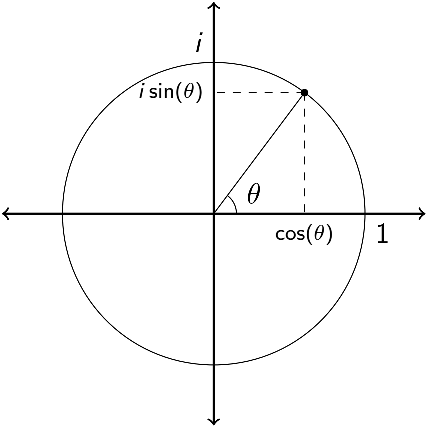
Creative Computing II

Christophe Rhodes
`c.rhodes@gold.ac.uk`

Autumn 2010, Wednesdays:
10:00–12:00: RHB307 & 14:00–16:00: WB316
Winter 2011, Wednesdays:
10:00–12:00: RHB307 & 14:00–16:00: WB316

Signals

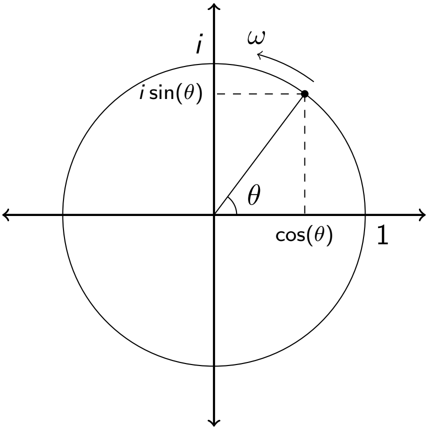
The Complex Exponential



$$e^{i\theta} = \cos(\theta) + i\sin(\theta);\ e^{-i\theta} = \cos(\theta) - i\sin(\theta).$$

Signals

The Complex Exponential



$$e^{i\omega t} = \cos(\omega t) + i\sin(\omega t);\ e^{-i\omega t} = \cos(\omega t) - i\sin(\omega t).$$

Signals

Sinusoidal functions

Functional relations:

- $\cos(\omega t) = \frac{e^{i\omega t} + e^{-i\omega t}}{2}$
- $\sin(\omega t) = \frac{e^{i\omega t} - e^{-i\omega t}}{2i}$

Identities:

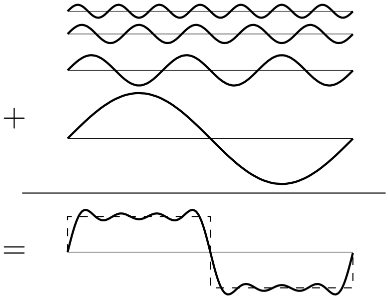
- $e^{i\pi} = -1$ (Euler's Identity)
- $e^{i\frac{\pi}{2}} = i$
- $e^{2\pi i} = 1$
- $(\cos(\theta) + i\sin(\theta))^n = \cos(n\theta) + i\sin(n\theta)$ (de Moivre's formula)

Signals

Fourier Series

Square wave:

$$s_f(t) = \sum_{k=1}^{\infty} \frac{1}{2k-1} \sin(2\pi(2k-1)ft)$$



Fourier Series:

- Any signal can be written as a weighted sum of sin and cos terms: a **Fourier Series**.
- For a signal of length L, all sinusoids have angular frequencies that are integer multiples of $\frac{2\pi}{L}$.
- For a real discrete-time signal at sample rate R, the maximum frequency component is the Nyquist frequency.

Fourier Series:

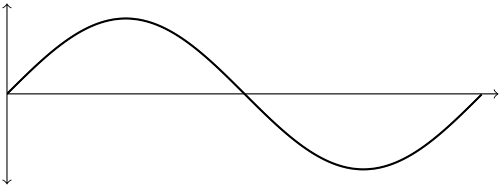
- Any signal can be written as a weighted sum of sin and cos terms: a **Fourier Series**.
- For a signal of length L, all sinusoids have angular frequencies that are integer multiples of $\frac{2\pi}{L}$.
- For a real discrete-time signal at sample rate R, the maximum frequency component is the Nyquist frequency.

Fourier Analysis of Signals:

- Extraction of frequency components for a given signal;
- Dot-product multiply by complex exponential signal;
- Magnitude and phase of result give magnitude and phase of corresponding sinusoid.

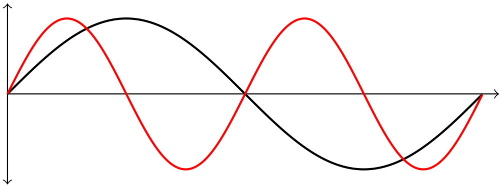
How does this work?

- dot-product of sinusoid with *exactly itself* gives a non-zero result;
- all other dot-products between sinusoids give zero.
- sinusoids are **orthogonal** basis functions.



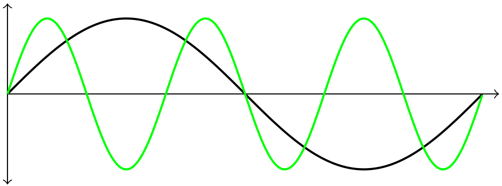
How does this work?

- dot-product of sinusoid with *exactly itself* gives a non-zero result;
- all other dot-products between sinusoids give zero.
- sinusoids are **orthogonal** basis functions.



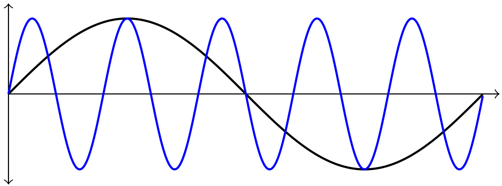
How does this work?

- dot-product of sinusoid with *exactly itself* gives a non-zero result;
- all other dot-products between sinusoids give zero.
- sinusoids are **orthogonal** basis functions.



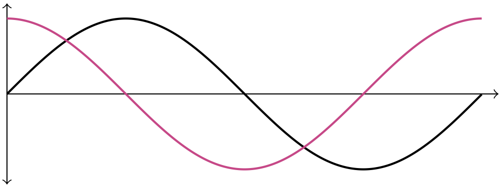
How does this work?

- dot-product of sinusoid with *exactly itself* gives a non-zero result;
- all other dot-products between sinusoids give zero.
- sinusoids are **orthogonal** basis functions.



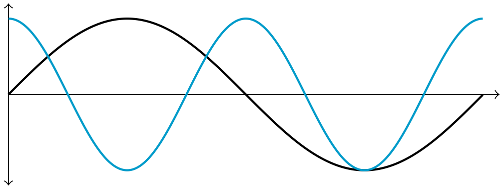
How does this work?

- dot-product of sinusoid with *exactly itself* gives a non-zero result;
- all other dot-products between sinusoids give zero.
- sinusoids are **orthogonal** basis functions.



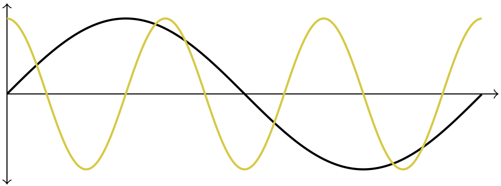
How does this work?

- dot-product of sinusoid with *exactly itself* gives a non-zero result;
- all other dot-products between sinusoids give zero.
- sinusoids are **orthogonal** basis functions.



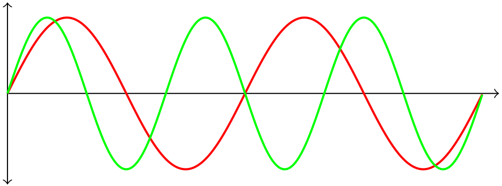
How does this work?

- dot-product of sinusoid with *exactly itself* gives a non-zero result;
- all other dot-products between sinusoids give zero.
- sinusoids are **orthogonal** basis functions.



How does this work?

- dot-product of sinusoid with *exactly itself* gives a non-zero result;
- all other dot-products between sinusoids give zero.
- sinusoids are **orthogonal** basis functions.



We can convert a signal into its *Fourier* representation by extracting *all* its frequency components: the **Fourier Transform**.

Signals

Fourier Transforms

We can convert a signal into its *Fourier* representation by extracting *all* its frequency components: the **Fourier Transform**.

$$\left(\mathcal{F}(x(k))\right)(\omega) = \sum_{k=0}^{L-1} x(k) e^{-i\omega k}$$

We can convert a signal into its *Fourier* representation by extracting *all* its frequency components: the **Fourier Transform**.

$$(\mathcal{F}(x(k)))(\omega) = \sum_{k=0}^{L-1} x(k) e^{-i\omega k}$$

- ω takes on values $\{0, \frac{2\pi}{L}, \frac{4\pi}{L}, ..., \pi, ..., \frac{2\pi(L-1)}{L}\}$
- L (real) signal values $\rightarrow \frac{L}{2}$ (complex) frequency components
- (L complex values $\rightarrow L$ complex frequency components)

We can convert a signal into its *Fourier* representation by extracting *all* its frequency components: the **Fourier Transform**.

$$\left(\mathcal{F}(x(k))\right)(\omega) = \sum_{k=0}^{L-1} x(k) e^{-i\omega k}$$

- ω takes on values $\{0, \frac{2\pi}{L}, \frac{4\pi}{L}, \ldots, \pi, \ldots, \frac{2\pi(L-1)}{L}\}$
- L (real) signal values $\rightarrow \frac{L}{2}$ (complex) frequency components
- (L complex values $\rightarrow L$ complex frequency components)

This is the **frequency spectrum** (sometimes just **spectrum**) of the signal.

We can convert a signal into its *Fourier* representation by extracting *all* its frequency components: the **Fourier Transform**.

$$(\mathcal{F}(x(k)))(\omega) = \sum_{k=0}^{L-1} x(k) e^{-i\omega k}$$

- ω takes on values $\{0, \frac{2\pi}{L}, \frac{4\pi}{L}, ..., \pi, ..., \frac{2\pi(L-1)}{L}\}$
- L (real) signal values $\to \frac{L}{2}$ (complex) frequency components
- (L complex values $\to L$ complex frequency components)

This is the **frequency spectrum** (sometimes just **spectrum**) of the signal.

Note: $\mathcal{F}(x)$ sometimes notated as $\tilde{x}$.

Fourier Transform: direct calculation is $O(L^2)$.

Fourier Transform: direct calculation is $O(L^2)$.

$$(\mathcal{F}(x(k)))(\omega) = \sum_{k=0}^{L-1} x(k) e^{-i\omega k}$$

Fourier Transform: direct calculation is $O(L^2)$.

$$(\mathcal{F}(x(k)))(\omega) = \sum_{k=0}^{L-1} x(k) e^{-i\omega k}$$

- $X \leftarrow \mathcal{F}(x)$

  $L \leftarrow \text{length}(x)$
  $X \leftarrow \text{newArray}(L,0)$
  for j from 0 below L **do**
      for k from 0 below L **do**
          $X_j \leftarrow X_j + x(k) \times e^{-\frac{2\pi ijk}{L}}$
      end for
  end for

Signals

The Fast Fourier Transform

$$\left(\mathcal{F}(x(k))\right)(\omega) = \sum_{k=0}^{L/2-1} x(2k)e^{-i\omega(2k)} + e^{-i\omega} \sum_{k=0}^{L/2-1} x(2k+1)e^{-i\omega(2k)}$$

Signals

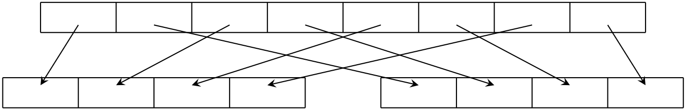
The Fast Fourier Transform

$$\left(\mathcal{F}(x(k))\right)(\omega) = \sum_{k=0}^{L/2-1} x(2k)e^{-i\omega(2k)} + e^{-i\omega} \sum_{k=0}^{L/2-1} x(2k+1)e^{-i\omega(2k)}$$

Signals

The Fast Fourier Transform

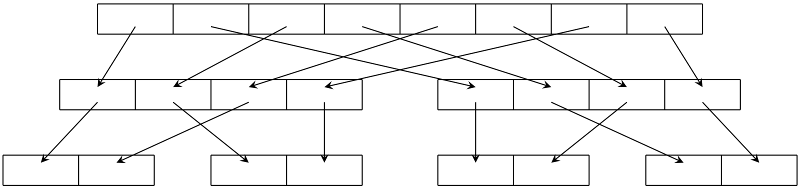
$$\left(\mathcal{F}(x(k))\right)(\omega) = \sum_{k=0}^{L/2-1} x(2k)e^{-i\omega(2k)} + e^{-i\omega} \sum_{k=0}^{L/2-1} x(2k+1)e^{-i\omega(2k)}$$



Signals

The Fast Fourier Transform

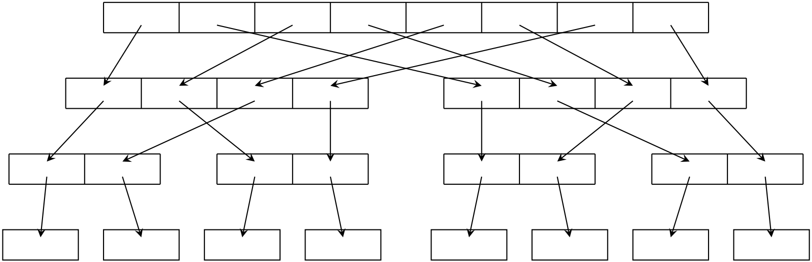
$$(\mathcal{F}(x(k)))(\omega) = \sum_{k=0}^{L/2-1} x(2k)e^{-i\omega(2k)} + e^{-i\omega} \sum_{k=0}^{L/2-1} x(2k+1)e^{-i\omega(2k)}$$



Signals

The Fast Fourier Transform

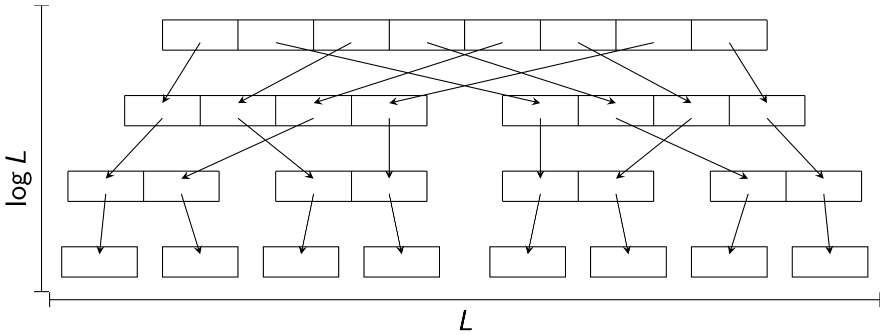
$$(\mathcal{F}(x(k)))\,(\omega) = \sum_{k=0}^{L/2-1} x(2k)e^{-i\omega(2k)} + e^{-i\omega} \sum_{k=0}^{L/2-1} x(2k+1)e^{-i\omega(2k)}$$



Signals

The Fast Fourier Transform

$$\left(\mathcal{F}(x(k))\right)(\omega) = \sum_{k=0}^{L/2-1} x(2k)e^{-i\omega(2k)} + e^{-i\omega} \sum_{k=0}^{L/2-1} x(2k+1)e^{-i\omega(2k)}$$



Signals

The Fast Fourier Transform

$$(\mathcal{F}(x(k)))(\omega) = \sum_{k=0}^{L/2-1} x(2k)e^{-i\omega(2k)} + e^{-i\omega} \sum_{k=0}^{L/2-1} x(2k+1)e^{-i\omega(2k)}$$

- $X \leftarrow \mathrm{FFT}(x)$

  $L \leftarrow \mathrm{length}(x)$
  if $L = 1$ **then**
  $\quad X \leftarrow x$
  else
  $\quad X \leftarrow \mathrm{newArray}(L,0)$
  $\quad E \leftarrow \mathrm{FFT}(X_{2k})$; $O \leftarrow \mathrm{FFT}(X_{2k+1})$
  $\quad$ **for** j from 0 below L **do**
  $\quad\quad X_j \leftarrow E_{j\% \frac{L}{2}} + e^{-\frac{2\pi ij}{L}} O_{j\% \frac{L}{2}}$
  $\quad$ **end for**
  end if

Notes:

- FFT has time complexity $O(N \log N)$
- Real algorithms are significantly more complicated
  - non-powers-of-two;
  - base case;
  - choice of radix;
  - exploit performance characteristics of processor and memory.

Notes:

- FFT has time complexity $O(N \log N)$
- Real algorithms are significantly more complicated
  - non-powers-of-two;
  - base case;
  - choice of radix;
  - exploit performance characteristics of processor and memory.

Octave: `fft` function.

- first element is zero-frequency (d.c.) or *constant* component;

Signals

The Fast Fourier Transform

Notes:

- FFT has time complexity $O(N \log N)$
- Real algorithms are significantly more complicated
  - non-powers-of-two;
  - base case;
  - choice of radix;
  - exploit performance characteristics of processor and memory.

Octave: `fft` function.

- first element is zero-frequency (d.c.) or *constant* component;
- next element is for angular frequency $\frac{2\pi}{L}$ (ordinary frequency $\frac{1}{L}$) slowest-moving or *fundamental* component;

Notes:

- FFT has time complexity $O(N \log N)$
- Real algorithms are significantly more complicated
  - non-powers-of-two;
  - base case;
  - choice of radix;
  - exploit performance characteristics of processor and memory.

Octave: `fft` function.

- first element is zero-frequency (d.c.) or *constant* component;
- next element is for angular frequency $\frac{2\pi}{L}$ (ordinary frequency $\frac{1}{L}$) slowest-moving or *fundamental* component;
- successive elements are for successive integer multiples of the fundamental, all the way up to the Nyquist frequency;

Notes:

- FFT has time complexity $O(N \log N)$
- Real algorithms are significantly more complicated
  - non-powers-of-two;
  - base case;
  - choice of radix;
  - exploit performance characteristics of processor and memory.

Octave: `fft` function.

- first element is zero-frequency (d.c.) or *constant* component;
- next element is for angular frequency $\frac{2\pi}{L}$ (ordinary frequency $\frac{1}{L}$) slowest-moving or *fundamental* component;
- successive elements are for successive integer multiples of the fundamental, all the way up to the Nyquist frequency;
- components above the Nyquist then continue all the way to angular frequency $\frac{2(L-1)\pi}{L}$ (regular frequency $\frac{L-1}{L}$).

Systems

Fourier Transforms and Convolution

Previously:

- system H response to signal x is $h * x$;
- direct convolution computation has complexity $O(N^2)$.

Systems

Fourier Transforms and Convolution

Previously:

- system H response to signal x is $h * x$;
- direct convolution computation has complexity $O(N^2)$.

Fourier Transform of a convolution is the product of the Fourier Transforms:

$$\mathcal{F}(h * x) = \mathcal{F}(h) \times \mathcal{F}(x)$$

so system output for signal x is

$$\mathcal{F}^{-1}(\mathcal{F}(h) \times \mathcal{F}(x))$$

Previously:

- system H response to signal x is $h * x$;
- direct convolution computation has complexity $O(N^2)$.

Fourier Transform of a convolution is the product of the Fourier Transforms:

$$\mathcal{F}(h * x) = \mathcal{F}(h) \times \mathcal{F}(x)$$

so system output for signal x is

$$\mathcal{F}^{-1}(\mathcal{F}(h) \times \mathcal{F}(x))$$

Octave:

- `ifft(fft(h,length([h x])-1).*fft(x,length([h x])-1))`

$$y = \mathcal{F}^{-1}(\mathcal{F}(h) \times \mathcal{F}(x))$$

so

$$\mathcal{F}(y) = \mathcal{F}(h) \times \mathcal{F}(x)$$

- $\mathcal{F}(h)$ is the **frequency response** of the system.
- the frequency spectrum of the output signal is the product of the spectrum of the input and the frequency response of the system.