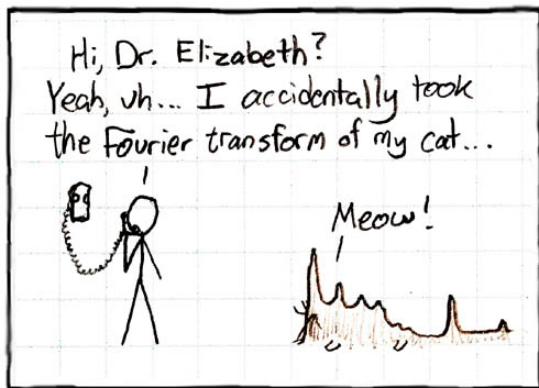


The Fast Fourier Transform

Fernando Guevara Vasquez
University of Utah



<http://xkcd.com/26>

(No cats were harmed in writing this talk, only transformed unitarily)

A little experiment

— Two tones are played in each experiment. —

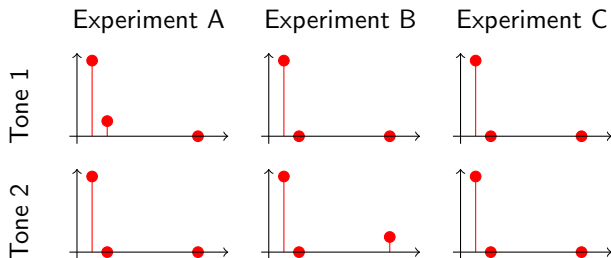
Please raise your hand if you hear a **difference** between the two tones.

Experiment A

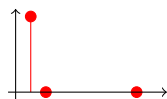
Experiment B

Experiment C

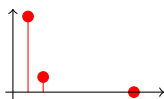
Experimental results



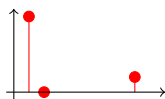
The different tones are:



$$\sin(2\pi F_1 t)$$



$$\sin(2\pi F_1 t) + \frac{1}{20} \sin(2\pi(F_1 + 2)t)$$



$$\sin(2\pi F_1 t) + \frac{1}{20} \sin(2\pi F_2 t)$$

with $F_1 = 440\text{Hz}$ (A4) and $F_2 = 1046.5\text{Hz}$ (C6).

Conclusion

To the human ear: a loud frequency can mask nearby frequencies.

Psychoacoustic coding

The principle behind MP3 (and other lossy sound compression formats) is to

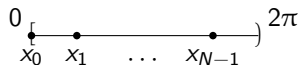
- Step 1. Separate the different frequencies of sound to compress
- Step 2. Eliminate frequencies that are masked by other frequencies
- Step 3. Compress everything with a lossless compression algorithm (à la zip)

For Step 1, an efficient way of sieving through the frequencies is the **Fast Fourier Transform** (1965 – Cooley and Tukey).



The Dark Side of the Moon, Pink Floyd

The Discrete Fourier Transform (DFT) problem



Find the coefficients a_0, a_1, \dots, a_n and b_1, b_2, \dots, b_n so that the function

$$f(x) = a_0 + a_1 \cos(x) + a_2 \cos(2x) + \dots + a_n \cos(nx) \\ + b_1 \sin(x) + b_2 \sin(2x) + \dots + b_n \sin(nx)$$

agrees with the data y_j at the equally spaced nodes

$$x_j \equiv \frac{2\pi j}{2n+1}, \quad j = 0, \dots, 2n$$

In other words we require that:

$$f(x_j) = y_j, \quad j = 0, \dots, 2n.$$

Complex numbers

Let x and y be real numbers and let $i \equiv \sqrt{-1}$.

- The number $z = x + iy$ a **complex number**.
- Complex numbers can be thought of as vectors in the xy plane.
- The **conjugate** of a complex number $z = x + iy$ is:

$$\bar{z} = x - iy.$$

- The **magnitude** of a complex number is:

$$\begin{aligned} |z|^2 &= \bar{z}z \\ &= (x - iy)(x + iy) \\ &= x^2 + y^2 \\ &= \text{length of } z \text{ as a vector, squared.} \end{aligned}$$

Euler's formula

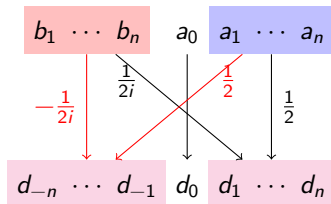
Using $e^{ix} = \cos x + i \sin x$

$$\cos x = \frac{e^{ix} + e^{-ix}}{2} \quad \text{and} \quad \sin x = \frac{e^{ix} - e^{-ix}}{2i},$$

and the sum of cosines and sines becomes:

$$\begin{aligned} f(x) &= \sum_{j=0}^n a_j \underbrace{\frac{1}{2}(e^{ijx} + e^{-ijx})}_{\cos(jx)} + \sum_{j=1}^n b_j \underbrace{\frac{1}{2i}(e^{ijx} - e^{-ijx})}_{\sin(jx)} \\ &= \sum_{j=-n}^n d_j e^{ijx}, \end{aligned}$$

The d_j are related to the a_j and b_j by



keep order: \rightarrow
reverse order: \rightarrow

The DFT problem (revisited)

Find the coefficients $d_{-n}, \dots, d_0, \dots, d_n \in \mathbb{C}$ so that for $j = 0, \dots, 2n$,

$$y_j = d_{-n}e^{-inx_j} + \dots + d_{-1}e^{-ix_j} + d_0e^{i0x_j} + \dots + d_n e^{inx_j}$$

The DFT problem (revisited)

Find the coefficients $d_{-n}, \dots, d_0, \dots, d_n \in \mathbb{C}$ so that for $j = 0, \dots, 2n$,

$$\begin{aligned}y_j &= d_{-n}e^{-inx_j} + \dots + d_{-1}e^{-ix_j} + d_0e^{i0x_j} + \dots + d_n e^{inx_j} \\ &= d_{-n}e^{i(n+1)x_j} + \dots + d_{-1}e^{i(2n)x_j} + d_0e^{i0x_j} + \dots + d_n e^{inx_j}\end{aligned}$$

We could make this simplification because

$$e^{-ikx_j} = e^{-2\pi i k j / (2n+1)} = e^{2\pi i (-k + (2n+1))j / (2n+1)} = e^{i(2n+1-k)x_j}.$$

The DFT problem (revisited)

Find the coefficients $d_{-n}, \dots, d_0, \dots, d_n \in \mathbb{C}$ so that for $j = 0, \dots, 2n$,

$$\begin{aligned} y_j &= d_{-n}e^{-inx_j} + \dots + d_{-1}e^{-ix_j} + d_0e^{i0x_j} + \dots + d_n e^{inx_j} \\ &= d_{-n}e^{i(n+1)x_j} + \dots + d_{-1}e^{i(2n)x_j} + d_0e^{i0x_j} + \dots + d_n e^{inx_j} \end{aligned}$$

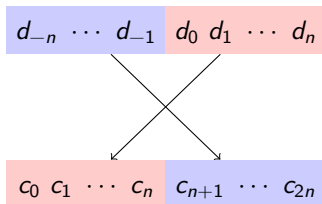
We could make this simplification because

$$e^{-ikx_j} = e^{-2\pi i k j / (2n+1)} = e^{2\pi i (-k + (2n+1))j / (2n+1)} = e^{i(2n+1-k)x_j}.$$

Thus it is equivalent to find c_0, \dots, c_{2n} so that

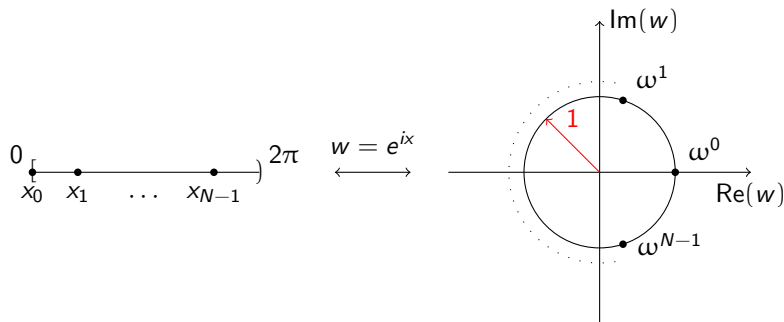
$$y_j = c_0 e^{i0x_j} + c_1 e^{ix_j} + \dots + c_{2n} e^{i(2n)x_j}, \quad \text{for } j = 0, \dots, 2n,$$

where the c_k and d_k are related by



The DFT problem and trigonometric polynomials

With $\omega = e^{2i\pi/N}$ we have $e^{ix_j} = e^{2ij\pi/N} = \omega^j$.



DFT problem \Leftrightarrow interpolating with trigonometric polynomials on the unit circle

$$\begin{aligned} f(x) &= c_0 + c_1 e^{ix} + c_2 e^{i2x} + \dots + c_{N-1} e^{i(N-1)x} \\ &= c_0 + c_1 w + c_2 w^2 + \dots + c_{N-1} w^{N-1} \quad \equiv p(w). \end{aligned}$$

Formulation of DFT as a linear system

Write the equations $p(\omega^j) = y_j$, $j = 0, \dots, N-1$ as a linear system:

$$\underbrace{\begin{bmatrix} \omega^{0 \times 0} & \omega^{0 \times 1} & \omega^{0 \times 2} & \dots & \omega^{0 \times (N-1)} \\ \omega^{1 \times 0} & \omega^{1 \times 1} & \omega^{1 \times 2} & \dots & \omega^{1 \times (N-1)} \\ \omega^{2 \times 0} & \omega^{2 \times 1} & \omega^{2 \times 2} & \dots & \omega^{2 \times (N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega^{(N-1) \times 0} & \omega^{(N-1) \times 1} & \omega^{(N-1) \times 2} & \dots & \omega^{(N-1) \times (N-1)} \end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{N-1} \end{bmatrix}}_{\mathbf{c}} = \underbrace{\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{N-1} \end{bmatrix}}_{\mathbf{y}}$$

Indeed the j -th row reads:

$$\begin{aligned} y_j &= c_0 \omega^{j \times 0} + c_1 \omega^{j \times 1} + c_2 \omega^{j \times 2} + \dots + c_{N-1} \omega^{j \times (N-1)} \\ &= c_0 (\omega^j)^0 + c_1 (\omega^j)^1 + c_2 (\omega^j)^2 + \dots + c_{N-1} (\omega^j)^{N-1} \\ &= p(\omega^j). \end{aligned}$$

Properties of the DFT matrix

Theorem

The matrix \mathbf{Q} of the DFT problem satisfies $\mathbf{Q}^* \mathbf{Q} = N\mathbf{I}$. Thus $\mathbf{Q}^{-1} = \frac{1}{N} \mathbf{Q}^*$, and the DFT problem admits a unique solution for any data \mathbf{y} .

Proof.

If \mathbf{q}_j is the j -th column of \mathbf{Q} , we need to show that

$$\mathbf{q}_n^* \mathbf{q}_m = \begin{cases} N & \text{if } n = m \\ 0 & \text{if } n \neq m. \end{cases}$$

We can compute this inner product explicitly

$$\mathbf{q}_n^* \mathbf{q}_m = \sum_{j=0}^{N-1} \omega^{-j \times n} \omega^{j \times m} = \sum_{j=0}^{N-1} \omega^{j(m-n)}$$



Properties of the DFT matrix

Theorem

The matrix \mathbf{Q} of the DFT problem satisfies $\mathbf{Q}^* \mathbf{Q} = N\mathbf{I}$. Thus $\mathbf{Q}^{-1} = \frac{1}{N} \mathbf{Q}^*$, and the DFT problem admits a unique solution for any data \mathbf{y} .

Proof.

If \mathbf{q}_j is the j -th column of \mathbf{Q} , we need to show that

$$\mathbf{q}_n^* \mathbf{q}_m = \begin{cases} N & \text{if } n = m \\ 0 & \text{if } n \neq m. \end{cases}$$

We can compute this inner product explicitly

$$\mathbf{q}_n^* \mathbf{q}_m = \sum_{j=0}^{N-1} \omega^{-j \times n} \omega^{j \times m} = \sum_{j=0}^{N-1} \omega^{j(m-n)} = \begin{cases} \frac{1 - \omega^{N(m-n)}}{1 - \omega^{m-n}} = 0, & \text{if } n \neq m, \\ N, & \text{if } n = m. \end{cases}$$



DFT Algorithm – version 1.0

$$\mathbf{c} = \frac{1}{N} \mathbf{Q}^* \mathbf{y}$$

- This is a **matrix-vector** product. **Cost** = N^2 (complex) operations.
- For 1 min of mono CD sound (44100 samples/sec) the signal has about

$$N \approx 2.6 \times 10^6 \text{ samples.}$$

- The number of operations to implement DFT version 1.0 is:

$$N^2 \approx 7.0 \times 10^{12}.$$

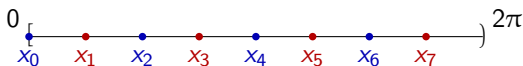
- A good smartphone can handle about 20 MFlops (million floating point ops/sec), so the DFT 1.0 would take about

$$3.5 \times 10^5 \text{ seconds} \approx 97 \text{ hours and 15 minutes.}$$

(this guesstimate is off by a factor of about 4, why?)

Divide and conquer

Let $N = 2M$ and $\omega = e^{2i\pi/N}$.



Cooley and Tukey idea:

Divide interpolation at all nodes into

$q(w) \equiv$ trig polynomial matching data at **even nodes**, $(q(\omega^{2k}) = y_{2k})$

$r(w) \equiv$ trig polynomial matching data at **odd nodes**, $(r(\omega^{2k}) = y_{2k+1})$

for $k = 0, \dots, M - 1$.

Theorem

The trig polynomial interpolating at all nodes is

$$p(w) = \left(\frac{1 + w^M}{2}\right) q(w) + \left(\frac{1 - w^M}{2}\right) r(w/\omega).$$

Proof.

Notice that

$$(\omega^j)^M = e^{(2i\pi/N)j(N/2)} = e^{ij\pi} = \begin{cases} 1 & \text{if } j \text{ is even} \\ -1 & \text{if } j \text{ is odd.} \end{cases}$$

For even $j = 2k$, $k = 0, \dots, M-1$:

$$\begin{aligned} p(\omega^{2k}) &= \underbrace{\left(\frac{1 + (\omega^{2k})^M}{2}\right)}_{=1} q(\omega^{2k}) + \underbrace{\left(\frac{1 - (\omega^{2k})^M}{2}\right)}_{=0} r(\omega^{2k}/\omega) \\ &= q(\omega^{2k}) = y_{2k}. \end{aligned}$$

For odd $j = 2k+1$, $k = 0, \dots, M-1$:

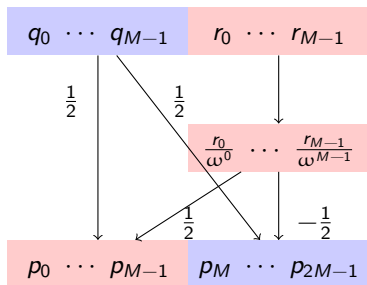
$$\begin{aligned} p(\omega^{2k+1}) &= \underbrace{\left(\frac{1 + (\omega^{2k+1})^M}{2}\right)}_{=0} q(\omega^{2k+1}) + \underbrace{\left(\frac{1 - (\omega^{2k+1})^M}{2}\right)}_{=1} r(\omega^{2k+1}/\omega) \\ &= r(\omega^{2k}) = y_{2k+1}. \end{aligned}$$

□

Show me the Cooley-Tukey DFT algorithm!

Rewrite the trig interpolant at all nodes as:

$$\begin{aligned}
 p(w) &= \left(\frac{1+w^M}{2}\right) q(w) + \left(\frac{1-w^M}{2}\right) r(w/\omega) \\
 &= \frac{1}{2}(q(w) + r(w/\omega)) + \frac{w^M}{2}(q(w) - r(w/\omega)) \\
 &= \frac{1}{2} \left(\underbrace{\sum_{j=0}^{M-1} (q_j + r_j \omega^{-j}) w^j}_{\text{defines } p_0, \dots, p_{M-1}} \right) + \frac{w^M}{2} \left(\underbrace{\sum_{j=0}^{M-1} (q_j - r_j \omega^{-j}) w^j}_{\text{defines } p_M, \dots, p_{2M-1}} \right).
 \end{aligned}$$



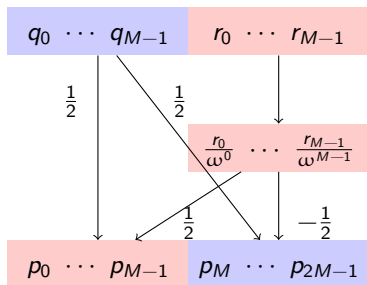
Let $\text{cost}(N) = \#$ multip. in FFT for $|\text{data}| = N$.
In general:

$$\text{cost}(2M) = \quad , \text{ for } M \geq 1.$$

Show me the Cooley-Tukey DFT algorithm!

Rewrite the trig interpolant at all nodes as:

$$\begin{aligned}
 p(w) &= \left(\frac{1+w^M}{2}\right) q(w) + \left(\frac{1-w^M}{2}\right) r(w/\omega) \\
 &= \frac{1}{2}(q(w) + r(w/\omega)) + \frac{w^M}{2}(q(w) - r(w/\omega)) \\
 &= \frac{1}{2} \underbrace{\left(\sum_{j=0}^{M-1} (q_j + r_j \omega^{-j}) w^j\right)}_{\text{defines } p_0, \dots, p_{M-1}} + \frac{w^M}{2} \underbrace{\left(\sum_{j=0}^{M-1} (q_j - r_j \omega^{-j}) w^j\right)}_{\text{defines } p_M, \dots, p_{2M-1}}.
 \end{aligned}$$



Let $\text{cost}(N) = \#$ multip. in FFT for $|\text{data}| = N$.
In general:

$$\text{cost}(2M) = 2\text{cost}(M) + 2M, \text{ for } M \geq 1.$$

```

function c = myfft(f)
N = length(f);
if (N==1)
    c = f;
else
    ce = myfft(f(1:2:N-1)); % even interpolant
    co = myfft(f(2:2:N)); % odd interpolant
    ph = exp(-2i*(0:(N/2-1))'*pi/N);
    c = 0.5*[ ce + ph.*co
              ce - ph.*co ];
end;

```

We have $\text{cost}(N) = N \log_2 N$. Indeed for $N = 2^m$:

$$\text{cost}(2^0) = 0 = 0 \times 2^0,$$

$$\text{cost}(2^1) = 2\text{cost}(2^0) + 2^1 = 2 = 1 \times 2^1$$

$$\text{cost}(2^2) = 2\text{cost}(2^1) + 2^2 = 8 = 2 \times 2^2$$

$$\text{cost}(2^3) = 2\text{cost}(2^2) + 2^3 = 24 = 3 \times 2^3$$

...

Cost comparison with version 1.0

- For 1 min of mono CD sound (44100 samples/sec) the signal has about

$$N \approx 2.6 \times 10^6 \text{ samples.}$$

- The number of operations to implement DFT version 2.0 is:

$$N \log_2 N \approx 5.6 \times 10^7.$$

- A good smartphone can handle about 20 MFlops (million floating point ops/sec), so the DFT 2.0 would take about

2.82 seconds (compare to 97 hours and 15 minutes for version 1.0)

Applications of FFT

- frequency analysis
- digital signal processing (e.g. low-pass and high-pass filters, noise suppression)
- compression of sound and image files
- large integer multiplication
- convolution
- MRI
- cats
- ...

FFTW3: Fastest Fourier Transform in the West

- A modern, widespread, free and open source package for computing FFT.
- Highly optimized, and works on many platforms.
- Works well when N has many small prime factors.



Thank you