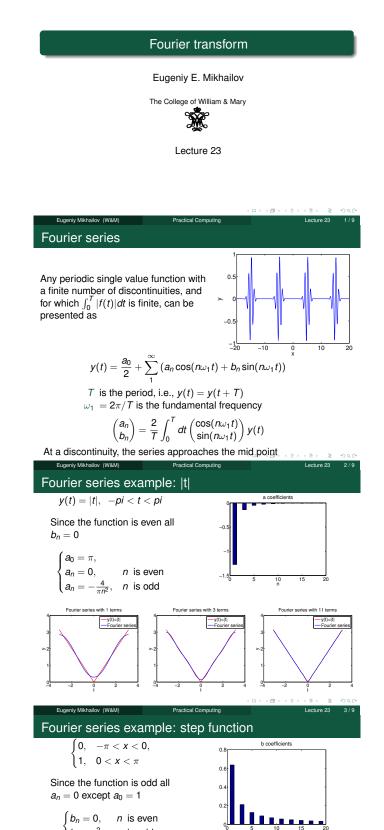
## Notes



# Notes

#### Notes

#### Notes

× 0.

 $b_n = \frac{2}{\pi n}$ , *n* is odd

-y(t)=|t| Fourie

> 0

9

× 0.

y(t)=|t| Fourie

Practical Computin

Lecture 23 4 / 9

ith 100 terms

## Complex representation

## Recall that

$$\exp(i\omega t) = \cos(\omega t) + i\sin(\omega t)$$

It can be shown that

$$y(t) = \sum_{n=-\infty}^{\infty} c_n \exp(in\omega_1 t)$$
  
$$c_n = \frac{1}{T} \int_0^T y(t) \exp(-i\omega_1 nt) dt$$

$$a_n = c_n + c_{-n}$$
  
$$b_n = i(c_n - c_{-n})$$

Practical Computing

1

Lecture 23

Lecture 23

- 31 6/9

# What to do if function is not periodic?

•  $T \to \infty$ 

Eugeniy Mikhailov (W&M)

- $\Sigma \to \int$
- discrete spectrum  $\rightarrow$  continuous spectrum •  $c_n \rightarrow c_\omega$

$$y(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} c_{\omega} \exp(i\omega t) d\omega$$
$$c_{\omega} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} y(t) \exp(-i\omega t) dt$$

Above requires that  $\int_{-\infty}^{\infty} dt y(t)$  exists and is finite.

Note that  $c_{\omega}$  has the extra  $\sqrt{2\pi}$  when compared to  $c_n$ , and T is gone.

Practical Computing

## Discrete Fourier transform (DFT)

#### In reality, we cannot have

Eugeniy Mikhailov (W&M)

- infinitely large interval
- infinite amount of points to calculate true integral

Assuming that y(t) has a period T and we took N equidistant points

 $y_1, y_2, y_3, \cdots, y_N$ 

y

data set We replace the integral in the Fourier series with the sum

$$y_k = \frac{1}{N} \sum_{n=0}^{N-1} c_n \exp(i\frac{2\pi(k-1)n}{N})$$
 inverse Fourier transform  

$$c_n = \sum_{k=1}^{N} y_k \exp(-i\frac{2\pi(k-1)n}{N})$$
 Fourier transform  

$$n = 0, 1, 2, \cdots, N-1$$

Confusion keeps increasing: where are the negative coefficients  $c_{-n}$ ? In DFT, they moved to the right end of the  $c_n$  vector :

 $c_{-n} = c_{N-n}$ 

Notes

## Notes

## Notes

Notes

## Fast Fourier transform (FFT)

Notes

Fast numerical realization of DFT is FFT. This is just the smart way to do DFT. Matlab has one built in

- y is a matlab vector of data points  $(y_k)$
- c=fft(y) Fourier transform
- y=ifft(c) inverse Fourier transform

Notice that fft does not normalize by *N*. So to get Fourier series  $c_n$ , you need to calculate fft (y) /N.

However y = ifft(fft(y))

Eugeniy Mikhailov (W&M)

Notice one more point of confusion: Matlab does not have index=0, so actual  $c_n = c_{matlab \ fft}(n-1)$ , so  $c_0 = c_{matlab \ fft}(1)$ 

Practical Computing

Notes

## Notes

#### Notes