem

The diameter of the contour is about 2.5. An interior Helmholtz problem with Dirichlet boundary data was solved using \( N = 6400 \) discretization points, with a prescribed accuracy of \( 10^{-10} \).

For \( k = 100.011027569 \cdots \), the smallest singular value of the boundary integral operator was \( \sigma_{\text{min}} = 0.00001366 \cdots \).

Time for constructing the inverse: 0.7 seconds.

Error in the inverse: \( 10^{-5} \).
Plot of $\sigma_{\text{min}}$ versus $k$ for an interior Helmholtz problem on the smooth pentagram. The values shown were computed using a matrix of size $N = 6400$. Each point in the graph required about 60s of CPU time.
Example 3:

An electrostatics problem in a dielectrically heterogeneous medium

\[ \varepsilon = 10^{-5} \quad N_{\text{contour}} = 25,600 \quad N_{\text{particles}} = 100,000 \]

Time to invert the boundary integral equation = 46 sec.

Time to compute the induced charges = 0.42 sec. (2.5 sec for the FMM)

Total time for the electro-statics problem = 3.8 sec.
A close-up of the particle distribution.
Example 4: Inversion of a “Finite Element Matrix”

A grid conduction problem (the “five-point stencil”).

The conductivity of each bar is a random number drawn from a uniform distribution on $[1, 2]$. 
<table>
<thead>
<tr>
<th>$N$</th>
<th>$T_{\text{invert}}$ (seconds)</th>
<th>$T_{\text{apply}}$ (seconds)</th>
<th>$M$ (kB)</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>5.93e-1</td>
<td>2.82e-3</td>
<td>3.82e+2</td>
<td>1.29e-8</td>
<td>1.37e-7</td>
<td>2.61e-8</td>
<td>3.31e-8</td>
</tr>
<tr>
<td>40000</td>
<td>4.69e+0</td>
<td>6.25e-3</td>
<td>9.19e+2</td>
<td>9.35e-9</td>
<td>8.74e-8</td>
<td>4.71e-8</td>
<td>6.47e-8</td>
</tr>
<tr>
<td>90000</td>
<td>1.28e+1</td>
<td>1.27e-2</td>
<td>1.51e+3</td>
<td>—</td>
<td>—</td>
<td>7.98e-8</td>
<td>1.25e-7</td>
</tr>
<tr>
<td>160000</td>
<td>2.87e+1</td>
<td>1.38e-2</td>
<td>2.15e+3</td>
<td>—</td>
<td>—</td>
<td>9.02e-8</td>
<td>1.84e-7</td>
</tr>
<tr>
<td>250000</td>
<td>4.67e+1</td>
<td>1.52e-2</td>
<td>2.80e+3</td>
<td>—</td>
<td>—</td>
<td>1.02e-7</td>
<td>1.14e-7</td>
</tr>
<tr>
<td>360000</td>
<td>7.50e+1</td>
<td>2.62e-2</td>
<td>3.55e+3</td>
<td>—</td>
<td>—</td>
<td>1.37e-7</td>
<td>1.57e-7</td>
</tr>
<tr>
<td>490000</td>
<td>1.13e+2</td>
<td>2.78e-2</td>
<td>4.22e+3</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>640000</td>
<td>1.54e+2</td>
<td>2.92e-2</td>
<td>5.45e+3</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>810000</td>
<td>1.98e+2</td>
<td>3.09e-2</td>
<td>5.86e+3</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1000000</td>
<td>2.45e+2</td>
<td>3.25e-2</td>
<td>6.66e+3</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

- $e_1$ The largest error in any entry of $\tilde{A}_n^{-1}$
- $e_2$ The error in $l^2$-operator norm of $\tilde{A}_n^{-1}$
- $e_3$ The $l^2$-error in the vector $\tilde{A}_nn^{-1}r$ where $r$ is a unit vector of random direction.
- $e_4$ The $l^2$-error in the first column of $\tilde{A}_nn^{-1}$.
$\frac{T_{\text{invert}}}{N}$ versus $N$
\frac{T_{\text{apply}}}{\sqrt{N}} \quad \text{versus} \quad N
$\frac{M}{\sqrt{N}}$ versus $N$. 
Existing fast direct solvers:

- 2D boundary integral equations. Very fast.
  Has proven capable of solving previously intractable problems.

- Certain 2D scattering problems.

- 2D finite element matrices.
  \(1\,000\,000 \times 1\,000\,000\) matrix inverted in 4 minutes using 7Mb of memory,
  subsequent solves take 0.03 seconds.

In development:

- Fast inversion schemes for 2D volume integral equations.

- 3D boundary integral equations.

- Applications to biochemical modelling.

- Applications to multiscale modelling.