Git/Github

- For those not use to it…
- First, install git on your computer
- Next, make account on github
- Click on “repositories” and follow the instructions!
- NB - you’ll need to upload an ssh key. ssh-keygen will make one for you if you don’t have one already.
  - Google “github ssh-keys” for useful hits

Generating Correlated Noise

- If d=A(m)+n, then $\chi^2=n^TN^{-1}n$, where $N_{ij}=<n_in_j>$.
- Obviously true for diagonal N. Can express non-diagonal with change of basis. Note N is symmetric and positive definite (no such thing as a negative variance)
- Cholesky useful factorization: $N=LL^T$ (or R^TR - always check what computer did) where L is triangular.
- $N-1=(LL^T)^{-1}=L^{T-1}L^{-1}$. $\chi 2$ then = $(L^{-1}n)^TI(L^{-1}n)$. So, $L^{-1}n$ must be Gaussian with standard deviation of 1.
- In that case, n=Lg, where g is unit variance Gaussians.
- Note - this will be (very!) important when we get to MCMC.

HW Example

- Suggested using $N_{ij}=1+\delta_{ij}$.
- Is this matrix stationary?
- yes - that means we can use Fourier to generate data
- What is correlation function?
- $\mathrm{corr}(n_i n_{i+j})=1+\delta(j)$.
- What is $\langle F(k)^* F(k) \rangle$?
- $=\mathrm{FT}(\mathrm{corr})=\mathrm{FT}(1)+\mathrm{FT}(\delta(j))$. $\mathrm{FT}(1)=N\delta(k)$. $\mathrm{FT}(\delta(j))=1$.

Random Walk

- As a second example, let’s use noise matrix to generate random walks in a stationary way.
- Random walk done by adding a random (gaussian) value to get to next point from current point. $r_{n+1}=r_n+g$
- What is the variance of r_n? Is it stationary?
- What is the variance of r_i-r_j?
- We can use this to find covariance of $<r_i r_j>$.

Random Walk

- Random walk done by adding a random (gaussian) value to get to next point from current point. $r_{n+1}=r_n+g$
- What is the variance of r_n? Is it stationary?
  - variance of r_n is n if we start with $r_0=0$ and use unit gaussians.
- What is the variance of r_i-r_j?
  - $var(r_i-r_j)=|i-j|$
- We can use this to find covariance of $<r_ir_j>$.
  - $var(r_i-r_j)=<(r_i-r_j)^2>-<r_i-r_j>^2=<r_i^2>-2<r_ir_j>+<r_j^2>-0=|i-j|$
  - so, $<r_ir_j>=(var(r_i)+var(r_j)-|i-j|)/2$

Pseudo-stationary

- Formally, random walks are not stationary. However, they “feel” stationary since any section is qualitatively indistinguishable from any other.
- One (only slightly illegal) trick: say we peg two ends of a long segment to zero, then look at small piece of middle.
- Then variance is more-or-less constant.
- Pick variance to be much larger than length, then diagonals are all shifted down.
- $N_{ij}=V-|i-j|/2$, and we have a stationary noise matrix.
- Of course, we could start with non-stationary as well…

Circulant

- Fourier transforms want things to be not just stationary but circulant, i.e. $<f(x)f(x+i)>=f(x)f(x+i\pm N)$ since they wrap around.
- That means if we want to use FFTs, we need to make sure we have thought about this.
- How could we make a *circulant* pseudo-stationary random walk N?

Circulant V1

- Well, one way is to make a covariance matrix for (i-j), and (i-j+N) and (i-j-N) and take the largest value.
- This works fine. See: make_data_rw_pseudostat_circ.py
- NB - our covariance matrix has gotten very close to singular. Cholesky is failing, but eigenvalues will work. Always want to check FFT values as well…

Pseuo-circulant

- If we’re creating data with FFTs, another way is to just paste a reversed copy of the covariance onto itself.

- Create double-sized fake data, then use half. See: make_data_rw_pseudostat_circ_v2.py

- We can also look at *power spectrum* in there, $\langle F(k)^2 \rangle$.

Finally…

- Now that we’ve got some familiarity with matched filters, correlated noise, and Fourier transforms applied to data, let’s look at LIGO data!

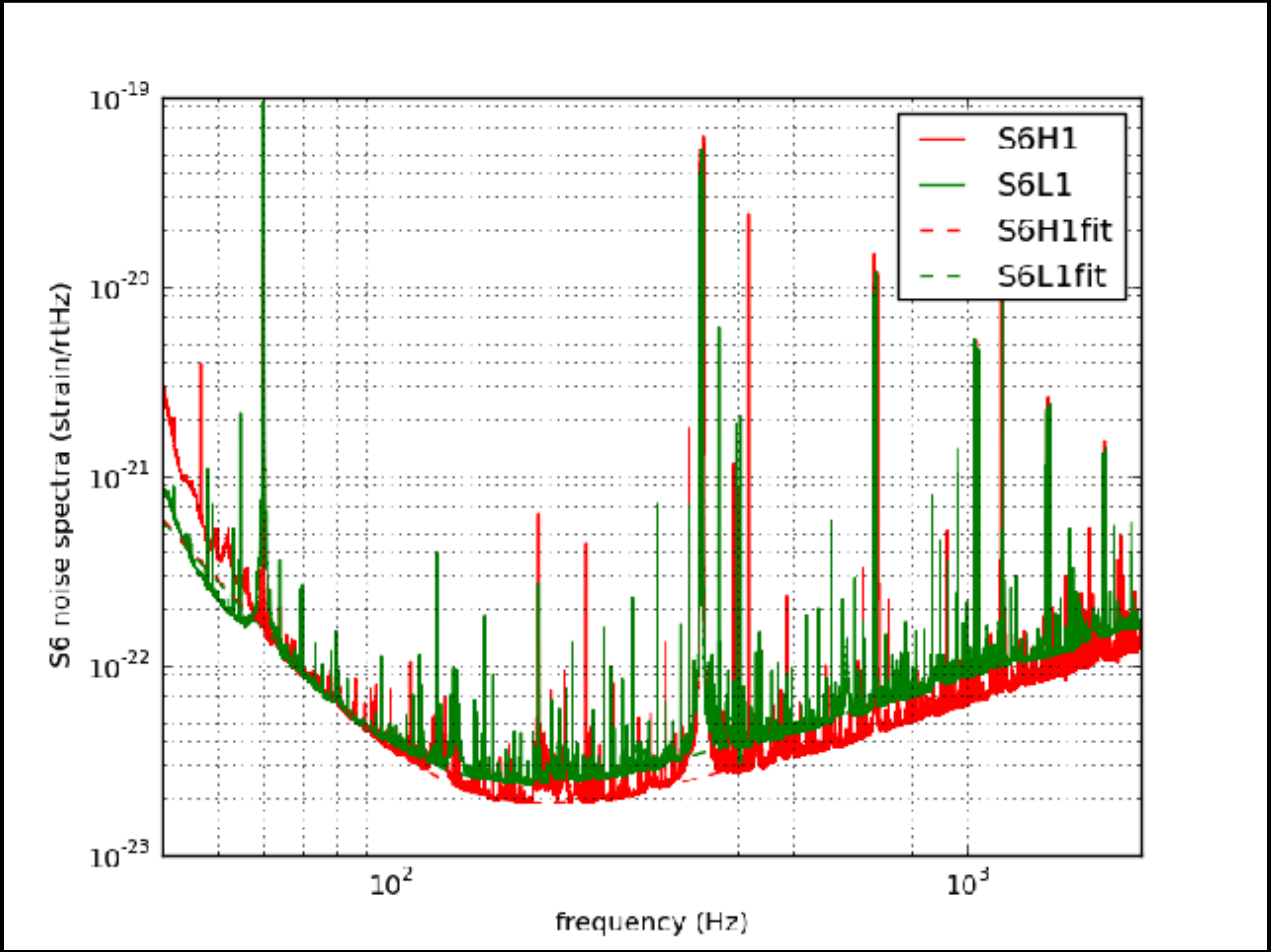
- Download stuff from LOSC (LIGO open science tutorial), but I’ve posted one on github.

- You can read data with “simple_read_ligo.py”

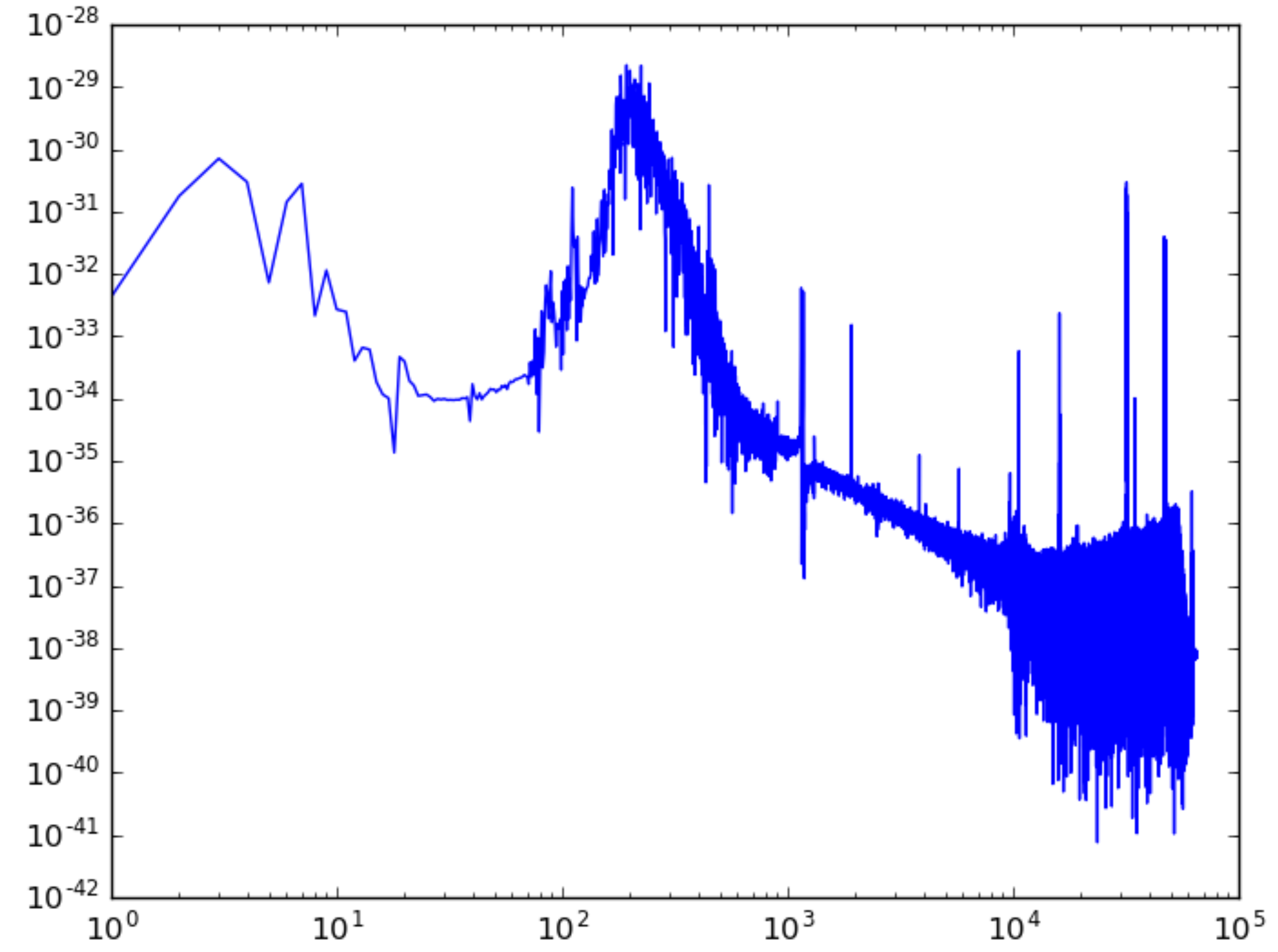
Look at data

- Strain is actual output of interferometer - a bit more on physics in coming lectures.
- Template is the signal we should see in the strain.
- How are we going to matched-filter this data?

First, what should we see for noise?



Let’s try to reproduce this



Well…

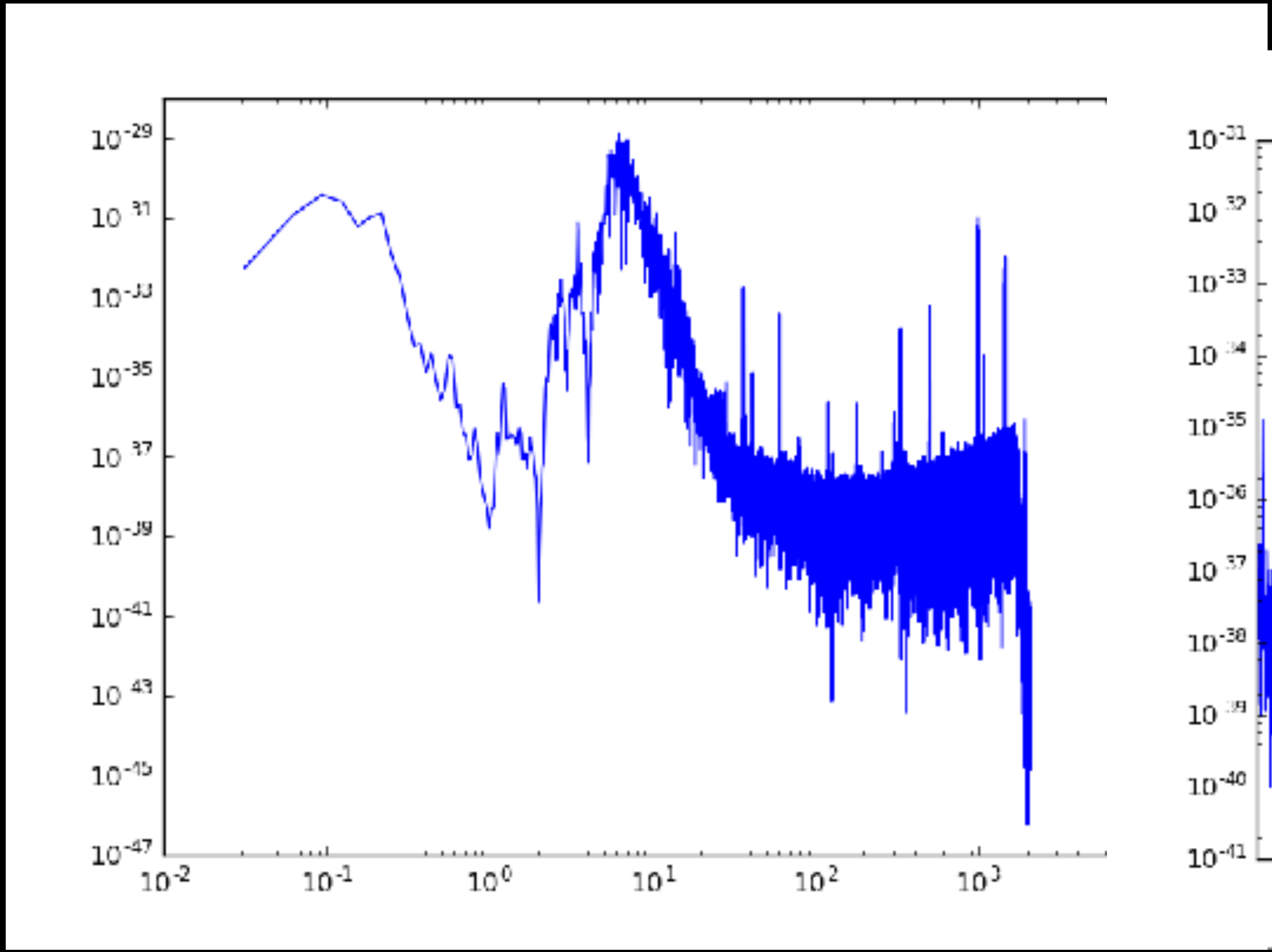
- They look nothing alike.
- Why? First off, what is FT of a line?
- Usual solution is to *window* data. Many types of windows possible (wikipedia lists ~30), but main thing is they go to zero (or very close) smoothly at the edges.
- Let’s try an FFT of the windowed version of data.
- Much better!

Spectral Resolution

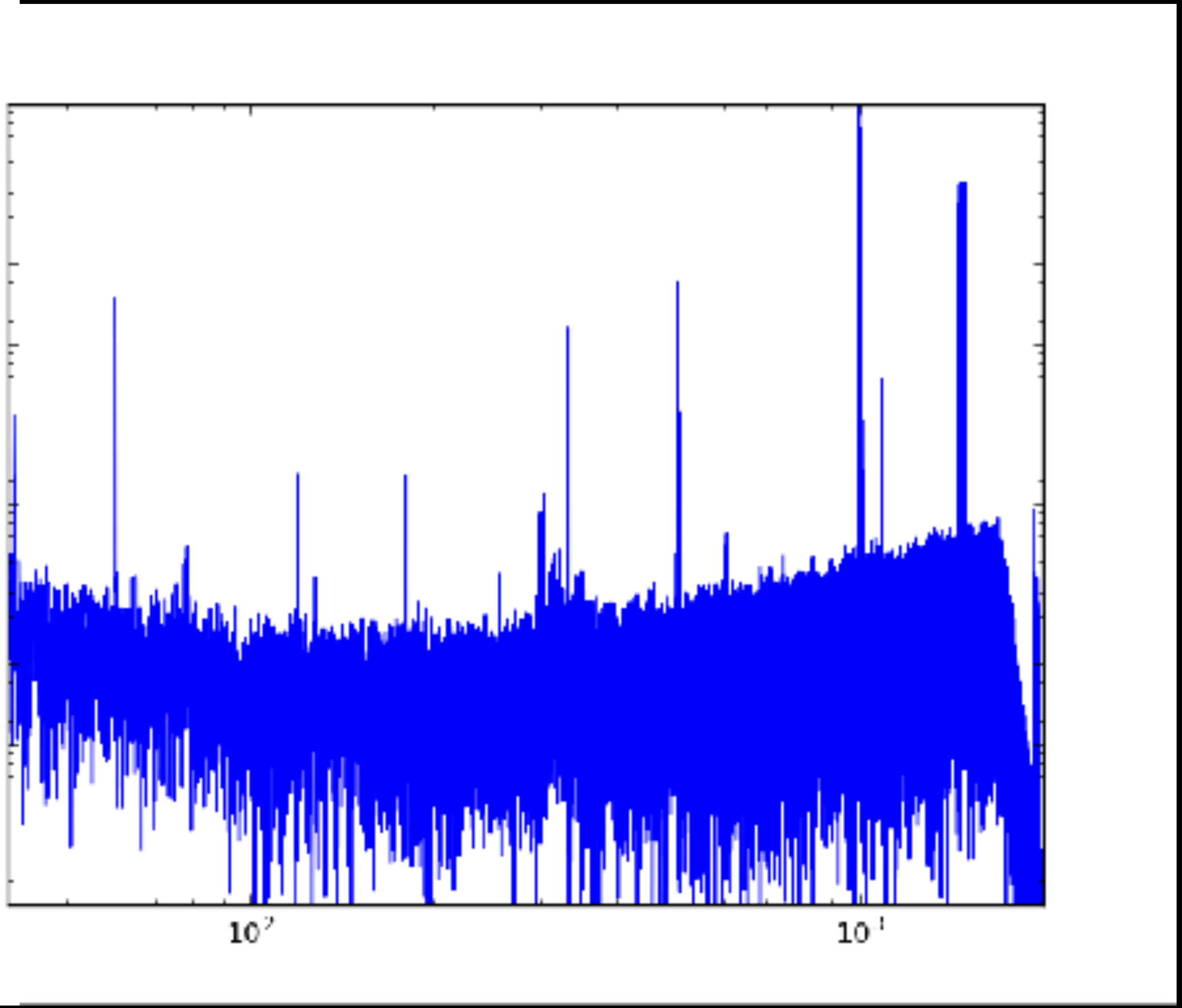
- I would like to assign real frequencies to my k-axis.
- What is the spectral resolution of my dataset?
- How do I get the true frequencies then?

Output

- Getting closer!



10^{-29}
10^{-31}
10^{-33}
10^{-35}
10^{-37}
10^{-39}
10^{-41}
10^{-43}
10^{-45}
10^{-47}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}



10^{-31}
10^{-32}
10^{-33}
10^{-34}
10^{-35}
10^{-36}
10^{-37}
10^{-38}
10^{-39}
10^{-40}
10^{-41}
10^{1}