Scientific Computation

Spring, 2019

Lecture 15

Imperial College
London

4 March, 2019

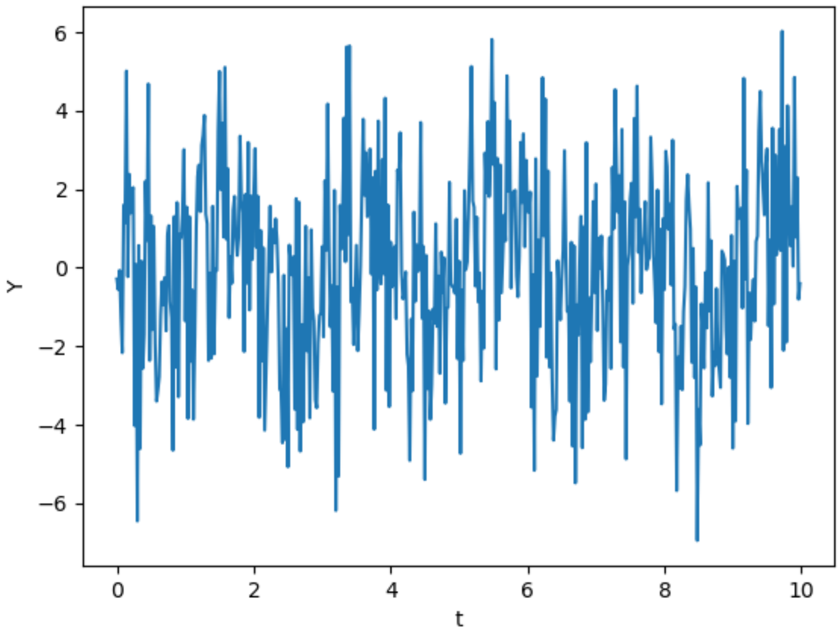
Mastery material

- The mastery material has a reading component (and coding+discussion components)
- I would like to release the reading component this Thursday along with HW3
- Please let me know as soon as possible if you object to this

Data analysis

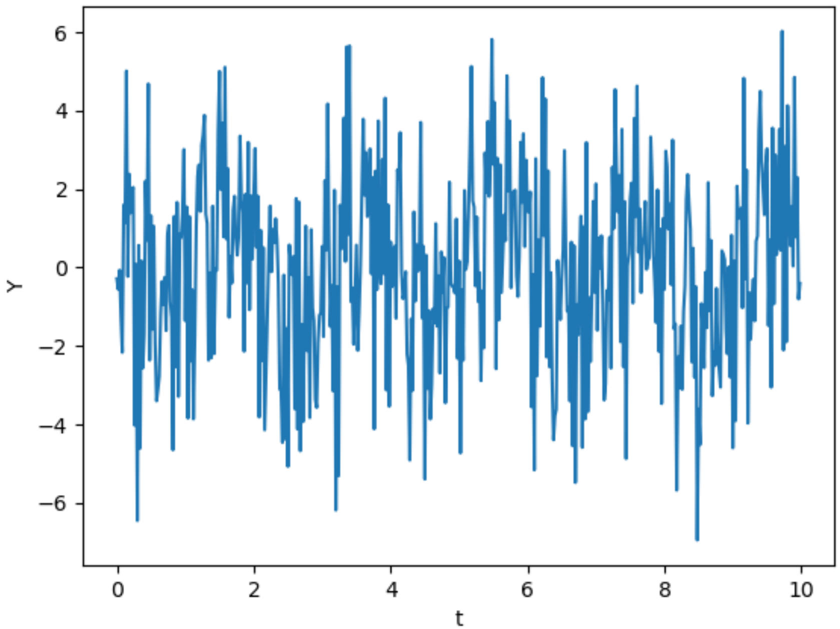
- **Last few lectures: analyze datasets arranged as tables or matrices**
- **Today (and tomorrow): Something simpler! We'll look at data arranged in 1D arrays**
- **Typical example, a time series:** $f(t_i)$, $t_i = i\ \Delta t$, $i = 0, 1, \ldots, N_t-1$
- **Could also have data in space (along a line):** $f(x_i)$, $x_i = i\ \Delta x$, $i = 0, 1, \ldots, N_x-1$
- **How do we extract trends or, more generally, "information", from such data?**

An example



- What should we do with a signal that looks like this?
- First step is to build a description
  - E.g. compute mean and rms (variance)
- But what next?

An example



- What should we do with a signal that looks like this?

- First step is to build a description
  - E.g. compute mean and rms (variance)

- But what next?
  - We can think about the *spectrum*
  - Underlying idea: decompose signal into waves with different frequencies
  - And check which frequencies hold the most "energy"

Fourier series

- **Periodic functions can be expanded as Fourier series:**

$$f(x) = \sum_{n=-\infty}^{\infty} c_n \exp(inx)$$

$$c_n = (1/2\pi) \int_{-\pi}^{\pi} f(x) \exp(-inx) dx$$

- **This expression considers a period =** 2π
- **This can be modified to** L **through a simple change of variables (**y = xL/(2π)**)**
- **Convergence theory considers general functions integrable on the interval [-**π, π)
- **In practice, Fourier series are only used for periodic functions**
- **For *infinitely differentiable functions* (*e.g.* a Gaussian), we have exponential convergence:** $c_n \sim exp(-\mu|n|)$, μ is a positive constant

Fourier series

- **Periodic functions can be expanded as Fourier series:**

$$f(x) = \sum_{n=-\infty}^{\infty} c_n \exp(inx)$$

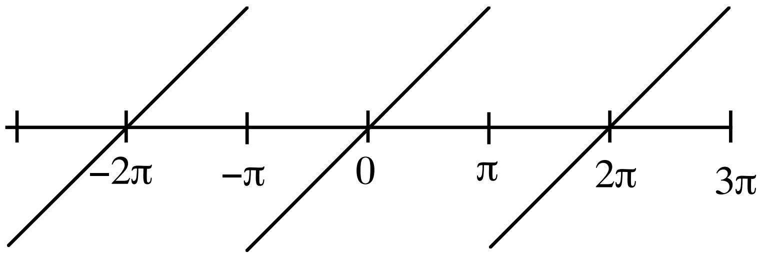
$$c_n = (1/2\pi) \int_{-\pi}^{\pi} f(x) \exp(-inx) dx$$

- **We will largely ignore the formal convergence theory and try to build intuition**

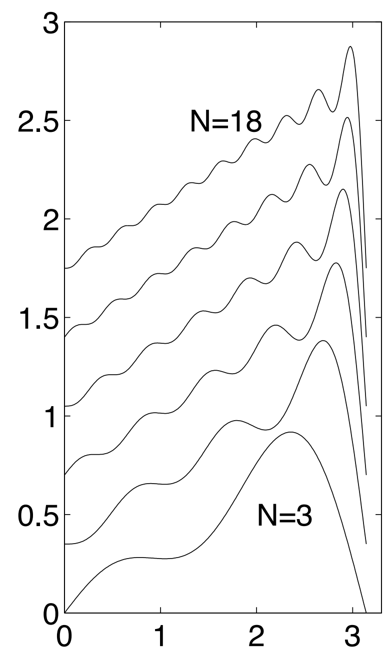
- **The rate-of-convergence of the series depends on the smoothness of the function**
  - **In the interior of the domain, we will simply assume the function and all of its derivatives are continuous**
  - **Let's look at a few examples with non-smooth behavior at the boundaries…**

Fourier series

- **Example 1: sawtooth function:**

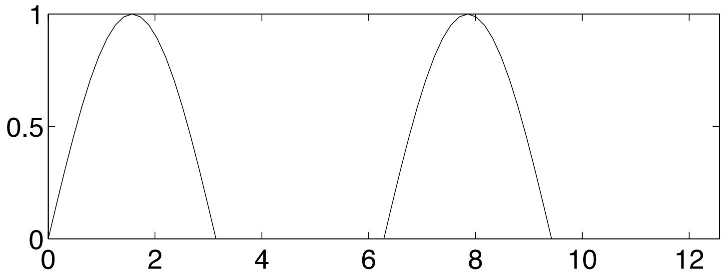


- **Function is discontinuous at "boundaries":** $f(-\pi) \neq f(\pi)$
- **"Convergence" is slow,** $|c_n| = \frac{1}{|n|}$
- **Figure shows approximation retaining 1st 2N terms in series (|n|<N)**



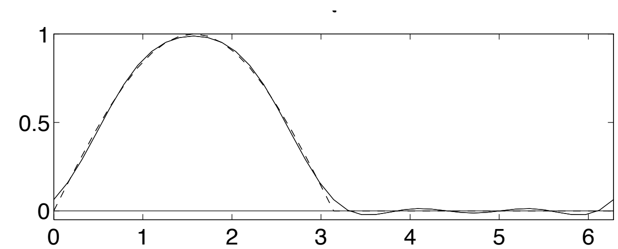
Fourier series

- **Example 2: half-wave rectifier:**



$$f(t) \equiv \begin{cases} \sin(t), & 0 < t < \pi \\ 0, & \pi < t < 2\pi \end{cases}$$

- **Function's derivative is discontinuous at "boundaries":** $f'(-\pi) \neq f'(\pi)$
- **Convergence is slow,** $|c_n| \sim \frac{1}{|n|^2}$
- **Figure shows approximation retaining 1st 4 terms in series (dashed curve is the approximation):**



Fourier series

- **A general result:**

- *If:*

  1.

$$f(\pi) = f(-\pi), f^{(1)}(\pi) = f^{(1)}(-\pi), ..., f^{(k-2)}(\pi) = f^{(k-2)}(-\pi)$$

  2. $f^{(k)}(x)$ *is integrable*

then the coefficients of the Fourier series

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

have the upper bounds

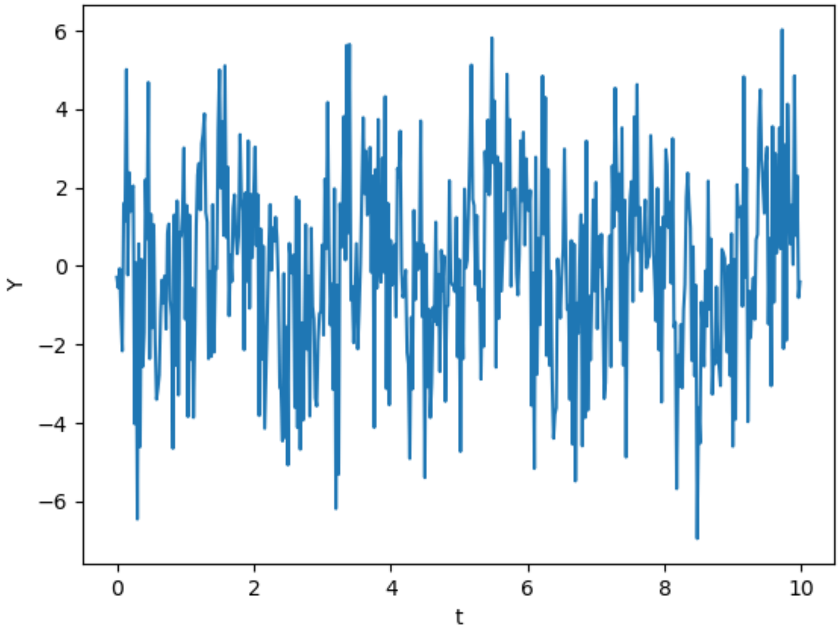
$$\mid a_n \mid \leq F/n^k; \qquad \mid b_n \mid \leq F/n^k$$

for some sufficiently large constant F, which is independent of n.

- **Proof: repeated application of integration by parts**

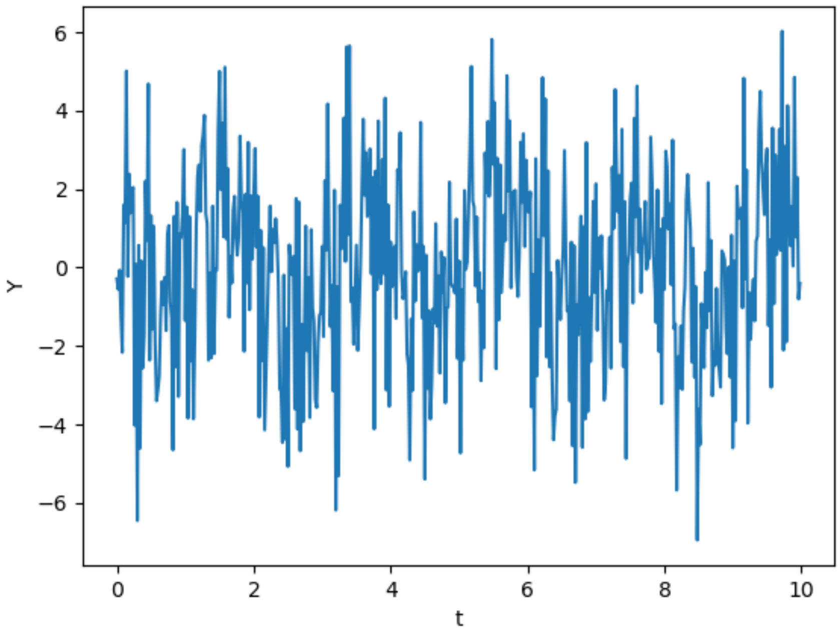
- **Note: this definition of a Fourier series is equivalent to previous with complex exponential**

An example



- **This signal is periodic (**T=10**)**
- **How do we compute its Fourier coefficients?**

An example



- **This signal is periodic**
- **How do we compute its Fourier coefficients?**
- **We will use the Discrete Fourier Transform (DFT)**
- **Available in** numpy.fft**, and** scipy.fftpack

Discrete Fourier transform

Now, our function is represented on a N-point discrete, equispaced grid:

$$t_j = j\Delta t,\ j = 0, 1, ..., N-1$$

We have a truncated Fourier series at the jth point:

$$f(t_j) = \sum_{n=-N/2}^{N/2-1} c_n exp(i2\pi n t_j/T) = \sum_{n=-N/2}^{N/2-1} c_n exp(i2\pi jn/N)$$

with NΔt = T, **and the inverse transform is now a discrete sum:**

$$c_n = \frac{1}{N}\sum_{j=0}^{N-1} f_j exp(-i2\pi n t_j/T) = \frac{1}{N}\sum_{j=0}^{N-1} f_j exp(-i2\pi jn/N)$$

- np.fft.fft **computes** c, **with the (1/N) factor omitted, but returns it in a … strange order:**

$$\mathrm{np.fft.fft(f)} = \mathrm{c_{np}} = \mathrm{N}\left(\mathrm{c_0, c_1, ..., c_{N/2-1}, c_{-N/2}, c_{-N/2+1}, ..., c_{-1}}\right)$$

Discrete Fourier transform

$$t_j = j\Delta t,\ j = 0, 1, ..., N-1$$

$$f(t_j) = \sum_{n=-N/2}^{N/2-1} c_n exp(i2\pi n t_j/T) = \sum_{n=-N/2}^{N/2-1} c_n exp(i2\pi jn/N)$$

$$c_n = \frac{1}{N}\sum_{j=0}^{N-1} f_j exp(-i2\pi n t_j/T) = \frac{1}{N}\sum_{j=0}^{N-1} f_j exp(-i2\pi jn/N)$$

- np.fft.fft **computes** c, **but returns it in a … strange order:**

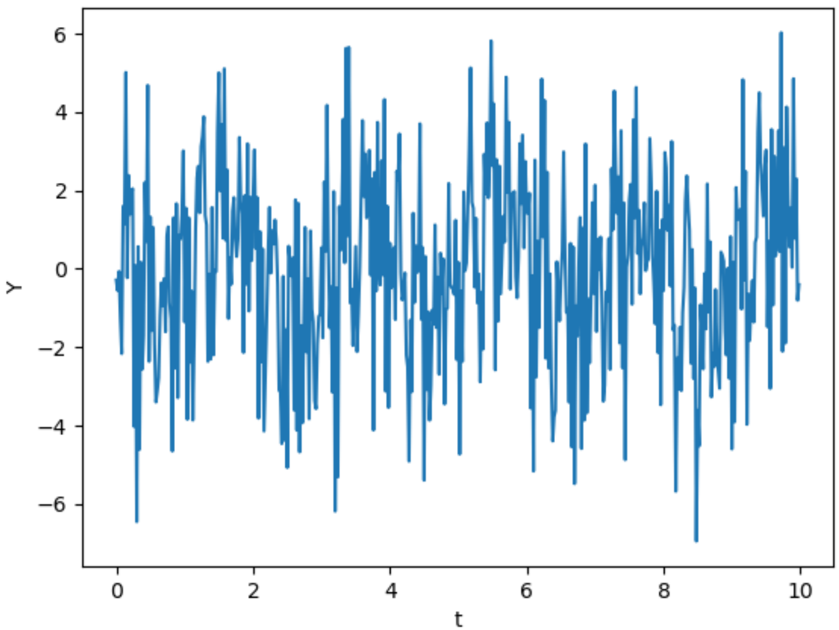
$$\mathrm{np.fft.fft(f)} = \mathrm{c_{np}} = \mathrm{N}\left(\mathrm{c_0, c_1, ..., c_{N/2-1}, c_{-N/2}, c_{-N/2+1}, ..., c_{-1}}\right)$$

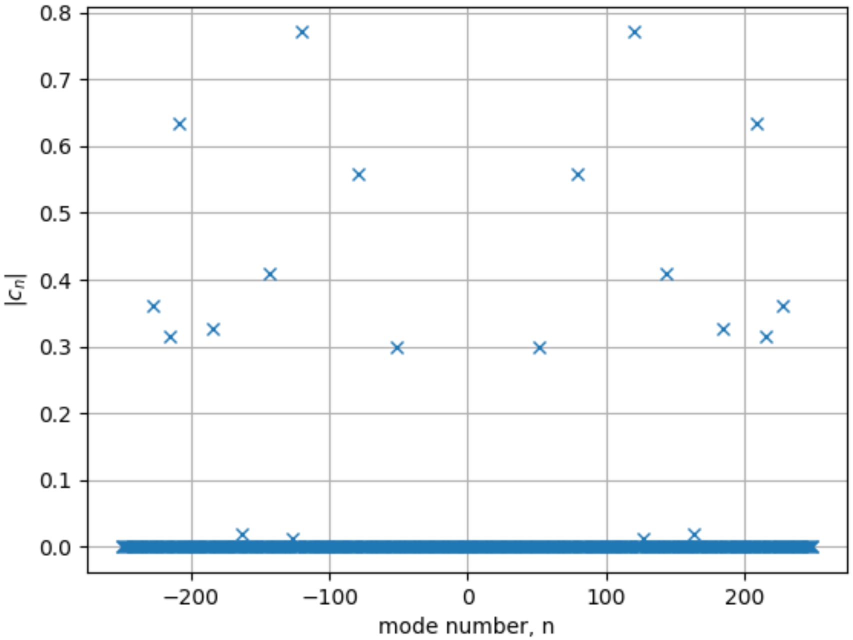
- **But we have a tool to help:**

$$\mathrm{np.fft.fftshift(c_{np})/N} = \mathrm{c_{-N/2}, c_{-N/2+1}, ..., c_{N/2-1}}$$

- **Let's go back to our example!**

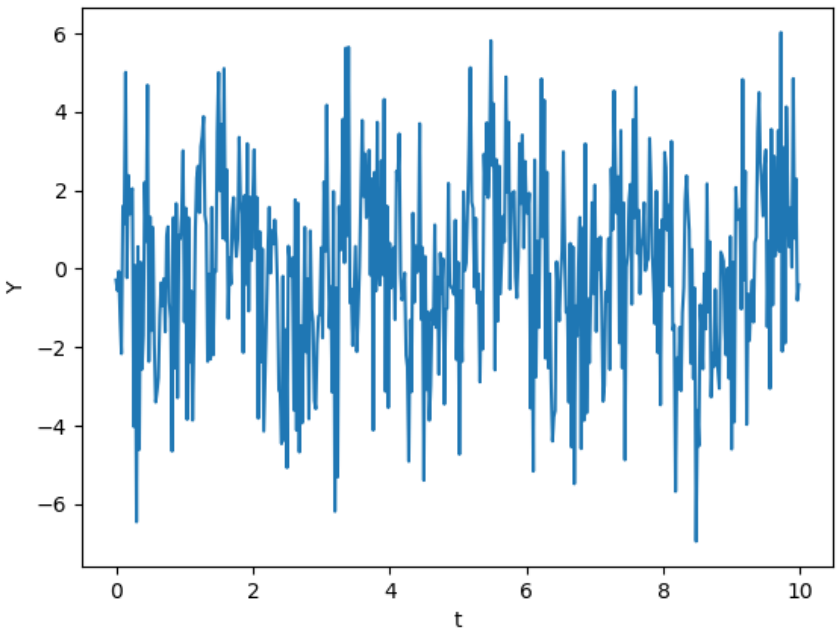
An example

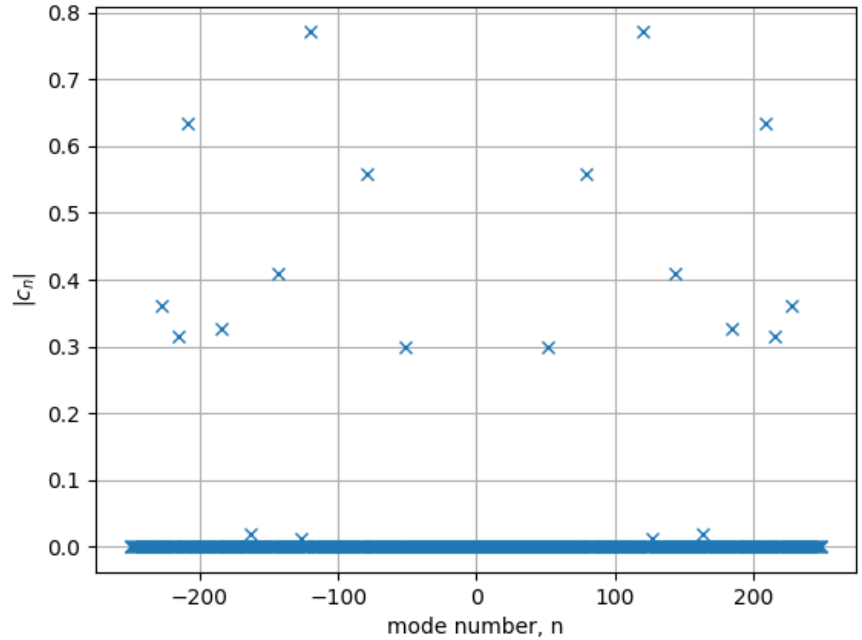





```
c = np.fft.fft(Y)
c = np.fft.fftshift(c)/Nt
n = np.arange(-Nt/2,Nt/2)

plt.figure()
plt.plot(n,np.abs(c),'x')
plt.xlabel('mode number, n')
plt.ylabel('$|c_n|$')
plt.grid()
```

An example






```python
c = np.fft.fft(Y)
c = np.fft.fftshift(c)/Nt
n = np.arange(-Nt/2,Nt/2)

plt.figure()
plt.plot(n,np.abs(c),'x')
plt.xlabel('mode number, n')
plt.ylabel('$|c_n|$')
plt.grid()
```

- **The signal is a superposition of 10 sine waves w/ different frequencies and with random amplitudes and phases:**

$$Y = \sum_{m=1}^{10} a_m sin(2\pi f_m t + \phi_m)$$

- f_m **is**: r/T, T=10, r **is randomly chosen integer,** 1<=r<Nt/2

An example

- **The signal is a superposition of 10 sine waves w/ different frequences and with random amplitudes and phases:**

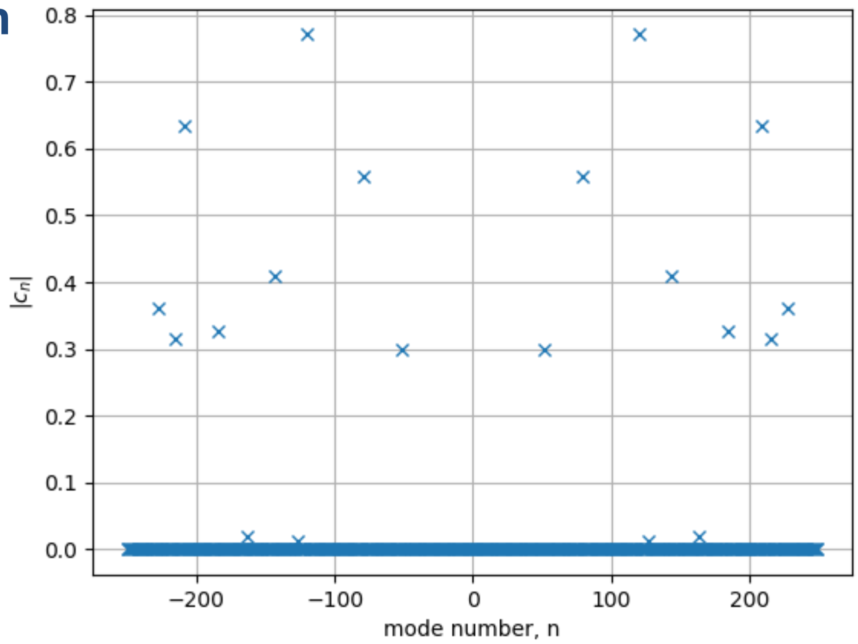
$$Y = \sum_{m=1}^{10} a_m sin(2\pi f_m t + \phi_m)$$

- f_m **is**: r/T**,** T=10, r **is randomly chosen integer,** 1<=r<Nt/2

- **Each** f_m **is a *frequency***

- $|c_n|^2$ **is the "energy" in a mode with frequency=** n/T

- **For real-valued data:** $c_n = c^*_{-n}$ **, so only n>=0 needed to be shown in figure (should use rfft instead of fft)**
- **It's essential that the timespan of the signal is an integer multiple of each frequency**



An example

- **The signal is a superposition of 10 sine waves w/ different frequences and with random amplitudes and phases:**

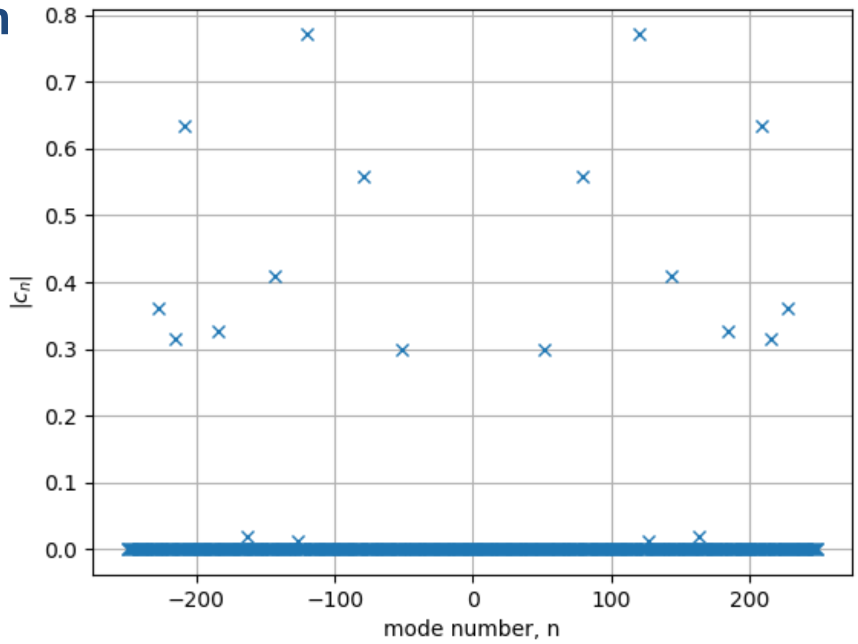
$$Y = \sum_{m=1}^{10} a_m sin(2\pi f_m t + \phi_m)$$

- f_m **is**: r/T, T=10, r **is randomly chosen integer,** 1<=r<Nt/2

- **Each** f_m **is a *frequency***

- $|c_n|^2$ **is the "energy" in a mode with frequency=** n/T

- **For real-valued data:** $c_n = -c_n^*$ **, so only n>=0 needed to be shown in figure (should use rfft instead of fft)**
- **It's essential that the timespan of the signal is an integer multiple of each frequency**



- (Nt/2-1)/T = Δt/2-1/T **is the highest frequency that can be "resolved"**

- **Rule-of-thumb: >2 points/period of highest-frequency component are needed (here, period = 1/f)**

Another example

- **Let's take two sine waves, with frequency** f=2/T **and** 2.5/T**. Here,** T **is the timespan of the signal and** 1/f **is the period of the wave:**

- **Problem setup:**

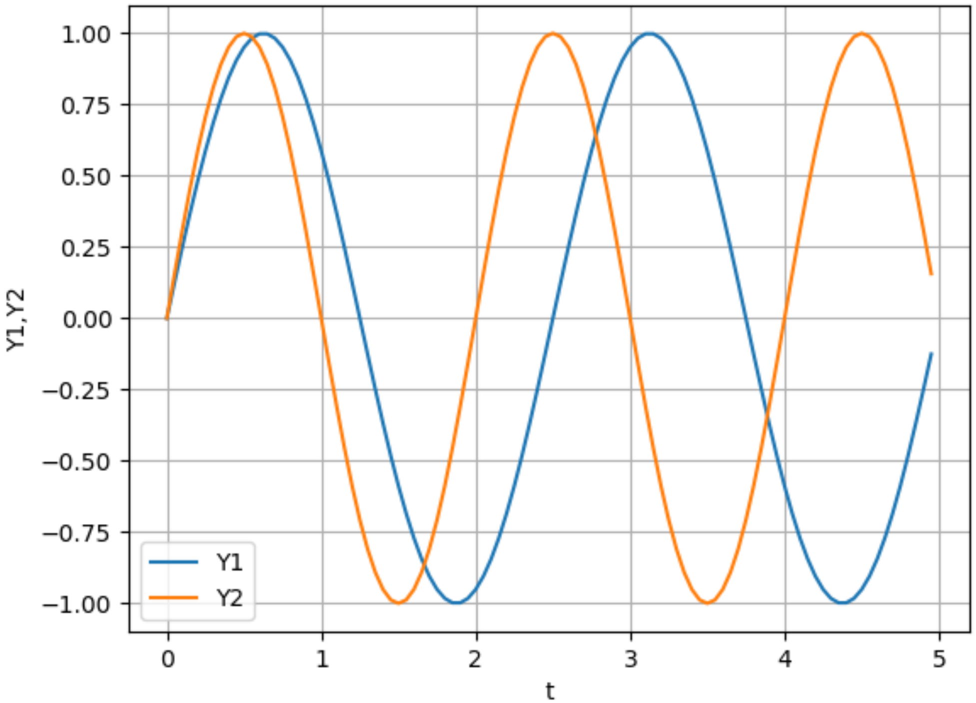
```
In [3]: T = 5

In [4]: Nt = 100

In [5]: t = np.linspace(0,T,Nt+1)

In [6]: t = t[:-1]

In [7]: f1 = 2/T

In [8]: f2 = 2.5/T

In [10]: Y1 = np.sin(2*np.pi*f1*t)

In [11]: Y2 = np.sin(2*np.pi*f2*t)

In [12]: c1 = np.fft.fft(Y1)/Nt

In [13]: c2 = np.fft.fft(Y2)/Nt
```

Another example

- **Let's take two sine waves, with frequency f=2/T and 2.5/T. Here, T is the timespan of the signal and 1/f is the period of the wave:**

- **Problem setup:**

```
In [3]: T = 5

In [4]: Nt = 100

In [5]: t = np.linspace(0,T,Nt+1)

In [6]: t = t[:-1]

In [7]: f1 = 2/T

In [8]: f2 = 2.5/T

In [10]: Y1 = np.sin(2*np.pi*f1*t)

In [11]: Y2 = np.sin(2*np.pi*f2*t)

In [12]: c1 = np.fft.fft(Y1)/Nt

In [13]: c2 = np.fft.fft(Y2)/Nt
```



The 2 waves

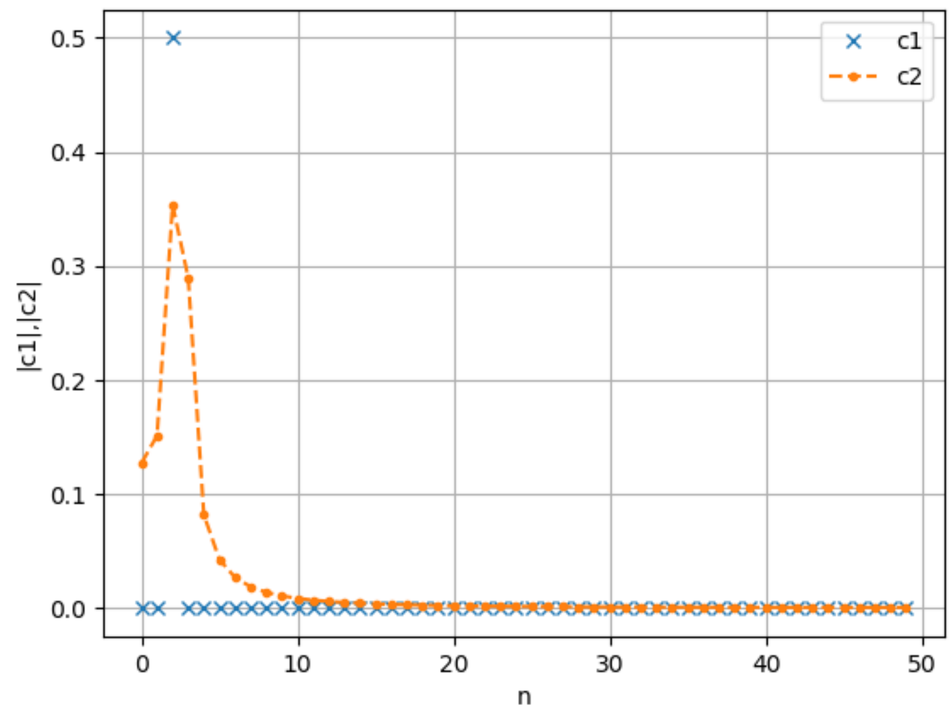
Imperial College
London

Another example

- **Let's take two sine waves, with frequency** f=2/T **and** 2.5/T**. Here,** T **is the timespan of the signal and** 1/f **is the period of the wave:**

- **Problem setup:**

```
In [3]: T = 5

In [4]: Nt = 100

In [5]: t = np.linspace(0,T,Nt+1)

In [6]: t = t[:-1]

In [7]: f1 = 2/T

In [8]: f2 = 2.5/T

In [10]: Y1 = np.sin(2*np.pi*f1*t)

In [11]: Y2 = np.sin(2*np.pi*f2*t)

In [12]: c1 = np.fft.fft(Y1)/Nt

In [13]: c2 = np.fft.fft(Y2)/Nt
```



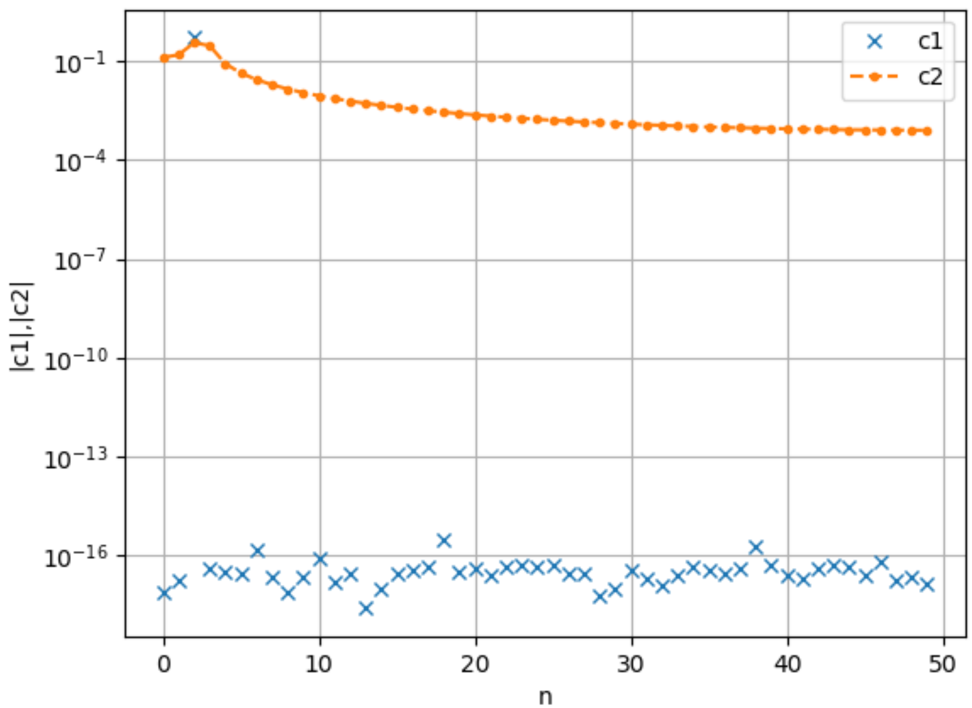
The 2 waves, note the "spread" of the energy across several frequencies for the 2nd wave

Another example

- **Let's take two sine waves, with frequency** f=2/T **and** 2.5/T**. Here,** T **is the timespan of the signal and** 1/f **is the period of the wave:**

- **Problem setup:**

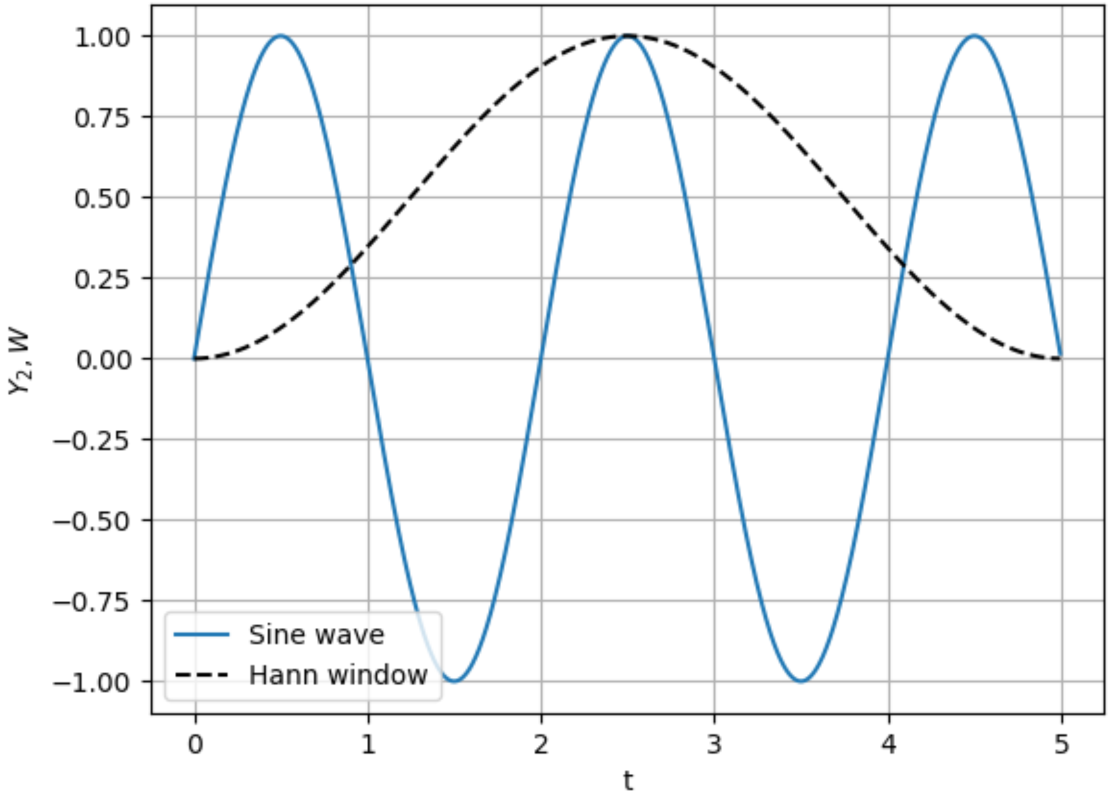
```
In [3]: T = 5

In [4]: Nt = 100

In [5]: t = np.linspace(0,T,Nt+1)

In [6]: t = t[:-1]

In [7]: f1 = 2/T

In [8]: f2 = 2.5/T

In [10]: Y1 = np.sin(2*np.pi*f1*t)

In [11]: Y2 = np.sin(2*np.pi*f2*t)

In [12]: c1 = np.fft.fft(Y1)/Nt

In [13]: c2 = np.fft.fft(Y2)/Nt
```



The 2 waves, note the "spread" of the energy across several frequencies for the 2nd wave and the (very) slow decay due to the "discontinuity" at t=5

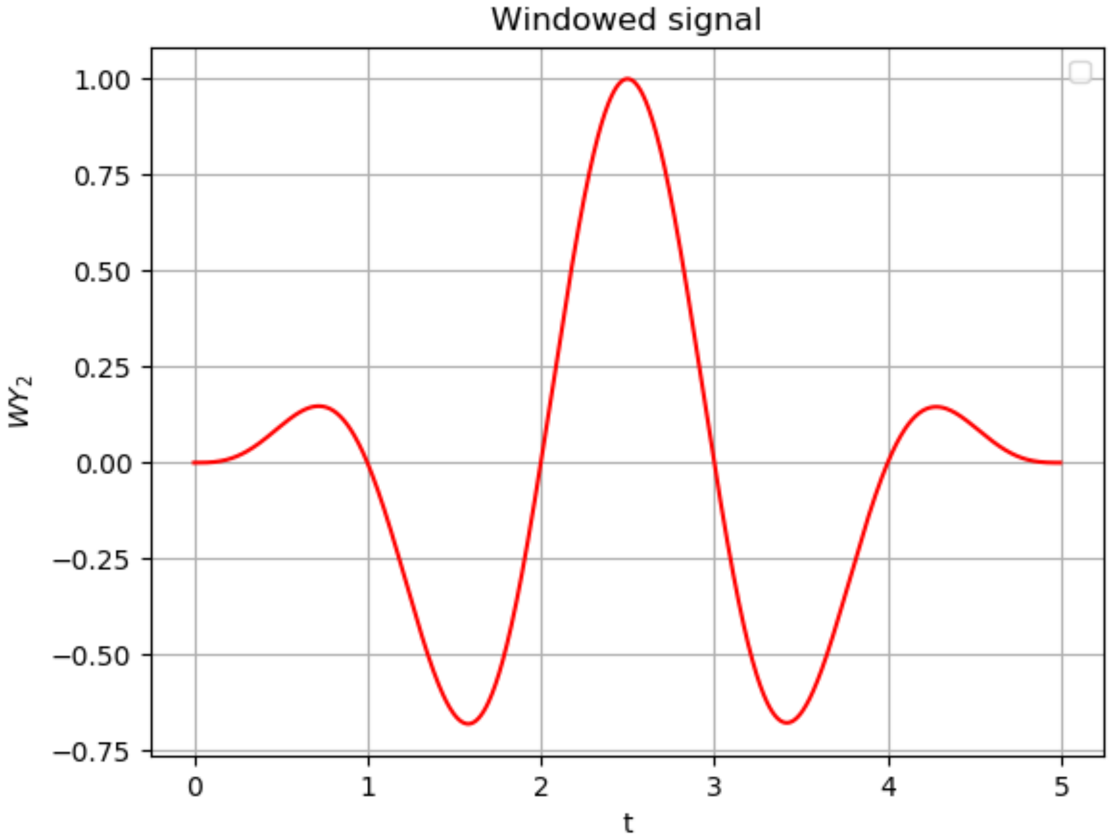
Another example

- There is no reason to generally expect signals to be periodic so this is an important issue
- The standard “fix” is to use windowing:
-



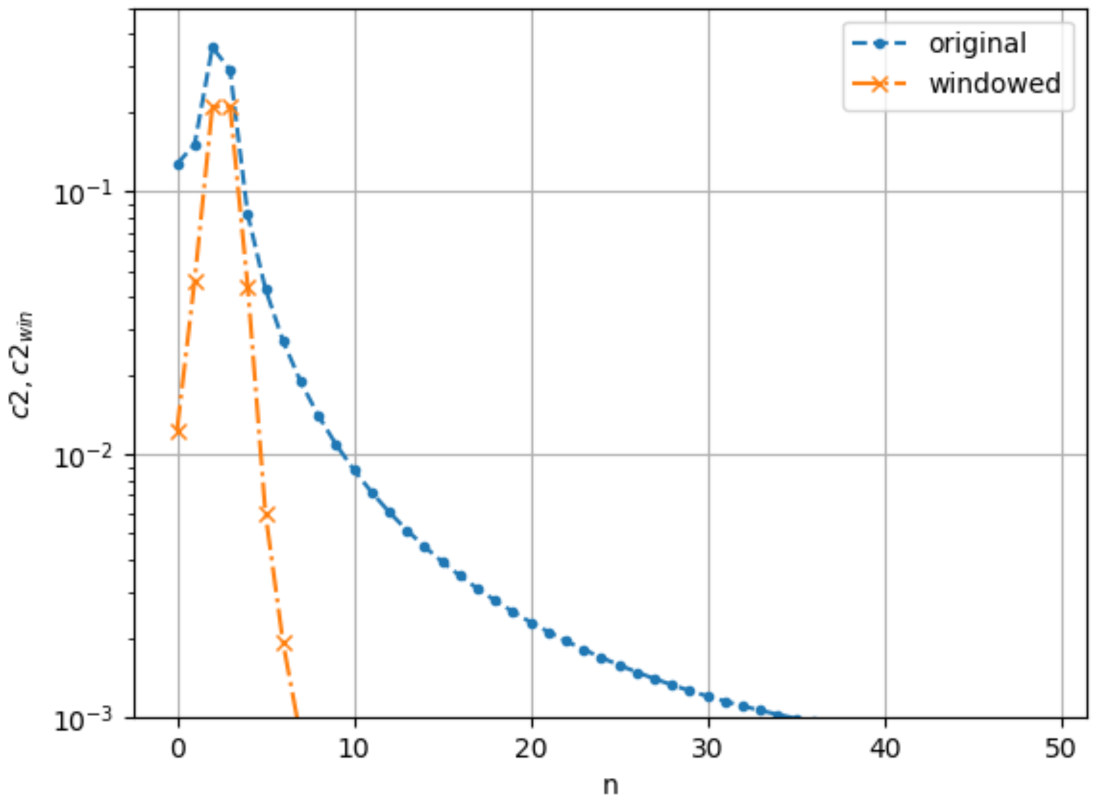
Another example

- There is no reason to generally expect signals to be periodic so this is an important issue
- The standard “fix” is to use windowing:
-



Another example

- There is no reason to generally expect signals to be periodic so this is an important issue
- The standard “fix” is to use windowing:
- The spectrum of the windowed signal “looks” more like a simple wave
- We have lost energy – there is no perfect solution
-



Data analysis

- In practice, we don't have to go through the windowing process ourselves
- Signal processing tools exist which:
  - Break the signal up into overlapping segments
  - Window the signal within each segment and compute the spectrum
  - Average the spectra from each segment (typically $|c|^2$ rather than $|c|$)
- This produces an estimate of the *autospectral density*
- And can be computed using *Welch's method*, scipy.signal.welch

```
In [201]: w2,Pxx2 = sig.welch(Y2)

In [203]: w2 = w2*Nt/T

In [206]: plt.semilogy(w2,Pxx2)
```

Data analysis

- In practice, we don't have to go through the windowing process ourselves

- Signal processing tools exist which:
  - Break the signal up into overlapping segments
  - Window the signal within each segment and compute the spectrum
  - Average the spectra from each segment (typically $|c|^2$ rather than $|c|$)

- This produces an estimate of the *autospectral density*

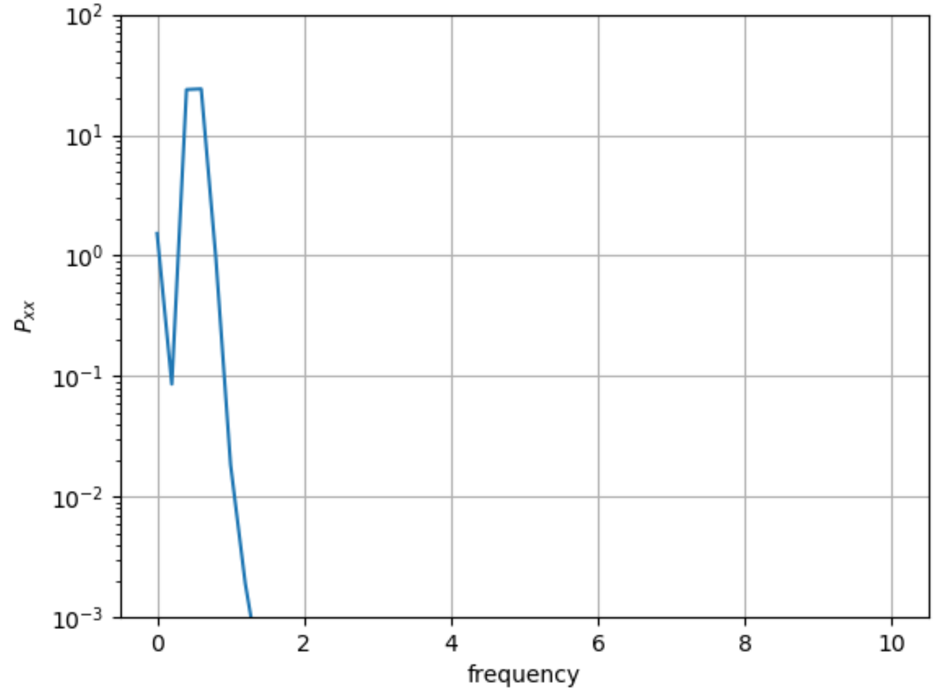
- And can be computed using *Welch's method*, scipy.signal.welch

```
In [201]: w2,Pxx2 = sig.welch(Y2)

In [203]: w2 = w2*Nt/T

In [206]: plt.semilogy(w2,Pxx2)
```

Data analysis

And can be computed using *Welch's method*, scipy.signal.welch

```
In [201]: w2,Pxx2 = sig.welch(Y2)

In [203]: w2 = w2*Nt/T

In [206]: plt.semilogy(w2,Pxx2)
```



Data analysis

Notes:

- **Consider a signal of length** T **with step** Δt **as a distribution of "energy" across a range of frequencies**
- **We need** $\Delta t < 2/f_{min}$ **to *resolve* the highest frequency components**
- **Also need** $T >> 1/f_{max}$ **to ensure "slow" components are contained within the signal**
  - **Not the case in our previous example**
  - **Often, slow components have the largest amplitude**

Data analysis

Notes:

- **Consider a signal of length** T **with step** Δt **as a distribution of "energy" across a range of frequencies**
- **We need** $\Delta t < 2/f_{min}$ **to *resolve* the highest frequency components**
- **Also need** $T >> 1/f_{max}$ **to ensure "slow" components are contained within the signal**
  - **Not the case in our previous example**
  - **Often, slow components have the largest amplitudes**
- **What about the running time?**
  - **Direct evaluation of the sum in the DFT requires** $O(N^2)$ **operations**
  - **But the FFT uses a divide and conquer approach and** $O(N\log_2 N)$ **operations are needed**