# Fast Poisson Solvers and the FFT

Suppose we have a *block TST* matrix with TST (Toeplitz symmetric tridiagonal) blocks:

$$A = \begin{pmatrix} S & T & & \\ T & \ddots & \ddots & \\ & \ddots & \ddots & T \\ & & T & S \end{pmatrix}, \quad S = \begin{pmatrix} \alpha & \beta & & \\ \beta & \ddots & \ddots & \\ & \ddots & \ddots & \beta \\ & & \beta & \alpha \end{pmatrix}, \quad T = \begin{pmatrix} \gamma & \delta & & \\ \delta & \ddots & \ddots & \\ & \ddots & \ddots & \delta \\ & & \delta & \gamma \end{pmatrix}. \tag{1}$$

If $A$ comes from the five-point finite difference approximation for Poisson's equation with Dirichlet boundary conditions on an $m_1$ by $m_2$ rectangular grid, then, with the natural ordering of equations and unknowns, $S$ and $T$ are $m_1$ by $m_1$ matrices given by

$$S = -\frac{1}{h^2}\begin{pmatrix} 4 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{pmatrix}, \quad T = \frac{1}{h^2} I.$$

For the nine-point operator,

$$S = -\frac{1}{6h^2}\begin{pmatrix} 20 & -4 & & \\ -4 & \ddots & \ddots & \\ & \ddots & \ddots & -4 \\ & & -4 & 20 \end{pmatrix}, \quad T = -\frac{1}{6h^2}\begin{pmatrix} -4 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & -4 \end{pmatrix}.$$

The eigenvalues and eigenvectors of TST matrices are known. It turns out that all TST matrices have the same eigenvectors; only their eigenvalues differ. If an $m$ by $m$ TST matrix has $\alpha$'s on its main diagonal and $\beta$'s on its first sub- and super-diagonals, then its eigenvalues are:

$$\lambda_j = \alpha + 2\beta\cos\left(\frac{\pi j}{m+1}\right), \quad j = 1, \ldots, m,$$

and the corresponding orthonormal eigenvectors are:

$$q_\ell^j = \sqrt{\frac{2}{m+1}}\sin\left(\frac{\pi j \ell}{m+1}\right), \quad \ell, j = 1, \ldots, m.$$

*Exercise:* Check that the above are indeed eigenvalues and eigenvectors of the $m$ by $m$ TST matrix with $\alpha$'s on the main diagonal and $\beta$'s on the first sub- and super- diagonals. Check that the eigenvectors are orthonormal: $\langle \mathbf{q}^j, \mathbf{q}^k \rangle = 0$ if $j \neq k$, $\langle \mathbf{q}^j, \mathbf{q}^j \rangle = 1$.

Let $A$ be the block TST matrix in (1) and consider the equation $A\mathbf{u} = \mathbf{f}$. Writing $\mathbf{u}$ and $\mathbf{f}$ as block vectors, $\mathbf{u} = (\mathbf{u}_1, \ldots, \mathbf{u}_{m_2})^T$ and $\mathbf{f} = (\mathbf{f}_1, \ldots, \mathbf{f}_{m_2})^T$, with blocks of length $m_1$ corresponding to one line in the $m_1$ by $m_2$ grid, this linear system takes the form:

$$T\mathbf{u}_{\ell-1} + S\mathbf{u}_\ell + T\mathbf{u}_{\ell+1} = \mathbf{f}_\ell, \quad \ell = 1, \ldots, m_2, \tag{2}$$

where $\mathbf{u}_0 = \mathbf{u}_{m_2+1} = \mathbf{0}$. Let $Q$ be the orthogonal matrix whose columns $\mathbf{q}^1, \ldots, \mathbf{q}^{m_1}$ are the eigenvectors of $S$ and $T$: $S = Q\Lambda^{(S)}Q^T$, $T = Q\Lambda^{(T)}Q^T$. Making these substitutions in (2) and multiplying each side by $Q^T$ gives

$$\Lambda^{(T)}Q^T\mathbf{u}_{\ell-1} + \Lambda^{(S)}Q^T\mathbf{u}_\ell + \Lambda^{(T)}Q^T\mathbf{u}_{\ell+1} = Q^T\mathbf{f}_\ell.$$

Defining $\mathbf{y}_\ell = Q^T\mathbf{u}_\ell$ and $\mathbf{g}_\ell = Q^T\mathbf{f}_\ell$, this becomes

$$\Lambda^{(T)}\mathbf{y}_{\ell-1} + \Lambda^{(S)}\mathbf{y}_\ell + \Lambda^{(T)}\mathbf{y}_{\ell+1} = \mathbf{g}_\ell, \quad \ell = 1, \ldots, m_2.$$

Let the entries in block $\ell$ of $\mathbf{y}$ be denoted $y_{\ell,1}, \ldots, y_{\ell,m_1}$, and similarly for those in each block of $\mathbf{g}$. Look at the equations for the $j$th entry in each block of $\mathbf{y}$:

$$\lambda_j^{(T)}y_{\ell-1,j} + \lambda_j^{(S)}y_{\ell,j} + \lambda_j^{(T)}y_{\ell+1,j} = g_{\ell,j}, \quad \ell = 1, \ldots, m_2.$$

Note that these equations decouple from those for all other entries. If, for each $j = 1, \ldots, m_1$, we define $\tilde{\mathbf{y}}_j = (y_{1,j}, \ldots, y_{m_2,j})^T$ and likewise $\tilde{\mathbf{g}}_j = (g_{1,j}, \ldots, g_{m_2,j})^T$, then we have $m_1$ independent tridiagonal systems, each involving $m_2$ unknowns:

$$\begin{pmatrix} \lambda_j^{(S)} & \lambda_j^{(T)} & & \\ \lambda_j^{(T)} & \ddots & \ddots & \\ & \ddots & \ddots & \lambda_j^{(T)} \\ & & \lambda_j^{(T)} & \lambda_j^{(S)} \end{pmatrix} \tilde{\mathbf{y}}_j = \tilde{\mathbf{g}}_j, \quad j = 1, \ldots, m_1.$$

The work to solve these tridiagonal systems is $O(m_1 m_2)$. This gives us the vectors $\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_{m_1}$. These can be rearranged to obtain the vectors $\mathbf{y}_1, \ldots, \mathbf{y}_{m_2}$, which were obtained from $\mathbf{u}_1, \ldots, \mathbf{u}_{m_2}$ by multiplying by $Q^T$. Thus the only remaining task is to recover the solution $\mathbf{u}$ from $\mathbf{y}$:

$$\mathbf{u}_\ell = Q\mathbf{y}_\ell, \quad \ell = 1, \ldots, m_2 \tag{3}$$

This requires $m_2$ matrix-vector multiplications, where the matrices are $m_1$ by $m_1$. Ordinarily, the amount of work required would be $O(m_2 m_1^2)$. A similar set of matrix-vector multiplications was required to compute $g$ from $f$: $g_\ell = Q^T f_\ell$, $\ell = 1, \ldots, m_2$. It turns out that because of the special form of the matrix $Q$, these matrix-vector multiplications can each be performed in time $O(m_1 \log_2 m_1)$ using the FFT. Thus the total work for performing these multiplications by $Q$ and $Q^T$, and the major part of the work in solving the original linear system, is $O(m_2 m_1 \log_2 m_1)$. This is almost the optimal order, $O(m_1 m_2)$.

*The Fast Fourier Transform.* To compute the entries of $\mathbf{u}_\ell$ in (3), we must compute sums of the form

$$u_{\ell,k} = \sum_{j=1}^{m_1} Q_{k,j} y_{\ell,j} = \sqrt{\frac{2}{m_1+1}} \sum_{j=1}^{m_1} \sin\left(\frac{\pi j k}{m_1+1}\right) y_{\ell,j}, \quad k = 1, \ldots, m_1.$$

Such sums are called discrete sine transforms, and they are the imaginary part of a *discrete Fourier transform* (DFT):

$$F_k = \sum_{j=0}^{N-1} e^{2\pi \imath jk/N} f_j, \quad k = 0, 1, \ldots, N-1. \tag{4}$$

While it would appear to require $O(N^2)$ work to compute the $N$ entries of the DFT, this can be done with $O(N \log_2 N)$ work using the *fast Fourier transform* (FFT) algorithm.

The key observation in computing the DFT of length $N$ is to note that it can be expressed in terms of two DFT's of length $N/2$, one involving the even terms and one involving the odd terms in (4). To this end, assume that $N$ is even, and define $w = e^{2\pi \imath/N}$. We can write

$$\begin{aligned} F_k &= \sum_{j=0}^{N/2-1} e^{2\pi\imath(2j)k/N} f_{2j} + \sum_{j=0}^{N/2-1} e^{2\pi\imath(2j+1)k/N} f_{2j+1} \\ &= \sum_{j=0}^{N/2-1} e^{2\pi\imath jk/(N/2)} f_{2j} + w^k \sum_{j=0}^{N/2-1} e^{2\pi\imath jk/(N/2)} f_{2j+1} \\ &= F_k^{(e)} + w^k F_k^{(o)}, \quad k = 0, 1, \ldots, N, \end{aligned} \tag{5}$$

where $F^{(e)}$ denotes the DFT of the even-numbered data points and $F^{(o)}$ the DFT of the odd-numbered data points. Note also that the DFT is periodic, with period equal to the number of data points. Thus $F^{(e)}$ and $F^{(o)}$ have period $N/2$: $F_{N/2+p}^{(e,o)} = F_p^{(e,o)}$, $p = 0, 1, \ldots, N/2-1$. Therefore once the $N/2$ entries in a period of $F^{(e)}$ and $F^{(o)}$ have been computed, we can obtain the length $N$ DFT by combining the two according to (5). If the $N$ coefficients $w^k$, $k = 0, 1, \ldots, N-1$ have already been computed, then the work to do this is $2N$ operations; we must multiply the coefficients $w^k$ by the entries $F_k^{(o)}$ and add to the entries $F_k^{(e)}$, $k = 0, 1, \ldots, N-1$.

This process can be repeated! For example, to compute the length $N/2$ DFT $F^{(e)}$ (assuming now that $N$ is a multiple of 4), we can write

$$\begin{aligned} F_k^{(e)} &= \sum_{j=0}^{N/4-1} e^{2\pi\imath(2j)k/(N/2)} f_{4j} + \sum_{j=0}^{N/4-1} e^{2\pi\imath(2j+1)k/(N/2)} f_{4j+2} \\ &= \sum_{j=0}^{N/4-1} e^{2\pi\imath jk/(N/4)} f_{4j} + w^{2k} \sum_{j=0}^{N/4-1} e^{2\pi\imath jk/(N/4)} f_{4j+2} \\ &= F_k^{(ee)} + w^{2k} F_k^{(eo)}, \quad k = 0, 1, \ldots, N/2-1, \end{aligned}$$

where $F^{(ee)}$ denotes the DFT of the even even data $(f_{4j})$ and $F^{(eo)}$ the DFT of the even odd data $(f_{4j+2})$. Since the length $N/4$ DFT's $F^{(ee)}$ and $F^{(eo)}$ are periodic with period $N/4$, once the $N/4$ entries in a period of these two transforms are known, we can combine them according to the above formula to obtain $F^{(e)}$. This requires $2N/2$ operations (if the powers of $w$ have already been computed), and the same number of operations is required to compute $F^{(o)}$, for a total of $2N$ operations.

Assuming that $N$ is a power of 2, this process can be repeated until we reach the length 1 DFT's. The DFT of length 1 is the identity, so it requires no work to compute these. There are $\log_2 N$ stages in this process and each stage requires $2N$ work to combine the DFT's of length $2^j$ to obtain those of length $2^{j+1}$. This gives a total amount of work that is approximately $2N \log_2 N$.

The only remaining question is how to keep track of which entry in the original vector of input data corresponds to each length 1 transform, e.g., $F^{(eooeoe)} = f_?$. This looks like a bookkeeping nightmare! But actually it is not so difficult. Reverse the sequence of e's and o's, assign 0 to e and 1 to o, and you will have the binary representation of the index of the original data point. Do you see why this works? It is because at each stage we separate the data according to the next bit from the right. The even data at stage one consists of those elements that have a 0 in the rightmost position of their binary index, while the odd data consists of those elements that have a 1 in the rightmost bit of their index. The even even data points have a 0 in the rightmost two bits of their index, while the even odd data points have a 0 in the rightmost bit and a 1 in the next bit from the right, etc. To make the bookkeeping really easy, we can initially order the data using bit reversal of the binary index. For example, if there are 8 data points, then we would order them as:

$$\begin{pmatrix} f_{0=000} \\ f_{1=001} \\ f_{2=010} \\ f_{3=011} \\ f_{4=100} \\ f_{5=101} \\ f_{6=110} \\ f_{7=111} \end{pmatrix} \longrightarrow \begin{pmatrix} f_{0=000} \\ f_{4=100} \\ f_{2=010} \\ f_{6=110} \\ f_{1=001} \\ f_{5=101} \\ f_{3=011} \\ f_{7=111} \end{pmatrix}.$$

Then the length 2 transforms are just linear combinations of neighboring entries, the length 4 transforms are linear combinations of neighboring length 2 transforms, etc.