LING 490 - SPECIAL TOPICS IN LINGUISTICS

Fundamentals of Digital Signal Processing

Yan Tang

Department of Linguistics, UIUC

Week 8

Last week…

- Linear combination equation:

$$y(t) = \sum_{j=0}^{M} b_j x(t-j) - \sum_{i=1}^{N} a_i y(t-i)$$

  - Input coefficients: $b_0, b_1, b_2 \ldots b_n$
  - Output coefficients: $a_0, a_1, a_2 \ldots a_n$

Last week…

- System and impulse response (IR)
  - Characterise system by knowing the IR
- Convolution: the way to generate the output of a system, given an input and the IR of the system

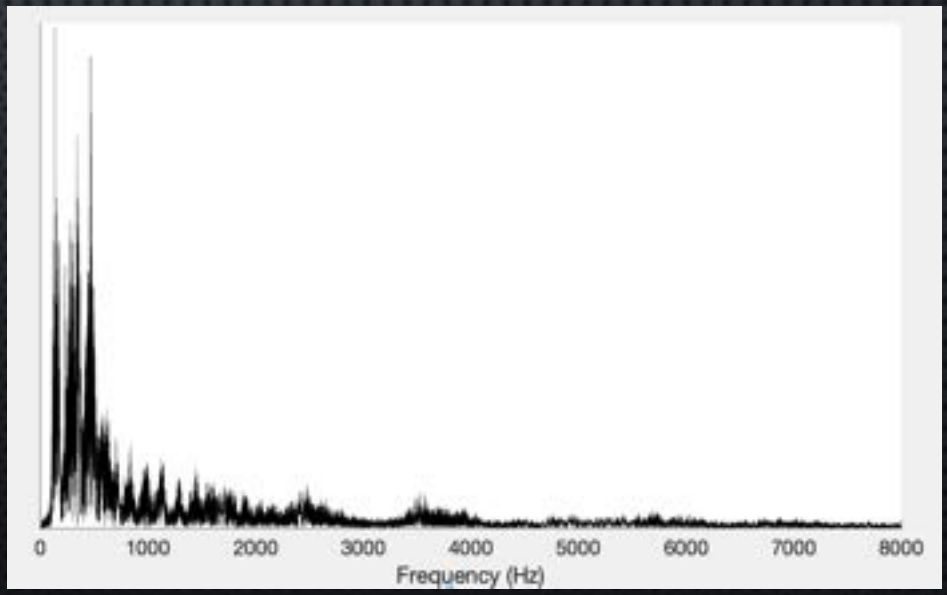
$$y(t) = \sum_i x(i)h(t-i) = \sum_i h(i)x(t-i)$$

$$y(t) = x(t) \otimes h(t) = h(t) \otimes x(t)$$

  - Time-domain linear convolution: slow!!!
  - Convolution to multiplication: go to the frequency domain…

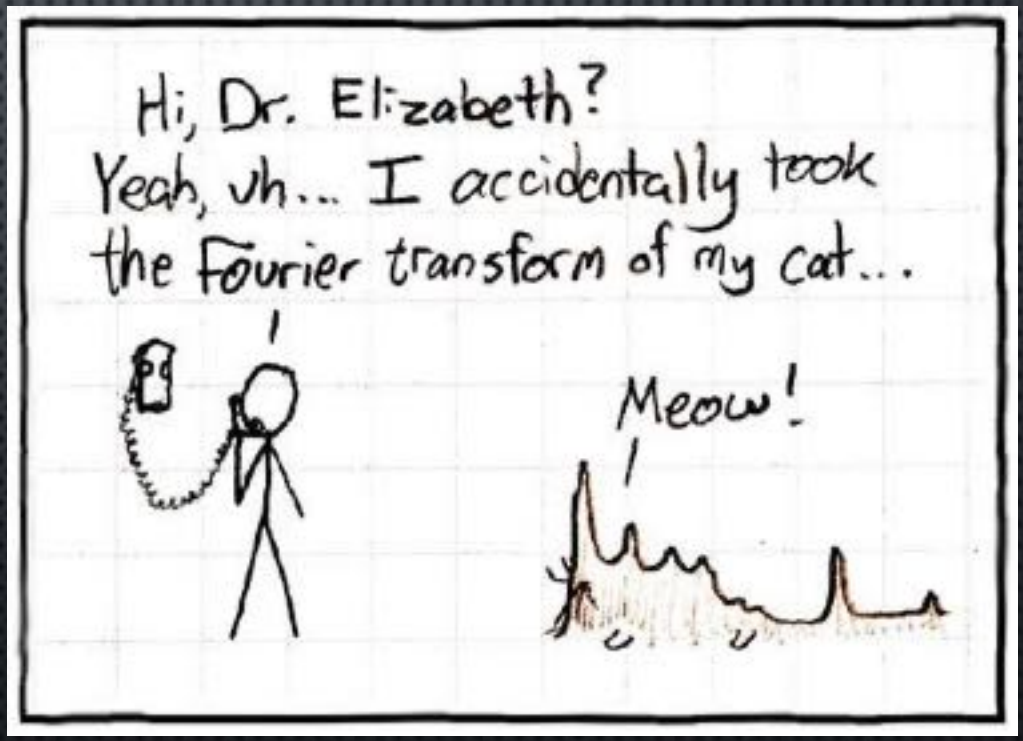
From time domain to frequency domain

- Time domain – waveform
  - Amplitude vs time
  - Intuitive and straightforward
  - Outside the box
- Frequency domain – spectra
  - Magnitude and phase vs frequency
  - The composition of the signal
  - Inside the box



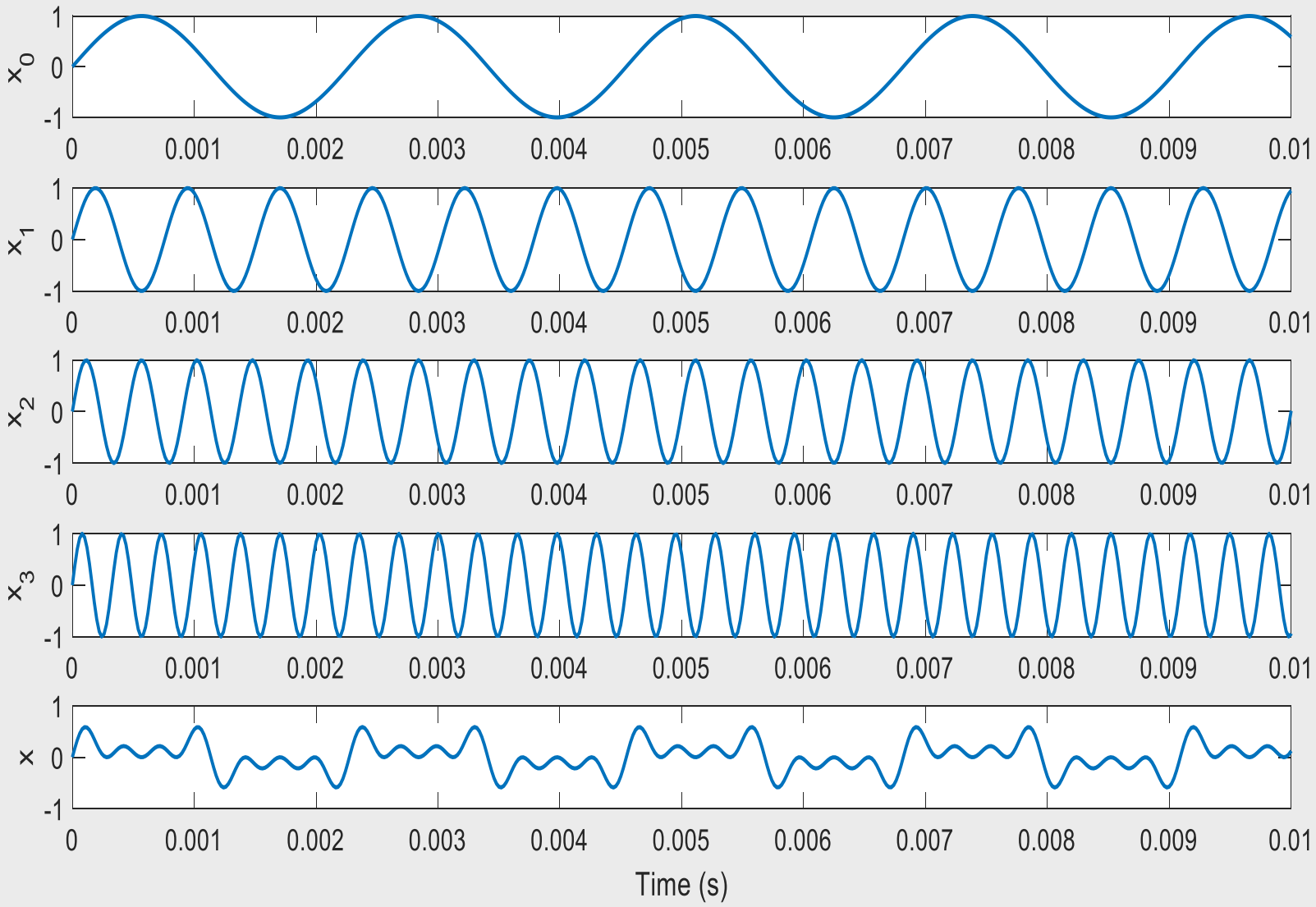
Fourier analysis

- History
- Overview, concepts
- Theory



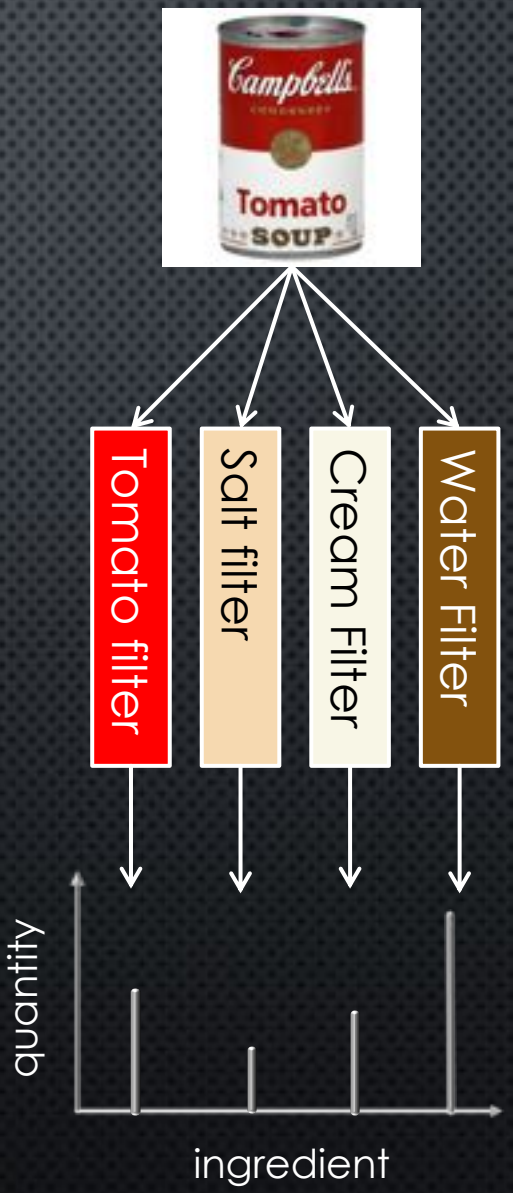
The transformation of the Fourier transform..

- Fourier analysis is one of the most widely used tools for analysing signals
- 1807 - Fourier Series - Joseph Fourier
  - A periodic function can be approximated by a sum of sinusoid functions
- 1822 - Fourier transform –Joseph Fourier
  - Expands the concept to allow non-period functions to be decomposed
- 1829 – Dirichlet establish conditions where a signal can be represented in this way


x0
x1
x2
x3
x
1
0
-1
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
Time (s)

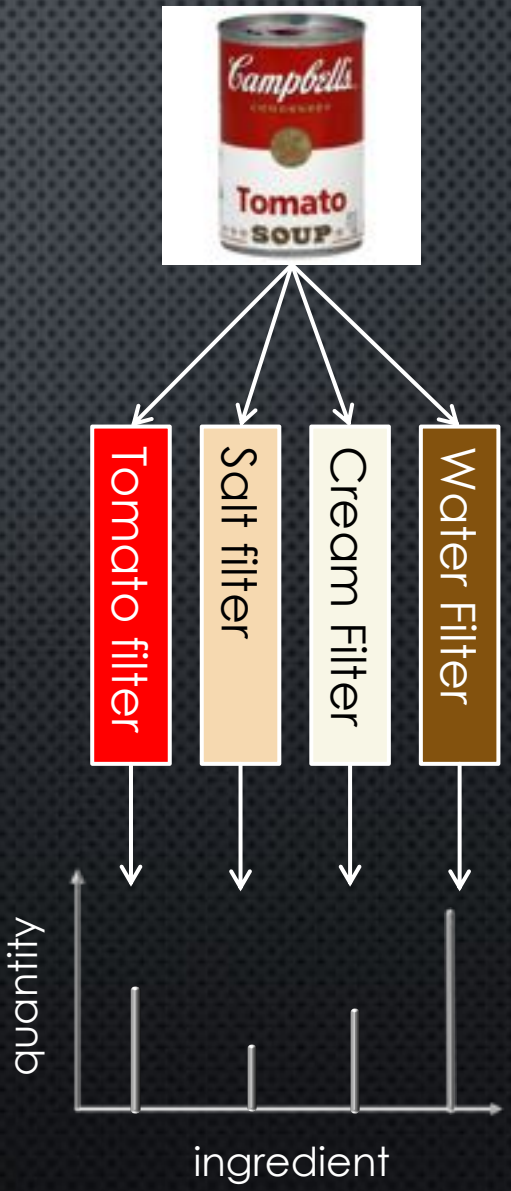
The "Souperior" transform

- **What does the Fourier Transform do?**
  - Given some soup, it finds the recipe
- **How?**
  - Run the soup through filters to extract each ingredient.
- **Why?**
  - Recipes are easier to analyze, compare, and modify than the soup itself
- **How do we make the soup again?**
  - Blend the ingredients in the amounts measured



The "Souperior" transform

- Filters are dedicated and exclusive
- Filtering must be thorough
- Obtained ingredients by filtering must be reusable reversely



Discrete Fourier Transform (DFT)

- In DSP, the transform derived by Fourier (which predated digital computers!) is not used
  - Problem: it is an infinite summation!
- Instead, DFT is used, which is finite and exact

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-i\varphi}, \varphi = \frac{2\pi kn}{N}$$

$x(n)$: time domain signal of length N

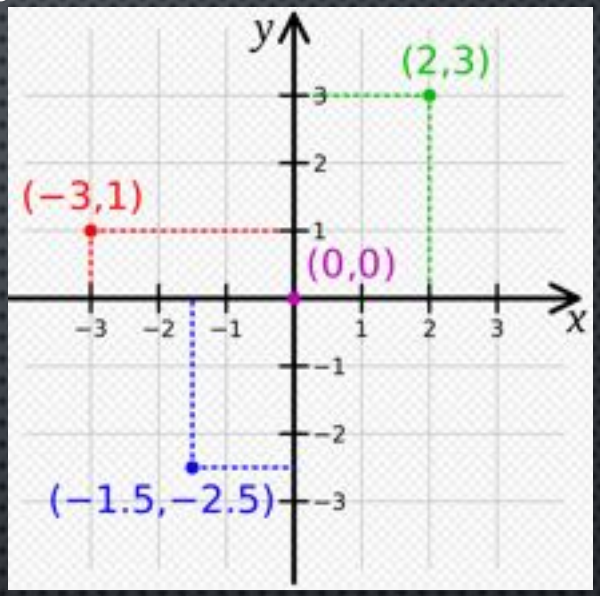
$X(k)$: amplitude and phase info at frequency bin k

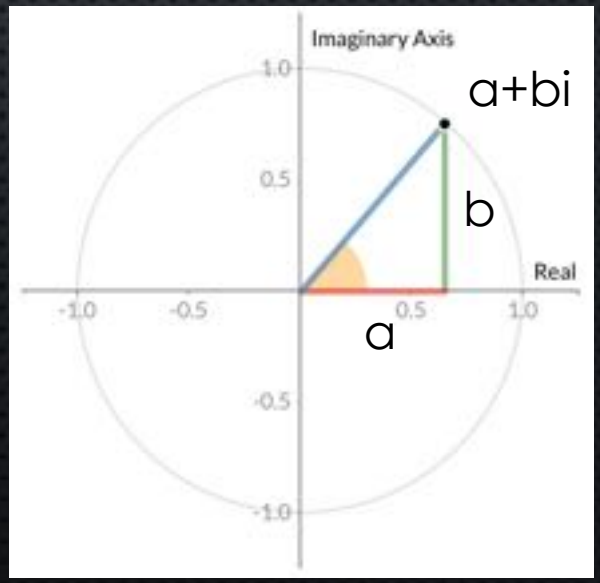
- Note that for $k = 0$, $X(0)$ is simply the sum of the signal samples

Complex numbers

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-i\varphi}, \varphi = k\frac{2\pi n}{N}$$

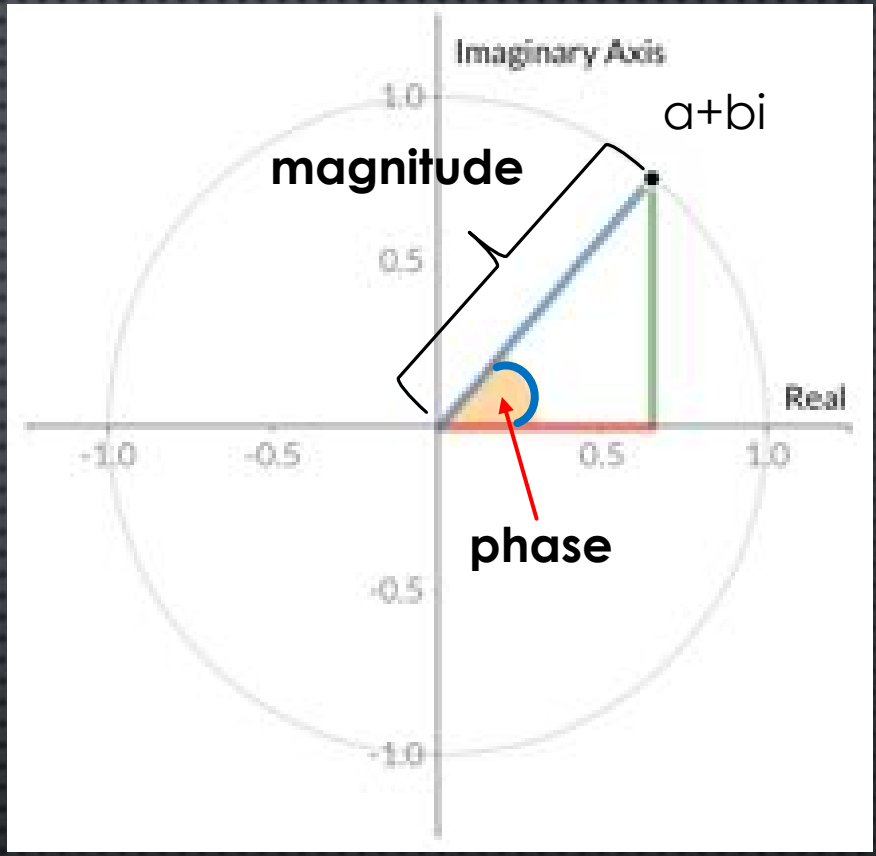
- What is "i" in there above "e"?
  - The imaginary part of a complex number produced by the Fourier transform
- What is complex number?
  - A pair of numbers which defines a point in a 2-dimensional space
  - a: real part; b: imaginary part
  - e.g. $3 + 4i$, $-5 - 8i$
  - $i^2 = -1$

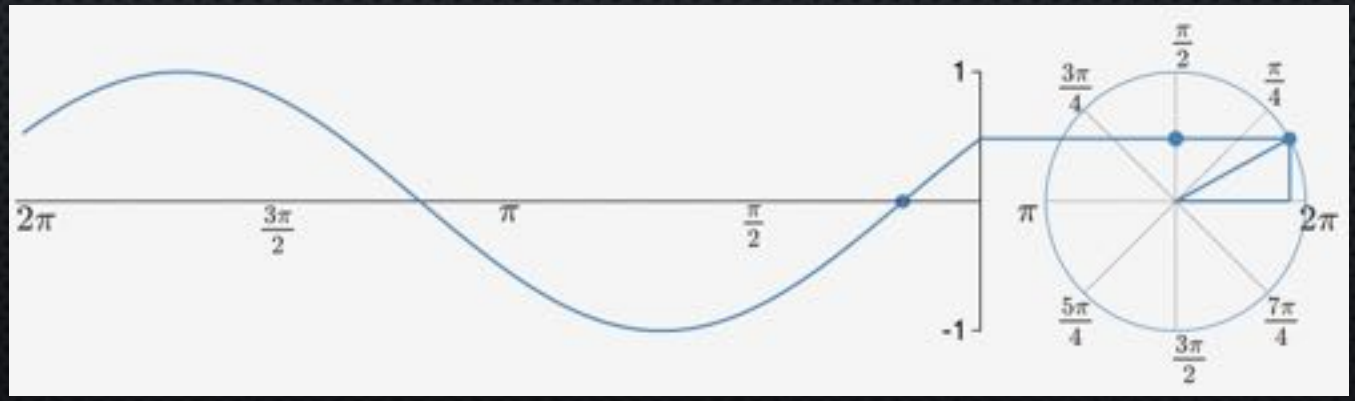




Complex numbers

- More information in complex numbers!
  - Magnitude
  - Phase
- $magnitude = \sqrt{a^2 + b^2}$
- $phase = \tan^{-1}\frac{b}{a}$





Euler's formula

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-i\varphi}, \varphi = \frac{2\pi kn}{N}$$

- The relationship between e and unit-circle on the complex plane

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

$$e^{i\varphi} = \cos(\varphi) + \sin(\varphi)i$$
$$e^{-i\varphi} = \cos(\varphi) - \sin(\varphi)i$$

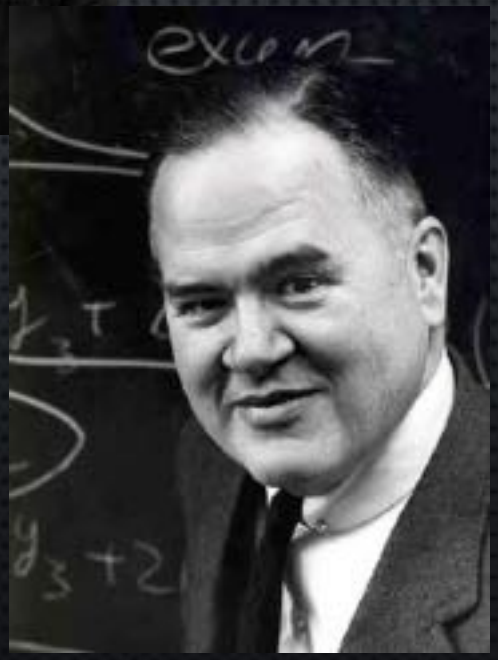
$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot (\cos(\varphi) - \sin(\varphi)i)$$

The Fast Fourier Transform (FFT)

- An algorithm for the efficient calculation of the DFT (not another transform)
- Discovered by Gauss in 1805

The Fast Fourier Transform

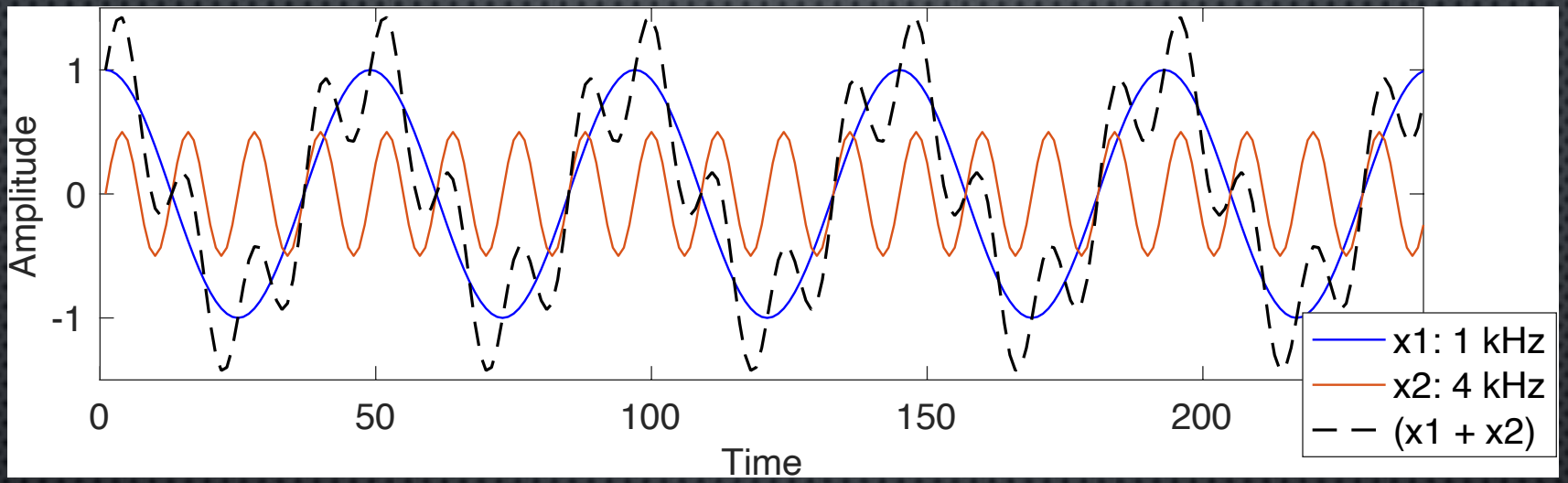
- An algorithm for the efficient calculation of the DFT (not another transform)
- Discovered by Gauss in 1805
- Rediscovered by Cooley and Tukey in the 1965



The Fast Fourier Transform

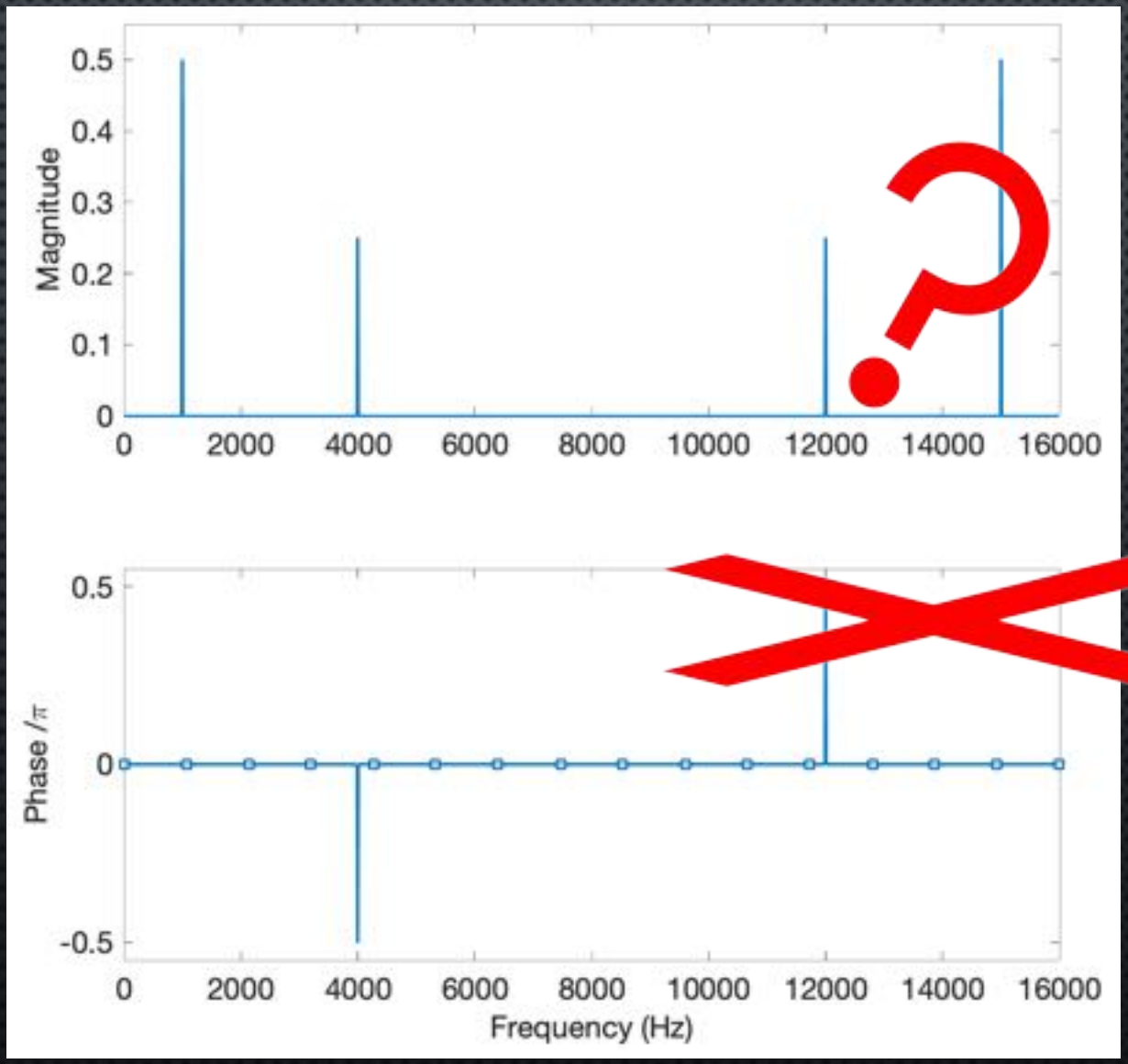
- An algorithm for the efficient calculation of the DFT (not another transform)
- Discovered by Gauss in 1805
- Rediscovered by Cooley and Tukey in the 1965
- Became popular because computers

Example: spectra of simple sinusoid



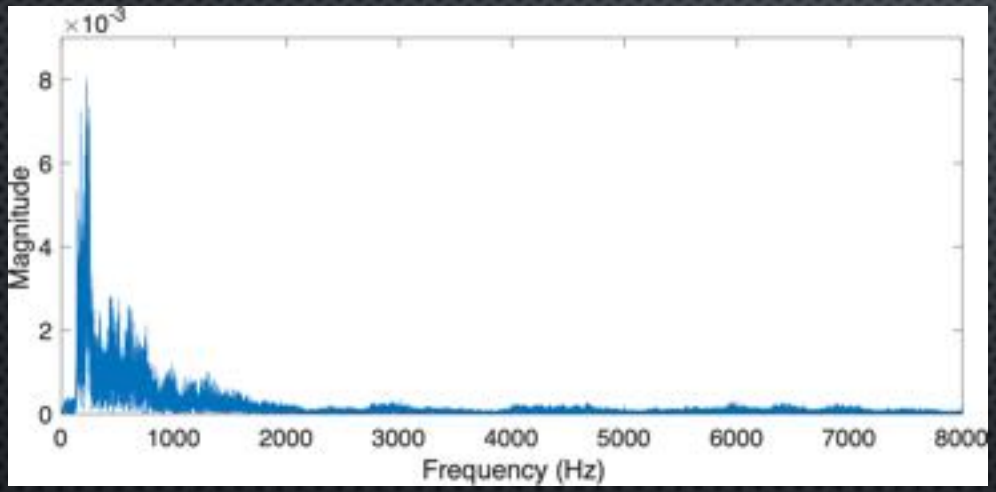
- Signals: fs = 16 kHz
  - $x = \sin(2\pi \cdot 1000t + \frac{\pi}{2}) + \sin(2\pi \cdot 4000t)$
- How many spectra are there for the signal after the Fourier transform?
- Can you envisage what the spectra look like?

Example: fs = 16 kHz

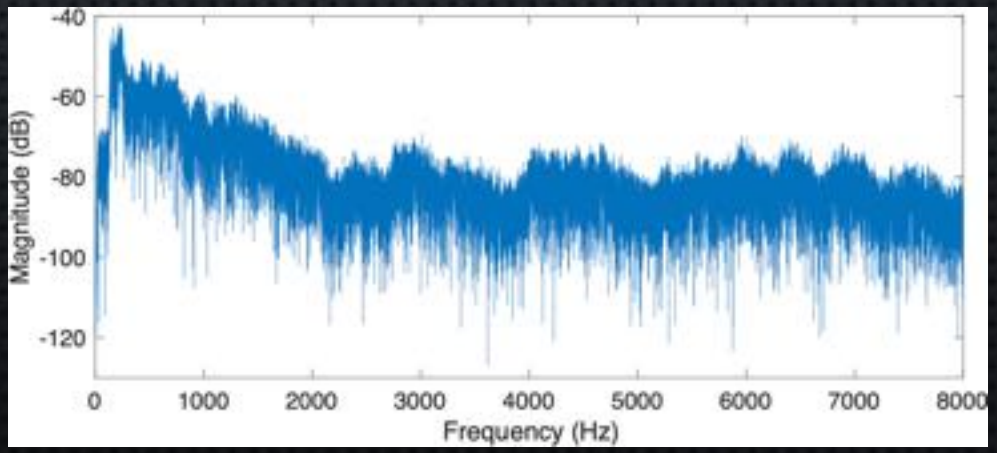


Produce magnitude spectrum

Fourier transform



Log magnitude in dB



Steps:

1. Perform Fourier Transform
2. Compute the magnitude of complex numbers
3. Use only the first half of the data
4. (Sometimes) convert to decibels

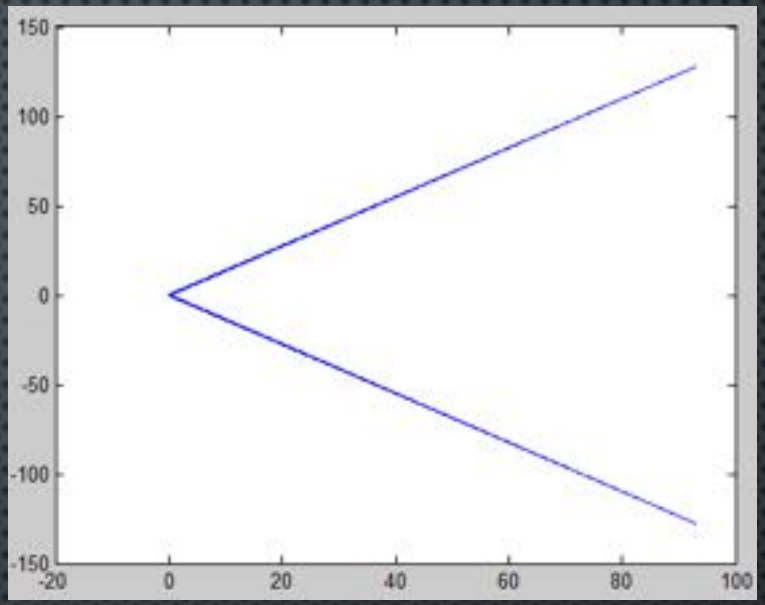
Illustrating the DFT

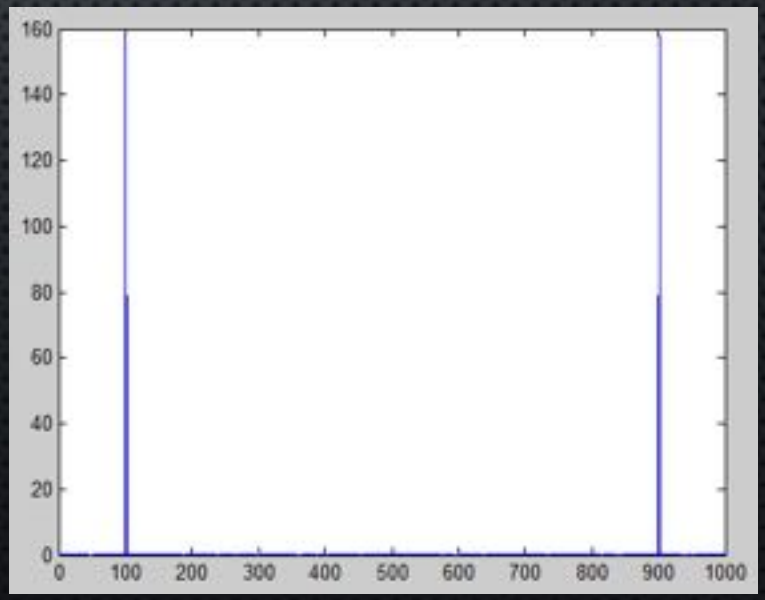
```
% create a 1 kHz tone at fs=10000
% duration = 100 ms

t = tone(1000, 100, 10000);

% compute DFT
dft_t = fft(t);
plot(dft_t)

% ooops – it's complex
% so plot magnitude instead
magdft_t = abs(dft_t);
plot(magdft_t)
```





magnitude

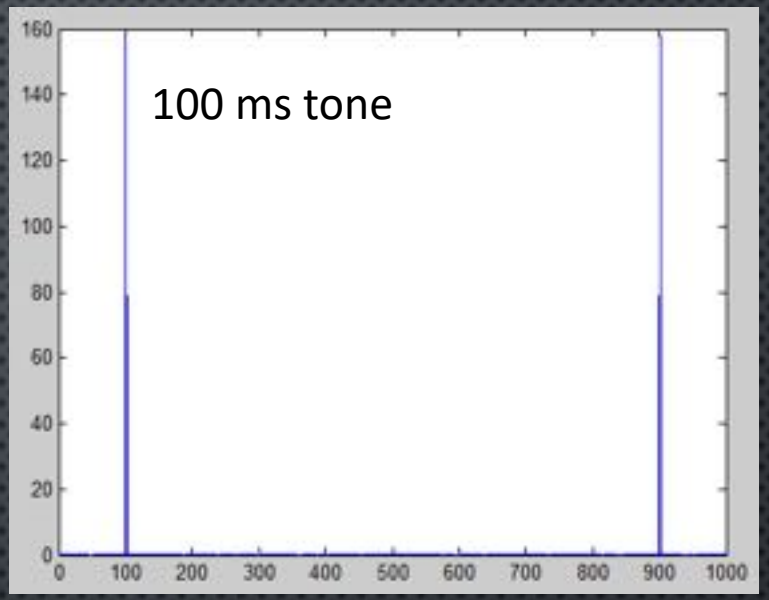
frequency →

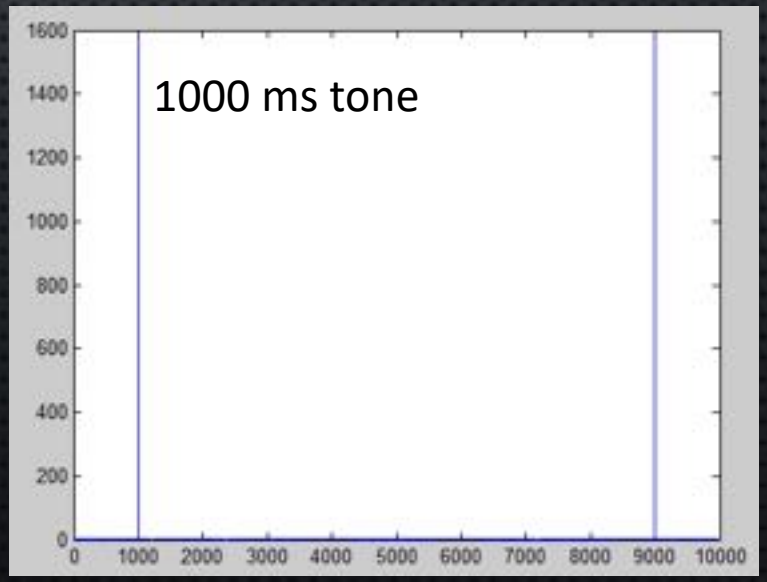
Exploring the frequency axis

```
% let's create one second of signal

t = tone(1000, 1000, 10000);
plot(abs(fft(t)))
```

- Tone 'peaks' are in the same place, yet the axis is labelled 1-1000 in the first case and 1-10000 in the second case
- The FFT function divides the frequency range [0-fs] into N, where N is the length of the input
- Each point on the frequency axis represents (fs/N) Hz

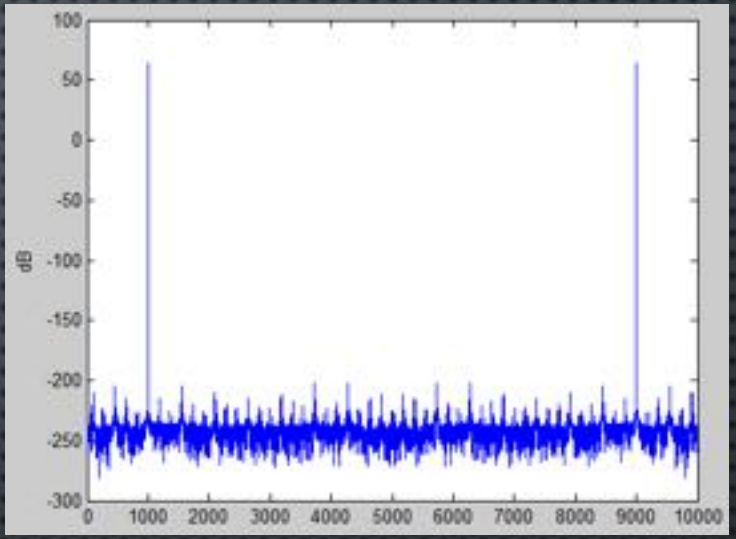


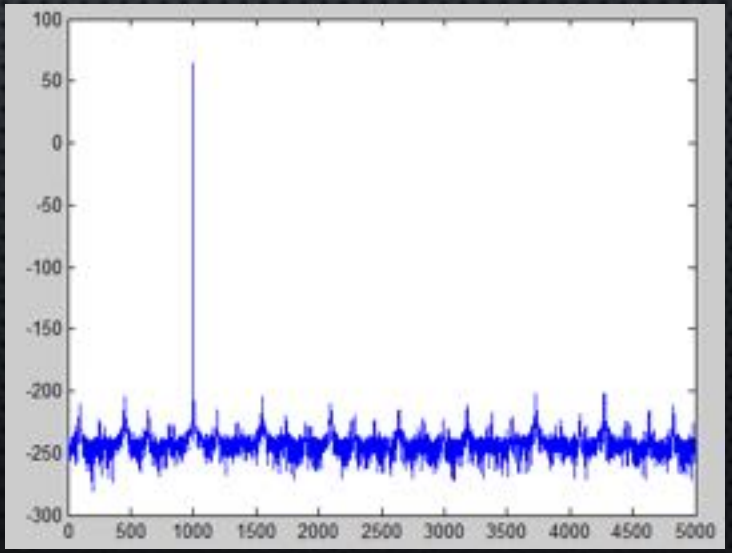


First, some simplifications

```
% convert magnitude to
   decibels
dbs = 20*log10(abs(fft(t)));
plot(dbs)
ylabel('dB')

% remove top-half of spectrum
npts=length(dbs);
dbs=dbs(1:round(npts/2));
plot(dbs)
```

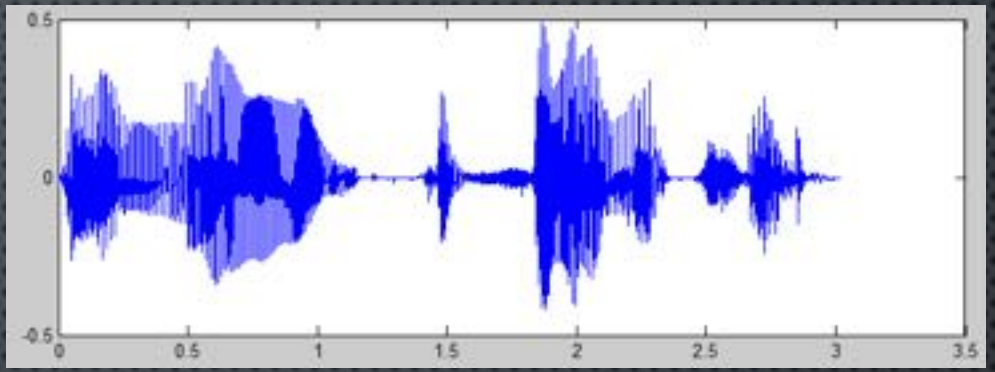


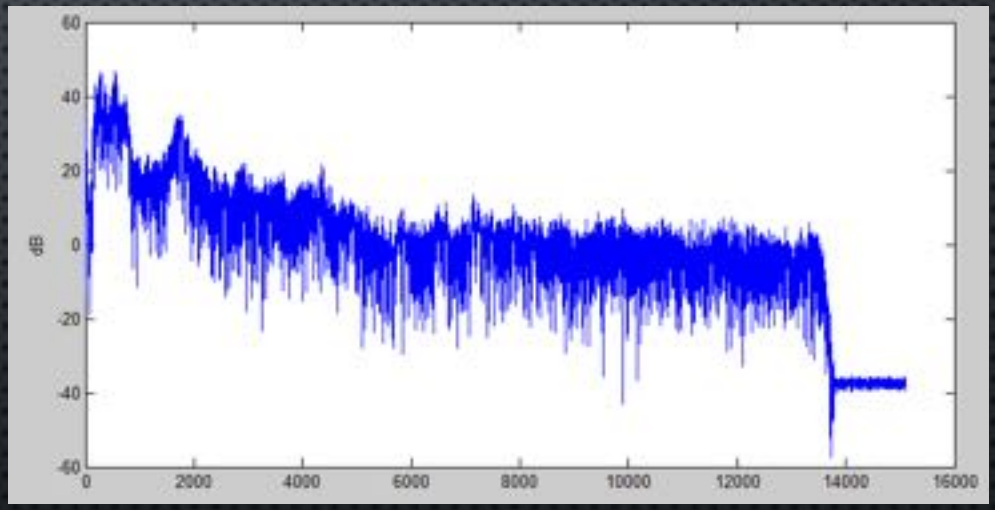


A speech example

```
% compute dB spectrum and plot
   lower half
f=fft(y);
dbs = 20*log10(abs(f));
npts=length(dbs);
dbs=dbs(1:round(npts/2));
plot(dbs)

% check that DC term
   represents
% sum of signal
f(1)    % outputs -63.6685
sum(y)   % outputs -63.6685
```





This is more like a frequency spectrum we are used to. But note, it is the spectrum of the entire signal.

FFT inputs and outputs

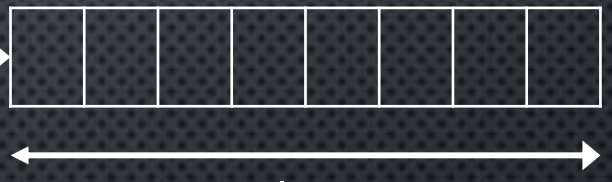
Array of time domain samples

N samples
T seconds

Real valued

FFT

Array of frequency domain samples



N samples
fs Hz

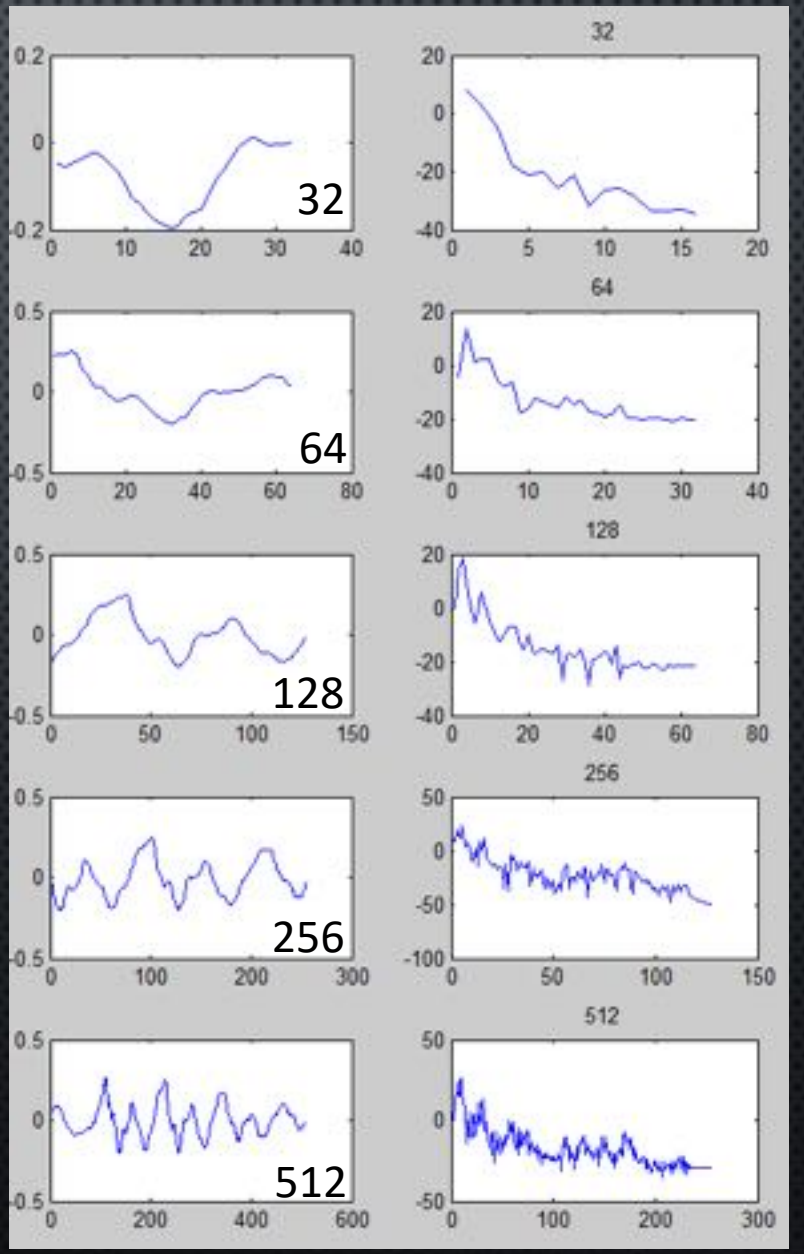
Complex valued (i.e. magnitude and phase)

$n = 0, 1, 2, 3, \ldots, N-1$
$t = 0, \Delta t, 2\Delta t, \ldots, (N-1)\Delta t$
$\Delta t = T/N$

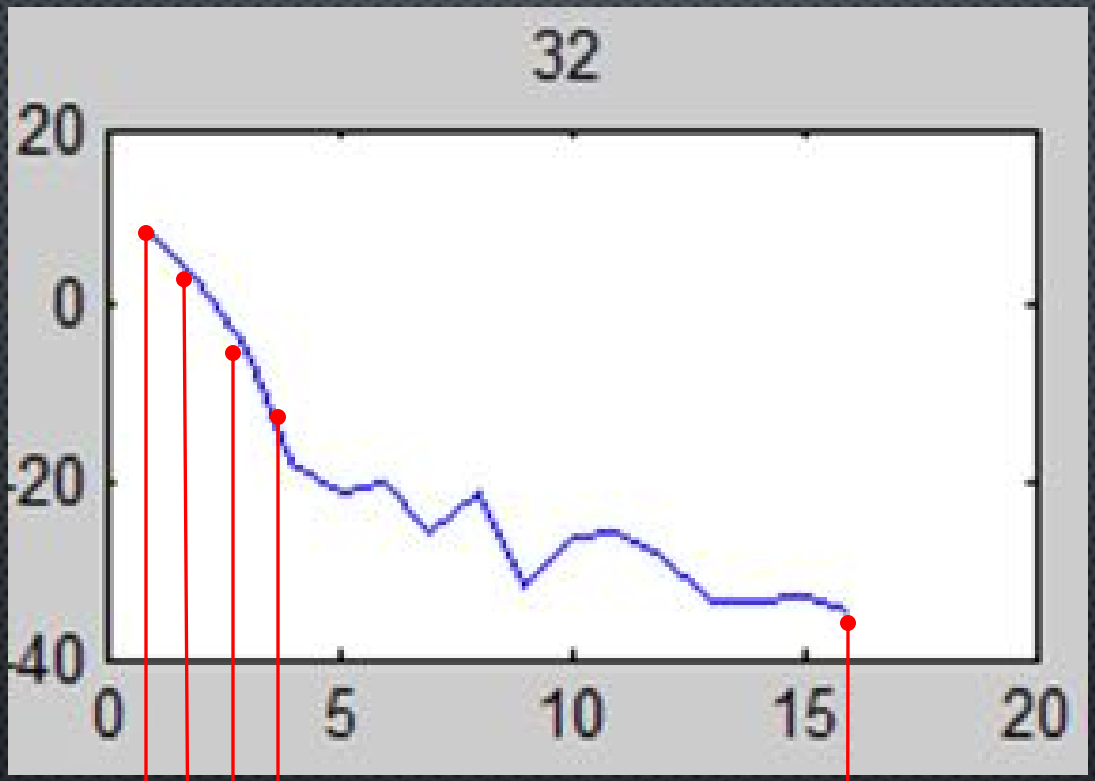
$k = 0, 1, 2, 3, \ldots, N-1$
$f = 0, \Delta f, 2\Delta f, \ldots, (N-1)\,\Delta f$
$\Delta f = fs/N$

Time/frequency resolution

- Regardless of the length of the time interval (here, from 32 to 512 samples), the Fourier transform stretches from 0 to fs Hz (fs = sampling frequency)
- Here we lose the top half, so the display is from 0 to fs/2 Hz
- Hence (reminder): each point on the frequency axis represents a distance of (fs/N) Hz
- So: the first point is 0 Hz, the 2nd is fs/N Hz, the 3rd point is (2*fs)/N, the 4th is (3*fs)/N, etc



N=32



0

fs/N

(2*fs)/N

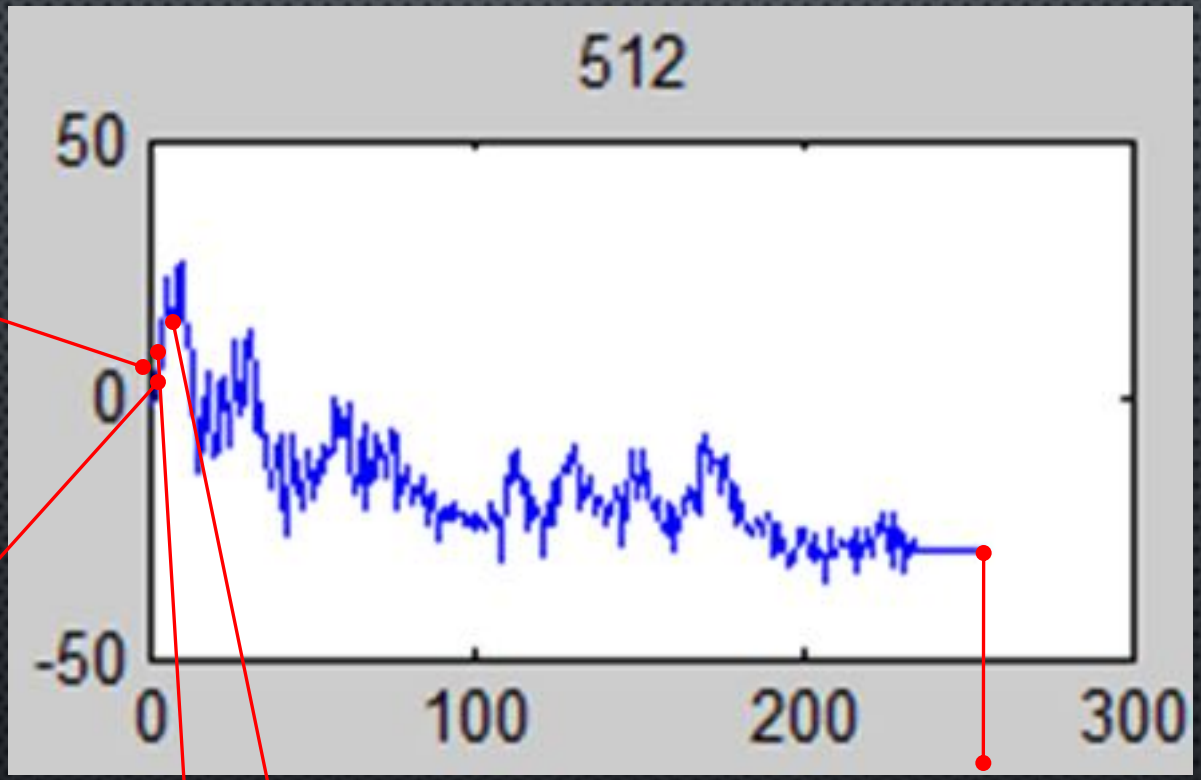
(3*fs)/N

(N/2)*(fs/N) = fs/2

for fs=10000 Hz: 0 313 625 938 ..... 5000 Hz

CRUDE frequency resolution!

Consider N=512



0

fs/N

(2*fs)/N

(3*fs)/N

(N/2)*(fs/N)

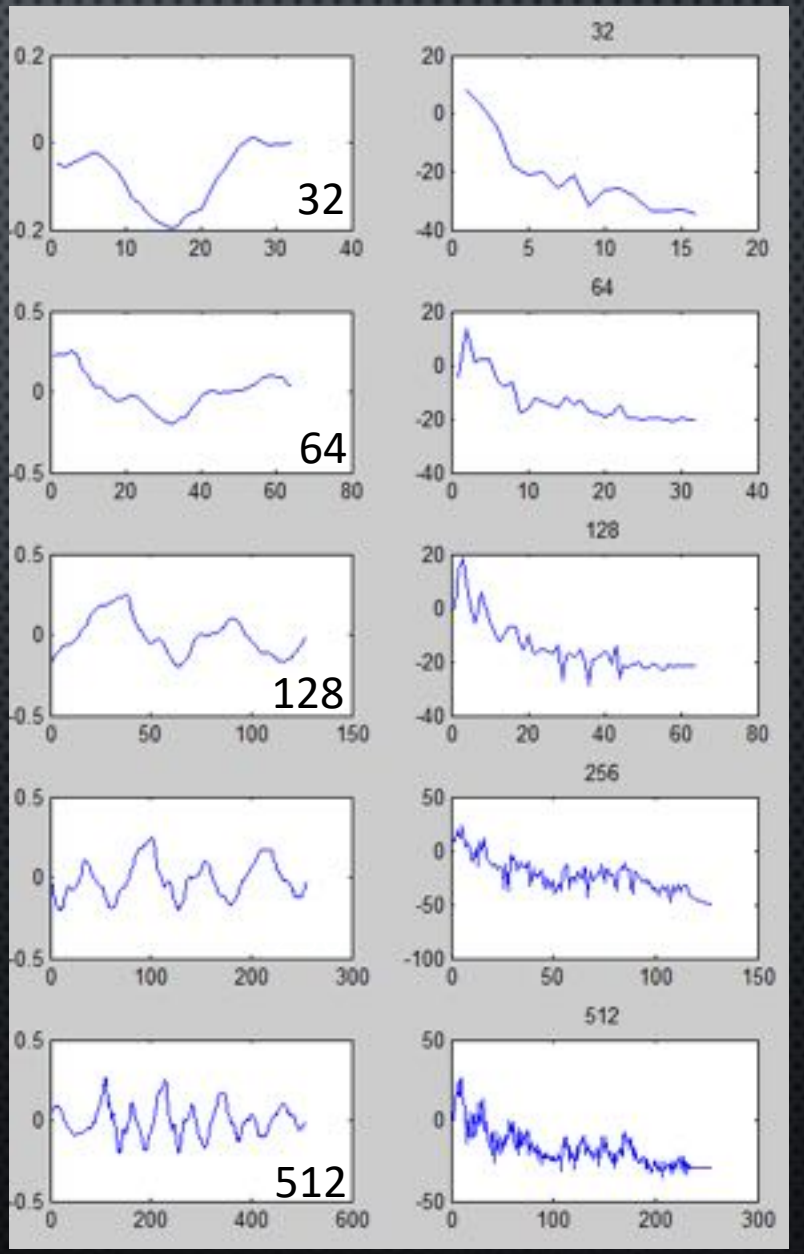
for fs=10000 Hz: 0 20 39 59 .... 5000 Hz

FINE frequency resolution!

Time/frequency resolution

- So: frequency resolution is proportional to 1/N, where N is the number of signal samples
- a short stretch of signal leads to poor frequency resolution
- a long stretch leads to very fine frequency resolution

SO: *why not always use long stretches of samples since this delivers very fine frequency resolution?*

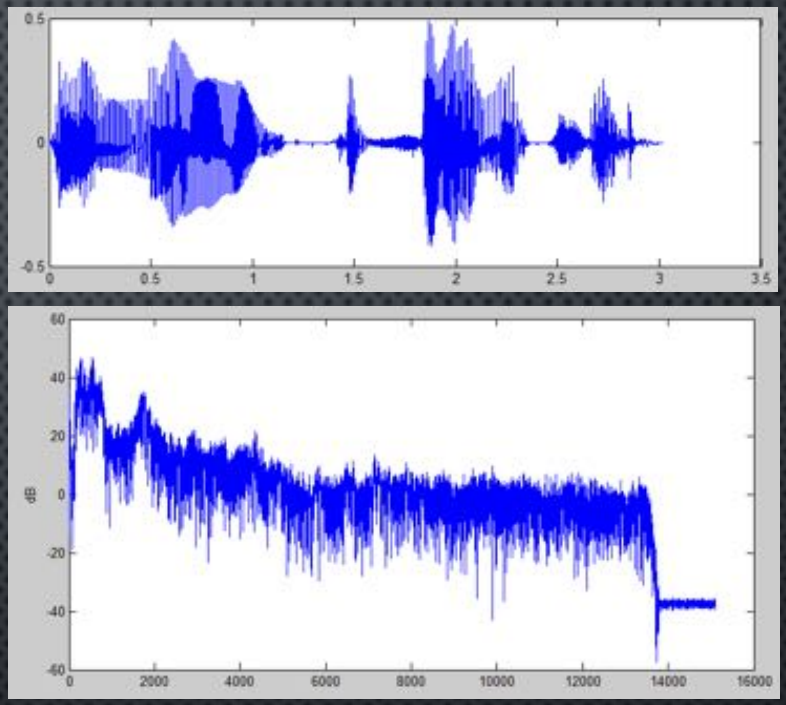


Time-frequency resolution tradeoff

Consider the extreme case: we transform the entire signal. This results in very fine frequency resolution (here, 0.73 Hz!!)

BUT: we only get ONE measurement of signal energy at each frequency. This is fine if the distribution of energy across frequency doesn't change ("Stationary signal").

Speech is certainly not like that!



THE TIME-FREQUENCY TRADEOFF

Frequency resolution is proportional to the **inverse** of time resolution (and vice versa)