CS 450: Numerical Analysis
Lecture 29 Chapter 12 Fast Fourier Transform
Fast Solvers: Multigrid and FFT

Edgar Solomonik

Department of Computer Science
University of Illinois at Urbana-Champaign

May 2, 2018
The Poisson equation serves as a model problem for numerical methods:

- the 2D Poisson problem and resulting Kronecker product linear system are a common benchmark,
- this system has the form $T \otimes I + I \otimes T$ where $T$ is tridiagonal.

Dense, sparse direct, iterative, FFT, and Multigrid methods provide increasingly good complexity for the problem:

- dense linear system solve costs $O(n^3)$ naively,
- nested dissection with Cholesky has $O(n^{3/2})$ complexity and $O(n \log n)$ memory
- Conjugate-Gradient gives $O(n^{3/2})$ complexity with $O(n)$ memory
- FFT achieves $O(n \log n)$ cost and multigrid achieves $O(n)$. 
Multigrid

- Multigrid employs a hierarchy of grids to accelerate iterative methods:
  - the residual equation $A\hat{x} = r$ on each fine grid, is approximately solved on the next coarser grid,
  - the equation is restricted by projection matrix $P$, so that $PAP^TP\hat{x} = Pr$
  - the interpolation operator (often given by $P^T$) is used to obtain an approximate $\hat{x}$ based on the coarse grid approximate solution,
  - at each level we perform some smoothing operations (e.g. Jacobi or Conjugate Gradient) before restriction and after interpolation,
  - at the coarsest level we typically solve directly.

- The multigrid method works by resolving high-frequency error components on finer-grids and low-frequency error components on coarser grids:
  - smoothers are usually effective at reducing local error, but slow at resolving global (low-frequency) components of the error,
  - on coarser grids, the low frequency error may be resolved more quickly.
Consider the Galerkin approximation with linear finite elements to the Poisson equation $u'' = f(t)$ with boundary conditions $u(a) = u(b) = 0$:

$$
\phi^{(h)}_i(t) = \begin{cases} 
\frac{(t - t_{i-1})}{h} : t \in [t_{i-1}, t_i] \\
\frac{(t_{i+1} - t)}{h} : t \in [t_i, t_{i+1}] \\
0 : \text{otherwise}
\end{cases}
$$

where $t_0 = t_1 = a$ and $t_{n+1} = t_n = b$. The weak form with grid spacing of $h$ is

$$
\int_a^b f(t)\phi^{(h)}_i(t)dt = -\sum_{j=1}^{n} x_j \int_a^b \phi^{(h)'}_j(t)\phi^{(h)'}_i(t)dt.
$$

In multigrid, we define a coarse grid basis of $(n - 1)/2$ functions, which are hat functions of twice the width,

$$
\phi^{(2h)}_i(t) = \frac{1}{2} \phi^{(h)}_{2i-2}(t) + \phi^{(h)}_{2i-1}(t) + \frac{1}{2} \phi^{(h)}_{2i}(t) = \begin{cases} 
\frac{(t - t_{i-2})}{2h} : t \in [t_{i-2}, t_i] \\
\frac{(t_{i+2} - t)}{2h} : t \in [t_i, t_{i+2}] \\
0 : \text{otherwise}
\end{cases}
$$
Coarse Grid Matrix

- Multigrid restricts the residual equation on the fine grid $A^{(h)} x = r^{(h)}$ to the coarse grid: Let $\phi^{(2h)} = \begin{bmatrix} \phi^{(2h)}_1 & \cdots & \phi^{(2h)}_{(n-1)/2} \end{bmatrix}$ and $\phi^{(h)} = \begin{bmatrix} \phi^{(h)}_1 & \cdots & \phi^{(h)}_n \end{bmatrix}$ and define restriction matrix $P$ so that $\phi^{(2h)} = P \phi^{(h)}$, i.e.,

$$P = \frac{1}{2} \begin{bmatrix} 1 & 2 & 1 \\ 1 & 2 & 1 \\ \vdots & \vdots & \vdots \end{bmatrix} = \begin{bmatrix} p^{(1)} \\ p^{(2)} \\ \vdots \end{bmatrix}.$$ 

The coarse grid stiffness matrix is given by

$$a^{(2h)}_{ij} = - \int_a^b \phi_j^{(2h)'}(t) \phi_i^{(2h)'}(t)dt$$

$$= -p^{(i)} \left( \int_a^b \phi^{(h)'}(t)\phi^{(h)'}T(t)dt \right) p^{(j)T},$$

$$A^{(2h)} = P A^{(h)} P^T.$$
Restricting the Residual Equation

Given the fine-grid residual $r^{(h)}$, we seek to use the coarse grid to approximate $x^{(h)}$ so that $Ax^{(h)} \approx r^{(h)}$

Given a function in the coarse grid basis, $u^{(2h)} = x^{(2h)T} \phi^{(2h)}$, we can express it in the fine-grid basis via

$$u^{(2h)} = x^{(2h)T} \underbrace{P \phi^{(h)}}_{\phi^{(2h)}} = \underbrace{x^{(2h)T}}_{x^{(h)T}} P \phi^{(h)}.$$

Consequently, the solution to the restricted residual equation $A^{(2h)}x^{(2h)} = r^{(2h)}$ will lead to an approximate residual equation solution on the fine grid with $x^{(h)} = P^T x^{(2h)}$. Noting this, we derive the form of the coarse grid residual,

$$r^{(2h)} = A^{(2h)}x^{(2h)}$$

$$= PA^{(h)} P^T x^{(2h)} = PA^{(h)} x^{(h)}$$

$$= Pr^{(h)}.$$
Discrete Fourier Transform

The solutions to hyperbolic PDEs like Poisson are wave-like and take on simple representations in the frequency basis, both for continuous and discretized equations. We define the discrete Fourier transform using

$$\omega_{(n)} = \cos(2\pi/n) - i \sin(2\pi/n) = e^{-2\pi i/n},$$

The DFT matrix $F \in \mathbb{R}^{n \times n}$ is given by $f_{ij} = \omega_{ij}^{(n)}$,

$$F = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega^{(4)} & \omega^2(4) & \omega^3(4) \\ 1 & \omega^{2(4)} & \omega^4(4) & \omega^6(4) \\ 1 & \omega^{3(4)} & \omega^6(4) & \omega^9(4) \end{bmatrix}$$

- it is complex and symmetric (not Hermitian),
- it is unitary modulo scaling $F^* = nF^{-1}$.

The discrete Fourier transform of vector $v$ is $Fv$. 
Fast Fourier Transform (FFT)

- Consider $b = Fa$, we have

$$\forall j \in [0, n - 1] \quad b_j = \sum_{k=0}^{n-1} \omega_{(n)}^{jk} a_k,$$

the FFT computes this recursively via 2 FFTs of dimension $n/2$, using

$\omega_{(n/2)} = \omega_{(n)}^2$,

$$b_j = \sum_{k=0}^{n/2-1} \omega_{(n)}^{j(2k)} a_{2k} + \sum_{k=0}^{n/2-1} \omega_{(n)}^{j(2k+1)} a_{2k+1}$$

$$= \sum_{k=0}^{n/2-1} \omega_{(n/2)}^{jk} a_{2k} + \omega_{(n)}^j \sum_{k=0}^{n/2-1} \omega_{(n/2)}^{jk} a_{2k+1}$$
Fast Fourier Transform Derivation

- The FFT leverages similarity between the first and second half of the output,

\[
b_j = \sum_{k=0}^{n/2-1} \omega_{(n/2)}^{jk} a_{2k} + \omega_{(n)}^{j} \sum_{k=0}^{n/2-1} \omega_{(n/2)}^{jk} a_{2k+1}
\]

\[
\underline{u_j}
\]

\[
\underline{v_j}
\]

corresponds closely to the entry shifted by \( n/2 \),

\[
b_{j+n/2} = \sum_{k=0}^{n/2-1} \omega_{(n/2)}^{(j+n/2)k} a_{2k} + \omega_{(n)}^{j+n/2} \sum_{k=0}^{n/2-1} \omega_{(n/2)}^{(j+n/2)k} a_{2k+1}
\]

Now \( \omega_{(n/2)}^{(j+n/2)k} = \omega_{(n/2)}^{jk} \) since \( (\omega_{(n/2)}^{n/2})^k = 1^k = 1 \) and using \( \omega_{(n)}^{n/2} = -1 \),

\[
b_{j+n/2} = \sum_{k=0}^{n/2-1} \omega_{(n/2)}^{jk} a_{2k} - \omega_{(n)}^{j} \sum_{k=0}^{n/2-1} \omega_{(n/2)}^{jk} a_{2k+1}
\]

\[
\underline{u_j}
\]

\[
\underline{v_j}
\]
FFT Algorithm Summary

- Let vectors $u$ and $v$ be two recursive FFTs, $\forall j \in [0, n/2 - 1]$
  \[
  u_j = \sum_{k=0}^{n/2-1} \omega_{(n/2)}^{jk} a_{2k}, \quad v_j = \sum_{k=0}^{n/2-1} \omega_{(n/2)}^{jk} a_{2k+1}
  \]

- Given $u$ and $v$ scale using "twiddle factors" $z_j = \omega_{(n)}^j \cdot v_j$

- Then it suffices to combine the vectors as follows $b = [u + z \ nob \ u - z]$

- The FFT has $O(n \log n)$ cost complexity:

  *There are two recursive calls of dimension $n/2$ and $O(n)$ work for application to twiddle factors and final summation, thus*

  \[
  T(n) = 2T(n) + O(n) = O(n \log n).
  \]
Applications of the FFT

- We can rapidly multiply degree $n - 1$ polynomials by considering their values $\omega^i_{(n)}$ for $i \in \{0, \ldots, n - 1\}$

$$p_c(\omega^i_{(n)}) = p_a(\omega^i_{(n)})p_b(\omega^i_{(n)})$$

Given coefficients of $p_a, p_b$ suffices to compute product with Vandermonde matrix where $v_{ij} = (\omega^i_{(n)})^j$, which is simply the DFT matrix. Interpolation to compute coefficients of $p_c$ is inverse DFT.

- More generally the DFT can be used to solve any Toeplitz linear system (convolution): A standard convolution has the form

$$\forall k \in [0, n - 1] \quad c_k = \sum_{j=0}^{k} a_j b_{k-j},$$

which is equivalent to multiplications of polynomials with degree $n/2 - 1$ and coefficients $a$ and $b$, where the convolution computes the coefficients $c$ of the product of the two polynomials.
The Fourier transform method for computing a convolution is given by

\[ c_k = \frac{1}{n} \sum_s \omega_n^{-ks} \left( \sum_j \omega_n^{sj} a_j \right) \left( \sum_t \omega_n^{st} b_t \right) \]

Rearrange the order of the summations to see what happens to every product of \( a \) and \( b \)

\[ c_k = \frac{1}{n} \sum_s \sum_j \sum_t \omega_n^{(j+t-k)s} a_j b_t \]

For any \( u = j + t - k \neq 0 \), we observe \( \sum_s (\omega_n^u)^s = 0 \)

When \( j + t - k = 0 \) the products \( \omega_n^{(s+t-j)k} = 1 \), so there are \( n \) nonzero terms \( a_j b_{k-j} \) in the summation
Solving Numerical PDEs with the FFT

- 1D finite-difference schemes on a regular grid correspond to convolutions:
  
  *1D model problem is simply convolution with vector $[1, -2, 1]$."

- For the 1D Poisson model problem, the eigenvectors of $T$ correspond to the imaginary part of a minor of a $2(n + 1)$-dimensional DFT matrix:

  In particular, $T = XD(X^{-1}$ where $x_{ij}$ is the imaginary part of $f_{i+1,j+1}$ with $X \in \mathbb{R}^{n \times n}$ and $F \in \mathbb{R}^{2(n+1) \times 2(n+1)}$. This means $T$ can be diagonalized and the overall system solved by FFT with $O(n \log n)$ cost.

- Multidimensional Poisson can be handled with multidimensional FFT:

  For example 2D FFT (1D FFT of each row then 1D FFT of each column) suffices to solve the 2D Poisson problem.