# Fourier Transforms

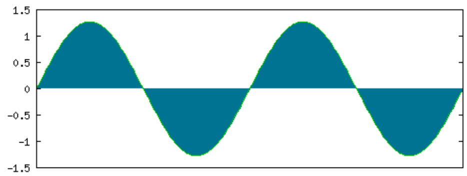
Mark Handley

## Fourier Series

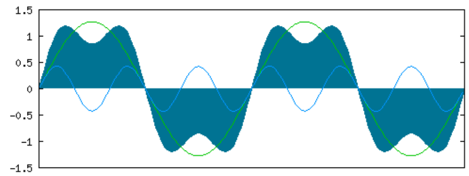
- Any periodic function can be expressed as the sum of a series of sines and cosines (or varying amplitudes)

## Square Wave

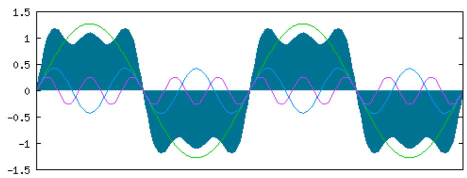
Frequencies: *f*



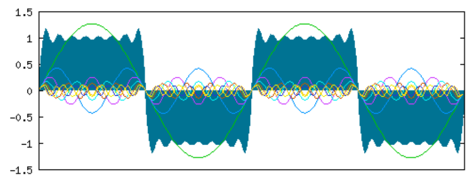

Frequencies: *f* + 3*f*




Frequencies: *f* + 3*f* + 5*f*

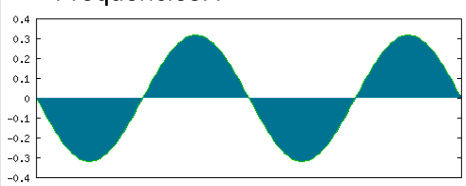



Frequencies: *f* + 3*f* + 5*f* + … + 15*f*

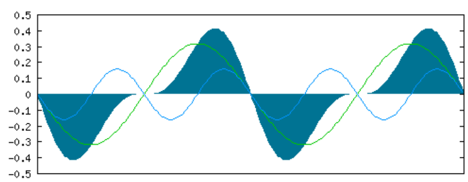



## Sawtooth Wave

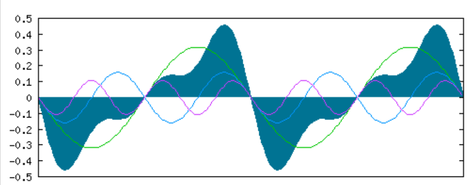
Frequencies: *f*



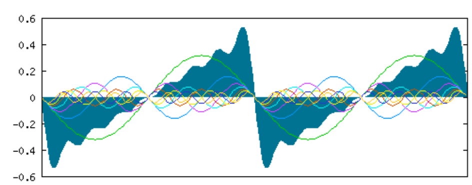

Frequencies: *f* + 2*f*




Frequencies: *f* + 2*f* + 3*f*




Frequencies: *f* + 2*f* + 3*f* + … + 8*f*



## Fourier Series

A function $f(x)$ can be expressed as a series of sines and cosines:

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n \cos(nx) + \sum_{n=1}^{\infty} b_n \sin(nx),$$

where:

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)dx$$

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(nx)dx$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(nx)dx$$

$n = 1, 2, 3, \ldots$

## Fourier Transform

- Fourier Series can be generalized to complex numbers, and further generalized to derive the *Fourier Transform*.

Forward Fourier Transform:

$$F(k) = \int_{-\infty}^{\infty} f(x)e^{-2\pi ikx}dk$$

Inverse Fourier Transform:

$$f(x) = \int_{-\infty}^{\infty} F(k)e^{2\pi ikx}dk$$

Note: $e^{xi} = \cos(x) + i\sin(x)$

## Fourier Transform

- Fourier Transform maps a time series (eg audio samples) into the series of frequencies (their amplitudes and phases) that composed the time series.
- Inverse Fourier Transform maps the series of frequencies (their amplitudes and phases) back into the corresponding time series.
- The two functions are inverses of each other.

## Discrete Fourier Transform

- If we wish to find the frequency spectrum of a function that we have *sampled*, the continuous Fourier Transform is not so useful.
- We need a discrete version:
  - *Discrete Fourier Transform*

## Discrete Fourier Transform

Forward DFT:

$$F_n = \sum_{k=0}^{N-1} f_k e^{-2\pi i n k/N}$$

The complex numbers $f_0 \ldots f_N$ are transformed into complex numbers $F_0 \ldots F_n$

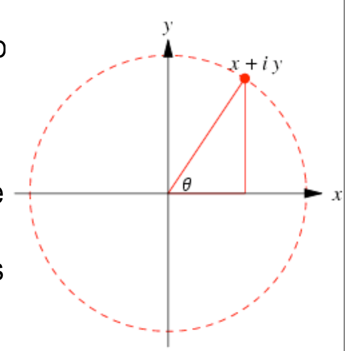
Inverse DFT:

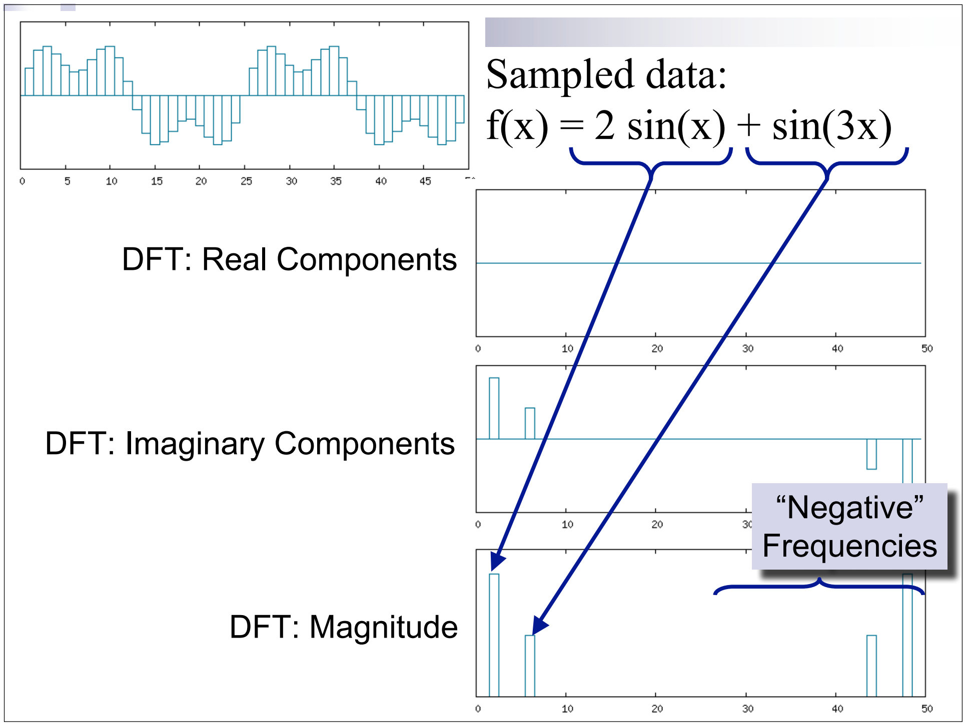
$$f_k = \frac{1}{N}\sum_{n=0}^{N-1} F_n e^{-2\pi i k n/N}$$

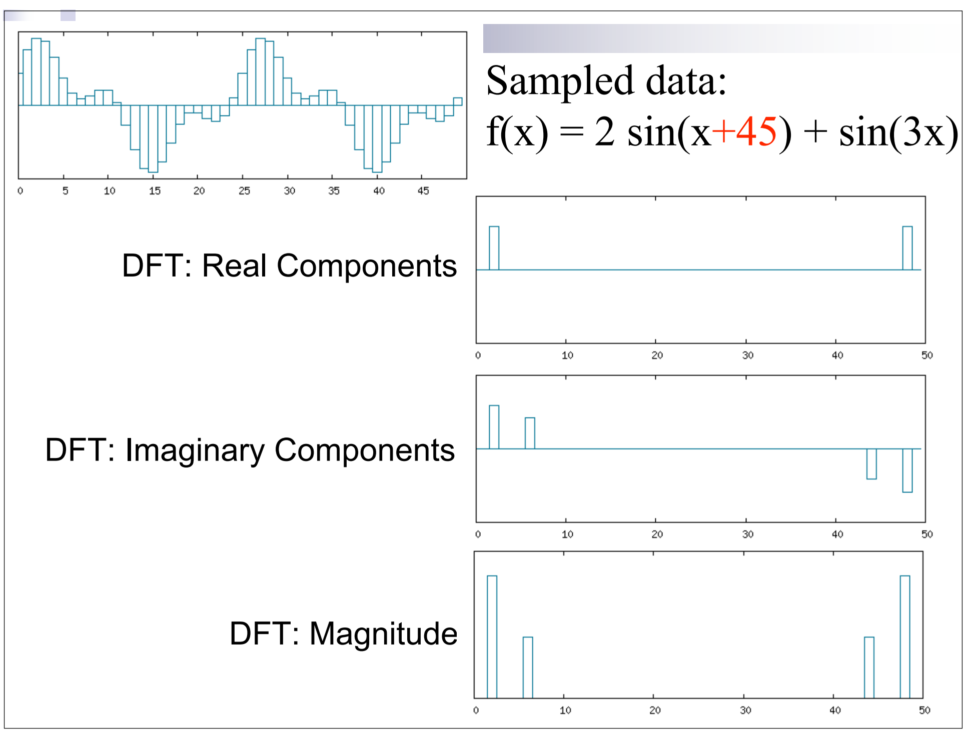
The complex numbers $F_0 \ldots F_n$ are transformed into complex numbers $f_0 \ldots f_N$

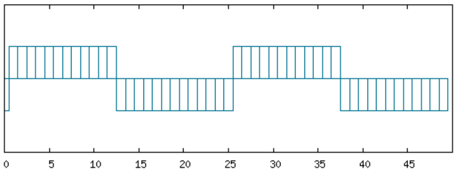
## DFT Example

- Interpreting a DFT can be slightly difficult, because the DFT of real data includes complex numbers.
- Basically:
  - The magnitude of the complex number fo a DFT component is the power at that frequency.
  - The phase $\theta$ of the waveform can be determined from the relative values of the real and imaginary coefficents.
- Also both positive and "negative" frequencies show up.



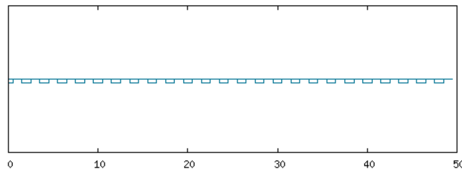

Sampled data:
f(x) = 2 sin(x) + sin(3x)
DFT: Real Components
DFT: Imaginary Components
DFT: Magnitude
“Negative”
Frequencies


Sampled data:
f(x) = 2 sin(x+45) + sin(3x)
DFT: Real Components
DFT: Imaginary Components
DFT: Magnitude

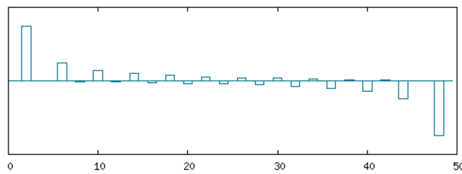



Sampled data:
Square wave

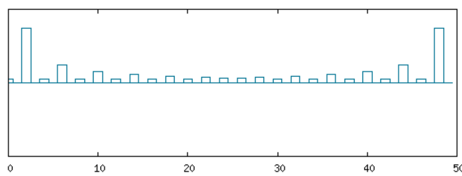
DFT: Real Components




DFT: Imaginary Components




DFT: Magnitude




## Fast Fourier Transform

- Discrete Fourier Transform would normally require $O(n^2)$ time to process for $n$ samples:

$$F_n = \sum_{k=0}^{N-1} f_k e^{-2\pi i n k / N}$$

- Don't usually calculate it this way in practice.
  - Fast Fourier Transform takes $O(n \log(n))$ time.
  - Most common algorithm is the Cooley-Tukey Algorithm.

## Fourier Cosine Transform

Any function can be split into even and odd parts:

$$f(x) = \frac{1}{2}[f(x) + f(-x)] + \frac{1}{2}[f(x) - f(-x)] = E(x) + O(x)$$

Then the Fourier Transform can be re-expressed as:

$$F(k) = \int_{-\infty}^{\infty} E(x)\cos(2\pi kx)dx \; - i\int_{-\infty}^{\infty} O(x)\sin(2\pi kx)dx$$

## Discrete Cosine Transform (DCT)

- When the input data contains only real numbers from an even function, the sin component of the DFT is 0, and the DFT becomes a *Discrete Cosine Transform* (DCT)
- There are 8 variants however, of which 4 are common.

## DCT Types

**DCT Type II**

- Used in JPEG, repeated for a 2-D transform.

$$f_j = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N} j \left(n + \frac{1}{2}\right)\right]$$

- Most common DCT.

## DCT Types

**DCT Type IV**

- Used in MP3.

$$f_j = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N} \left(j + \frac{1}{2}\right)\left(n + \frac{1}{2}\right)\right]$$

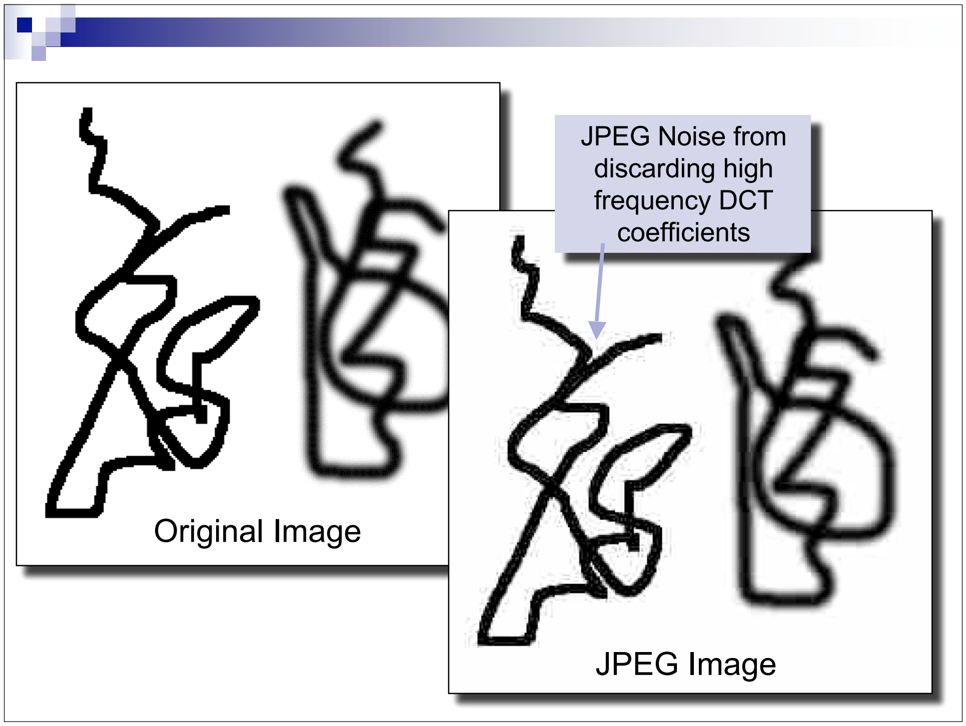
- In MP3, the data is overlapped so that half the data from one sample set is reused in the next.
  - Known as Modified DCT or MDCT
  - This reduces boundary effects.

## Why do we use DCT for Multimedia?

- For audio:
  - Human ear has different dynamic range for different frequencies.
  - Transform to from time domain to frequency domain, and quantize different frequencies differently.
- For images and video:
  - Human eye is less sensitive to fine detail.
  - Transform from spacial domain to frequency domain, and quantize high frequencies more coarsely (or not at all)
  - Has the effect of slightly blurring the image - may not be perceptable if done right.

## Why use DCT/DFT?

- Some tasks are much easier to handle in the frequency domain that in the time domain.
- Eg: graphic equalizer. We want to boost the bass:
  1. Transform to frequency domain.
  2. Increase the magnitude of low frequency components.
  3. Transform back to time domain.


JPEG Noise from discarding high frequency DCT coefficients
Original Image
JPEG Image