Unit 5:

Fourier Theory

Representing data using frequencies. Why bother?

FT-01: Images and Signals

Goal: To introduce the idea of image/signal compression.

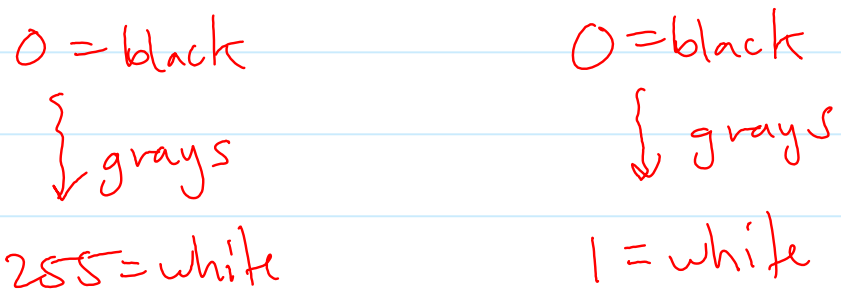
Signal and Image Processing

Digital images are stored as arrays of "picture elements", or pixels.

- Images are stored as an array of numbers

Graylevel ("Intensity") Images

The stored numbers refer to the shade of gray.



With 256 shades of gray, a pixel can be represented by an 8-bit integer. a.k.a. "char" or "uint8"

Colour Images

Consists of 3 intensity images: one for each of red, green, and blue. (why only 3?)

Images (and videos) can take a lot of memory to store.

eg. 1024x1024 pixel image, 24-bit colour

(1 megapixel) = 3 MB

Data versus Information

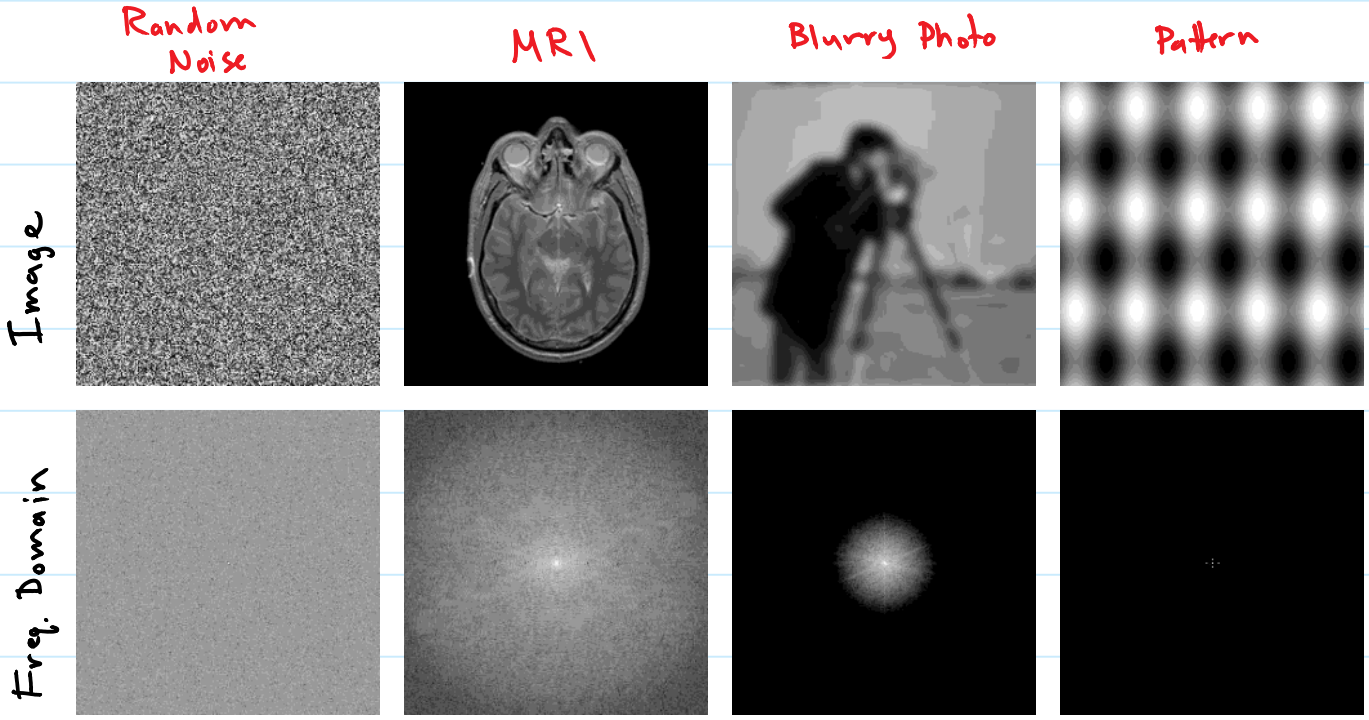
It would be nice to represent the image with less data. In reality, most images hold far less information than the raw memory requirements would suggest. The trick is to find a way to get rid of the unnecessary data.

Eg. Consider the space of all 256x256 pixel 8-bit images. How

many are there?

256 graylevels / 256^2 pixels = 256^{65536} = HUGE!

Included in that set of images...



JPEG Compression

JPEG = "Joint Photographic Experts Group"

- Standard for storing digital image information
- file extensions .jpg or .jpeg

MPEG Compression

MPEG = "Moving Picture Experts Group"

- Standard for storing digital videos.
- MP3 = "MPEG Audio Layer 3"

All these technologies use the Fourier transform to sort out what data to store, and what data to discard.

(audio compression demos)

FT-02: Fourier Series

Goal: To introduce the use of trigonometric functions to approximate other functions.
To review complex numbers.

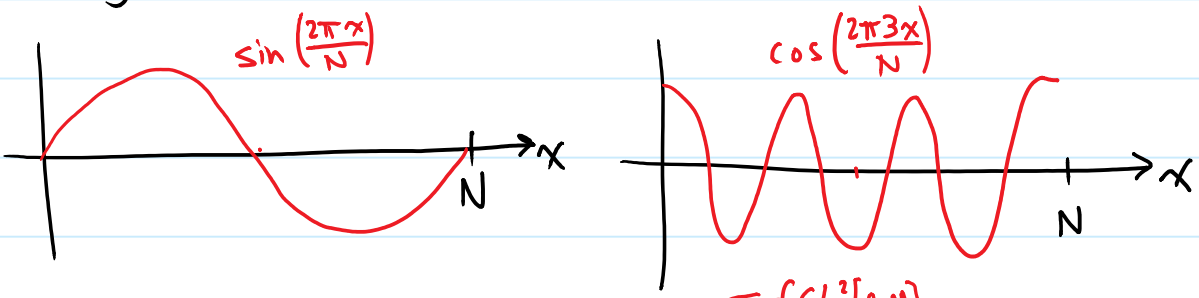
Fourier Series

Consider the trigonometric functions of x,

$$\sin\left(\frac{2\pi k x}{N}\right) \quad \text{and} \quad \cos\left(\frac{2\pi k x}{N}\right) \qquad k \in \mathbb{Z}$$

They repeat when x increases by $\frac{N}{k}$.

Or, they repeat k times in $0 \le x \le N$.



Theorem: Suppose f is a "nice" N-periodic function. ($f \in L^2[0,N]$)
There exist coefficients a_k & b_k such that

$$f(x) = a_0 + \sum_{k=1}^{\infty}\left[a_k \cos\left(\frac{2\pi k x}{N}\right) + b_k \sin\left(\frac{2\pi k x}{N}\right)\right]$$

This is known as a Fourier Series.

In practice, we approximate f with a truncated Fourier Series,

$$f(x) = a_0 + \sum_{k=1}^{m}\left[a_k \cos\left(\frac{2\pi k x}{N}\right) + b_k \sin\left(\frac{2\pi k x}{N}\right)\right] \quad (*)$$

$$f(x) = a_0 + \sum_{k=1} \left[a_k \cos\left(\frac{2\pi k x}{N}\right) + b_k \sin\left(\frac{2\pi k x}{N}\right) \right] \quad (*)$$

Instead of treating the a's and b's separately, we can use the more sophisticated and compact complex notation,

$$f(x) = \sum_{k=-m}^{m} C_k \left(\cos\left(\frac{2\pi k x}{N}\right) + i \sin\left(\frac{2\pi k x}{N}\right) \right)$$

i is complex number

Notice the sum is now from $-m$ to m. Here's why. If $f(x) \in \mathbb{R}$, then we need to make sure all the imaginary parts cancel out.

$$f(x) = a_0 + \sum_{k=1}^{m} \left[C_k \cos\frac{2\pi k x}{N} + i C_k \sin\frac{2\pi k x}{N} + C_{-k} \cos\frac{-2\pi k x}{N} + i C_{-k} \sin\frac{-2\pi k x}{N} \right]$$

$$= a_0 + \sum_{k=1}^{m} \left[(C_k + C_{-k}) \cos\frac{2\pi k x}{N} + i (C_k - C_{-k}) \sin\frac{2\pi k x}{N} \right]$$

Comparing to (*) above...

$$\begin{cases} a_k = C_k + C_{-k} & (1) \\ b_k = i(C_k - C_{-k}) & (2) \end{cases}$$

$(1) + i(2) \Rightarrow \quad a_k + i b_k = 2C_{-k} \Rightarrow C_{-k} = \dfrac{a_k + i b_k}{2}$

$(1) - i(2) \Rightarrow \quad a_k - i b_k = 2C_k \Rightarrow C_k = \dfrac{a_k - i b_k}{2}$

Notice, then, that $c_k = \overline{c_{-k}}$ (complex conjugates)

A Fourier series is a representation of a signal as a linear combination of waves of varying frequencies. The Fourier coefficients a_k, b_k (or c_k) tell us the amplitude of each frequency. The Fourier transform is a frequency decomposition of a signal.

(Matlab demo - fourier_series_demo.m)

<u>Uses of Frequency Decomposition</u>

Compression: Many of the types of data we perceive are dominated by low-frequency information.

eg. audio, images, videos

Filtering: eg. denoising/smoothing
Pitch Adjustment: eg. "auto-tune"
Ranging: eg. RADAR, SONAR
Medical imaging, astronomy, WiFi/Cell phones, scientific analysis, etc.

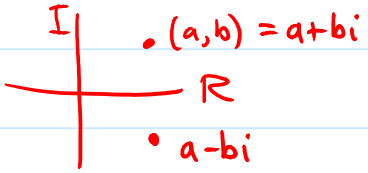
<u>Complex Numbers</u> (remember those?)

Basic facts:

(1) $z = a + bi$ a = real part b = imaginary part

$i = \sqrt{-1}$

(2) $\bar{z} = a - bi$ complex conjugate



(3) $z_1 + z_2 = (a_1 + b_1 i) + (a_2 + b_2 i)$

$= (a_1 + a_2) + (b_1 + b_2) i$

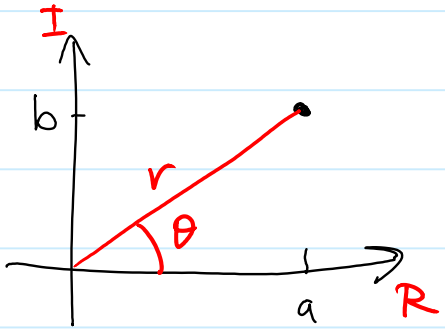
(4) $z_1 z_2 = (a_1 + b_1 i)(a_2 + b_2 i)$

$= a_1 a_2 \; a_1 b_2 i + a_2 b_1 i + b_1 b_2 \underbrace{i^2}_{=-1}$

$= (a_1 a_2 - b_1 b_2) + (a_1 b_2 + a_2 b_1) i$

(5) $|z|^2 = z\bar{z} = a^2 + b^2$

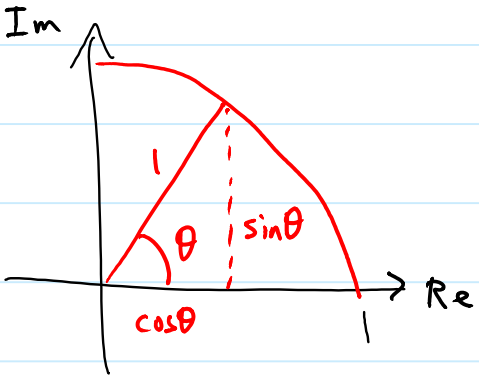
$|z|$ = modulus of z



(6) $z = re^{i\theta} = a + bi$

Euler's Identity

$$e^{i\theta} = \cos\theta + i\sin\theta$$



Conversely

$$\begin{cases} \cos\theta = \dfrac{e^{i\theta} + e^{-i\theta}}{2} \\ \sin\theta = \dfrac{e^{i\theta} - e^{-i\theta}}{2i} \end{cases}$$

This polar form is often very convenient. For example, let's multiply two complex numbers.

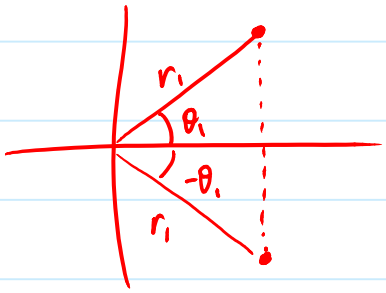
$$\left.\begin{aligned} z_1 &= r_1 e^{i\theta_1} \\ z_2 &= r_2 e^{i\theta_2} \end{aligned}\right\} \quad z_1 z_2 = (r_1 e^{i\theta_1})(r_2 e^{i\theta_2})$$

$$= r_1 r_2 e^{i(\theta_1 + \theta_2)}$$

How about taking the complex conjugate?

How about taking the complex conjugate?

$$\bar{z}_1 = r_1 e^{-i\theta_1}$$



FT-03: Trigonometric Orthogonality

Goal: To derive the orthogonality behind what makes Fourier theory so useful.

Complex Inner Product

Let a and b be complex vectors of length N.

Then $\langle a, b \rangle = \sum_{k=1}^{N} a_k \bar{b}_k$

Thus $\langle a, b \rangle = \overline{\langle b, a \rangle}$

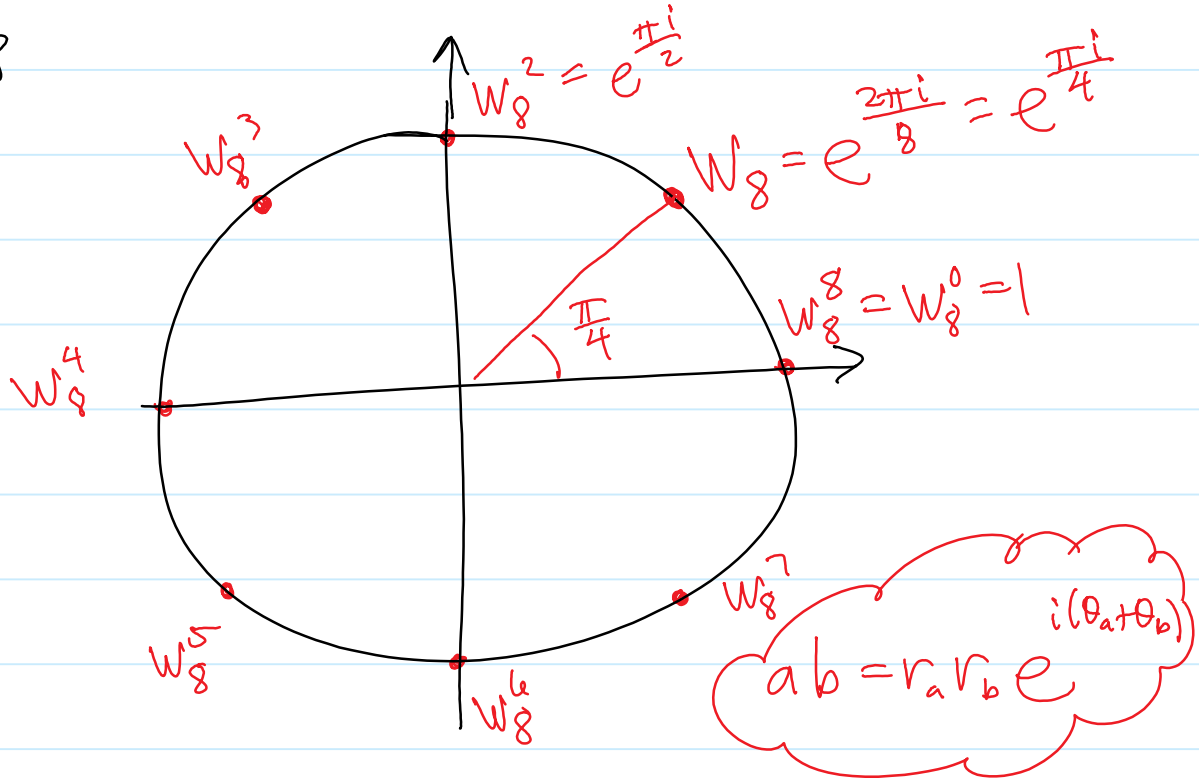
The N^{th} Root of Unity

Define the N^{th} root of unity, W_N, as

$$W_N = e^{\frac{2\pi i}{N}}$$

Then, $W_N^N = \left(e^{\frac{2\pi i}{N}}\right)^N = e^{2\pi i} = 1$

eg. $N = 8$



Consider the vector defined by

Consider the vector defined by

$$W_N(n) = \left(1, W_N^n, W_N^{2n}, \cdots, W_N^{(N-1)n}\right)$$

Note: $W_N^n = \left(e^{\frac{2\pi i}{N}}\right)^n = e^{\frac{2\pi i n}{N}}$

$$W_N(n)_k = W_N^{nk} = e^{\frac{2\pi i n k}{N}}$$

Orthogonality of $W(n)$ and $W(m)$.

$$\langle W_N(n), W_N(m) \rangle = \sum_{k=0}^{N-1} W_N(n)_k \overline{W_N(m)_k}$$

$$= \sum_{k=0}^{N-1} e^{\frac{2\pi i n k}{N}} e^{\frac{-2\pi i m k}{N}}$$

$$= \sum_{k=0}^{N-1} \left[e^{\frac{2\pi i (n-m)}{N}} \right]^k$$

= geometric sum

Formula: $\sum_{k=0}^{N-1} r^j = \frac{1-r^N}{1-r}$ $\quad r \neq 1$

In our case, $r = e^{\frac{2\pi i (n-m)}{N}}$ (assume $n \neq m$)

Then

$$\langle W_N(n), W_N(m) \rangle = \frac{1 - e^{\frac{2\pi i (n-m) \cancel{N}}{\cancel{N}}}}{1 - e^{\frac{2\pi i (n-m)}{N}}} = \frac{1-1}{1-\text{blah}} = 0$$

If $n = m$,

$$\langle W_N(n), W_N(m) \rangle = \sum_{k=0}^{N-1} 1 = N$$

$$\therefore \langle W_N(n), W_N(m) \rangle = \begin{cases} 0 & \text{if } n \neq m \\ N & \text{if } n = m \end{cases}$$

Goal: To derive the Discrete Fourier Transform (DFT) and its inverse.

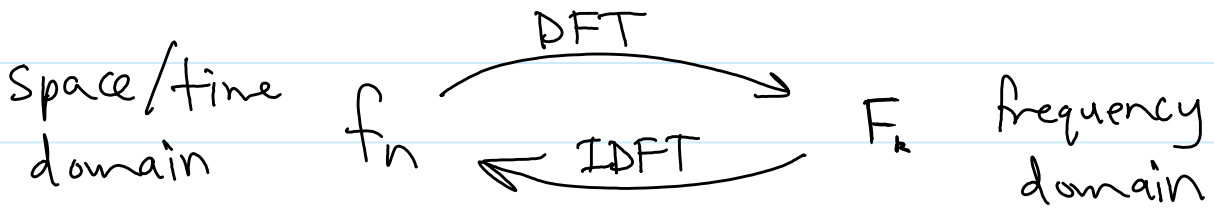
Discrete Fourier Transform

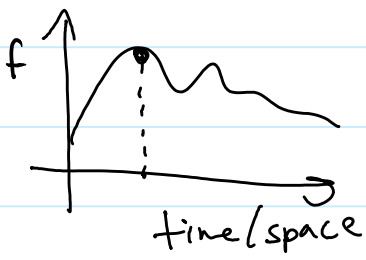
Consider the sampled signal $\{f_n \in \mathbb{C} \mid n = 0, \ldots, N-1\}$. We can think of the signal as an N-dimensional vector $\vec{f}$. We can represent f as a linear combination of our Fourier basis vectors,

$$f_n = \frac{1}{N} \sum_{k=0}^{N-1} F_k W_N^{nk} = \frac{1}{N} \langle \vec{F}, \overline{W_N(n)} \rangle \quad \text{IDFT}$$

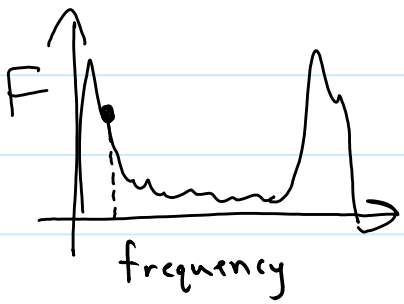
where $W_N = e^{\frac{2\pi i}{N}}$

$$F_k = \sum_{n=0}^{N-1} f_n \overline{W}_N^{nk} = \langle \vec{f}, W_N(k) \rangle \quad \text{DFT}$$





Measures the signal at each time or location



Measures the contribution of each freq. to the overall signal.

(run fftgui.m and compression_1D_demo.m)

Matrix Form of the DFT

$$F_k = \sum_{n=0}^{N-1} f_n \bar{W}^{nk} \qquad k = 0, \ldots, N-1$$

$$F_0 = f_0 \bar{W}^{0\cdot 0} + f_1 \bar{W}^{1\cdot 0} + \cdots + f_{N-1} \bar{W}^{(N-1)\cdot 0} = \sum f_n$$

$$F_1 = f_0 \bar{W}^{0\cdot 1} + f_1 \bar{W}^{1\cdot 1} + \cdots + f_{N-1} \bar{W}^{(N-1)\cdot 1}$$

$$\vdots$$

$$F_{N-1} = f_0 \bar{W}^{0\cdot(N-1)} + f_1 \bar{W}^{1\cdot(N-1)} + \cdots + f_{N-1} \bar{W}^{(N-1)\cdot(N-1)}$$

Using matrix-vector notation…

$$\begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ \vdots \\ F_{N-1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \bar{W}^1 & \bar{W}^2 & & \bar{W}^{N-1} \\ 1 & \bar{W}^2 & \bar{W}^4 & & \bar{W}^{2(N-1)} \\ 1 & \bar{W}^3 & \bar{W}^6 & & \bar{W}^{3(N-1)} \\ \vdots & & & & \\ 1 & \bar{W}^{N-1} & \bar{W}^{2(N-1)} & \cdots & \bar{W}^{(N-1)(N-1)} \end{bmatrix}}_{M \text{ (symmetric)}} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{N-1} \end{bmatrix}$$

Hence, we get $F = Mf$

Recall that $\overline{W_N(n)} = \left[\bar{W}_N^{ok} \right]$

Recall that $\overline{W_N}(n) = \begin{bmatrix} \overline{W}_N^{0k} \\ \overline{W}_N^{1k} \\ \overline{W}_N^{2k} \\ \vdots \\ \overline{W}_N^{(N-1)k} \end{bmatrix}$

So $M = \left[\overline{W_N}(0) \middle| \overline{W_N}(1) \middle| \cdots \middle| \overline{W_N}(N-1) \right]$

Notice that $M = M^T = \begin{bmatrix} \overline{W}(0)^T \\ \vdots \\ \overline{W}(N-1)^T \end{bmatrix}$

Consider $M^T M = \begin{bmatrix} W(0)^T \\ \vdots \\ W(N-1)^T \end{bmatrix} \left[\overline{W}(0) \middle| \cdots \middle| \overline{W}(N-1) \right]$

$$= \begin{bmatrix} \langle W(0), W(0) \rangle & \langle W(0), W(1) \rangle & \cdots \\ \langle W(1), W(0) \rangle & \langle W(1), W(1) \rangle & \cdots \\ \vdots & \vdots & \\ & & \end{bmatrix}$$

$$= N \begin{bmatrix} 1 & & & 0 \\ & 1 & & \\ & & \ddots & \\ 0 & & & 1 \end{bmatrix} = N I$$

$$\bar{M}^T M = N I \Rightarrow \left(\frac{1}{N}\bar{M}\right) M = I$$

$$\therefore M^{-1} = \frac{1}{N}\bar{M}$$

Recall the DFT is $F = Mf$

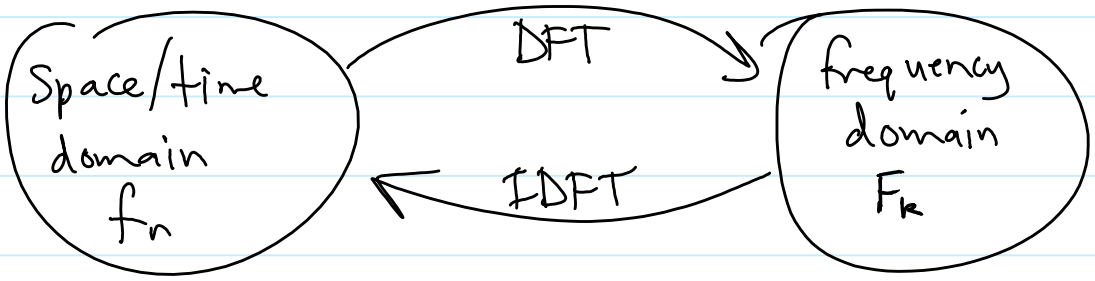
$$\Rightarrow f = M^{-1}F = \frac{1}{N}\bar{M}F$$

This is the inverse DFT (IDFT). It takes Fourier coefficients, and converts them to the spatial or temporal domain. So, the IDFT can be written

$$\begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \end{bmatrix} = \frac{1}{N} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W^1 & W^2 & & W^{N-1} \\ 1 & W^2 & W^4 & & W^{2(N-1)} \\ \vdots & & & & \\ 1 & W^{N-1} & \cdots & & W^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} F_0 \\ F_1 \\ \vdots \\ F_{N-1} \end{bmatrix}$$

In summation notation,

$$f_n = \frac{1}{N} \sum_{k=0}^{N-1} F_k W^{nk} \qquad n = 0, \ldots, N-1$$



Given the Fourier coefficients, $\{F_k \in \mathbb{C}, k = 0, \ldots, N-1\}$,
the IDFT gives the signal composed of the frequencies with amplitudes F_k.

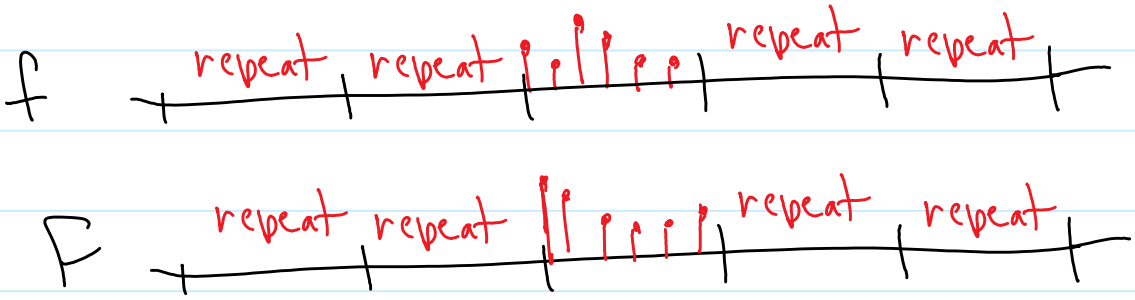
(Matlab demo - dftMatrix_demo.m)

FT-05: Properties of the Fourier Transform

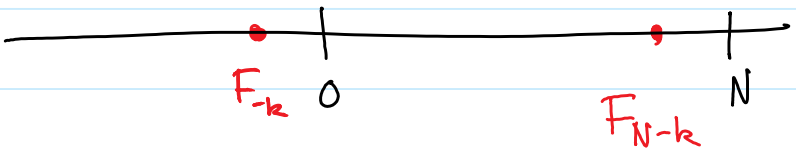
Goal: To see the amazing things that the Fourier Transform can do.

Periodicity

By the nature of the DFT, both f and F are periodic. However, we only store and manipulate one period.



Proof: I will show that $F_k = F_{N-k}$



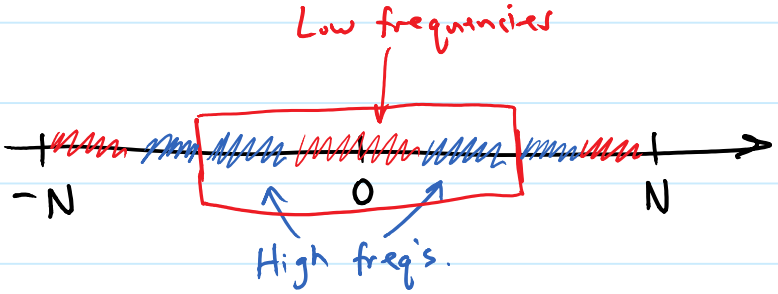
$$F_{N-k} = \sum_{n=0}^{N-1} f_n W_N^{n(N-k)}$$

$e^{\frac{-2\pi i n N}{N}} = 1$

$$= \sum_{n=0}^{N-1} f_n W_N^{nN} W_N^{-nk}$$

$$= \sum_{n=0}^{N-1} f_n W_N^{n(-k)} = F_{-k}$$

QED



When looking at the Fourier coefficients, we can look at $[0, N-1]$ or we can look at

N odd: $\left[\frac{-(N-1)}{2}, \frac{N-1}{2}\right]$ eg. $\underset{N=5}{[0\ 1\ 2\ 3\ 4]} \to [-2\ -1\ 0\ 1\ 2]$

N even: $\left[-\frac{N}{2}, \frac{N}{2}-1\right]$ eg. $[0\ 1\ 2\ 3\ 4\ 5] \to [-3\ -2\ -1\ 0\ 1\ 2]$

Matlab's commands "fftshift" and "ifftshift" do this transformation for you.

(Matlab demo - sound_demos.m)

Conjugate Symmetry

If your signal f is real-valued,

i.e. $f_n = \overline{f_n}$, then...

$$F_{N-k} = \sum_{n=0}^{N-1} f_n \overline{W_N^{n(N-k)}}$$

$$= \sum_{n=0}^{N-1} f_n \overline{W_N^{nN}}\ \overline{W_N^{(-nk)}} \qquad e^{\frac{-2\pi i(-nk)}{N}} = W_N^{nk}$$

$$= \sum_{n=0}^{N-1} f_n W_N^{nk}$$

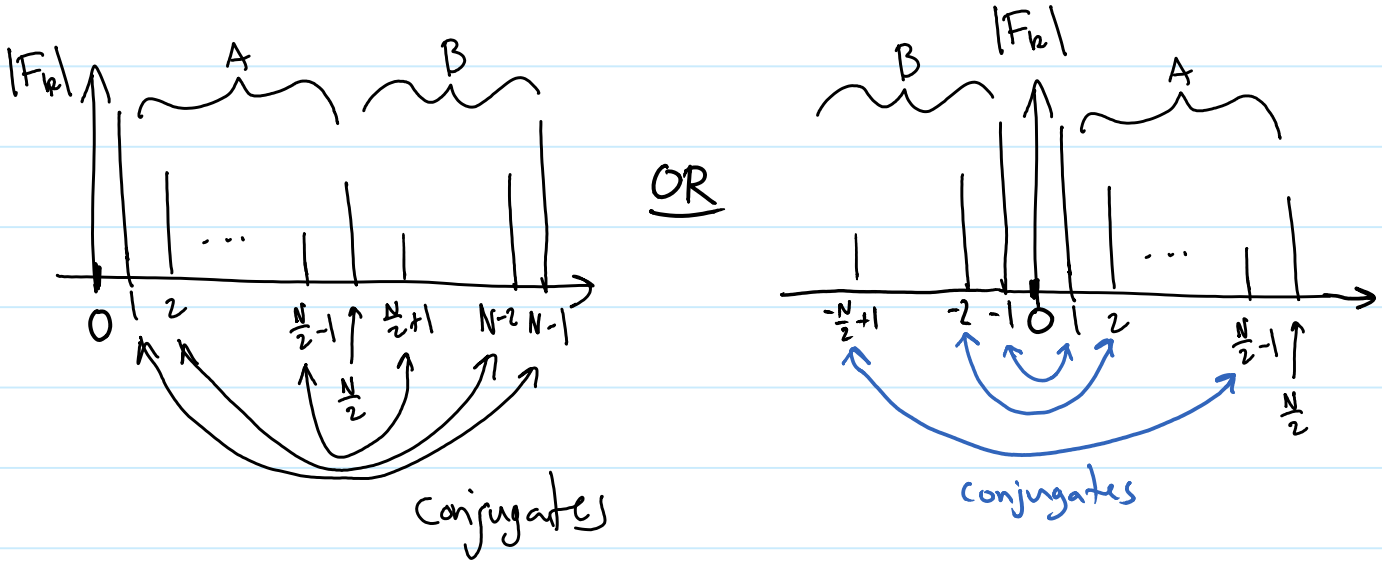
$$= \overline{\sum_{n=0}^{N-1} f_n \overline{W_N^{nk}}}$$

$$= \overline{F_k}$$

F_n-k=F_-k (always)
= conjugate F_k (if f is real)

So, $f_{N-k} = \overline{F_k}$.

Consequently, $|F_{N-k}| = |F_{-k}| = |\overline{F_k}|$



2-Dimensional DFT

A graylevel image is represented by a 2-D array

$$f_{nj} \qquad n = 0, 1, \ldots, N-1$$
$$j = 0, 1, \ldots, M-1$$

The 2-D DFT is defined as

$$\text{2D-DFT}: \quad F_{kl} = \sum_{n=0}^{N-1} \sum_{j=0}^{M-1} f_{nj} \overline{W_N}^{nk} \overline{W_M}^{jl}$$

$$\text{2D-IDFT}: \quad f_{nj} = \frac{1}{MN} \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} F_{kl} W_N^{nk} W_M^{jl}$$

How to Compute a 2-D DFT Using 1-D DFT

$$F_{kl} = \sum_{n=0}^{N-1} \sum_{j=0}^{M-1} f_{nj} \bar{W}_N^{nk} \bar{W}_M^{jl}$$

$$= \sum_{n=0}^{N-1} \bar{W}_N^{nk} \underbrace{\left[\sum_{j=0}^{M-1} f_{nj} \bar{W}_M^{jl} \right]}_{H_{nl} \leftarrow \text{n is fixed (row)}}$$

H_{nl} = DFT of n^{th} row of f, $n = 0, 1, \ldots, N-1$

$\Rightarrow$ N 1D-DFT's

After that,

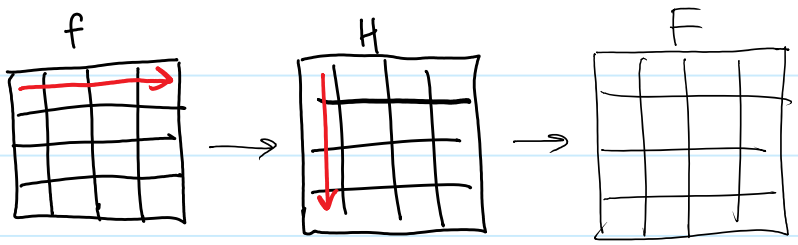
$$F_{kl} = \sum_{n=0}^{N-1} H_{nl} \bar{W}_N^{nk}$$

l is fixed

$k = 0, 1, \ldots, M-1$

This is the DFT of the l^{th} column of H

$\Rightarrow$ M 1D-DFT's



In MatLab, use fft2 & ifft2.

(Fourier image demos)

FT-06: More Properties of the Fourier Transform

Goal: To see the amazing things that the Fourier Transform can do.

Convolution

A convolution is a sum (integral) of the form

$$(f * g)_a = \sum_{n=0}^{N-1} f_n \cdot g_{a-n}$$

$$\sum_{b=0}^{N-1} g_b \bar{W}_N^{(b+n)k} = \sum_{b=0}^{N-1} g_b \bar{W}^{bk} \bar{W}^{nk}$$

Just for fun, let's take the DFT of $(f*g)_a$.

$$DFT(f*g)_k = \sum_{a=0}^{N-1} (f*g)_a \bar{W}^{ak}$$

$$= \sum_{a=0}^{N-1} \sum_{n=0}^{N-1} f_n g_{a-n} \bar{W}^{ak} \times \overbrace{\bar{W}^{nk} \bar{W}^{-nk}}^{=1}$$

$$= \sum_{n=0}^{N-1} \left[f_n \bar{W}^{nk} \sum_{a=0}^{N-1} g_{a-n} \bar{W}^{ak-nk} \right]$$

Change of variables: Let $b = a - n \Rightarrow a = b + n$

$$= \sum_{n=0}^{N-1} f_n \bar{W}^{nk} \sum_{b=-n}^{N-1-n} g_b \bar{W}^{bk}$$

$$= \sum_{n=0}^{N-1} f_n \bar{W}^{nk} \sum_{b=0}^{N-1} g_b \bar{W}^{bk}$$

$$= F_k \cdot G_k$$

Thus, we can evaluate a convolution using the Fourier transform.

Since, $DFT(f*g)_k = F_k G_k$

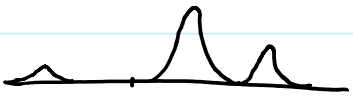
$$(f*g)_a = DFT^{-1}(FG)$$

element-wise multiplication $\Rightarrow O(N)$

Example 1: One thing that convolution is used for is blurring.

$f =$  spikes

$g =$  Gaussian function $g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$

$f * g =$ 

Example 2: It can also be used to compute the correlation coefficient.

$$CC(a; f,g) = \frac{\sum_{n=0}^{N-1} f_n g_{n-a}}{\|f\|^2 \|g\|^2} = \frac{(f * \bar{g})_a}{\|f\|^2 \|g\|^2}$$

$\bar{g}_n = g_{-n}$

If CC = 0 $\Rightarrow$ signals are uncorrelated (orthogonal)

If CC = ± 1 $\Rightarrow$ signals are (negatively) correlated

$f \propto g \iff f = Kg$ for some $K \neq 0$

Try taking the DFT of a DFT.

Goal: To learn how the FFT algorithm works to compute the DFT so efficiently.

Fast Fourier Transform

The Fast Fourier Transform is one of the most important computational algorithms today. Here's how it works…

Divide and Conquer

$$F_k = \sum_{n=0}^{N-1} f_n \bar{W}_N^{nk}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} f_n \bar{W}_N^{nk} + \underbrace{\sum_{n=\frac{N}{2}}^{N-1} f_n \bar{W}_N^{nk}}_{(*)}$$

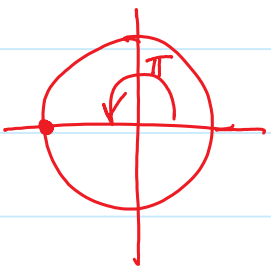
Change of index: $m = n - \frac{N}{2}$ for $(*)$

$$(*) = \sum_{m=0}^{\frac{N}{2}-1} f_{m+\frac{N}{2}} \bar{W}_N^{(m+\frac{N}{2})k}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} f_{n+\frac{N}{2}} \bar{W}_N^{nk} \bar{W}_N^{\frac{Nk}{2}}$$

$$F_k = \sum_{n=0}^{\frac{N}{2}-1} \left(f_n + f_{n+\frac{N}{2}} \bar{W}_N^{\frac{Nk}{2}} \right) \bar{W}_N^{nk}$$

$$\bar{W}_N^{\frac{Nk}{2}} = e^{\frac{-2\pi i N k}{2N}} = e^{-\pi i k} = (-1)^k$$



For even indices (of the form $2 \cdot \ell$)

For even indices (of the form $2j$)

$$F_{2j} = \sum_{n=0}^{\frac{N}{2}-1} \left(f_n + f_{n+\frac{N}{2}}\right) \overline{W}_N^{2nj} \qquad j = 0, 1, \ldots, \frac{N}{2}-1$$

For odd indices (of the form $2j+1$)

$$F_{2j+1} = \sum_{n=0}^{\frac{N}{2}-1} \left(f_n - f_{n+\frac{N}{2}}\right) \overline{W}_N^{n(2j+1)}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} \left[\left(f_n - f_{n+\frac{N}{2}}\right) \overline{W}_N^{n} \right] \overline{W}_N^{2nj} \qquad j = 0, \ldots, \frac{N}{2}-1$$

How do we interpret $\overline{W}_N^{2nj}$?

$$\overline{W}_N^{2nj} = e^{\frac{-2\pi i 2nj}{N}} = e^{\frac{-2\pi i nj}{N/2}} = \overline{W}_{\frac{N}{2}}^{nj}$$

Let

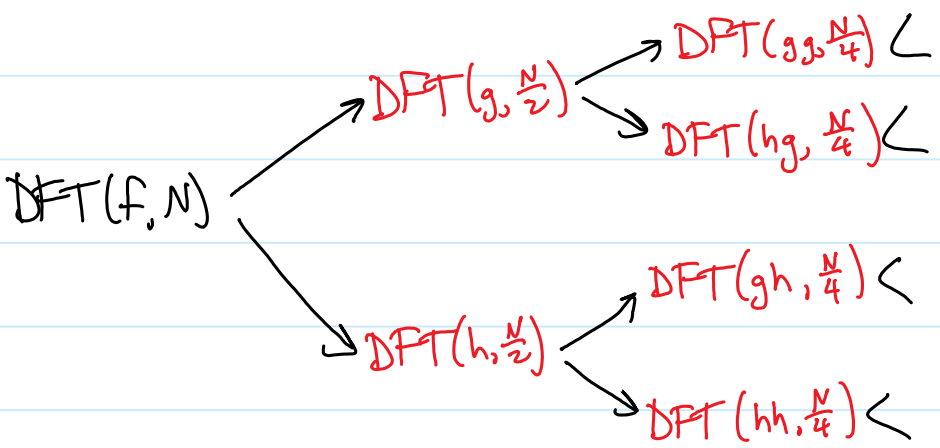
$$\left.\begin{aligned} g_n &= f_n + f_{n+\frac{N}{2}} \\ h_n &= \left(f_n - f_{n+\frac{N}{2}}\right) \overline{W}_N^{n} \end{aligned}\right\} \quad n = 0, \ldots, \frac{N}{2}-1$$

Then

$$F_{2j} = \sum_{n=0}^{\frac{N}{2}-1} g_n \overline{W}_{\frac{N}{2}}^{nj} = \mathrm{DFT}\left(g, \tfrac{N}{2}\right)$$

$$F_{2j+1} = \sum_{n=0}^{\frac{N}{2}-1} h_n \overline{W}_{\frac{N}{2}}^{nj} = \mathrm{DFT}\left(h, \tfrac{N}{2}\right)$$

$$F_{2j+1} = \sum_{n=0}^{} h_n W_{\frac{N}{2}}^{\cdot} = DFT\left(h, \frac{N}{2}\right)$$



Eventually, you get to a DFT of a vector of length 1… which is **just the vector itself**.

i.e. $c \in \mathbb{C} \Rightarrow DFT(c, 1) = c$

Recursive FFT Algorithm

function $\{F_k\} = FFT(\{f_n\}, N)$

if $N = 1$

$F_0 = f_0$

else

for $n = 0$ to $\frac{N}{2} - 1$

$g_n = f_n + f_{n+\frac{N}{2}}$

$h_n = \left(f_n - f_{n+\frac{N}{2}}\right) W_N^n$

end

$\{F_{2k}\} = FFT\left(\{g_n\}, \frac{N}{2}\right)$

$\{F_{2k+1}\} = FFT(\{h_n\}, \frac{N}{2})$

end

Complexity

DFT by direct computation is $O(N^2)$.

Using FFT algorithm:

→ $\log_2 N$ steps of recursion

→ $O(N)$ flops per step

FFT Butterfly Diagram