FFT: The Fast Fourier Transform

- The fast Fourier Transform is the discrete version of Fourier Series, *roughly speaking*.
- There are many versions of it!
- Modern technology, of course, is discrete.
- The recursive version of the FFT goes back to Gauss (1777-1855)
- According to the wikipedia he invented it around 1805 and used it to interpolate the trajectories of the asteroids Pallas and Juno. He published his work posthumously in Latin.
- However, the basis of modern versions is a famous paper:
- Cooley and Tukey rediscovered and popularized the FFT 160 years after Gauss.
- We will look at a version that I believe is particularly compelling and clear on first exposure to the FFT. Modern implementations are
recursive, binary based, and of course available in numerous implementations.

• This description is taken from section 6.6-1 the classic Ralston and Rabinowitz, A First Course in Numerical Analysis, Dover Publications, ISBN 0-486-41454-X, first published in 1965. (This is the same source as we used for the Simplex Method earlier this semester.)

• Ralston and Rabinowitz is an excellent reference, mathematically thorough, with great exercises, and still in print and available as an inexpensive paperback.

• Recall Euler’s Formula, which combines the exponential and trigonometric functions:

\[ e^{i\theta} = \cos \theta + i \sin \theta \]

Query: how can that be true?

• It follows from this formula that

\[ e^{-i\theta} = \cos \theta - i \sin \theta \]

and thus

\[ \cos \theta = \frac{1}{2} (e^{i\theta} + e^{-i\theta}) \quad \text{and} \]

\[ \sin \theta = \frac{1}{2i} (e^{i\theta} - e^{-i\theta}) \]

• Our starting point is to use these formulas to take a fresh look at the Fourier Series

\[ F(t) = \frac{a_0}{2} + \sum_{j=1}^{n} (a_j \cos jt + b_j \sin jt) \]
where

\[ a_j = \frac{1}{2}(G_j + \bar{G}_j), \]
\[ b_j = \frac{1}{2i}(G_j - \bar{G}_j), \]
\[ G_j = \int_{-\pi}^{\pi} e^{ijt} f(t) dt = a_j + ib_j, \quad i^2 = -1 \]

- To be consistent with Ralston/Rabinowitz let’s change \( f \) to \( g \), and change the periodicity to 1:

\[ G_j = \int_{-\pi}^{\pi} g(t)e^{2\pi ijt} dt \quad \text{where} \quad i^2 = -1. \]  
(1)

- Integration by substitution introduces an extra factor \( 2\pi \), we ignore this detail.

- We suppose further that we can evaluate \( g \) only at \( N \) evenly spaced points in the interval \([0, 1]\):

\[ g_k = g(x_k), \quad x_k = \frac{k}{N}, \quad k = 0, 1, \ldots, N - 1. \]

- Replacing the integral in (1) with a sum leads to

\[ G_j = \sum_{k=0}^{N-1} g_k e^{\frac{2\pi ijk}{N}}, \quad j = 0, \ldots, N - 1. \]
We want to compute $N$ quantities $G_j$. Each sum has $N$ terms. A straightforward implementation would require $N^2$ terms and $N^2$ products.

The FFT reduces this effort top $N \log N$, which of course is a substantial reduction.

Rewrite $G_j$ as

$$G_j = \sum_{k=0}^{N-1} g_k w^{jk} \quad \text{where} \quad w = e^{\frac{2\pi i}{N}}$$

Note that $w^j$ is $N$-periodic. The powers of $w$ can be computed once and then stored for future reference.

In practice, $N$ is a power of 2, but the FFT is easier to explain in terms of a prime factorization that has all factors distinct. So suppose

$$N = r_1 r_2 \ldots r_t$$

We now define $t$-tuples $(j_1, \ldots, j_t)$ and $(k_1, \ldots, k_t)$ such that for $s = 1, 2, \ldots, t$, $j_s = 0, 1, \ldots, r_s - 1$, and $k_s = 0, 1, \ldots, r_s - 1$

$$j = j_1 + r_1 j_2 + r_1 r_2 j_3 + \ldots r_1 r_2 \ldots r_{t-1} j_t \quad (2)$$

and

$$k = k_t + r_t k_{t-1} + r_t r_{t-1} k_{t-2} + \ldots r_t r_{t-1} \ldots r_2 k_1. \quad (3)$$
• The $j$’s and $k$’s are the digits of $j$ and $k$. If all the $r_i$ were the same (in praxis they are all equal to 2) then the digits would be the ordinary base $r$ digits. The representations (2) and (3) are called a mixed radix representation.

• Let’s consider the example

\[ N = r_1 r_2 r_3 = 2 \times 3 \times 5 = 30, \]
\[ j = j_1 + r_1 j_2 + r_1 r_2 j_3 = j_1 + 2 j_2 + 6 j_3, \]
\[ k = k_3 + r_3 k_2 + r_3 r_2 k_1 = k_3 + 5 k_2 + 15 k_1. \]

\[ \hat{j} = 7 = 1 \cdot 6 + 0 \cdot 2 + 1 \quad \hat{j}_1 = 1 = \hat{j}_3 \quad \hat{j}_2 = 0 \]
\[ \hat{k} = 7 = 0 \cdot 15 + 5 \cdot 1 + 2 \quad \hat{k}_3 = 2 \quad \hat{k}_2 = 5 \quad \hat{k}_1 = 0 \]
• Now suppose that the \( r_i \) are general, but \( t = 3 \) to keep the algebra simple.

• Recall that \( w^n = 1 \).

• We get

\[
G_j = \sum_{k=0}^{N-1} g_k w^{jk} \\
= \sum_{k=0}^{N-1} g_k w^j(k_3+k_2r_3+k_1r_2r_3) \\
= \sum_{k_3=0}^{r_3-1} \sum_{k_2=0}^{r_2-1} \sum_{k_1=0}^{r_1-1} g_k w^{k_1j} r_3 r_2 w^{k_2j} r_3 w^{k_3j} \\
\]

• Next, note that

\[
w^{k_1j} r_3 r_2 = w^{k_1(j_1+r_1j_2+r_1r_2)} r_3 r_2 \\
= w^{k_1j_1} r_3 r_2 \underbrace{w^{k_1j_2} r_3 r_2}_{=1} \underbrace{w^{k_1r_2} r_3 r_2}_{=1} \\
= w^{k_1j_1} r_3 r_2 \\
\]

and

\[
w^{k_2j} r_3 = w^{k_2(j_1+r_1j_2+r_1r_2)} r_3 = w^{k_2(j_1+r_1j_2)} r_3 .
\]

• We can thus rewrite \( G_j \) as

\[
G_j = \sum_{k_3=0}^{r_3-1} \left( \sum_{k_2=0}^{r_2-1} \left( \sum_{k_1=0}^{r_1-1} g_k w^{k_1j_1} r_3 r_2 \right) w^{k_2(j_1+j_2)} r_3 \right) w^{jk_3} .
\]
• This sum can be computed as follows:

1. Compute the innermost sum for all triples \((j_1, k_2, k_3)\) (requiring \(r_1N\) multiplications).
2. Compute the middle sum for all triples \((j_1, j_2, k_3)\) (requiring \(r_2N\) multiplications).
3. Compute the outermost sum for all triples \((j_1, j_2, j_3)\) (requiring \(r_3N\) multiplications).

• Schematically, we compute the \(G_j\) as follows:

\[
\begin{align*}
    f_0(k_1, k_2, k_3) & \leftarrow g_k \\
    f_1(j_1, k_2, k_3) & \leftarrow \sum_{k_1=0}^{r_1-1} f_0(k_1, k_2, k_3) w^{k_1j_1} w^{k_2} w^{k_3} \\
    f_2(j_1, j_2, k_3) & \leftarrow \sum_{k_2=0}^{r_2-1} f_1(j_1, k_2, k_3) w^{k_2(j_1+j_2)} w^{k_3} \\
    G_j = f_3(j_1, j_2, j_3) & \leftarrow \sum_{k_3=0}^{r_3-1} f_2(j_1, h_2, k_3) w^{j_3} 
\end{align*}
\]

• The total number of terms in these sums equals \(N(r_1 + r_2 + r_3)\). In our small example we get \(30 \times (2 + 3 + 5) = 300\) operations instead of \(30^2 = 900\).

• In general, if \(N = r^t\) we get \(N \times rt\) instead of \(N^2\) operations.

• For example, if \(N = 2^{14}\) (corresponding to about 16 kHz, the limit of human hearing) we
get $2^{14} \times 28$ instead of $2^{28}$ operations. This amounts to savings by a factor

$$\frac{2^{28}}{28 \times 2^{14}} \approx 585.$$ 

- If we consider $N = r^t$, what are the best values of $r$ and $t$?
- Suppose we keep $N = r^t$ fixed and we want to minimize

$$E = N \times t \times r.$$ 

- Writing

$$t = \log_r N = \frac{\ln N}{\ln r}$$

we get

$$E = (N \ln N) \frac{r}{\ln r}$$

- $N \ln N$ is a constant. Differentiating and setting equal to zero gives

$$\frac{d}{dr} \frac{r}{\ln r} = \frac{\ln r - 1}{\ln^2 r} = 0.$$ 

- Thus $\ln r = 1$ which means the “optimal” value of $r$ is $e$. Of course, $r$ needs to be a positive integer.
- Examining $r = 2$ and $r = 3$ gives

$$\frac{3}{\ln 3} \approx 2.73 \quad \text{and} \quad \frac{2}{\ln 2} \approx 2.89.$$
So, in principle, $r = 3$ would be the most effective choice for the FFT. However, since computers are binary based, $r = 2$ is the usual choice.

- Next: making the approximation local, wavelets.