FFT: The Fast Fourier Transform

- The fast Fourier Transform is the discrete version of Fourier Series.
- There are many versions of it!
- Modern technology, of course, is discrete.
- The recursive version of the FFT goes back to Gauss\(^{-1}\).
- 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.
- However, modern implementations are recursive, binary based, and of course available in numerous implementations.
- The FFT has even been hardwired (rather than coded).

\(^{-1}\) Carl Friedrich Gauss, 1777–1855
• 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\(^{-2}\), which combines the exponential and trigonometric functions:

\[
e^{i\theta} = \cos \theta + i \sin \theta \quad \text{where} \quad i^2 = -1.
\]

Query: how can that be true?
• It follows from this formula that

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

and thus

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

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

• 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

\[
a_j = \int_{-\pi}^{\pi} f(t) \cos t dt = \frac{1}{2} (G_j + \bar{G}_j),
\]

\[
b_j = \int_{-\pi}^{\pi} f(t) \sin t dt = \frac{1}{2i} (G_j - \bar{G}_j),
\]

where

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

\(^{-2}\) Leonard Euler, 1707–1783

Math 5600 Spring 2021 Notes of 2/26/21 page 2
• To be consistent with Ralston/Rabinowitz let’s change $f$ to $g$, and change the periodicity to 1:

$$G_j = \int_0^1 g(t) e^{2\pi i j t} dt \quad \text{where} \quad i^2 = -1. \quad (1)$$

• 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

$$G_j = \sum_{k=0}^{N-1} g_k e^{\frac{2\pi i j k}{N}}, \quad j = 0, \ldots, N - 1.$$

• 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 to $N \log N$, 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**

$$N = r_1 r_2 r_3 = 2 \times 3 \times 5 = 30,$$

$$j = j_1 + r_1 j_2 + r_1 r_2 j_3 = j_1 + 2j_2 + 6j_3,$$

$$k = k_3 + r_3 k_2 + r_3 r_2 k_1 = k_3 + 5k_2 + 15k_1.$$

<table>
<thead>
<tr>
<th>$j$</th>
<th>$j_1$</th>
<th>$j_2$</th>
<th>$j_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$k$</td>
<td>$k_1$</td>
<td>$k_2$</td>
<td>$k_3$</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$j$</th>
<th>$j_1$</th>
<th>$j_2$</th>
<th>$j_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$k$</td>
<td>$k_1$</td>
<td>$k_2$</td>
<td>$k_3$</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
• Now suppose that the $r_i$ are general, but $t = 3$ to keep the algebra simple.

• Recall that

\[ w^N = 1. \]

• We get

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

• Next, note that

\[
w^{k_1 j r_3 r_2} = w^{k_1 (j_1 + r_1 j_2 + r_1 r_2 j_3) r_3 r_2} \\
= w^{k_1 j_1 r_3 r_2} w^{k_1 r_1 j_2 r_3 r_2} w^{k_1 r_1 r_2 j_3 r_3 r_2} = 1 = 1 \\
= w^{k_1 j_1 r_3 r_2}
\]

and

\[
w^{k_2 j r_3} = w^{k_2 (j_1 + r_1 j_2 + r_1 r_2 j_3) r_3} = w^{k_2 (j_1 + r_1 j_2) r_3}.
\]

• We can thus rewrite $G_j$ as

\[
G_j = \sum_{k_3=0}^{r_3-1} \left( \sum_{k_2=0}^{r_2-1} \left( \sum_{k_1=0}^{r_1-1} g_k w^{k_1 j_1 r_3 r_2} w^{k_2 (j_1 + j_2 r_1) r_3} \right) w^{jk_3} \right).
\]

• This sum can be computed as follows:

1. Compute the innermost sum for all triples $(j_1, k_2, k_3)$ (requiring $r_1 N$ 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:

\[
\begin{align*}
  f_0(k_1, k_2, k_3) &\leftarrow g_k \\
  f_1(j_1, k_2, k_3) &\leftarrow \sum_{k_1=0}^{r_1-1} f_0(k_1, k_2, k_3)w^{k_1j_1r_2r_3} \\
  f_2(j_1, j_2, k_3) &\leftarrow \sum_{k_2=0}^{r_2-1} f_1(j_1, k_2, k_3)w^{k_2(j_1+j_2r_1)r_3} \\
  G_j = f_3(j_1, j_2, j_3) &\leftarrow \sum_{k_3=0}^{r_3-1} f_2(j_1, h_2, k_3)w^{jk_3}
\end{align*}
\]

- Thus at the end of the process the original array containing the data \(g_j\) is overwritten by the coefficients \(G_k\).

- However, the sequence of the \(G_k\) differs from that of the \(g_j\). For details of this digit reversal see Ralston and Rabinowitz.

- The total number of terms in the above 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

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

• 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.