Fast direct solvers by multilevel matrix compression

Kenneth L. Ho

Courant Institute of Mathematical Sciences and Program in Computational Biology
New York University, New York, NY, USA
ho@courant.nyu.edu

Introduction

Current state-of-the-art solvers for large dense structured linear systems, e.g., as arising from boundary integral discretizations of many partial differential equations in physics, are typically iterative, coupling Krylov methods with algorithms to apply the system matrix rapidly. Such techniques are "fast", often with $\mathcal{O}(kN)$ complexity, where k is the number of iterations required and N is the order of the system, and have enabled enormous breakthroughs in computational science and engineering; however, several severe shortfalls remain. Here, we present instead a fast direct solver that overcomes these limitations. The core algorithm compresses the system matrix via the multilevel application of the interpolative decomposition. The compressed representation allows efficient storage, fast matrix-vector multiplication, and fast matrix factorization and inverse application via embedding into a highly structured sparse matrix. For boundary integral equations, the solver typically has complexity $\mathcal{O}(N)$ in 2D and $\mathcal{O}(N^{3/2})$ in 3D.

Advantages of direct solvers
- Insensitivity to ill conditioning.
- Fast inverse applications.
- Efficient low-rank updates.

This is joint work with Leslie Greengard.

Low-rank structures

We assume that the system matrix can be block-partitioned such that each row or column block, with the diagonal portion deleted, is low-rank (Figure 1). This structure is common to many physical problems. As an example, consider Laplace's equation in 3D with Dirichlet boundary conditions:

$$\Delta u = 0 \quad \text{in } \Omega, \qquad u = f \quad \text{on } \partial\Omega.$$

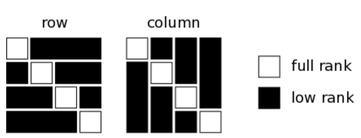

Figure 1: *Matrix rank structure.*

Potential theory suggests the double-layer representation

$$u(x) = \int_{\partial\Omega} \frac{\partial G}{\partial n_y}(x,y)\,\sigma(y)\,dS_y, \quad x \in \Omega,$$

where

$$G(x,y) = \frac{1}{4\pi|x-y|}$$

is the Green's function, n_y is the unit outer normal at $y \in \partial\Omega$, and σ is an unknown surface charge density. As $x \to \xi \in \partial\Omega$,

$$u(x) \to -\frac{1}{2}\sigma(\xi) + \int_{\partial\Omega} \frac{\partial G}{\partial n_y}(\xi,y)\,\sigma(y)\,dS_y,$$

so enforcing the boundary condition gives the second-kind integral equation

$$-\frac{1}{2}\sigma(\xi) + \int_{\partial\Omega} \frac{\partial G}{\partial n_y}(\xi,y)\,\sigma(y)\,dS_y = f(\xi),$$

or, upon discretization, the linear system $A_{ij}\sigma_j = f_i$. Sorted according to an octree ordering so that nearby discretization nodes are grouped together, the matrix A has precisely the rank structure shown in Figure 1 due to the smoothness of the far field.

Matrix compression

We compress each low-rank matrix block by using the interpolative decomposition (ID), resulting in a skeleton matrix of dimension $K \ll N$ (Figure 2). Since the cost of direct inversion is cubic in the matrix dimension, this provides a significant acceleration. If the matrix is endowed with a tree such that the low-rank structure holds at each level, e.g., via an octree, then we can further compress the matrix by repeating this procedure hierarchically (Figure 3). The multilevel compression algorithm computes a representation of the form

ID $A = BP$
- B is a submatrix of A.
- P is a projection matrix containing the identity.

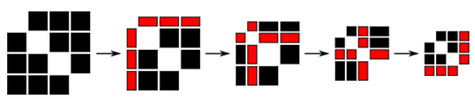

Figure 2: *Compression of low-rank matrix blocks.*

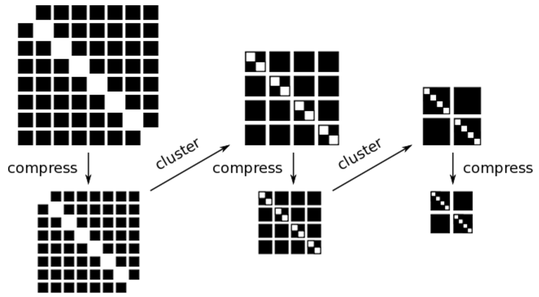

Figure 3: *Multilevel matrix compression.*

$$A = A_{ii} + L_i S_{ij} R_j,$$

where A_{ii} is the block-diagonal part of A; L_i and R_j are block-diagonal row and column interpolation matrices, consisting of ID projection matrices; and S_{ij} is the skeleton matrix, which itself is compressed recursively in the same form.

Observe that the compressed representation admits fast matrix-vector multiplication; thus, the current work may also be regarded as a kernel-independent fast multipole method (FMM).

Matrix inversion

To compute a compressed matrix inverse, we note that the compressed system $Ax = b$ can be written as

$$b_i = A_{ii}x_i + \sum_{j\neq i} A_{ij}x_j = A_{ii}x_i + L_i \sum_{j\neq i} S_{ij}R_j x_j,$$

which is equivalent to the expanded system

$$A_{ii}x_i + L_i y_i = b_i, \quad R_j x_j = z_j, \quad \sum_{j\neq i} S_{ij} z_j = y_j,$$

i.e.,

$$\begin{pmatrix} A & L & \\ R & & -I \\ & -I & S \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} b \\ 0 \\ 0 \end{pmatrix},$$

where the linear system with matrix S can itself be expanded in the same form. The resulting matrix is sparse and highly structured, and can be factored rapidly using standard sparse techniques. Once the factors have been computed, any given system can be solved at minimal cost.

Numerical results

As examples, we compressed the unscaled Laplace potential interaction matrices

$$A_{ij} = (1-\delta_{ij})\log r_{ij} \quad \text{in 2D}, \qquad A_{ij} = \frac{(1-\delta_{ij})}{r_{ij}} \quad \text{in 3D}$$

between surface points to an accuracy of $\epsilon = 10^{-6}$, and compared the application and solve times against using LAPACK and the FMM, where appropriate (Tables 1 and 2). Sparse matrix factorization and inverse application were performed with UMFPACK. All computations were run on a single core of a 2.66 GHz processor.

The data reflect the $\mathcal{O}(N)$ and $\mathcal{O}(N^{3/2})$ scalings in 2D and 3D, respectively. Furthermore, for the problem sizes considered, the breakeven point against using the FMM is quite small, on the order of fifty matrix applications, or about ten solves.

Conclusion

We have constructed a fast direct solver based on multilevel matrix compression that achieves optimal or near-optimal scalings for boundary integral equations in 2D and 3D. We expect that our method will have significant impact in the context of optimization and design, where the large costs of matrix compression and factorization can be amortized against the many systems to be solved.

Acknowledgments

We thank Zydrunas Gimbutas for the sparse inverse formulation, and Mark Tygert for the many helpful discussions. KLH acknowledges support from the NYU MacCracken and NSF IGERT (DGE 0333389) programs.

Table 1: *Data for $A_{ij} = (1-\delta_{ij})\log r_{ij}$ on the unit circle at $\epsilon = 10^{-6}$.*

N	k		t_{cm} (s)	t_{mv} (s)			t_{LU} (s)		t_{inv} (s)		storage (MB)	
	row	col		uncomp	comp	FMM	uncomp	comp	uncomp	comp	uncomp	comp
4096	77	77	0.24E+00	0.23E-01	0.10E-02	0.49E-01	0.49E+01	0.15E+00	0.58E-01	0.30E-02	134.22	2.59
8192	83	83	0.50E+00	0.96E-01	0.20E-02	0.87E-01	0.39E+02	0.28E+00	0.23E+00	0.50E-02	536.87	4.42
16384	87	87	0.98E+00	(0.38E+00)	0.60E-02	0.18E+00	(0.31E+03)	0.49E+00	(0.91E+00)	0.11E-01	2147.48	8.80
32768	94	93	0.20E+01	(0.15E+01)	0.12E-01	0.37E+00	(0.25E+04)	0.98E+00	(0.36E+01)	0.21E-01	8589.93	17.46
65536	99	100	0.42E+01	(0.61E+01)	0.25E-01	0.76E+00	(0.20E+05)	0.20E+01	(0.15E+02)	0.43E-01	34359.74	34.56
131072	102	102	0.79E+01	(0.25E+02)	0.44E-01	0.14E+01	(0.16E+06)	0.41E+01	(0.58E+02)	0.82E-01	137438.95	68.46

Table 2: *Data for $A_{ij} = (1-\delta_{ij})/r_{ij}$ on the unit sphere at $\epsilon = 10^{-6}$.*

N	k		t_{cm} (s)	t_{mv} (s)			t_{LU} (s)		t_{inv} (s)		storage (MB)	
	row	col		uncomp	comp	FMM	uncomp	comp	uncomp	comp	uncomp	comp
4096	1580	1583	0.47E+01	0.25E-01	0.80E-02	0.45E+00	0.49E+01	0.21E+01	0.55E-01	0.14E-01	134.22	38.50
8192	2294	2289	0.16E+02	0.12E+00	0.21E-01	0.91E+00	0.38E+02	0.68E+01	0.26E+00	0.37E-01	536.87	94.52
16384	3221	3226	0.40E+02	(0.49E+00)	0.44E-01	0.20E+01	(0.30E+03)	0.20E+02	(0.10E+01)	0.92E-01	2147.48	219.87
32768	4469	4477	0.11E+03	(0.20E+01)	0.99E-01	0.37E+01	(0.24E+04)	0.51E+02	(0.41E+01)	0.22E+00	8589.93	499.78
65536	6256	6234	0.31E+03	(0.79E+01)	0.21E+00	0.10E+02	(0.19E+05)	(0.15E+03)	(0.17E+02)	(0.47E+00)	34359.74	1132.28
131072	8682	8729	0.85E+03	(0.31E+02)	0.48E+00	0.18E+02	(0.15E+06)	(0.46E+03)	(0.66E+02)	(0.11E+01)	137438.95	2497.66