Recap: Matched Filter

- If we have some template A that we want to find in our data, then our best-fit value for the amplitude is: $m=A^TN^{-1}d/A^TN^{-1}A$.

- If the noise is stationary an we want to shift the template, we can do so by taking the cross-correlation of A with $N^{-1}d$. Modulo edge effects, $A^TN^{-1}A$ is constant and so only needs to be computed once.

SNR

- Very often we want not just the amplitude, but to know if the matched filter has found something statistically significant.

- The error on m is just sqrt($(A^T N^{-1} A)^{-1}$), so the matched filter SNR is just correlation of $(A, N^{-1} d)/\mathrm{sqrt}(A^T N^{-1} A)$.

- Of course, if your errors aren't right, maybe you want to get the error by looking at the RMS of the matched filter.

How about changing noise?

- Often we’re searching for things where noise is varying - say, searching for sources in a map that is deeper in the middle.
- Top: $A^T N^{-1} d$ is easy - correlate A with $N^{-1}d$.
- Bottom: $A^T N^{-1} A$ we can correlate N^{-1} with A^2.
- Amplitude stays ratio of the two (bottom is now a vector instead of a scalar), and SNR is top/√(bottom)

Fourier Stuff

- Matched filters usually require Fourier transforms.
- Fourier transforms on the computer are discrete, since we don't have infinite computing power/RAM…
- Understanding how the computer does them is important.
- Usual discrete Fourier transform (DFT) is $F(k)=\sum f(x)\exp(-2\pi ikx/N)$ for N data points (not the noise matrix for now) where x goes from 0 to N-1, and k goes from 0 to N-1.

DFT Examples

- What is the DFT of $f(x)=1$ if $x=0$, otherwise 0?
- What is the DFT of $f(x)=1$ everywhere?
- If we have $DFT(f(x))$, what is $DFT(f(x+j))$?
- If $f(x)$ is real, what can we say about $F(k)$?

DFT Examples

- What is the DFT of $f(x)=1$ if $x=0$, otherwise 0?
  - $F(k)=1$ everywhere
- What is the DFT of $f(x)=1$ everywhere?
  - $F(0)=n$, otherwise 0
- If we have DFT(f(x)), what is DFT(f(x+j))?
  - $F_{new}(k)\text{->}\exp(-2\pi ikj/N)F(k)$
- If f(x) is real, what can we say about F(k)?
  - $F(-k)=F^*(k)$. NB - numpy.fft.rfft/irfft will do real-to-complex transforms.

DFT as Rotation

- if $F(k)=\sum f(x)\exp(-2\pi ikx/N)$
- For a single k value, this is just a dot product: $F(k_j)=f(x)\cdot\exp(-2\pi ik_jx/N)$
- Once again, we can write this down using linear algebra: $F=Tf$ where $T_{mn}=\exp(-2\pi imn/N)$ (switching from i,j to m,n to avoid confusion with sqrt(-1))

DFT as Rotation 2

- First, note that T is symmetric, since swapping n&m doesn't change anything.
- What is the dot product of columns of T? Note that we want the dot product of a vector with itself to be the length, which implies we conjugate one copy.
- So, want $\sum \exp(2\pi imn/N)\exp(-2\pi imn'/N)$
  - This equals $N\delta(m-n)$
- What is T^{-1} then? just T^*/N, so $T^*T=NI$.
- Tells us that the inverse transform is just $f(x)=1/N\sum F(k)\exp(2\pi ikx/N)$

Normalizations

- NB - a more natural mathematical definition would be $T=\exp(-2\pi imn/N)/\mathrm{sqrt}(N)$. Then T^{-1} would be T^{*}.
- Not a good idea to put in extra square roots on a computer though…
- Becomes important when estimating noise in Fourier space.
- What is Parseval's theorem using usual DFT definitions?

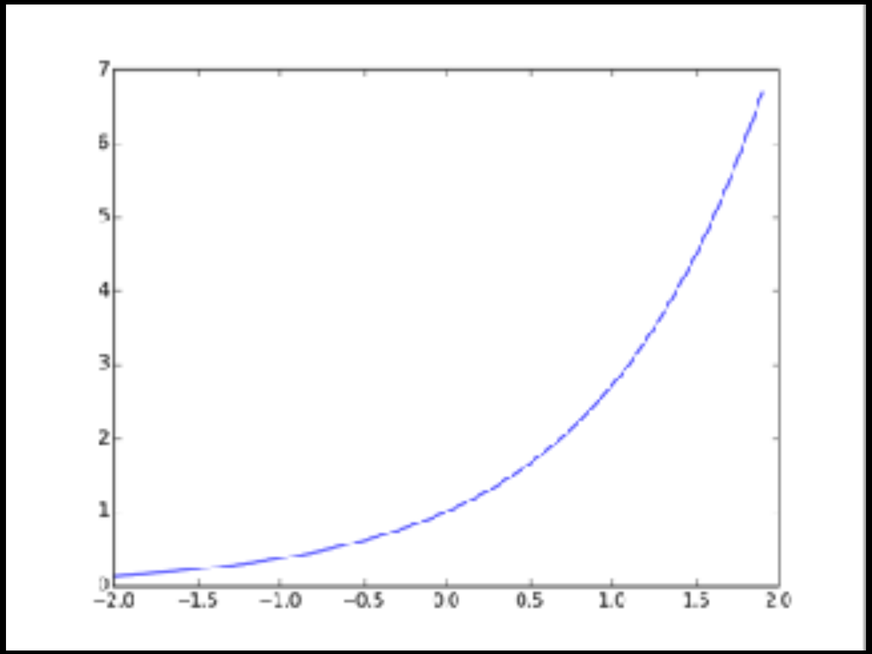
Periodicity

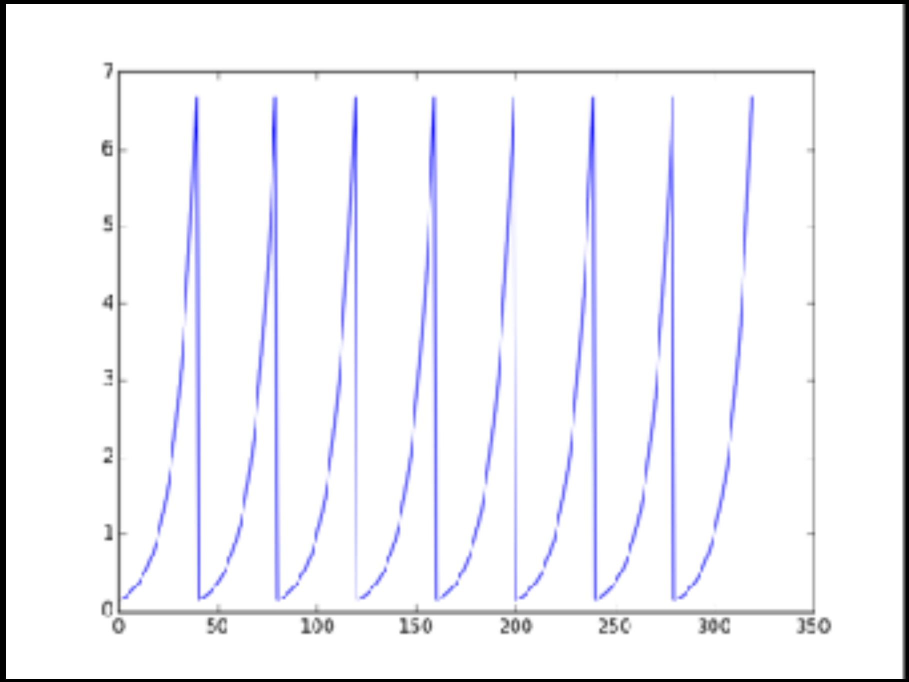
- $f(x)=\sum F(k)\exp(2\pi ikx/N)$
- What is $f(x+N)$? $\sum F(k)\exp(2\pi ik(x+N)/N)$
- $=\sum F(k)\exp(2\pi ik)\exp(2\pi ikx/N)$.
- $\exp(2\pi ik)=1$ for integer k, so $f(x+N)==f(x)$
- DFT's are periodic - they just repeat themselves ad infinitum.

Periodicity

```
import numpy
from matplotlib import pyplot as plt

x=numpy.arange(-2,2,0.1)
y=numpy.exp(x)
plt.plot(x,y)
plt.savefig('fft_exp')
plt.show()
yy=numpy.concatenate((y,y))
yy=numpy.concatenate((yy,yy))
yy=numpy.concatenate((yy,yy))
plt.plot(yy)
plt.savefig('fft_exp_repeating')
plt.show()
```





- You may think you're taking top transform. You're not - you're taking the bottom one.
- In particular, jumps from right edge to left will strongly affect DFT

Aliasing

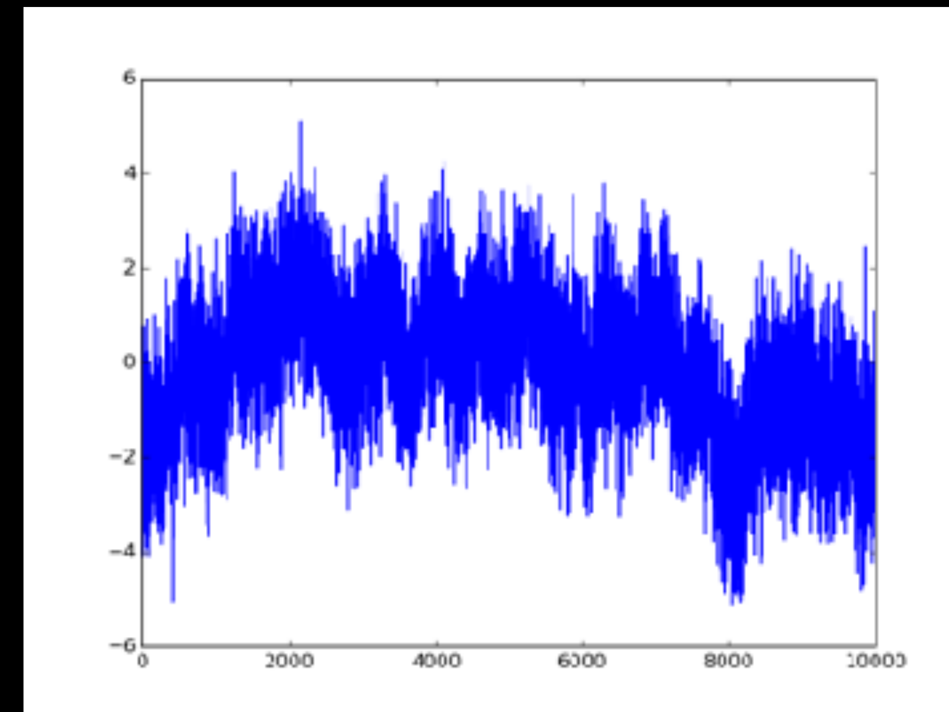
- $f(x)=\sum F(k)\exp(2\pi ikx/N)$
- What if I had higher frequency, k>N? let $k^*=k-N$ (i.e. k^* low freq.)
- $f(x)=\sum F(k)\exp(2\pi i(k^*+N)x/N=\sum F(k)\exp(2\pi ix)\exp(2\pi ik^*x/N)$
- for x integer, middle term goes away: $\sum F(k^*+N)\exp(2\pi ik^*x/N)$
- High frequencies behave exactly like low frequencies - power has been *aliased* into main frequencies of DFT.
- Always keep this in mind! Make sure samples are fine enough to prevent aliasing.

Negative Frequencies

- All frequencies that are N apart behave identically
- DFT has frequencies up to (N-1).
- Frequency (N-1) equivalent to frequency (-1). You will do better to think of DFT as giving frequencies (-N/2,N/2) than frequencies (0,N-1)
- *Sampling (Nyquist) theorem*: if function is band-limited - highest frequency is ν - then I get full information if I sample *twice* per frequency, $dt=1/(2\nu)$. Factor of 2 comes from aliasing.

Non-white noise

- Let's say we wanted to make some non-white noise. Typical case: variance is say 1/f plus a constant (white noise).
- Characterized by *knee frequency*, where 1/f part matches white noise, and index since not always exactly 1/f^1.0.
- Often called pink noise, flicker noise, or just 1/f noise. In astro, 1/f usually doesn't mean index is -1, just negative.




```
import numpy as np
from matplotlib import pyplot as plt

#specify what sort of noise/data we want
n=10000
alpha=-2.0
knee=100.0

#one easy thing to do is generate white noise, then scale its Fourier transform
dat=np.random.randn(n)
datft=np.fft.rfft(dat)
nuvec=np.arange(len(datft))+1 #the plus one is so the bottom frequency isn't zero


filtvec=np.sqrt(1+(nuvec/knee)**alpha)
datft=datft*filtvec
dat_pink=np.fft.irfft(datft)

plt.ion()
plt.clf()
plt.plot(dat_pink)
```