Fast direct inversion of boundary integral equation matrices.

Thomas Ben Thompson

I propose to study $O(n)$ direct methods for the inversion of the dense hierarchically low-rank matrices that arise in the discretization of boundary integral equations. Classical direct methods, like Gaussian elimination, solve the problem $Ax = b$, with $A \in \mathbb{R}^{nxn} \quad x, b \in \mathbb{R}^n$ via the inversion of the dense matrix, A, which requires $O(n^3)$ time and $O(n^2)$ space. Iterative methods reduce the time cost to $O(kV)$, where k is the number of iterations and V is the cost of a matrix-vector product. Generally $V = O(n^2)$, but, as I will discuss, the special matrices I am studying can be approximated such that $V = O(n)$. The difficulty with iterative methods is two-fold: first, the rate of convergence is dependent on the the ratio of largest to smallest singular values (the condition number), while the time cost for classical direct methods is independent of the matrix; second, for each new vector b, the entire solution process is repeated, whereas a direct method inverts the matrix A independently of the b vector. Recently, research has produces direct methods that are able to take advantage of special matrix structure, specifically a hierarchically low-rank property, to reduce the time cost to $O(n^{3/2})$ or $O(n)$ [5, 4, 1]. An off-diagonal block, $B \in \mathbb{R}^{mxm}$ of a hierarchically low rank matrix has rank $o(m)$. Despite this, the matrix A, taken in full, has complete rank n.

In my research, I solve the partial differential equations describing Earth's crustal deformation in order to estimate seismic hazard and understand the mechanics of earthquakes. However, one difficulty that slows progress is that complex geometries are hard to represent using commonly used methods like the finite element method. To avoid this problem, I translate the partial differential equations into a convolution over the boundary of the domain – a boundary integral equation [2]. Because the integral equation explicitly consider the boundaries of the domain instead of the volume, treating faults and other surfaces within the Earth becomes much easier.

$$\int_{\delta\Omega} U^*(x,y)t(y)dy = u(x) + \int_{\delta\Omega} T^*(x,y)u(y)dy \tag{1}$$

where Ω is the region of interest, $U^*(x,y) = O(\frac{1}{\|x-y\|})$ is an operator that represents the displacement at x given a force at y, $T^*(x,y) = O(\frac{1}{\|x-y\|^2})$ is an operator that represents the displacement at x given a displacement at y, $t(x)$ are the forces and $u(x)$ are the displacements in the region. Given knowledge of either $u(x)$ or $t(x)$, the integral equation can be solved for the other. Enforcing the integral at n points along the boundary and using quadrature methods to to representing the integrals as the sum of n carefully placed point evaluations, the integral equation is transformed into a dense linear algebra problem. The matrices produced by this method have the hierarchically low-rank property because the kernel functions $U^*(x,y)$ and $T^*(x,y)$ decay rapidly when x and y move apart. This is simple to understand physically: for large $\|x-y\|$, a small perturbation in x is negligible; as a result, in terms of the influence on y, two displacements at $x+\epsilon$ and x appear almost the same as one displacement of twice the magnitude at x.

I currently solve this problem using an iterative methodology where matrix vector products are computed in $O(n)$ time using a method called the Fast Multipole Method (FMM), which takes advantage of the hierarchically low-rank property [3]. The FMM decomposes space into a heirarchical tree, and approximates far-field interactions (matrix blocks) via a low rank decomposition, $B = LTM$, with $B \in R^{pxq}$ as the block of the original matrix and $L \in \mathbb{R}^{pxk}, T \in \mathbb{R}^{kxk}, M \in \mathbb{R}^{kxq}$ as the low-rank decomposition. The near-field interactions/blocks are computed directly, because they are not low-rank. An analysis of this algorithm demonstrates that it requires $O(n)$ time and space to compute a matrix vector product [3].

However, as mentioned above, iterative methods have significant disadvantages for repeated solution with different right hand sides. This is important for time-dependent simulation where at each time step a linear system must be solved. If t time steps are taken, the time cost for an iterative method

based on the FMM is $C_{\mathrm{m}}tkn$. For a typical simulation of an earthquake, $t = O(10^5)$ and for sufficient accuracy with typical linear solvers, $k = O(100)$. In contrast, for a $O(n)$ direct method, the cost would be $C_{\mathrm{m}}tn + C_i n$, because the matrix must be inverted once and then multiplied once at each time step (I include the constants to show that C_m would be the same for both cases). For $t \gtrsim 50$, previous work has found the direct inversion approach to be beneficial (CITE).

I will focus on the algorithm presented in [1] as it achieves linear time while remaining relatively conceptually simple. The algorithm has two primary steps: first, create an extended sparse matrix ($O(n)$ entries) from the dense matrix by replacing dense blocks with their FMM low-rank approximation; second, apply a modified Gaussian elimination process that approximates any fill-in (setting a zero entry to a non-zero value) by piping the value through the low-rank approximation. I will make these ideas more concrete.

Representing the dense matrix by an extended sparse matrix is straightforward given the low-rank decompositions. Replace each low-rank block $B = LTM$, by an extended block. Suppose B represents the interaction of the index set I influencing the index set J. Hence the extended block looks like $\widetilde{B} = \begin{pmatrix} 0 & 0 & L \\ M & -I & 0 \\ 0 & T & -I \end{pmatrix}$ with an extended vector of unknowns ($2k$ new unknowns): $\begin{pmatrix} x_J \\ y \\ z \end{pmatrix}$. This modification represents the step-by-step application of the low-rank representation:

$$y = Mx_J \quad z = Ty \quad Lz + \text{contribution of other blocks} = b_I \tag{2}$$

Because L, T, M are all sparse, the block, as a whole is sparse.

Looking at the above $\widetilde{B}$ matrix, there four possible cases for fill-in during elimination. For example, a fill-in in the B_{11} block represents an interaction between x_J and x_I, which requires increasing the respective entry in the M matrix. I skip over the details here and the very important detail of how to order the matrices so that the entries that are modified have not yet been eliminated. But, assuming this is possible, each elimination step adds one row to one other row. Because, the matrix is sparse, each row has $O(1)$ entries and produces $O(1)$ fill-ins. With $O(n)$ elimination steps total, the time cost is linear.

Two interesting open problems in the study of hierarchically low-rank direct inversion methods:

- I know of no literature that has attempted to parallelize or distribute this algorithm or any similar algorithm. Distribution may require substantial modifications. For example, an efficient implementation might involve local inversion of a given node's sub-block of the global matrix. These inverted sub-blocks could be merged in a suitable fashion using theorems on blockwise matrix inversion.
- The spatial decomposition imposed by the algorithm described in [1] is restrictive – an octree with uniform depth. Broadening the range of allowed spatial decompositions would help apply the method to new problems, including the integral equations I use.

I will implement the algorithm for a simple case and see if I can make any progress on either of these problems. If I fail, I will look for ways to speed up the algorithm, for example by improving cache locality.

References

[1] Sivaram Ambikasaran and Eric Darve. The inverse fast multipole method. *arXiv preprint*, pages 1–25, 2014.

[2] TA Cruse. Numerical solutions in three dimensional elastostatics. *International Journal of Solids and Structures*, 5:1259–1274, 1969.

[3] L Greengard and V Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 348:325–348, 1987.

[4] Leslie Greengard, Denis Gueyffier, Per-Gunnar Martinsson, and Vladimir Rokhlin. *Fast direct solvers for integral equations in complex three-dimensional domains*, volume 18. May 2009.

[5] P.G. Martinsson and V. Rokhlin. A fast direct solver for boundary integral equations in two dimensions. *Journal of Computational Physics*, 205(1):1–23, May 2005.