Boundary Element Methods

Lecture 6:
Fast Methods for Integral Equations

November 8, 2017

Outline

- Cost of Boundary Element Methods

- Circulant and Toeplitz matrices and the Fast Fourier Transform (FFT)

- Fast Multipole Method

How expensive are integral equation methods?

Let the number of unknowns ($\propto$ #elements) be n.

Let the typical element size be h.

Then $n = \mathcal{O}(1/h^{d-1})$, where $d = 2, 3$ is the problem dimension.

Naïve approach

Assemble entire dense matrix - n^2 entries.

Invert matrix by Gaussian elimination - $\mathcal{O}(n^3)$ operations.

Less naïve approach

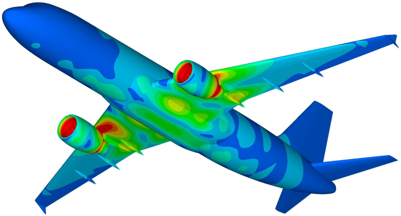
Iterative method - $\mathcal{O}(n^2)$ operations if well-conditioned (iteration count $m \ll n$), and using standard matrix-vector product.

Main bottleneck: dense matrix storage.

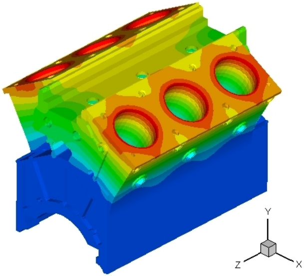
Computational cost - some example numbers

Typical industrial application: $>$ ($\gg$)100,000 unknowns.

More than 160 Gb to store matrix (my laptop has 16 Gb).



Noise control on Airbus



Heat conduction in engine

[Images from "FastBEM"]

Fast algorithms

- Fast algorithms rely on iterative methods (GMRES, CG).
- Need to compute matrix-vector product (MVP) $\mathbf{Ax}$.
- Dense MVP costs $\mathcal{O}(n^2)$.
- "Sparse" MVP can cost $\mathcal{O}(n)$ or $\mathcal{O}(n \log n)$.
- Furthermore, requires $\mathcal{O}(n)$ storage.
- Can solve 10,000 of previous problem on my laptop!
- 2 main types of sparse MVP: FFT and FMM.

Consider the exterior Dirichlet problem we solved last time.

The arising matrix is **circulant**

$$\mathbf{C} = \begin{pmatrix} c_0 & c_{n-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & & c_2 \\ \vdots & c_1 & c_0 & \ddots & \vdots \\ c_{n-2} & & \ddots & \ddots & c_{n-1} \\ c_{n-1} & c_{-2} & \cdots & c_1 & c_0 \end{pmatrix}$$

and is defined by the n-vector $\mathbf{c} = [c_0, c_1, \ldots, c_{n-1}]$.

$\mathbf{C}$ has the property of being diagonalized by the DFT matrix

$$\mathbf{C} = \mathbf{F}^{-1}\mathbf{\Lambda}\mathbf{F},$$

where $\mathbf{F}$ is the DFT matrix and $\mathbf{\Lambda} = \mathrm{diag}(\mathbf{F}\mathbf{c})$ is diagonal.

Hence MVP is

$$\mathbf{Cx} = \mathbf{F}^{-1}\mathrm{diag}(\mathbf{Fc})\mathbf{Fx}.$$

This can be computed using the following four steps:

i) compute $\mathbf{f} = \mathrm{FFT}(\mathbf{x})$,

ii) compute $\mathbf{g} = \mathrm{FFT}(\mathbf{c})$,

iii) compute element-wise vector-vector product $\mathbf{h} = \mathbf{g}. * \mathbf{f}$,

iv) compute $\mathbf{y} = \mathrm{IFFT}(\mathbf{h})$ to obtain $\mathbf{Cx}$.

FFT and IFFT can be done in $\mathcal{O}(n \log n)$ operations.

Also, since we only need the defining vector $\mathbf{c}$, storage is $\mathcal{O}(n)$.

So far shown applicability to

Non-circular geometries

Circulant matrices will not arise in general.

Consider another simple geometry, a straight line.

The matrix for this problem is not circulant, but Toeplitz:

$$\mathbf{T} = \begin{pmatrix} a_0 & a_{-1} & a_{-2} & \cdots & \cdots & a_{-(n-1)} \\ a_1 & a_0 & a_{-1} & & & a_{-(n-2)} \\ a_2 & a_1 & \ddots & & & \vdots \\ \vdots & & & \ddots & a_{-1} & a_{-2} \\ a_{n-2} & \ddots & & a_1 & a_0 & a_{-1} \\ a_{n-1} & \cdots & \cdots & a_2 & a_1 & a_0 \end{pmatrix}.$$

Matrix is determined by first column and row $\rightarrow 2n-1$ entries.

Fast MVP with Toeplitz matrix

Embed Toeplitz matrix in a $2n \times 2n$ circulant matrix

$$\mathbf{C}_{2n} = \begin{pmatrix} \mathbf{T}_n & \mathbf{S}_n \\ \mathbf{S}_n & \mathbf{T}_n \end{pmatrix},$$

where

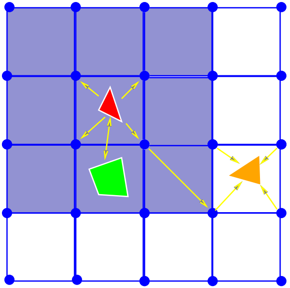
$$\mathbf{S}_n = \begin{pmatrix} 0 & a_{n-1} & \cdots & \cdots & a_1 \\ a_{-(n-1)} & \ddots & \ddots & & a_2 \\ \vdots & \ddots & & & \vdots \\ a_{-2} & & \ddots & \ddots & a_{n-1} \\ a_{-1} & a_{-2} & \cdots & \cdots & 0 \end{pmatrix}.$$

Then

$$\mathbf{T}_n\mathbf{x} = (\mathbf{I}_n \quad \mathbf{0}_n)\mathbf{C}_{2n}(\mathbf{x} \quad \mathbf{0}_n)^{\mathrm{T}}.$$

So can use FFT again! Cost $\mathcal{O}(2n\log(2n))$.

What about shapes that aren't circles or lines?



Algorithm outline

(i) Project panel charges onto grid

(ii) Calculate grid-charge potentials on grid

(iii) Interpolate grid potentials onto panels

(iv) Local corrections (compute nearby interactions directly)

Exploit translation invariance of the uniform grid by using FFT.

Overheads of projection and interpolation are not expensive so achieve $\mathcal{O}(n \log n)$.

Fast multipole method

Pioneering work by Greengard and Rokhlin in 1980s. Regarded as "one of the top ten algorithms of the 20th century" - Cipra.

Problem:

Wish to evaluate the sum

$$u(\mathbf{x}_i) = \sum_{j=1}^{n} G(\mathbf{x}_i, \mathbf{x}_j) g_j, \quad j = 1, \dots n,$$

where

$\{\mathbf{x}_i\}_{i=1}^n$ is a set of points in the plane,

$\{\mathbf{q}_i\}_{i=1}^n$ is a set of real numbers called *sources*,

$u(\mathbf{x})$ is called the *potential.*

The kernel is our Green's function. In 2D, $G(\mathbf{x}, \mathbf{y}) = \log |\mathbf{x} - \mathbf{y}|$.

Fast multipole method

Evaluating the sum

$$u(\mathbf{x}_i) = \sum_{j=1}^{n} G(\mathbf{x}_i, \mathbf{x}_j) g_j, \quad j = 1, \ldots n.$$

Direction evaluation - $\mathcal{O}(n^2)$.

Single-level fast multipole - $\mathcal{O}(n^{3/2})$

Multi-level fast multipole - $\mathcal{O}(n)$.

Implementing the FMM is not easy, and the constant in the $\mathcal{O}(n)$ can sometimes be large.

Let's discuss the ideas behind the single-level FMM.

For more details, see "A short course on fast multipole methods" - Beatson and Greengard.

Preconditioning large systems

- FFT- and FMM-accelerated methods rely on iterative methods.
- For them to be efficient, we require that the number of iterations is relatively low.
- Thus require matrix to be well conditioned.
- Provides further motivation for second kind equations which tend to more better conditioned than first kind.
- Can still become ill-conditioned for large systems. “Preconditioning” becomes necessary.
- Solve modified problem $M^{-1}Ax = M^{-1}b$, for example, where M is the preconditioner.

Summary

- Motivation for fast solvers.
- Use iterative solvers - concentrate on accelerating matrix-vector product.
- Circulant and Toeplitz matrices - FFT acceleration for the matrix-vector product.
- FFT methods for more general geometries.
- FFT relies on **tranlsation invariance** of grid.
- Fast multipole method uses **multi-resolution**.
- Can achieve $\mathcal{O}(n \log n)$ or $\mathcal{O}(n)$ cost instead of $\mathcal{O}(n^2)$.