

Lecture 6

Stationary noise applied to , non-linear
least-squares, maybe matched filters?

Stationary Noise

- A common case for correlated noise is noise depends only on separation of points, $N_{ij}=f(|i-j|)$
- This is called *stationary* noise, since on average, noise properties don't change with time.
- We can invoke the power of Fourier transforms to enormously simplify stationary noise.
- Note - computer takes the discrete Fourier transform (DFT):
 $F(k)=\sum f(x)\exp(-2\pi i k x/N)$. I suggest you commit this to memory.
- IDFT: $f(x)=1/N \sum F(k)\exp(2\pi i k x/N)$
- Note: x, k integers, go from 0 to $N-1$.
- Also note: $\sum \exp(-2\pi i k x/N)=0$ for integer k, x , unless $k=0$, in which case it's N . This is the discrete equivalent of a Dirac- δ

Stationary Noise 2

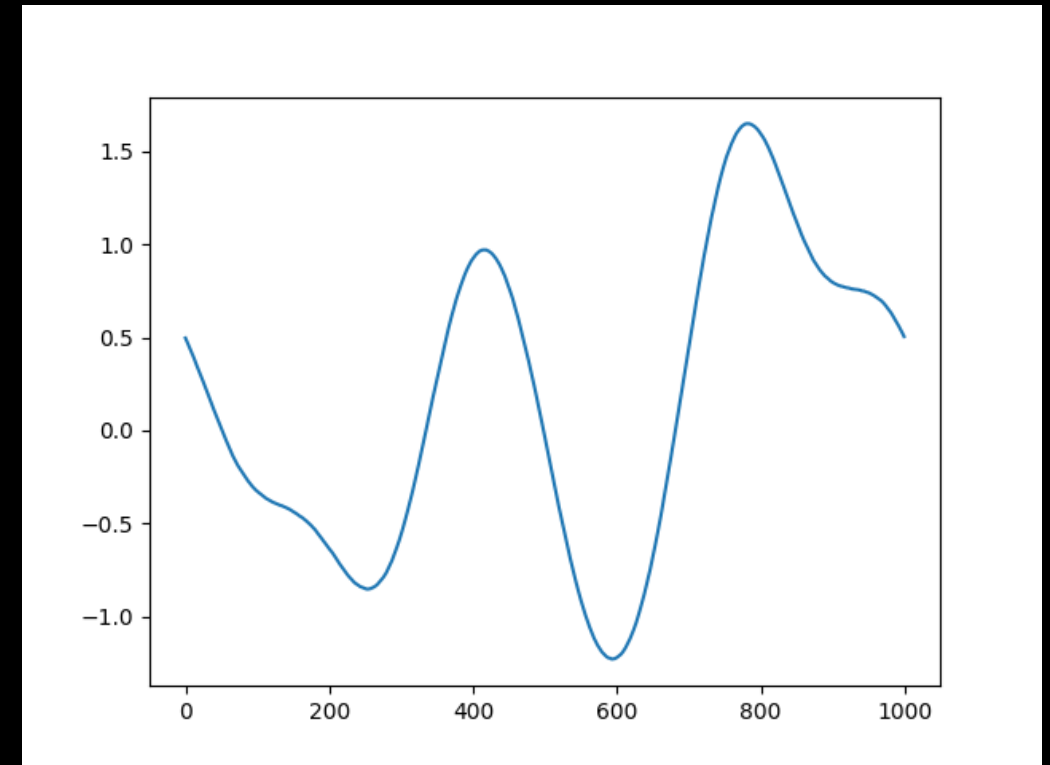
- Say we have noise where $\langle f(x)f(x+dx) \rangle = g(dx)$ (not of x)
- Fourier space: $\langle F(k)F^*(k') \rangle = \langle \sum f(x)\exp(-2\pi i k x/N) \sum f(x')\exp(2\pi i k' x'/N) \rangle$
- Can swap x' for $x+dx$, since sum is over all points, can sum over dx :
- $\langle F(k)F^*(k') \rangle = \langle \sum f(x)\exp(-2\pi i k x/N) \sum f(x+dx)\exp(2\pi i k' (x+dx)/N) \rangle$
- Reorder sum over x then dx : $\langle \sum \exp(2\pi i k' dx/N) \sum f(x)f(x+dx)\exp(2\pi i x(k'-k)/N) \rangle$
- Now $\langle f(x)f(x+dx) \rangle = g(dx)$ (by assumption), can come out. $\sum \exp(2\pi i k' dx/N) g(dx) \sum \exp(2\pi i x(k'-k)/N)$
- Interior goes to N for $k'=k$, left with $N \sum g(dx) \exp(2\pi i k dx/N) \delta_{kk'}$, so Fourier transform of noise is diagonal.
- Further, variance of $F(k)$ given by Fourier transform of correlation function $g(dx)$.

Correlated Noise via DFT

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

n=1000
x=np.fft.fftfreq(n)*n
sig=100
mycorr=np.exp(-0.5*x**2/sig**2)
mysps=np.fft.rfft(mycorr)

dat=np.random.randn(n)
datft=np.fft.rfft(dat)
dat_corr=np.fft.irfft(datft*np.sqrt(mysps))
```



- Code is much shorter!
- And much (much) faster!

Correlated Noise Summary

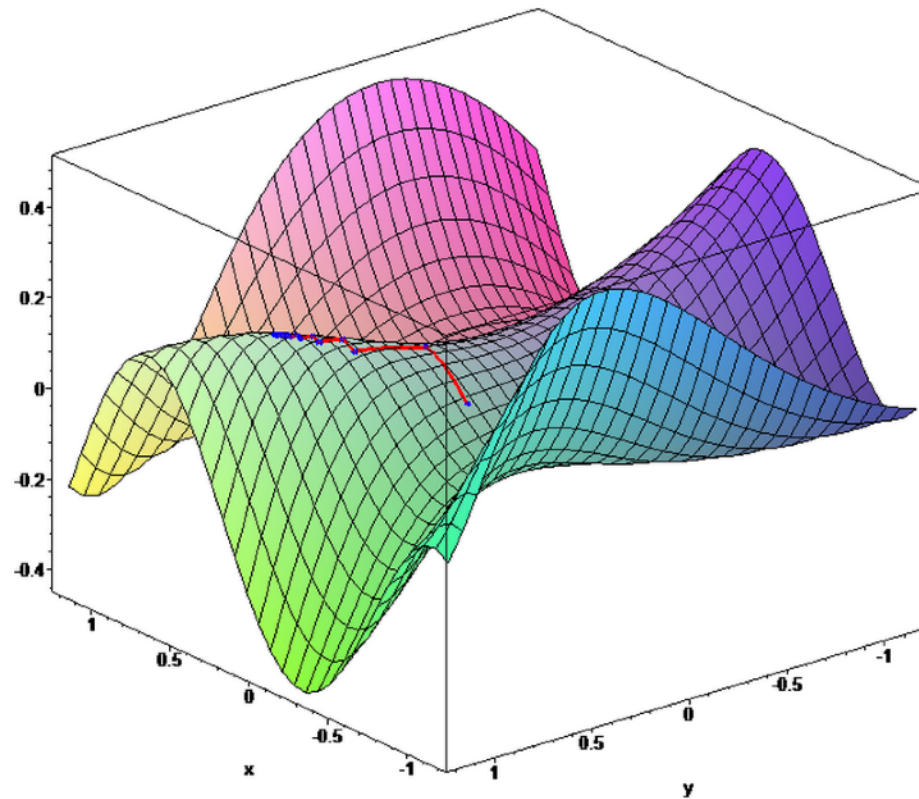
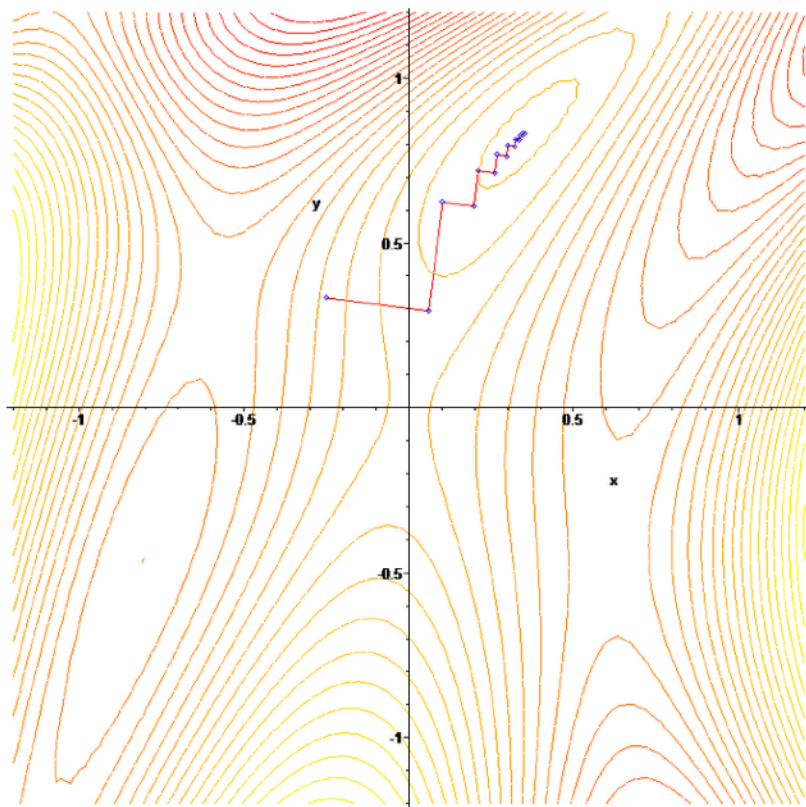
- In real life, we frequently have noise that is correlated between data points.
- Our entire least-squares framework carries over, as long as $N_{ij} = \langle n_i n_j \rangle$
- We can generate correlated noise using e.g. Cholesky (or eigenvalues, but Cholesky faster)
- If the noise is stationary, the noise is diagonal in Fourier space.
- Can generate correlated stationary noise realization very fast with Fourier transforms, since $\langle F(k)^2 \rangle = \text{DFT}(g(dx))$ where $g(dx) = \langle f(x)f(x+dx) \rangle$

Nonlinear Fitting

- Sometimes data depend non-linearly on model parameters
- Examples are Gaussian and Lorentzian ($a/(b+(x-c)^2)$)
- Often significantly more complicated - cannot reason about global behaviour from local properties. May be multiple local minima
- Many methods reduce to how to efficiently find the “nearest” minimum.
- One possibility - find steepest downhill direction, move to the bottom, repeat until we’re happy. Called “steepest descent.”
- How might this end badly?

Steepest Descent

The "Zig-Zagging" nature of the method is also evident below, where the gradient ascent method is applied to $F(x, y) = \sin\left(\frac{1}{2}x^2 - \frac{1}{4}y^2 + 3\right) \cos(2x + 1 - e^y)$.



From wikipedia. Zigagging is inefficient.

Worked Example

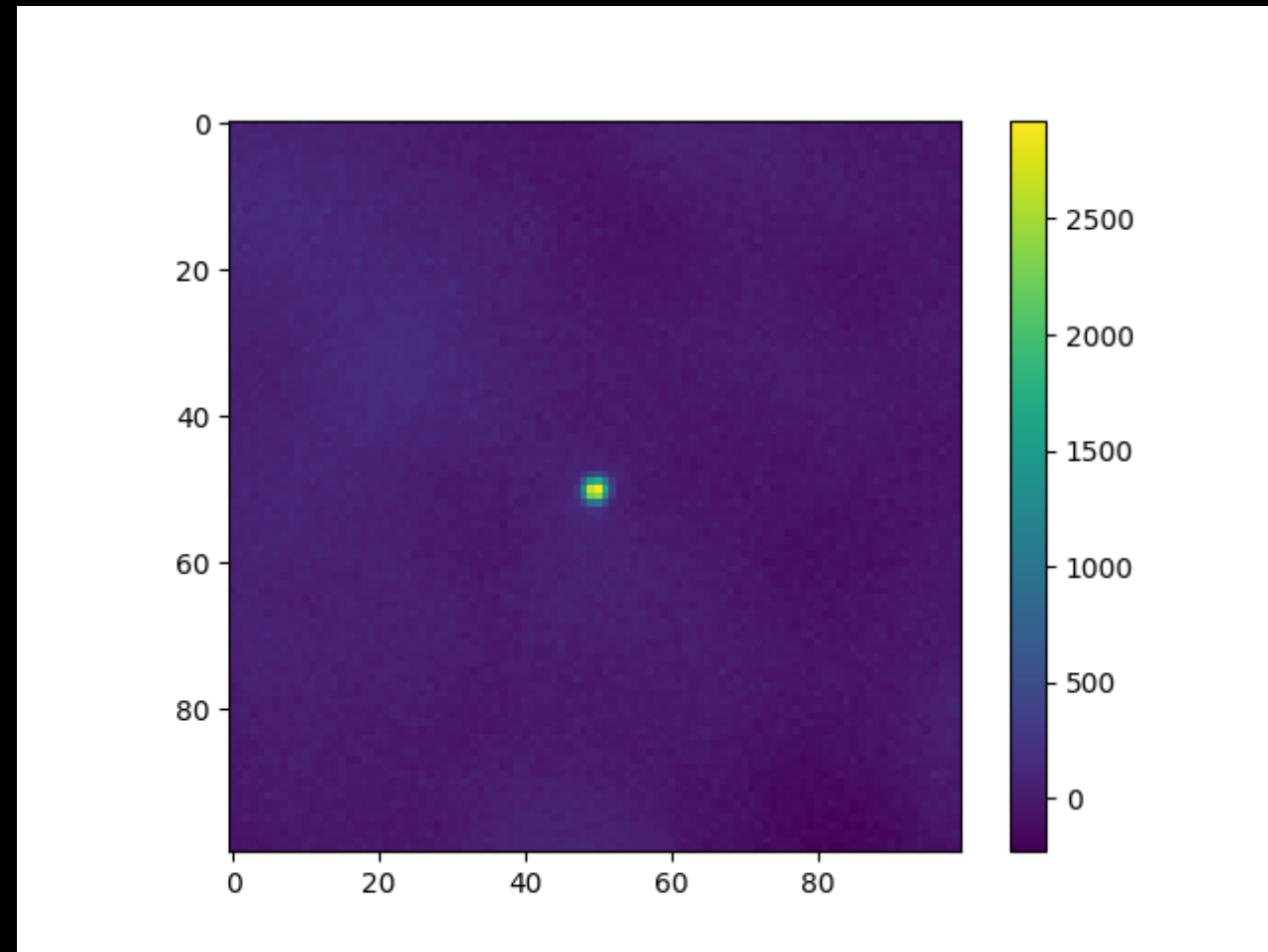
- What is the best-fit mean and error for a set of uncorrelated gaussian variables with same mean but individual errors?
- $A=?$ Show that $A^T N^{-1} A = \sum (\sigma_i^{-2})$, $A^T N^{-1} d = \sum d_i / \sigma_i^2$.
- Define weights $w_i = \sigma_i^{-2}$. Then $m = \sum w_i d_i / \sum w_i$. Variance of our estimator is $1 / \sum w_i$.

Worked Example 2

- Let's assume that N is constant and diagonal, and we have a single parameter.
- Show $LHS = \sum(m_i^2/\sigma^2)$
- $RHS = \sum(d_i m_i/\sigma^2)$
- Best-fit amplitude is $RHS/LHS = \sum(d_i m_i)/\sum m_i^2$
- $Error = 1/\sqrt{RHS} = \sigma/\sqrt{\sum m_i^2}$. If there are $\sim n$ model points with value ~ 1 , this turns into σ/\sqrt{n} , as roughly expected.

Example - Source in ACT Data

- Let's fit the amplitude of a source in ACT data.
- Look at `find_act_flux.py`.
- Let's try to guess a Gaussian, fit amplitude to it.
- Map is saved as FITS.
Ability to read/write/
manipulate FITS images
extremely useful in
astronomy!



Better: Newton's Method

- linear: $\langle d \rangle = A m$. Nonlinear: $\langle d \rangle = A(m)$ $\chi^2 = (d - A(m))^T N^{-1} (d - A(m))$
- If we're "close" to minimum, can linearize. $A(m) = A(m_0) + \partial A / \partial m * \delta m$
- Now have $\chi^2 = (d - A(m_0) - \partial A / \partial m \delta m)^T N^{-1} (d - A(m_0) - \partial A / \partial m \delta m)$
- What is the gradient?

Newton's Method ctd

- Gradient trickier - $\partial A/\partial m$ depends in general on m , so there's a second derivative
- Two terms: $\nabla \chi^2 = (-\partial A/\partial m)^T N^{-1} (d - A(m_0) - \partial A/\partial m \delta m) - (\partial^2 A/\partial m_i \partial m_j \delta m)^T N^{-1} (d - A(m_0) - \partial A/\partial m \delta m)$
- If we are near solution $d \approx A(m_0)$ and δm is small, so first term has one small quantity, second has two. Second term in general will be smaller, so usual thing is to drop it.
- Call $\partial A/\partial m$ A_m . Call $d - A(m_0)$ r . Then $\nabla \chi^2 \approx -A_m^T N^{-1} (r - A_m \delta m)$
- We know how to solve this! $A_m^T N^{-1} A_m \delta m = A_m^T N^{-1} r$

How to Implement

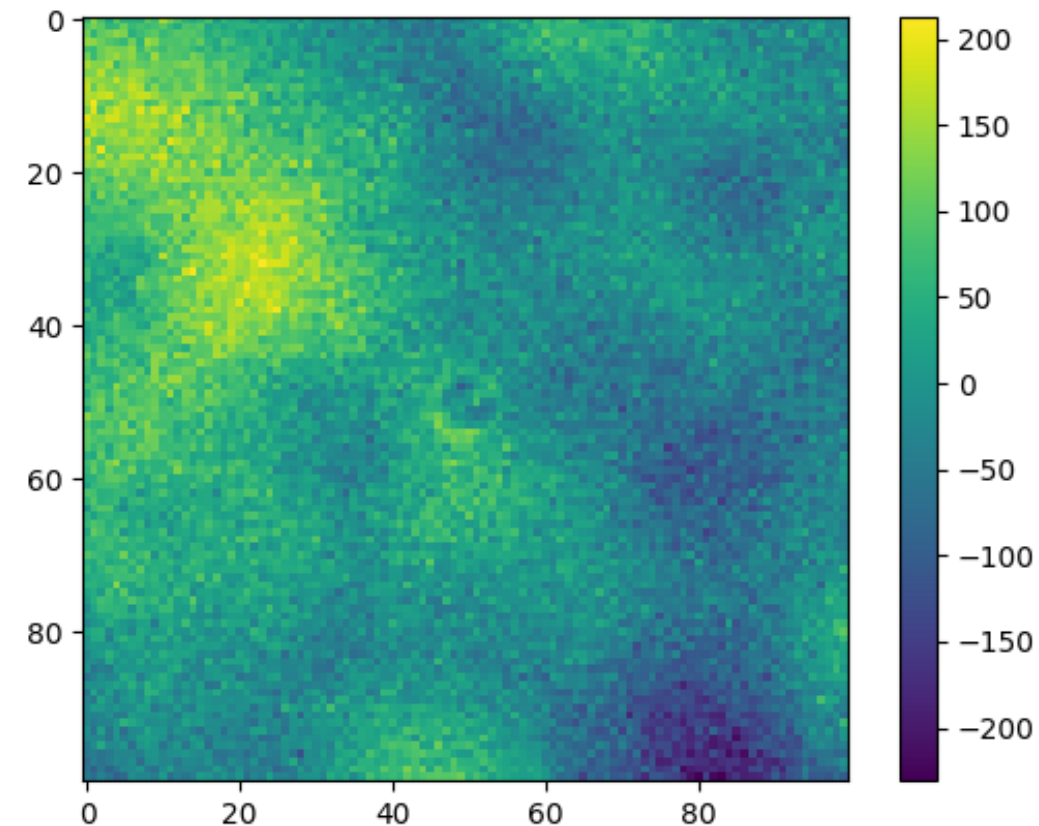
- Start with a guess for the parameters: m_0 .
- Calculate model $A(m_0)$ and local gradient A_m . Gradient can be done analytically, but also often numerically.
- Solve linear system $A_m^T N^{-1} A_m \delta m = A_m^T N^{-1} r$
- Set $m_0 \rightarrow m_0 + \delta m$.
- Repeat until δm is “small”. For χ^2 , change should be $\ll 1$ (why?).

Levenbert-Marquardt

- Newton's method can sometimes (often) fail, if the step is too large.
- In such cases, the curvature estimate is not a good approximation over the region we care about.
- We want a shorter step that is more downhill. One way to do that is to introduce a parameter λ , and increase the diagonal of the LHS by a factor of $(1+\lambda)$.
- If $\lambda=0$, we have Newton's method. If λ is large, LHS inverse is small, so step is short. It is also guaranteed to be in a downhill direction.
- Levenberg-Marquardt: take a step. If step fails, increase λ and try new, shorter step. If step succeeds, accept the step and decrease λ .
- Lots of ways to do this, NR method is silly. If $\lambda \ll 1$, just make it 0. If you ever fail, set λ to be order unity right away.

ACT Map Example

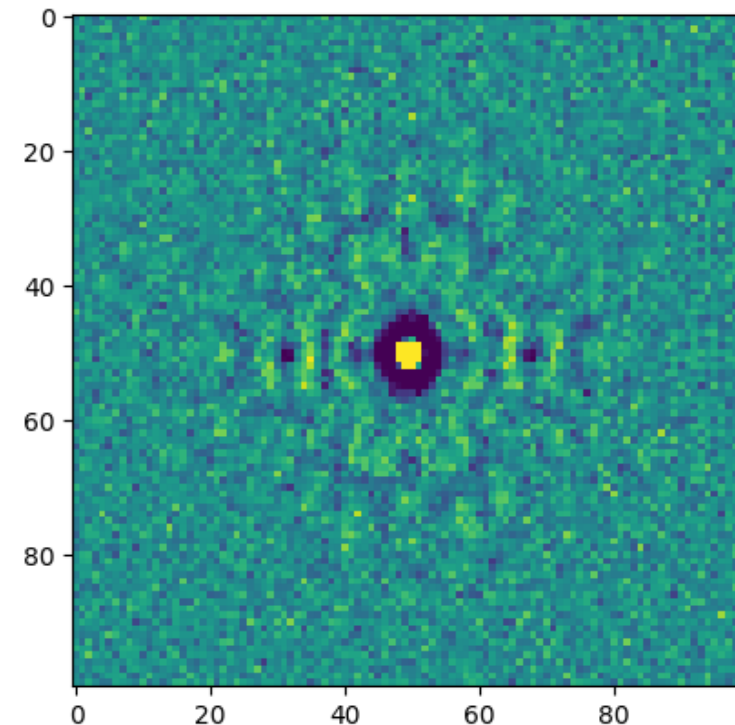
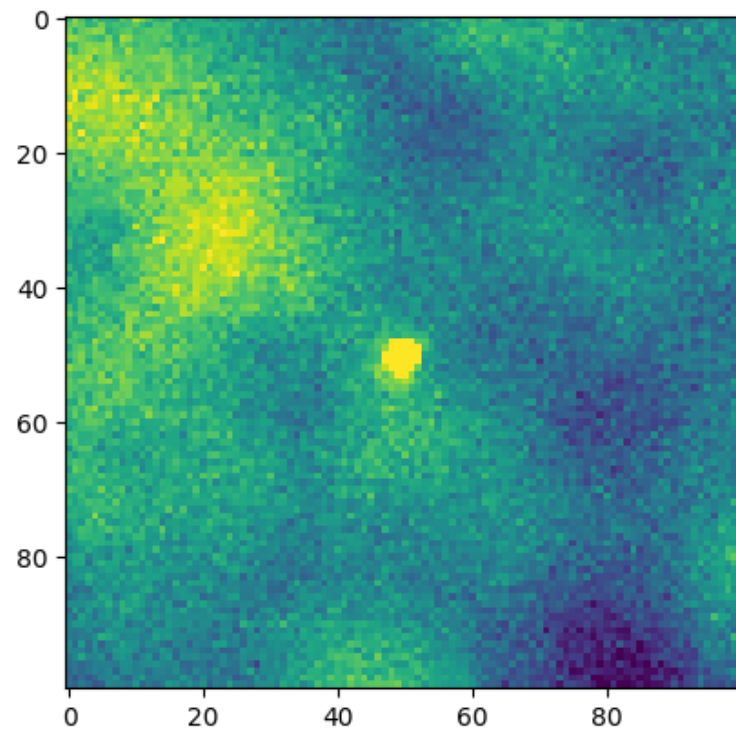
- Look at `fit_act_flux_newton.py`
- This implements numerical derivatives w/ Newton's method to fit a Gaussian (including sigma, dx, dy) to the ACT data.
- How should we estimate the noise, and hence the parameter uncertainties?
- Think about how we would do this accounting for the correlated noise?



ACT w/ Correlated Noise

- Now we have everything we need to get a real error bar!
- We will assume noise is stationary inside patch
- Noise model will be $|FT|^2$, with some smoothing.
- LHS: $A^T N^{-1} A$. For $N^{-1} A$, we can FT A , then divide by our (smoothed) FT of the patch, and FT back.
- RHS: $A^T N^{-1} d$. We could FT d , but we already have $N^{-1} A$, so $(N^{-1} A)^T d$ is just the dot product of something we already have with the data. Sanity check is that $A^T (N^{-1} d)$ gives same answer.
- NB - there are some normalization factors to do with FT, plus I make 4 reflected copies to avoid edge effects. (More discussion soon...)

Output: left original, right N-1d



```
best-fit improvement is 9258.461141888942
best-fit amplitude is 3131.6195091919985
rhs two ways is 45247.759297511904 45247.759297511904
amp/errs are 0.9799892049119797 0.004653847062334384 0.00994763245208032 0.4678
3464153434196
in temperature: 3068.953312899911 14.574078253202229
```

Output: The full noise model has 2x smaller errorbar than pretending noise is white. The fitting has figured out that noise is on large scales, but we have signal on small scales and taken advantage of that.

Matched Filters

- Let's say we know what a signal looks like, but we don't know *where* to look for it.
- In 1D, this would be $A(x-t)^T N^{-1} A(x-t) m = A(x-t)^T N^{-1} d(x)$
- If noise is stationary, I can let $x' = x - t$, then N stays the same.
 $A^T N^{-1} A m = A^T N^{-1} d(x' + t)$. LHS is just a number, RHS I get $\sum (N^{-1} A) d(x' + t)$.
- This is a cross-correlation, and can be done efficiently in Fourier space.
- Called matched filters, but never forget underneath you are doing a least-squares fit for a model against every location in the data.

Convolution Theorem

- Convolution defined to be $\text{conv}(y) = f \otimes g = \int f(x)g(y-x)dx$
- $\sum_x \sum F(k) \exp(2\pi i k x) \sum \text{conj}(G(k')) \exp(2\pi i k' x) \exp(-2\pi i k' y/N)$
- Reorder sum: $\sum \sum F(k) \text{conj}(G(k')) \exp(-2\pi i k' y/N) \sum_x \exp(2\pi i (k+k')x)$
- equals zero unless $k' = -k$. Cancels one sum, conjugates G
- $f \otimes g = \sum F(k)G(k) \exp(2\pi i k y/N) = \text{ift}(\text{dft}(f) * \text{dft}(g))$
- So, to convolve two functions, multiply their DFTs and take the IFT
- Cross correlation similar - we get $f(x+y)$ instead, and we get $\text{corr}(f,g) = \text{IDFT}(FG^*)$ (i.e. take the conjugate of one DFT).

Matched Filters Redux

- With convolution/correlation theorems in hand, we can now efficiently do a least-squares fit at every point in our data.
- FT the signal. FT the data. Conjugate one, multiply, and IDFT. If you want normalized amplitude, divide by sum of model².
- Adding correlated noise trivial! We've already FFT'd, so after we FT the data/model, we divide by the noise power spectrum (and similar for the LHS).
- NB - easy to get your factors of N , \sqrt{N} etc. wrong, so do be careful if you want properly normalized...
- Often, we don't know exactly what the noise is, in which case make the RHS using your best guess, but then estimate noise by looking at scatter in MF output.