A fast direct solver for boundary integral equations in two dimensions

P.G. Martinsson and V. Rokhlin

Yale University

Thanks to: M. Tygert

Boundary equation representation of a PDE

Example: Let us represent the solution of the boundary value problem

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

as a double layer potential,

$$v(x) = \int_\Gamma \big(n(y) \cdot \nabla_y \log |x - y|\big) u(y)\, ds(y), \qquad x \in \Omega,$$

where $n(y)$ is the outward pointing unit normal of Γ at y. Then the boundary charge distribution u satisfies the boundary integral equation

(BIE) $$\frac{1}{2} u(x) + \int_\Gamma \big(n(y) \cdot \nabla_y \log |x - y|\big) u(y)\, ds(y) = f(x), \qquad x \in \Gamma.$$

- (BIE) and (BVP) are in a strong sense equivalent.
- (BIE) is appealing mathematically (2^{nd} kind Fredholm equation).

BIE AS FOUNDATION FOR NUMERICS

Suppose that we wish to numerically solve the integral equation

$$u(x) + \int_\Gamma K(x,y)u(y)\,ds(y) = f(x), \qquad x \in \Gamma.$$

We first discretize the contour into n points

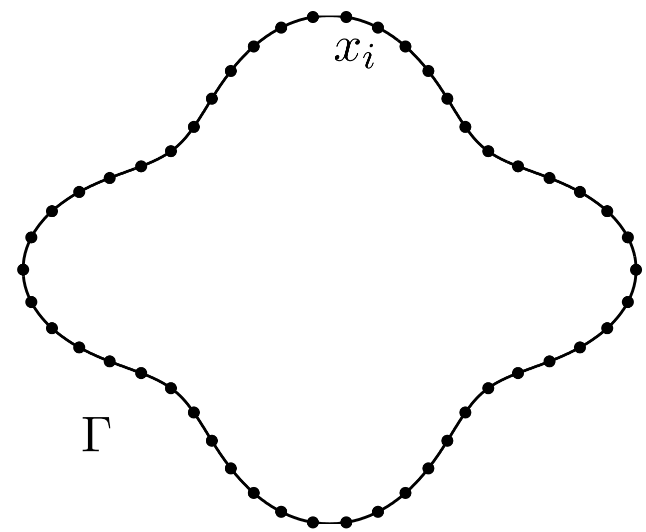
$$\Gamma \sim [x_1, \dots, x_n].$$

Then the operator

$$\int_\Gamma K(x,y)u(y)\,ds(y)$$

turns into a matrix A with entries (sort of)

$$A_{ij} = K(x_i, x_j), \qquad i, j = 1, \dots, n.$$



Since A is **dense**, it appears that

the cost for constructing A is $O(n^2)$ (with a large constant),

the cost for solving $(I + A)u = f$ is $O(n^3)$.

Fast solution of boundary integral equations

We let A denote the dense $n \times n$ matrix discretizing the operator

$$\int_\Gamma K(x,y)u(y)\,ds(y).$$

There exist $O(n \log^q n)$ algorithms $(q = 0, 1, 2)$ that evaluate the map

$$u \mapsto Au.$$

These include the Fast Multipole Method, Panel Clustering, multigrid, wavelets,...
Developed circa 1980 – 1985.

Using iterative (Krylov) methods, the equation

$$(I + A)u = f$$

can be solved using $O(\sqrt{\kappa} \cdot n \log^q n)$ operations, where κ is the condition number of $I + A$.

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 $$\mathbf{\Delta u} = \nabla p, \qquad \nabla \cdot \mathbf{u} = 0,$$

Helmholtz $$(-\Delta - k^2)u = f,$$

Schrödinger $$(-\Delta + V)\,\Psi = i\frac{\partial \Psi}{\partial t},$$

Maxwell $$\begin{cases} \nabla \cdot \mathbf{E} = \rho & \nabla \times \mathbf{E} = -\dfrac{\partial \mathbf{B}}{\partial t} \\ \nabla \cdot \mathbf{B} = 0 & \nabla \times \mathbf{B} = \mathbf{J} + \dfrac{\partial \mathbf{E}}{\partial t} \end{cases}$$

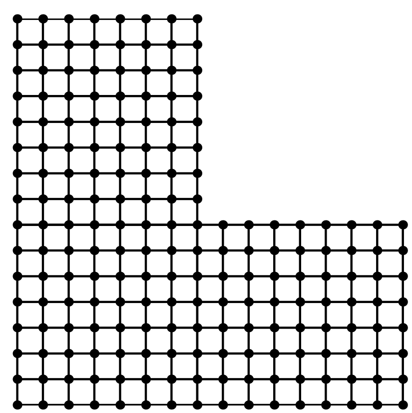
The lattice Laplace equation

$$\begin{cases} -\Delta \mathbf{u}(m) = 0, & m \in \Omega \subset \mathbb{Z}^2, \\ \mathbf{u}(m) = \mathbf{g}(m), & m \in \Gamma = \partial\Omega. \end{cases}$$

where $m = (m_1, m_2)$ is an integer index and

$$\begin{aligned} \Delta \mathbf{u}(m_1, m_2) =& \mathbf{u}(m_1 - 1, m_2) - 2\mathbf{u}(m_1, m_2) + \mathbf{u}(m_1 + 1, m_2) + \\ & \mathbf{u}(m_1, m_2 - 1) - 2\mathbf{u}(m_1, m_2) + \mathbf{u}(m_1, m_2 + 1), \end{aligned}$$

has a well-conditioned boundary formulation on Γ.



Spitzer (1969), M. and Rodin (2002), M. and Babuška (2002).

When fast boundary methods work, they are:

Accurate: The computational error should be roughly $\kappa\,\varepsilon$, where ε is the machine precision and κ is the actual condition number.

Optimally efficient: The CPU time requirement should be roughly proportional to N, the actual complexity of the problem.

Robust: The computation should be black-box with no need for fine-tuning, parameter-selection, etc. In particular, delicate mesh-requirements currently form a major obstacle.

The technique of solving boundary value problems (BVP) via fast iterative techniques for solving boundary integral equations (BIE) works best when

(1) there is no body force,

(2) the differential operator has constant coefficients,

(3) the BVP is linear,

(4) the BIE that is used is well-conditioned.

Over the last 20 years, much progress has been made in extending the technique to overcome the apparent obstreperousness of problems violating conditions (1), (2) and (3).

However, to overcome (4), we need a . . .

direct solver.

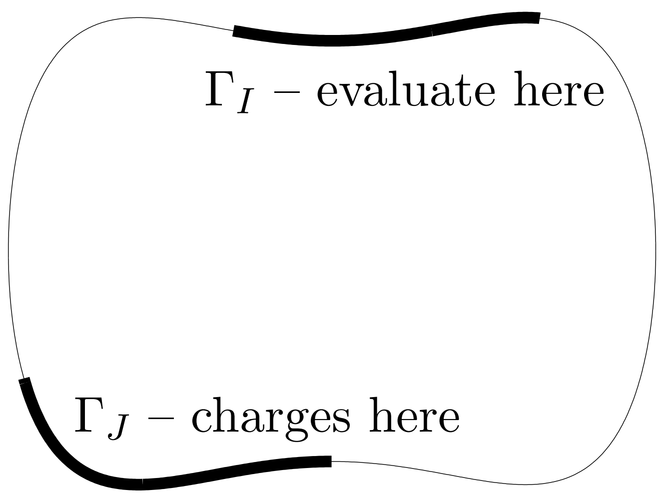
Benefits of a direct solver:

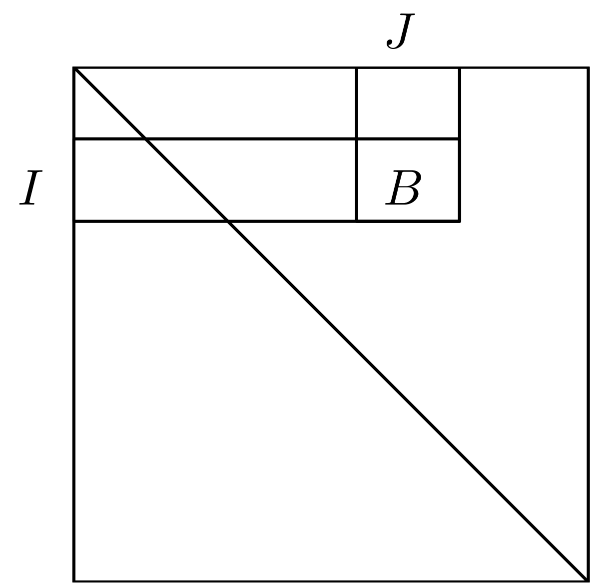
- **ill-conditioned problems**,
- multiple right-hand sides,
- up-dating a known solution to find the solution of another problem that is "close",
- constructing the SVD and other factorizations of the matrix.

There exist partial results in this direction:

- E. Michielssen, A. Boag and W.C. Chew (1996),
- G. Beylkin and N. Coult (1998),
- $\mathcal{H}$-matrix methods (ca. 1998), W. Hackbusch, M. Bebendorf, S. Börm, L. Grasedyck, S. Sauter.
- Y. Chen (2002).

For non-oscillatory problems, the off-diagonal blocks of the matrix A that discretizes the integral operator have low rank.





A contour ...   ...and the dense $n \times n$ matrix.

The sub-matrix B has elements

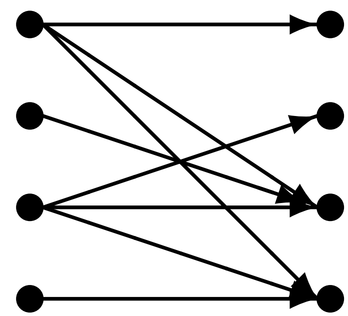
$$B_{ij} = K(x_i,\, x_j), \qquad i \in I,\ j \in J.$$

Compression of rank-deficient matrices

Let B be an $m \times n$ matrix with rank k.

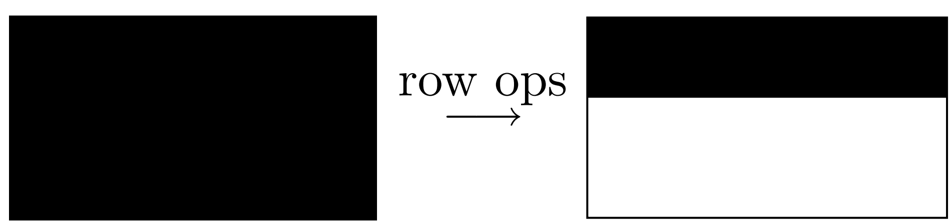
Using $O(mnk)$ arithmetic operations, we can find k rows of B that form a well-conditioned basis for the row-space of B. In other words, there exists a well-conditioned set of row operations with the following effect:

$$\begin{array}{c} \longrightarrow \\ \\ \longrightarrow \\ \end{array} \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix}$$



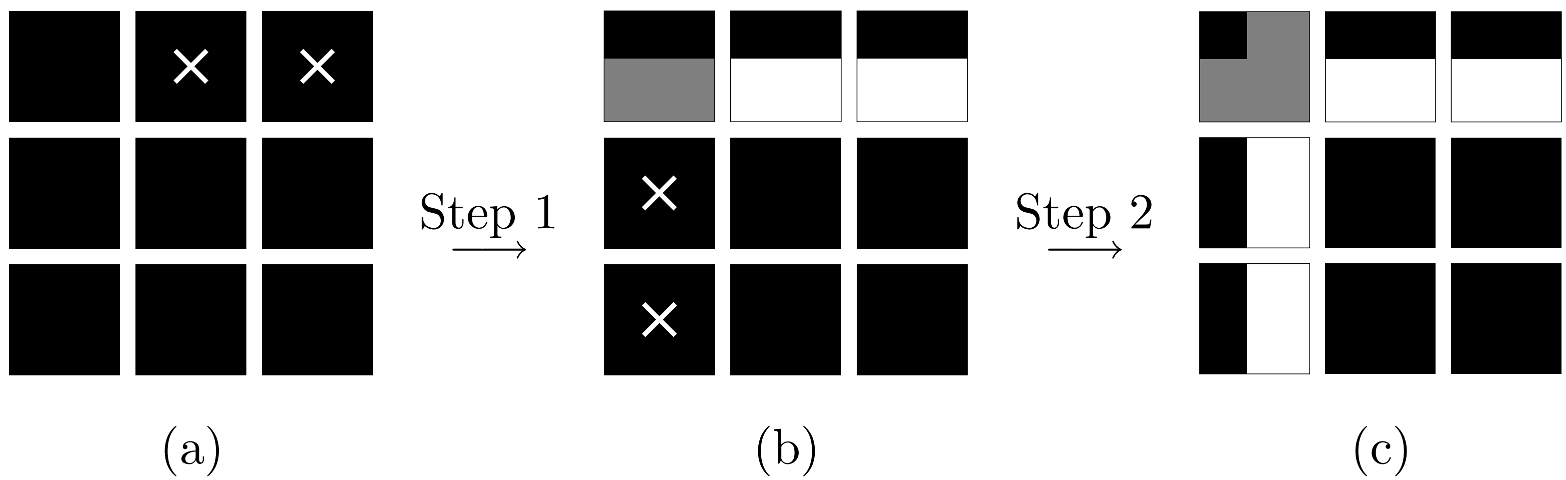
$$\begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

For large matrices, we illustrate such row operations as follows:



The results presented here follow from results by Gu and Eisenstat (1996).

<u>Single-level compression</u>

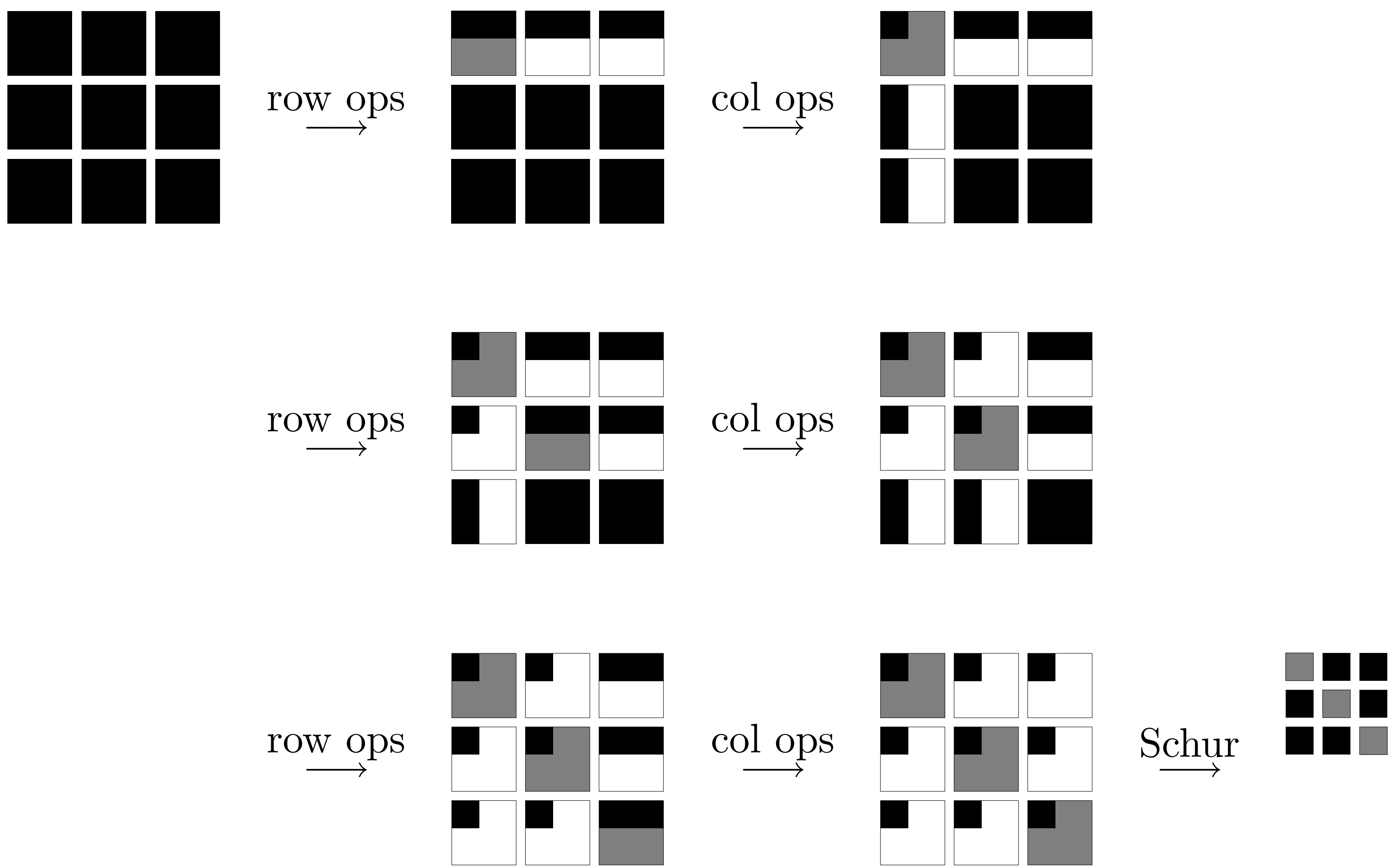


Step 1: Apply row operations in the top row of blocks to introduce zeros in the blocks marked with 'x' in (a).

Step 2: Apply column operations in the left column of blocks to introduce zeros in the blocks marked with 'x' in (b).

Important: The non-zero off-diagonal blocks that remain in (c) are *sub-matrices* of the off-diagonal blocks in (a).

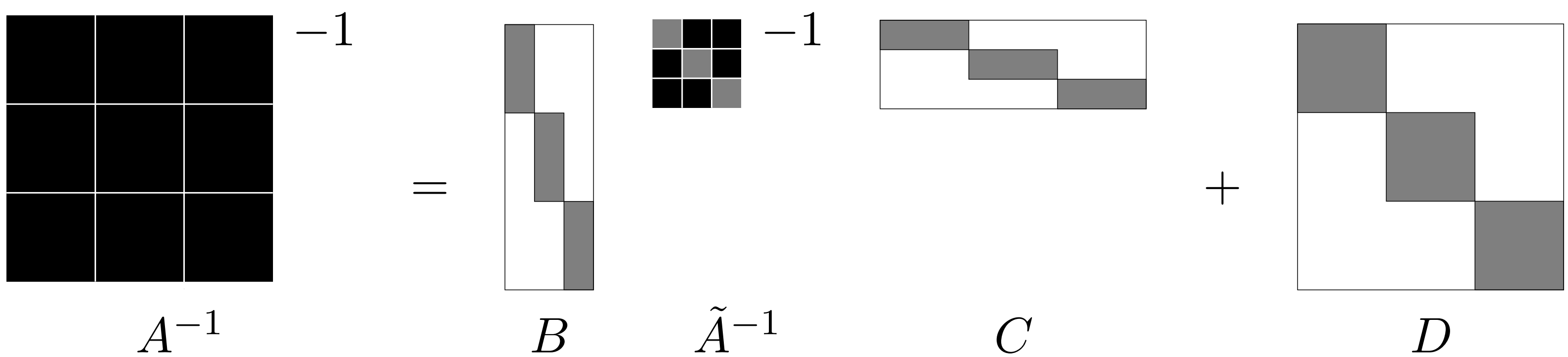
Single-level compression, more ...



The single-level compression can be represented as a matrix factorization:

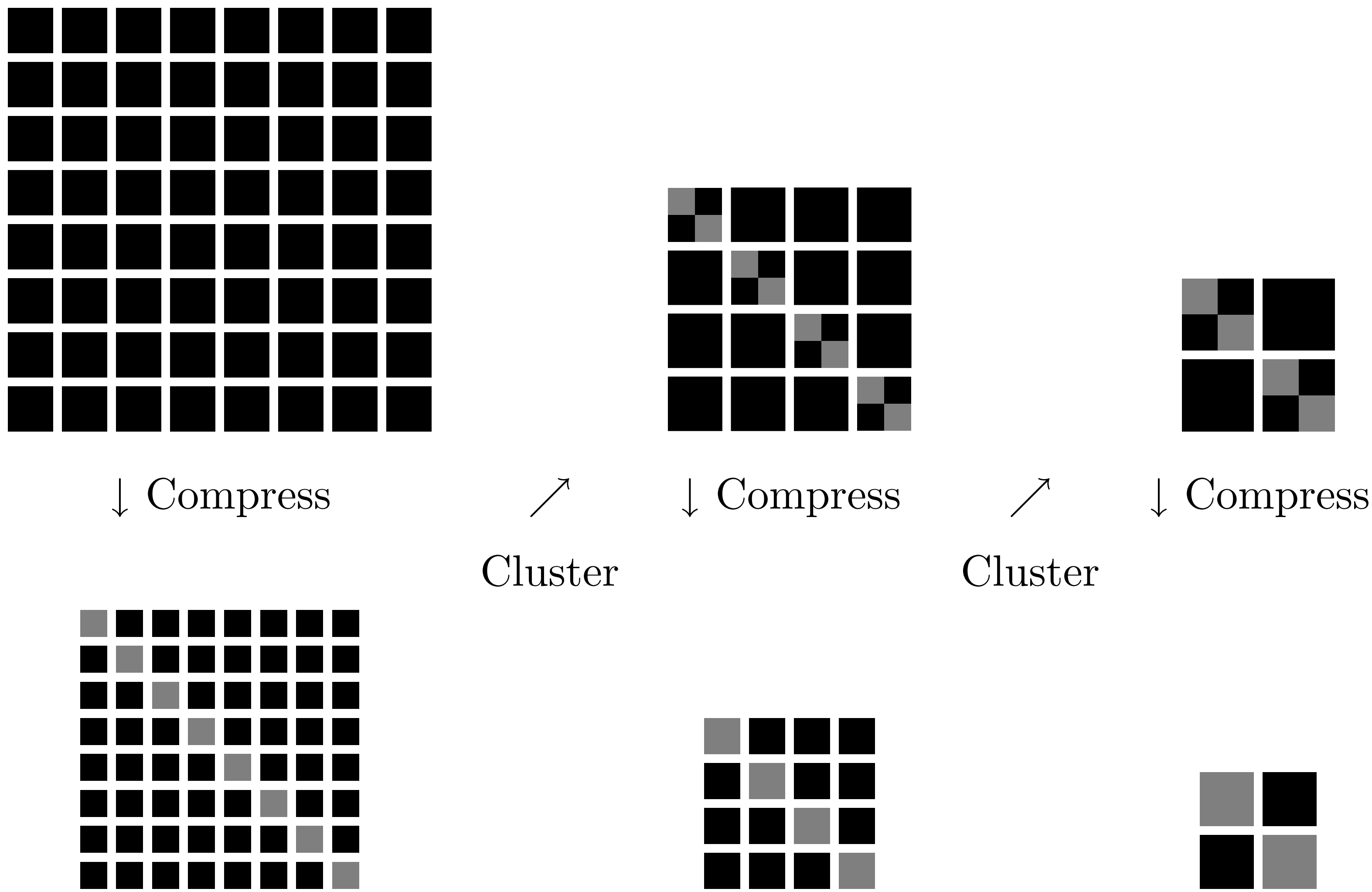
$$A^{-1} = B\,\tilde{A}^{-1}\,C + D,$$

where $\tilde{A}$ is the compressed matrix and B, C and D are block-diagonal.



Important: The off-diagonal blocks of $\tilde{A}$ are submatrices of the off-diagonal blocks of A.

Hierarchical Compression



Expressed in terms of matrix algebra, the hierarchical scheme is obtained by setting $A = A^{(1)}$ and telescoping the factorization

$$\left[A^{(1)}\right]^{-1} = B^{(1)} \left[\tilde{A}^{(1)}\right]^{-1} C^{(1)} + D^{(1)}. \tag{1}$$

That is, we set $A^{(2)} = \tilde{A}^{(1)}$ (with blocks clustered) and compress $A^{(2)}$;

$$\left[A^{(2)}\right]^{-1} = B^{(2)} \left[\tilde{A}^{(2)}\right]^{-1} C^{(2)} + D^{(2)}. \tag{2}$$

Combining (1) and (2), we obtain

$$\left[A^{(1)}\right]^{-1} = B^{(1)} \left(B^{(2)} \left[\tilde{A}^{(2)}\right]^{-1} C^{(2)} + D^{(2)}\right) C^{(1)} + D^{(1)}.$$

The process is continued recursively.

The output of the scheme is:

- A compressed block matrix $\tilde{A}$.
- A sequence of block-diagonal matrices $\{B^{(q)},\, C^{(q)},\, D^{(q)}\}_{q=1}^{Q}$.

Some comments:

So far, the scheme is generic — it depends only on rank considerations.

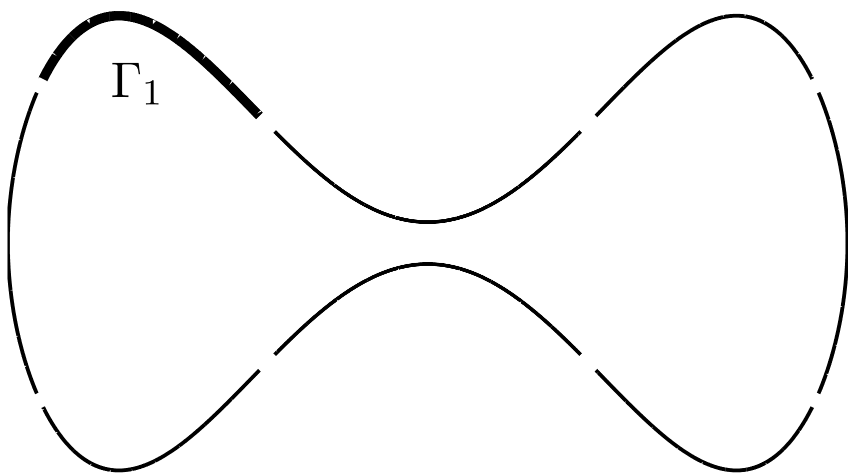
When applied to a matrix representing a non-oscillatory contour integral equation, the complexity of the scheme is $O(n^2 \log n)$.

Instead of the skeleton compression, we could have used the SVD.

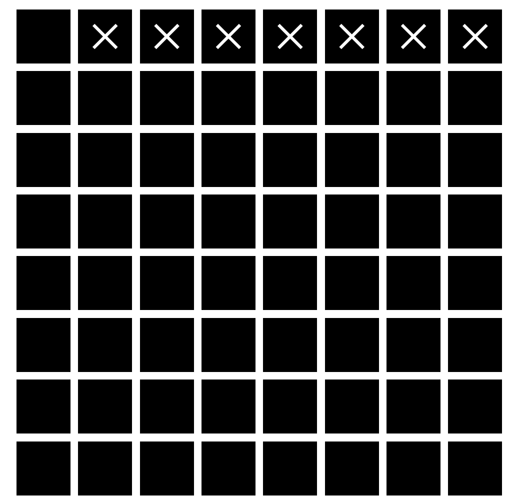
However, the fact that the skeletons preserve the structure of the matrix in a certain sense allows for an improvement that leads to $O(n \log^q n)$ complexity (for $q = 1, 2$).

A SCHEME TAILORED FOR BOUNDARY INTEGRAL EQUATIONS

The most expensive part of the computation is the compression of the off-diagonal blocks.

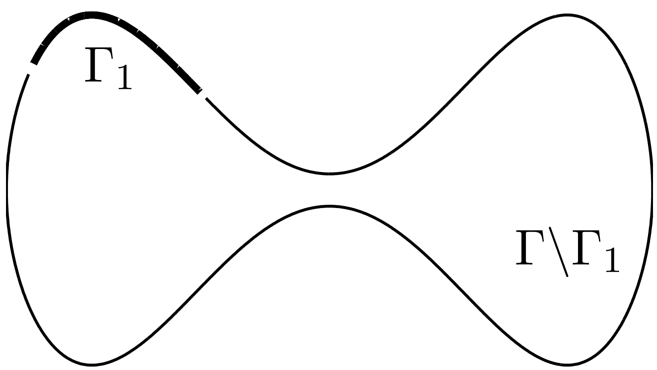


A segmented contour ...

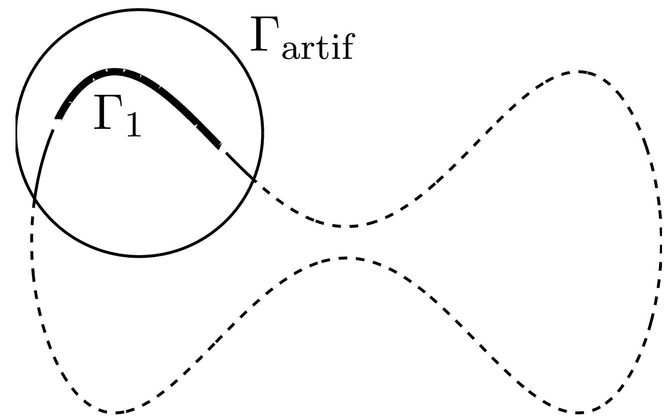


...and the corresponding matrix.

Since the off-diagonal blocks are not updated, we only need to determine the row and column operations and apply them to the diagonal block.
This is a local operation!



Instead of compressing the large matrix representing the interaction between Γ_1 and all of the rest of the contour...



... it is sufficient to compress only the interaction between Γ_1 and the artificial contour Γ_{artif}.

Caveat emptor:

One step in the scheme relies on the following matrix factorization (which is a version of the Schur complement):

$$\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & 0 \\ X_{12} & 0 & X_{33} \end{bmatrix}^{-1} =$$

$$\begin{bmatrix} I & 0 \\ -X_{22}^{-1}X_{21} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} X_{11} - X_{12}X_{22}^{-1}X_{21} & X_{13} \\ X_{31} & X_{33} \end{bmatrix}^{-1} \begin{bmatrix} I & -X_{12}X_{22}^{-1} & 0 \\ 0 & 0 & I \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & X_{22}^{-1} & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

For this relation to be useful, the quantity

$$||X_{22}^{-1}||_2$$

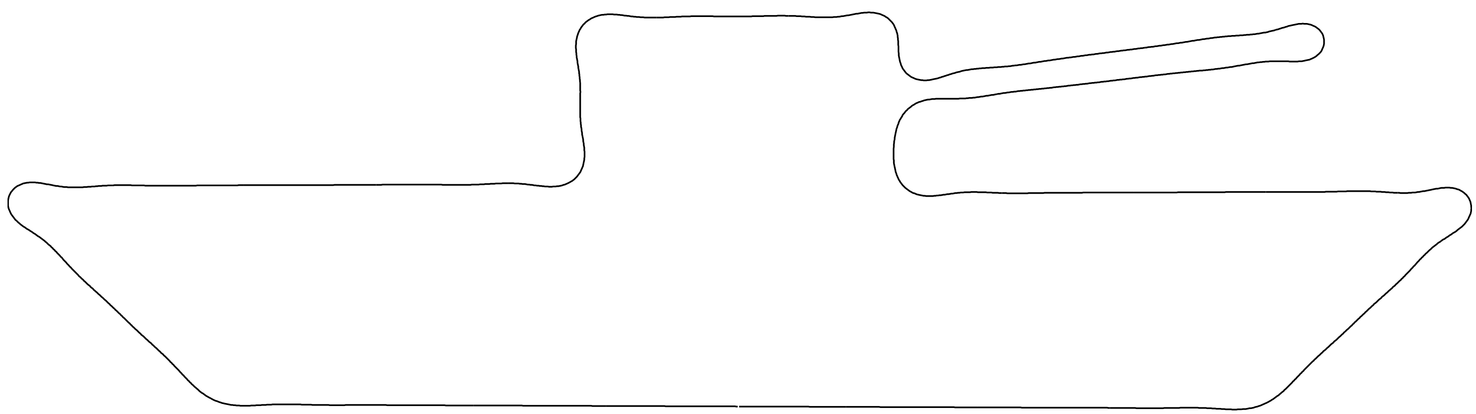
must not be excessively large. Extensive numerical experiments indicate that in the context of boundary integral equations it is not. However, this has not been proved.

Numerical examples

The algorithm was implemented in Matlab (using mex-programs for the skeletonization).

The experiments were run on a Pentium IV with a 2.8Ghz processor and 512 Mb of RAM.

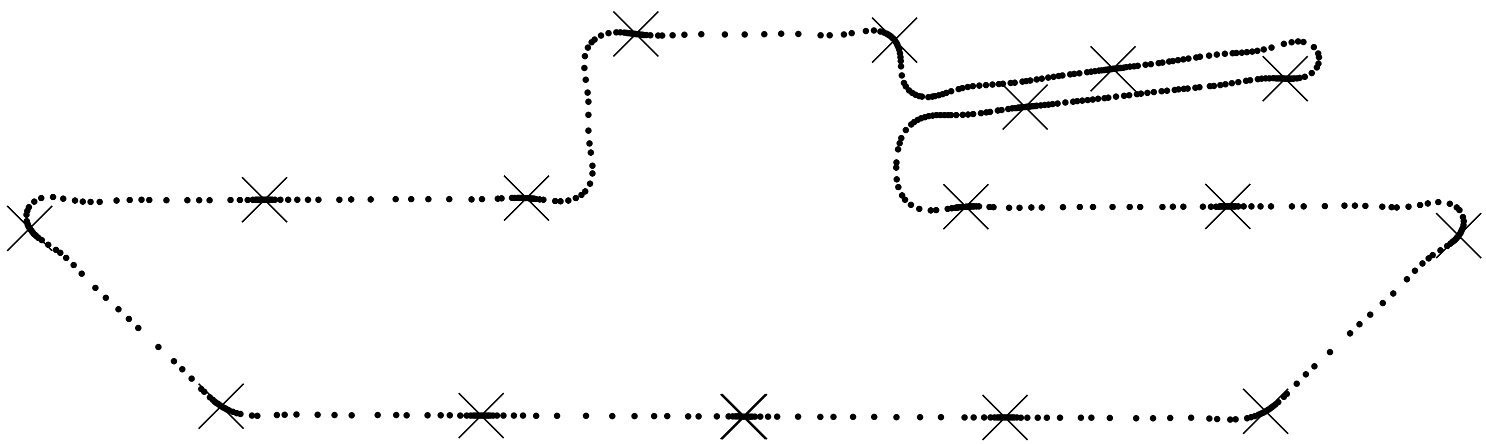
Example 1



A smooth contour. Its length is roughly 15 and its horizontal width is 2.

N_{start}	N_{final}	t_{tot}	t_{solve}	E_{actual}	E_{res}	E_{pot}	c_{top}	σ_{min}	M
800	335	1.2	0.01	4.0e-09	3.4e-9	5.6e-10	1.9e+2	1.5e-2	4250
1600	368	4.2	0.02	1.1e-09	2.4e-9	4.4e-10	3.6e+2	1.4e-2	6627
3200	369	8.4	0.05	—	2.7e-9	6.4e-10	3.6e+2	1.4e-2	8987
6400	369	9.0	0.07	—	3.3e-9	1.6e-10	8.1e+2	1.4e-2	13301
12800	369	13	0.12	—	2.6e-9	6.6e-10	1.6e+3	1.4e-2	21991
25600	370	16	0.24	—	3.8e-9	2.7e-10	1.6e+3	1.4e-2	39970
51200	371	35	0.49	—	2.9e-9	4.9e-10	2.0e+3	—	76346
102400	375	90	0.98	—	1.3e-9	—	1.2e+4	—	151571

Computational results for the double layer potential associated with an exterior Laplace Dirichlet problem.

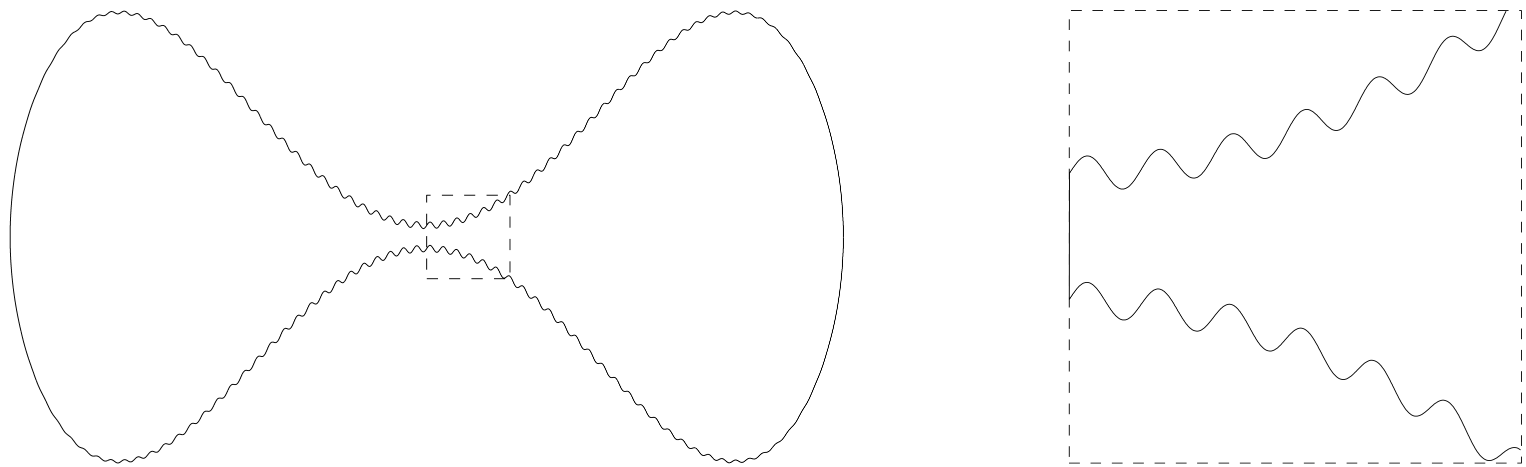


The points left after two rounds of compression. The crosses mark the boundary points between adjacent clusters.

k	N_{start}	N_{final}	t_{tot}	t_{solve}	E_{res}	E_{pot}	$\sigma_{\min}$	M
21	800	435	1.5e+01	3.3e-02	9.7e-08	7.1e-07	6.5e-01	12758
40	1600	550	3.0e+01	6.7e-02	6.2e-08	4.0e-08	8.0e-01	25372
79	3200	683	5.3e+01	1.2e-01	5.3e-08	3.8e-08	3.4e-01	44993
158	6400	870	9.2e+01	2.0e-01	3.9e-08	2.9e-08	3.4e-01	81679
316	12800	1179	1.8e+02	3.9e-01	2.3e-08	2.0e-08	3.4e-01	160493
632	25600	1753	4.3e+02	7.5e+00	1.7e-08	1.4e-08	3.3e-01	350984
1264	51200	2864	(1.5e+03)	(2.3e+02)	9.5e-09	—	—	835847

Computational results for an exterior Helmholtz Dirichlet problem discretized with 10^{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}).

Example 2



(a) (b)

(a) A rippled contour. (b) A close-up of the area marked by a dashed rectangle in (a). The horizontal axis of the contour has length 1 and the number of ripples change between the different experiments to keep a constant ratio of 80 discretization nodes per wavelength.

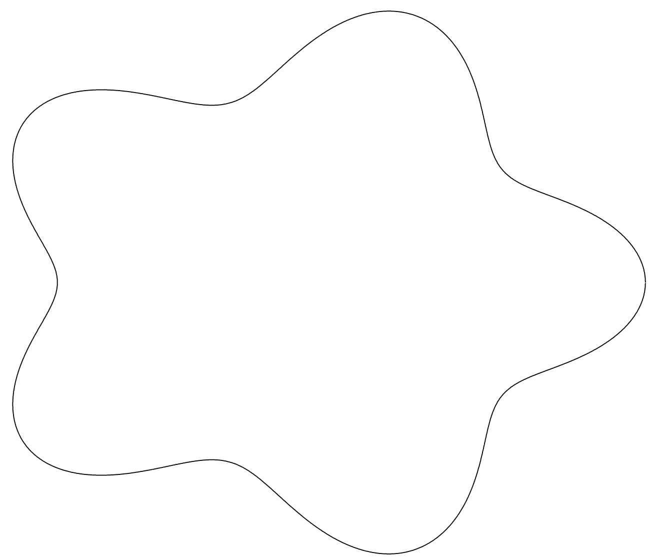
N_{start}	N_{final}	t_{tot}	t_{solve}	E_{actual}	E_{res}	E_{pot}	σ_{min}	M
400	160	2.4e-01	4.6e-03	2.3e-09	2.0e-09	1.2e-09	4.0e-02	954
800	214	4.7e-01	8.9e-03	2.3e-09	2.5e-09	2.8e-10	3.1e-02	2110
1600	286	7.5e+00	2.6e-02	1.9e-09	2.1e-09	9.8e-11	2.2e-02	4710
3200	361	1.1e+01	3.7e-02	—	1.4e-09	1.8e-10	1.8e-02	9781
6400	437	1.5e+01	7.2e-02	—	2.0e-09	1.3e-10	1.5e-02	20484
12800	508	2.1e+01	1.5e-01	—	1.6e-09	9.2e-11	1.4e-02	42307
25600	559	3.7e+01	2.9e-01	—	2.0e-09	1.3e-10	1.3e-02	86481
51200	599	8.0e+01	6.1e-01	—	1.8e-09	2.8e-10	—	177442
102400	634	1.9e+02	1.2e+00	—	1.4e-09	—	—	365495

Computational results for the double layer potential associated with an exterior Laplace Dirichlet problem on the rippled contour.

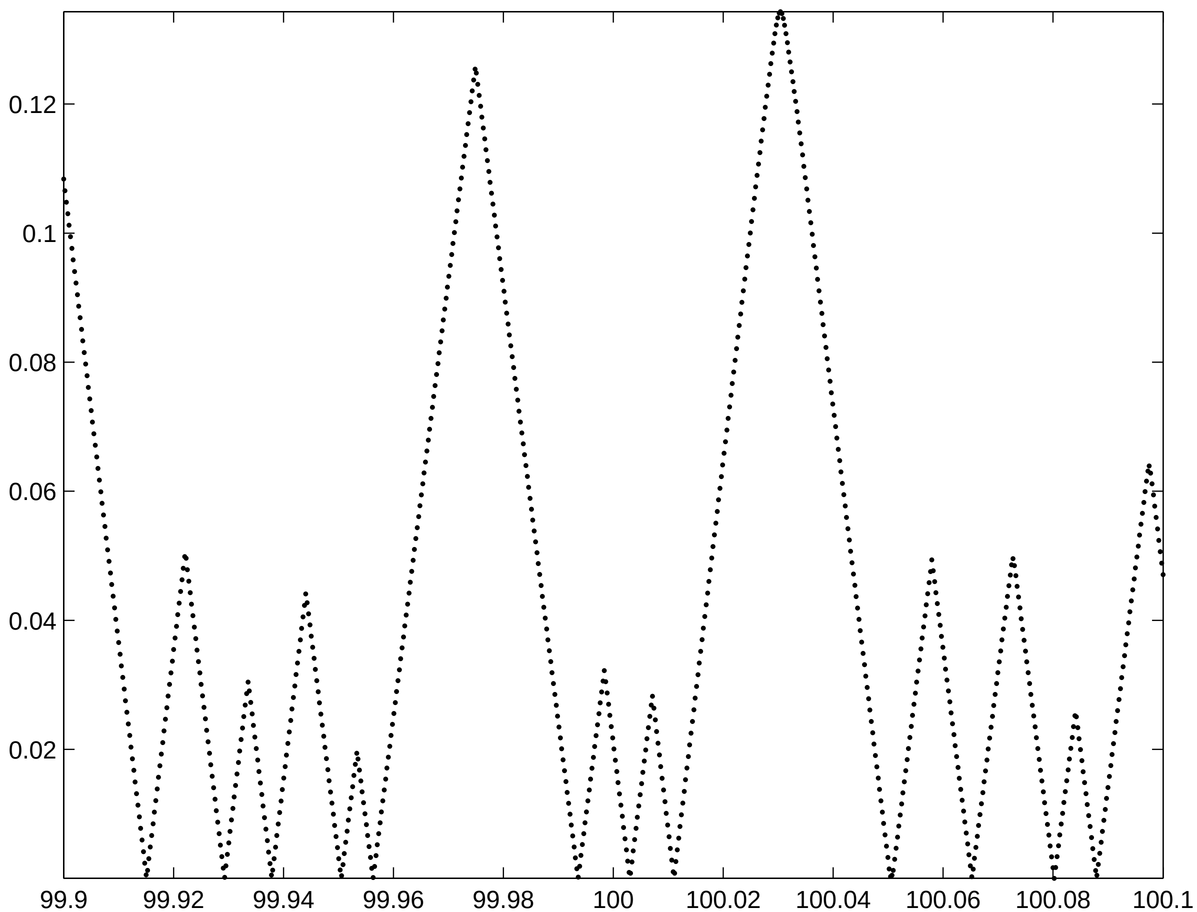
k	N_{start}	N_{final}	t_{tot}	t_{solve}	E_{res}	E_{pot}	$\sigma_{\min}$	M
7	400	224	2.9e+00	9.0e-03	6.9e-08	9.4e-07	7.9e-01	3241
15	800	320	7.7e+00	1.9e-02	7.4e-08	1.2e-07	7.9e-01	8233
29	1600	470	2.1e+01	4.6e-02	6.7e-08	8.1e-08	7.8e-01	20469
58	3200	704	6.1e+01	1.1e-01	5.2e-08	6.4e-08	8.0e-01	49854
115	6400	1122	1.4e+02	2.9e-01	4.8e-08	7.5e-08	8.0e-01	126576
230	12800	1900	(4.7e+02)	(2.5e+01)	5.5e-08	7.5e-08	8.0e-01	341054
461	25600	3398	—	—	—	—	—	983061

Computational results for an exterior Helmholtz Dirichlet problem on the rippled contour. The Helmholtz parameter k was chosen to keep the number of discretization points per wavelength constant at roughly 55 points per wavelength (resulting in a quadrature error about 10^{-12}).

Example 3



A smooth pentagram. Its diameter is 2.5 and its length is roughly 8.3.

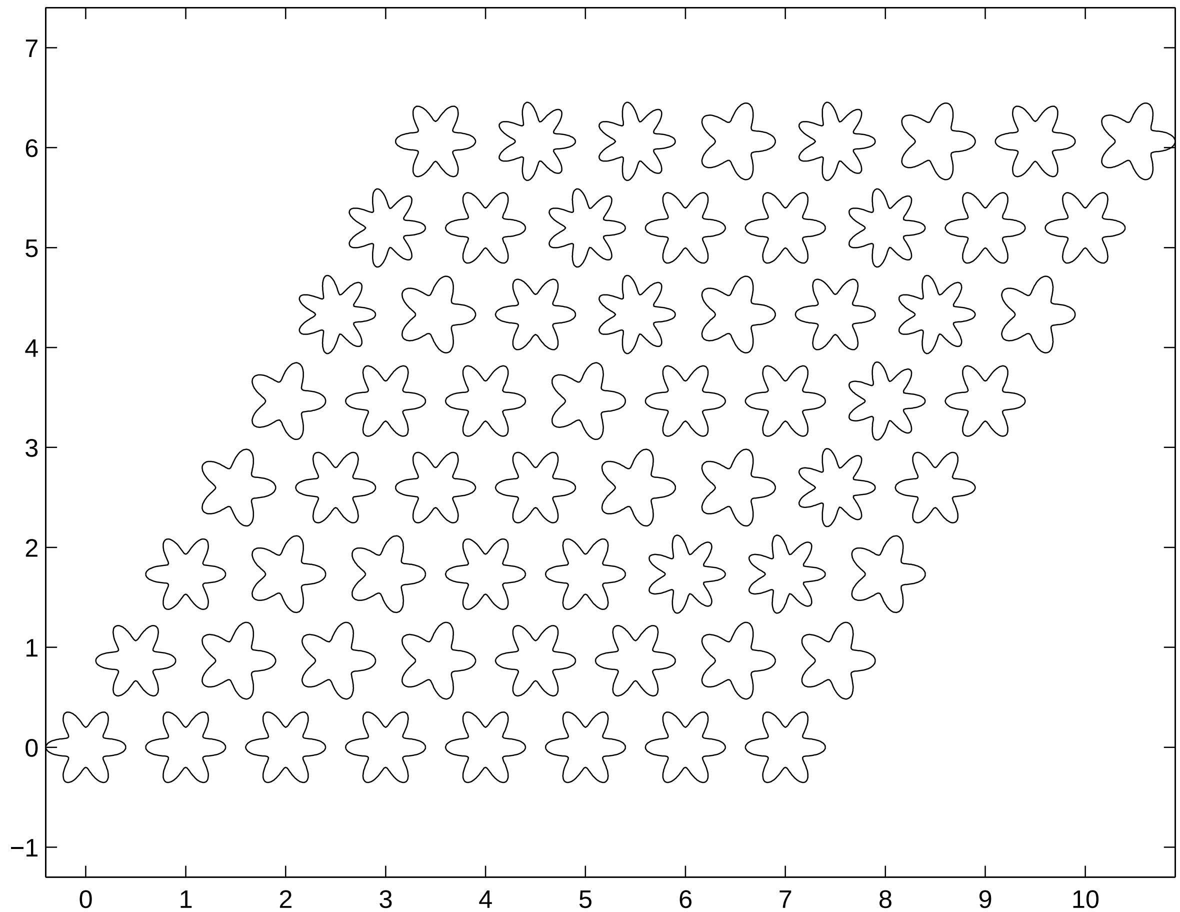


Plot of $\sigma_{\min}$ versus k for an interior Helmholtz problem on the smooth pentagram. The values shown were computed using a matrix of size $N = 6400$. Each point in the graph required about $60s$ of CPU time.

j	p_j	n_j	γ_j	t_j	$\|\|C^{(j)}\|\|_\infty$	$\|\|B^{(j)}\|\|_\infty$	$\|\|D^{(j)}\|\|_\infty$
1	128	50.00	0.76	15.50	1.12e+00	1.12e+00	4.20e-02
2	64	76.00	0.59	14.32	3.27e+01	3.27e+01	1.75e+00
3	32	89.72	0.60	8.94	1.63e+01	1.62e+01	9.28e-01
4	16	107.00	0.64	6.27	9.09e+00	9.17e+00	2.41e+00
5	8	138.00	0.72	5.97	7.32e+00	7.31e+00	3.64e+00
6	4	199.50	0.80	7.76	3.22e+00	3.23e+00	3.86e+00

Interior Helmholtz Dirichlet problem on a smooth pentagram for the case $N = 6\,400$, $k = 100.011027569\cdots$ and $\sigma_{\min} = 0.00001366\cdots$. For each level j, the table shows the number of clusters p_j on that level, the average size of a cluster n_j, the compression ratio γ_j, the time required for the factorization t_j and the size of the matrices $B^{(j)}$, $C^{(j)}$ and $D^{(j)}$ in the maximum norm. For this computation, $E_{\rm res} = 2.8 \cdot 10^{-10}$ and $E_{\rm pot} = 3.3 \cdot 10^{-5}$.

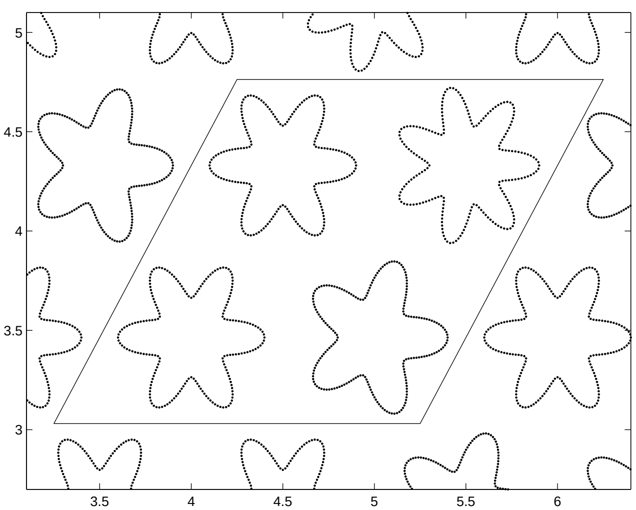
Example 4



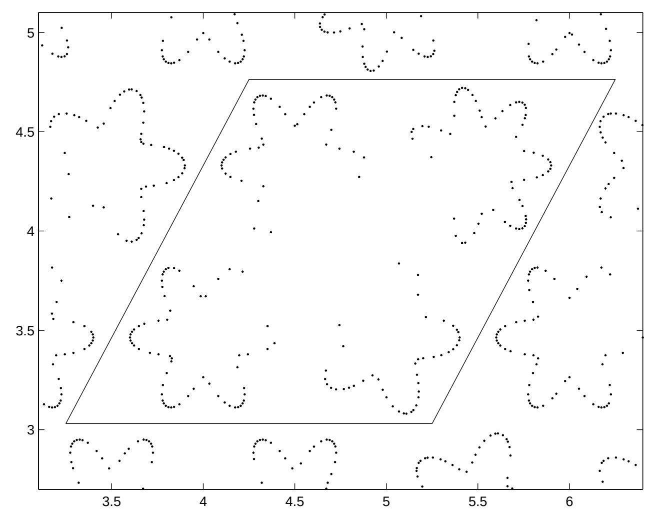
Contour:	t_{tot}	N_{start}	N_{final}	M
Rippled dumb-bell	37s	25 600	559	86Mb
Star-fish lattice	172s	25 600	1202	210Mb

Test results for two experiments concerning the matrix obtained by discretizing a double layer Laplace Dirichlet problem.

For the lattice problem, the computational complexity turns out to be $O(N^{3/2})$.



(a)



(b)

Fig. (a) shows a close-up of the star-fish lattice. Fig. (b) shows the nodes remaining after the interaction between the cluster formed by the points inside the parallelogram and the remainder of the contour has been compressed.

Summary

We have presented an $O(n \log n)$ direct solver for contour integral equations with non-oscillatory (or moderately oscillatory) kernels.

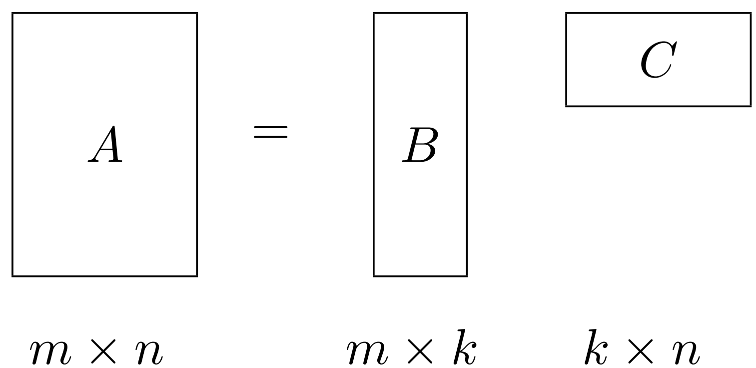
Work in progress:

- Applications of the scheme.
- Computing standard factorizations (SVD) of a dense matrix.
- Integral equations defined on surfaces rather than curves.
- Highly oscillatory problems.

<u>Mini-lecture on factorizations of low-rank matrices</u>

Suppose that A is an $m \times n$ matrix of rank k.

By definition, there exist matrices B and C such that

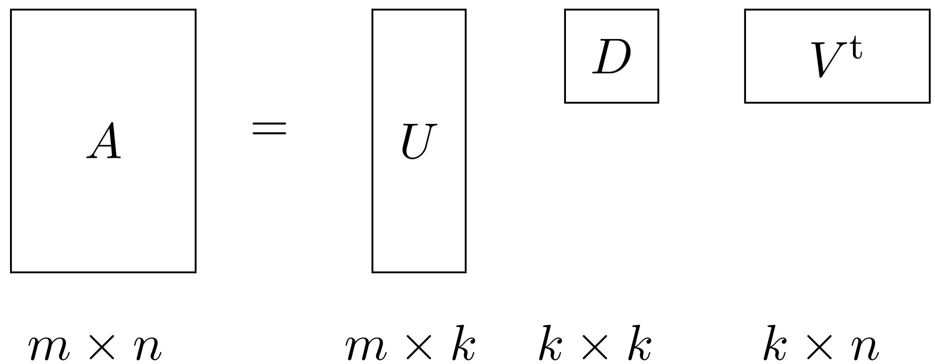


Such decompositions are useful for:

- storing A,
- applying A cheaply to a vector or a matrix,
- gaining heuristic understanding.

Not all factorizations are created equal.

The least equal is the Singular Value Decomposition (SVD),



- U and V have orthonormal columns, *i.e.* $U^{\mathrm{t}}U = V^{\mathrm{t}}V = I_k$,
- D is diagonal,
- D has (by definition) the same spectral properties as A,
- if A does not have rank k, then the rank-k SVD is the optimal rank-k approximation.

Defying the many advantages of the SVD, let us try something different.

If the top left $k \times k$ submatrix A_{11} of A is non-singular, then

$$\underset{m \times n}{\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}} = \underset{m \times k}{\begin{bmatrix} I \\ A_{21}A_{11}^{-1} \end{bmatrix}} \underset{k \times k}{\begin{bmatrix} A_{11} \end{bmatrix}} \underset{k \times n}{\begin{bmatrix} I & A_{11}^{-1}A_{12} \end{bmatrix}}$$

This "SVD-like" decomposition preserves the geometry of A in the sense that the middle factor is a *sub-matrix* of A.

However, it is useless if A_{11} is singular or ill-conditioned.

To overcome the potential problems of ill-conditioning, let us consider a pivoted factorization:

$$\underset{m \times n}{P_{\mathrm{L}}\, A\, P_{\mathrm{R}}} = \underset{m \times k}{\begin{bmatrix} I \\ S \end{bmatrix}} \; \underset{k \times k}{\tilde{A}} \; \underset{k \times n}{\begin{bmatrix} I & T \end{bmatrix}}$$

where

- P_{L} and P_{R} are permutation matrices,
- $\tilde{A}$ is a *sub-matrix* of $P_{\mathrm{L}} A P_{\mathrm{R}}$.

$$(1)\qquad \underset{m\times n}{\boxed{P_{\mathrm{L}}\,A\,P_{\mathrm{R}}}} \;=\; \underset{m\times k}{\left[\begin{array}{c} I \\ \hline S \end{array}\right]} \;\; \underset{k\times k}{\boxed{\tilde{A}}} \;\; \underset{k\times n}{\left[\begin{array}{c|c} I & T \end{array}\right]}$$

There exists an algorithm that computes a factorization of the form (1) such that

- S and T are moderate in size.
- The spectrum of $\tilde{A}$ approximates the spectrum of A.

The algorithm requires at most $O(m\,n\,\min(m,n))$ arithmetic operations; but typically requires only $O(mnk)$.

The statement relies on results by M. Gu and S.C. Eisenstat (1996).

Comparison

SVD

$$A = U \; D \; V^{\mathrm{t}}$$

- U and V are unitary
- $\sigma(D) = \sigma(A)$
- mixes coordinates
-

"Skeletonization"

$$A' = \begin{bmatrix} I \\ S \end{bmatrix} \; \tilde{A} \; \begin{bmatrix} I & T \end{bmatrix}$$

- S and T are small but not zero
- $\sigma(\tilde{A}) \sim \sigma(A)$
- preserves the geometry of A
- cheaper to apply to a vector

The skeleton-factorization has one more advantage:

It can be computed through a Gram-Schmidt process.

The complexity of this algorithm is $O(mnk)$ and the code is simple (but must be implemented very carefully).

This cheat *may in principle fail* but we have never seen it happen.

So what about the claim that given an $m \times n$ matrix B of rank k, there exists a well-conditioned $m \times m$ matrix L such that

$$\boxed{\;L\;}\;\boxed{\quad B \quad} = \begin{array}{|c|} \hline \hat{B} \\ \hline 0 \\ \hline \end{array}$$

where $\hat{B}$ is a $k \times m$ matrix consisting of k of the rows of B?

The $m \times n$ matrix B of rank k has the skeletonization

$$(1) \qquad P_{\rm L} B P_{\rm R} = \left[\begin{array}{c} I \\ S \end{array}\right] \tilde{B}\,[I\ T] \qquad \Leftrightarrow \qquad P_{\rm L} B = \left[\begin{array}{c} I \\ S \end{array}\right] \underbrace{\left[\tilde{B}\ \tilde{B}T\right] P_{\rm R}^{\rm t}}_{=:\hat{B}}.$$

The matrix $\hat{B}$ thus constructed is necessarily formed by k of the rows of B. Defining

$$L = \left[\begin{array}{cc} I & 0 \\ -S & I \end{array}\right] P_{\rm L},$$

we find that

$$LB = \left[\begin{array}{cc} I & 0 \\ -S & I \end{array}\right] P_{\rm L} B = \left[\begin{array}{cc} I & 0 \\ -S & I \end{array}\right] \left[\begin{array}{c} I \\ S \end{array}\right] \hat{B} = \left[\begin{array}{c} \hat{B} \\ 0 \end{array}\right].$$

NB: This is not how the row operations are implemented in practise.