Math 476/667: The Fast Fourier Transform

The Fourier transform was originally developed by Joseph Fourier [3] for the study of heat transfer and vibrations. Fourier transforms are currently used in the study of differential equations, approximation theory, quantum mechanics, time-series analysis, implementation of high precision arithmetic, digital signal processing, GPS, sound and video compression, digital telephony and encryption. The fast Fourier transform is a divide and conquer algorithm developed by Cooley and Tukey [1] to efficiently compute a discrete Fourier transform on a digital computer. In 2000 Dongarra and Sullivan listed the fast Fourier transform among the top 10 algorithms of the 20th century [2].

Joseph Fourier of École Polytechnique, James Cooley of IBM Watson Laboratories and John Tukey of Princeton University and Bell Laboratories.

The Discrete Fourier Transform

The discrete Fourier transform is given by the matrix-vector multiplication Ax where A is an $N \times N$ matrix with general term given by $a_{kl} = e^{-i2\pi kl/N}$ with $k = 0, 1, \ldots, N-1$ and $l = 0, 1, \ldots, N-1$. While standard mathematical notation for matrices and vectors use index variables which range from 1 to N, we have shifted the indices by one so that the first column and first row of A are given by $k = 0$ and $l = 0$. Shifting indices in this way is both the natural for the C programming language and the mathematics. This shifted notation for indices will be used throughout our computational study of linear algebra.

Define $\overline{A}$ to be the matrix whose entries are exactly the complex conjugates of the entries of A. Our first result is

The Fourier Inversion Theorem. *Let A be the $N \times N$ Fourier transform matrix defined above. Then*

$$A^{-1} = \frac{1}{N}\overline{A}.$$

To see why this formula is true we first prove

The Orthogonality Lemma. *Suppose $l, p \in \{0, 1, \ldots, N-1\}$, then*

$$\sum_{q=0}^{N-1} e^{i2\pi(l-p)q/N} = \begin{cases} N & \textit{for } l = p \\ 0 & \textit{otherwise.} \end{cases}$$

Proof of The Orthogonality Lemma. Since

$$0 \le l \le N-1 \qquad \text{and} \qquad -(N-1) \le -p \le 0,$$

then $-(N-1) \le l-p \le N-1$ and consequently

$$-2\pi\Big(1-\frac{1}{N}\Big) \le 2\pi(l-p)/N \le 2\pi\Big(1-\frac{1}{N}\Big).$$

Define $\omega = e^{i2\pi(l-p)/N}$. Since the only time $e^{i\theta} = 1$ is when θ is a multiple of 2π, we conclude that

$$\omega = 1 \qquad \text{if and only if} \qquad l = p.$$

Clearly, if $l = p$ then

$$\sum_{q=0}^{N-1} e^{i2\pi(l-p)q/N} = \sum_{q=0}^{N-1} w^q = \sum_{q=0}^{N-1} 1 = N.$$

On the other hand, if $l \neq p$ then $\omega \neq 1$. In this case,

$$\omega^N = e^{i2\pi(l-p)} = 1$$

and the geometric sum formula yields that

$$\sum_{q=0}^{N-1} e^{i2\pi(l-p)q/N} = \sum_{q=0}^{N-1} \omega^q = \frac{1-\omega^N}{1-\omega} = \frac{1-1}{1-\omega} = 0.$$

This finishes the proof of the lemma. ////

We are now ready to explain the Fourier inversion theorem.

Proof of The Fourier Inversion Theorem. Let $b = Ax$ and $c = \frac{1}{N}\overline{A}b$. Claim that $c = x$. By definition

$$b_k = \sum_{l=0}^{N-1} e^{-i2\pi kl/N} x_l \qquad \text{and} \qquad c_p = \frac{1}{N}\sum_{q=0}^{N-1} e^{i2\pi pq/N} b_q.$$

Substituting yields

$$\begin{aligned} c_p &= \sum_{q=0}^{N-1} e^{-i2\pi pq/N}\Big(\frac{1}{N}\sum_{l=0}^{N-1} e^{i2\pi ql/N} x_l\Big) = \frac{1}{N}\sum_{l=0}^{N-1}\Big\{\sum_{q=0}^{N-1} e^{i2\pi(l-p)q/N}\Big\} x_l \\ &= \frac{1}{N}\sum_{l=0}^{N-1}\left\{\begin{array}{cl} N & \text{for } l = p \\ 0 & \text{otherwise} \end{array}\right\} x_l = \frac{N}{N} x_p = x_p. \end{aligned}$$

This finishes the proof of the theorem. ////

The Fast Fourier Transform

While a factor 18 speedup was easy to obtain by parallelizing the slow algorithm, in the case of the Fourier transform much more significant gains can be achieved by using a conquer and divide approach. This is possible because the matrix A corresponding to the Fourier transform has a significant number of symmetries in it based on the factors of the length N of the transform. For simplicity we will assume that $N = 2^n$ for some positive integer n. Thus, N is divisible by 2 and we can write $2K = N$. It follows that

$$\begin{aligned}\sum_{l=0}^{N-1} e^{-i2\pi kl/N} x_l &= \sum_{l \text{ even}} e^{-i2\pi kl/N} x_l + \sum_{l \text{ odd}} e^{-i2\pi kl/N} x_l \\ &= \sum_{p=0}^{K-1} e^{-i2\pi kp/K} x_{2p} + e^{-i2\pi k/N} \sum_{p=0}^{K-1} e^{-i2\pi kp/K} x_{2p+1}\end{aligned} \tag{1}$$

Note that the original Fourier transform of size N has been rewritten as two smaller Fourier transforms of size K which then need to be combined. The combining is done by multiplying the second transform by the factor $e^{-i2\pi k/N}$ for $k = 0, 1, \ldots, N-1$ which results in N additional multiplications. Therefore, the total number of operations has been reduced to

$$K^2 + N + K^2 = 2\left(\frac{N}{2}\right)^2 + N = \frac{1}{2}N^2 + N$$

which is a reduction of almost half the original N^2.

We are now ready to prove

The Fast Fourier Transform Theorem. *Suppose $N = 2^n$, then the Fourier transform can be computed in $N \log_2 N$ number of operations.*

Proof of The Fast Fourier Transform Transform Theorem. Consider the minimal number of operations T_n needed to perform a discrete Fourier transform of size 2^n. By the conquer and divide step described above, we know that

$$T_n \le 2T_{n-1} + 2^n \qquad \text{and similarly} \qquad T_{n-1} \le 2T_{n-2} + 2^{n-1}.$$

Substituting the latter in to the former yields $T_n \le 2^2 T_{n-2} + 2(2^n)$ and by induction it follows that

$$T_n \le 2^n T_0 + n2^n.$$

Since the transform of length one is the identity then $T_0 = 0$. Consequently, $T_n \le N \log_2 N$. This shows the Fourier transform can be computed in $N \log_2 N$ operations. ////

We remark that $N \log_2 N$ number of operations can be much smaller than N^2 when N is large. When $N = 8192$, as used for our previous numerical test, it follows that

$$N \log_2 N = 106496 \qquad \text{and} \qquad N^2 = 67108864.$$

Since $67108864/106496 \approx 630$, using the fast Fourier transform has the performance advantage of about 630 additional processor cores when $N = 8192$. For larger values of N the advantages are even more pronounced. When $N = 65536$ the slow algorithm takes an impractically long time; for values of N corresponding to vectors that are sized to the limits of available memory, the fast algorithm is the only way to complete the computation.

References

1. James Cooley and John Tukey, An algorithm for the machine calculation of complex Fourier series, *Math. Comput.*, Vol. 19, 1965.
2. Jack Dongarra and Francis Sullivan, Top Ten Algorithms of the Century, *Computing in Science and Engineering*, 2000.
3. Joseph Fourier, Théorie analytique de la chaleur, *Firmin Didot Père at Fils*, 1822.