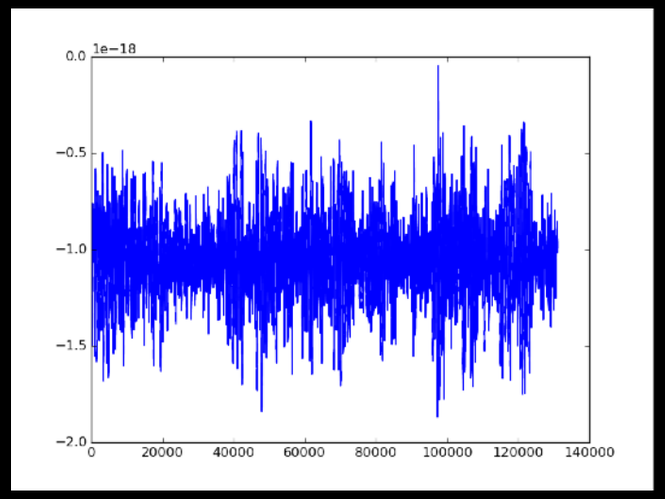
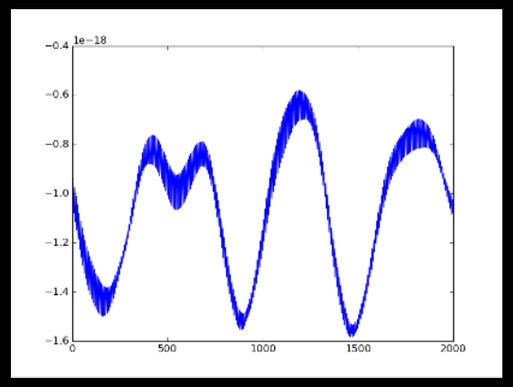
Lecture 5

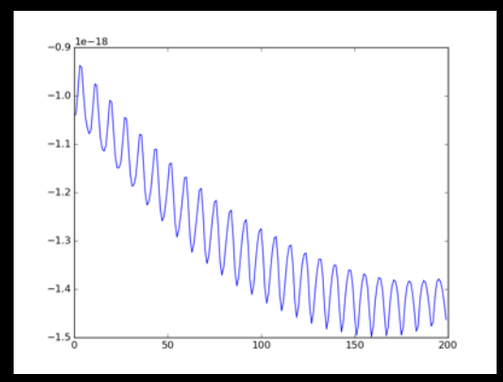
Correlated noise ctd., stationary noise,
non-linear least-squares

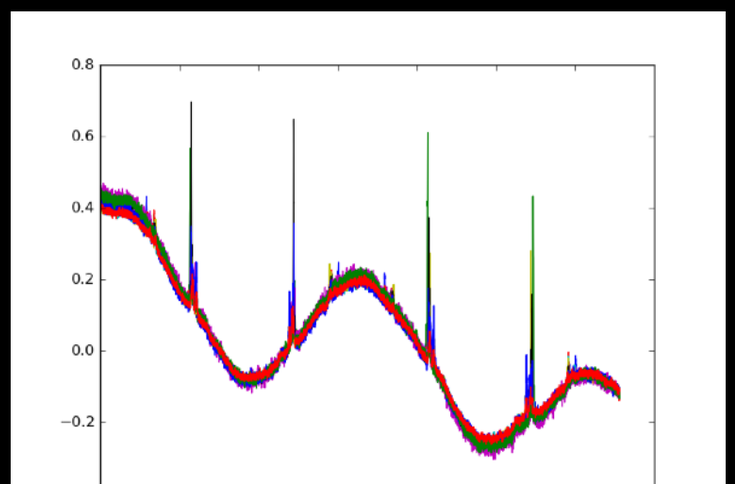
Correlated Noise

- So far, we have assumed that the noise is independent between data sets.
- Life is sometimes that kind, but very often not. We need tools to deal with this.









Right:LIGO data, with varying levels of zoom.
Left: detector timestreams from Mustang 2 camera @GBT

Fortunately…

- Linear algebra expressions for χ^2 already can handle this.
- Let V be an orthogonal matrix, so $VV^T=V^TV=I$, and d-Am=r (for residual)
- $\chi 2=r^TN^{-1}r = r^TV^TVN^{-1}V^TVr$. Let r->Vr, N->$VNV^T$, and $\chi 2$ expression is unchanged in new, rotated space.
- Furthermore, (fairly) easy to show that $\langle N_{ij}\rangle=\langle r_i r_j\rangle$.
- So, we can work in this new, rotated space without ever referring to original coordinates. Just need to calculate noise covariances N_{ij}.

Generating Correlated Noise

- Say we have a noise matrix N and want to create realizations from it. How do we do this?

- Same trick in reverse. If $d_{new}=Vd_{old}$, then I can generate d_{old} and rotate to get d_{new}.

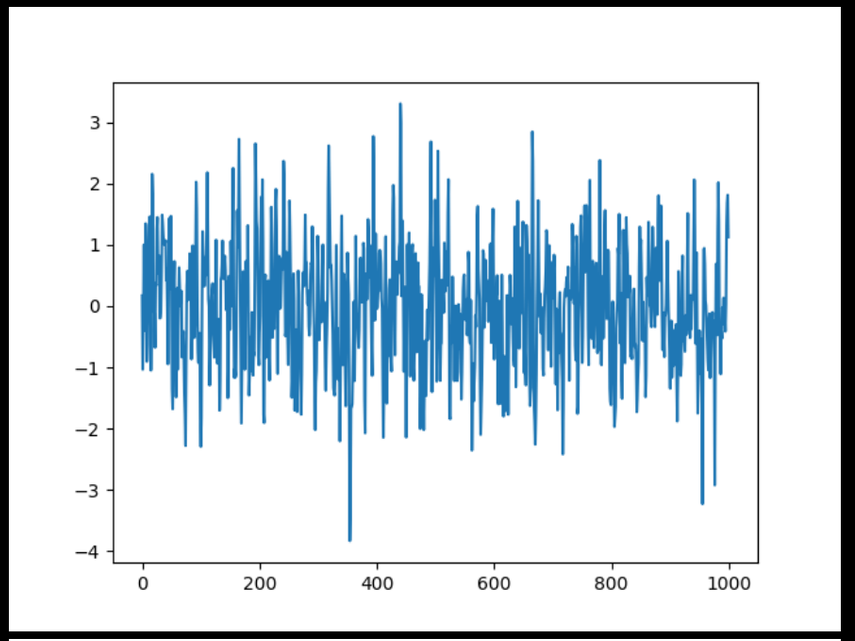
- We can pick any matrix that diagonalizes N, since we know how to generate uncorrelated data.

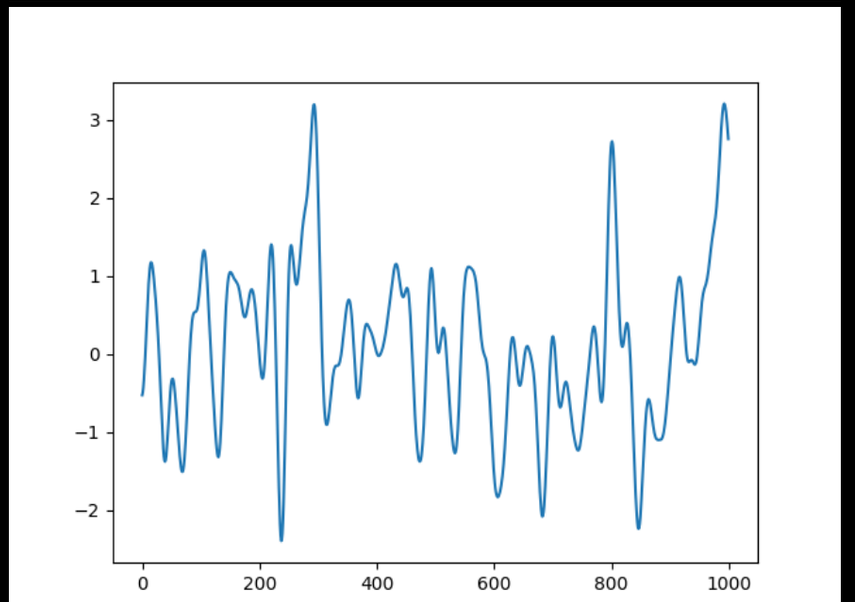
- A particularly useful one is Cholesky (LU equivalent for positive-definite): $N=LL^T$. We can generate simulated data just by taking L*g*, where *g* is a vector of zero-mean, unit-variance Gaussian random deviates.

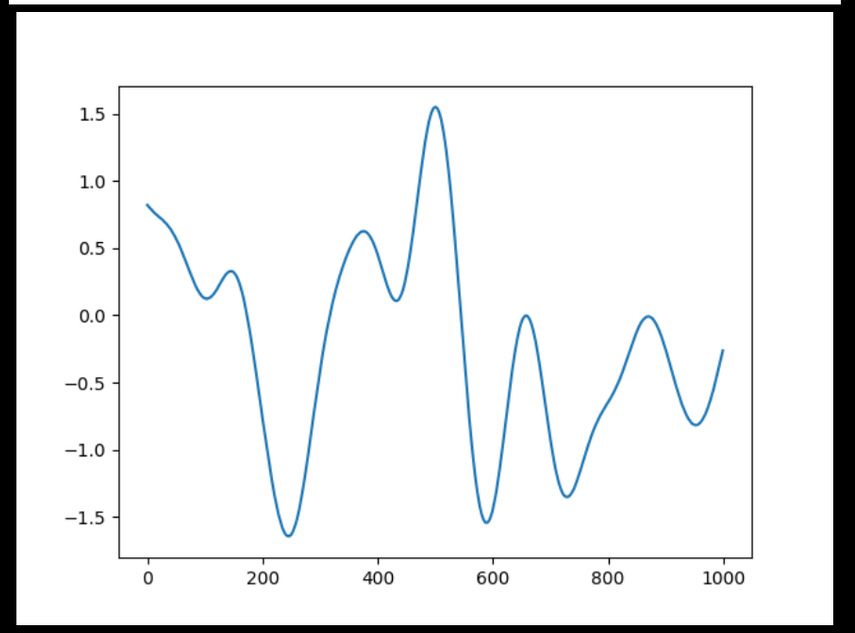
- Often a good idea to generate many simulations, average their outer products, and check empirical matrix agrees with expected.

Gaussian Correlated Noise

- Look at gauss_corrnoise.py
- $\langle f(x)f(x+dx)\rangle = \exp(-x^2/2\sigma^2)$
- top: σ=1, middle σ=10, bottom σ=50







60 Hz Example

- In class we saw 60 Hz noise with a pure cos term. Can we do generic 60 Hz noise $\cos(120\pi t+\phi)$?

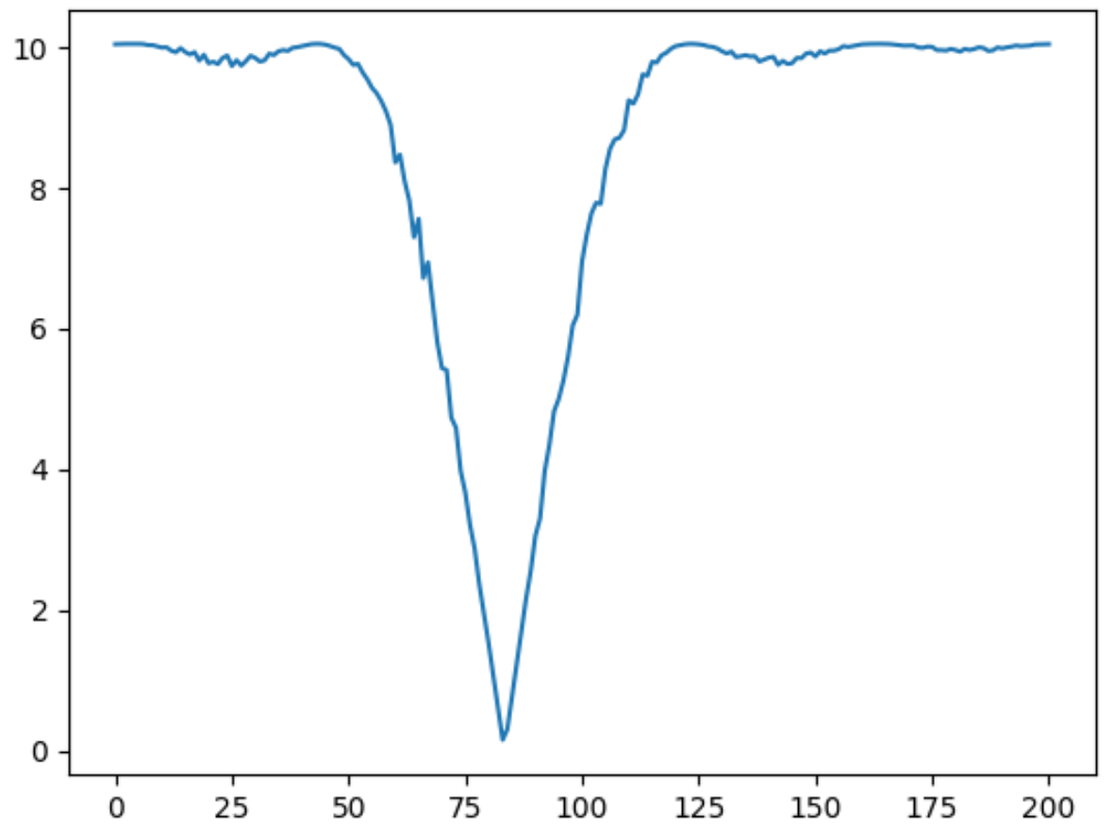
- It might look nonlinear, but we can turn into $\cos(\phi)\cos(120\pi t)+\sin(\phi)\sin(120\pi t)$, which is linear.

- How about errors? Let's say we fit sine waves, how should errors behave as function of frequency?

```
Ninv=np.linalg.inv(N)
nuvec=np.linspace(35,95,201)
myerrs=0.0*nuvec #this will be the correct error bar
myerrs2=0*myerrs #this will be the error bar ignoring our correlated noise
Ninv2=np.linalg.inv(np.diag(np.diag(N)))
for i in range(len(nuvec)):
    nu=nuvec[i]
    phi=2*np.pi*np.random.rand(1)
    myvec=np.cos(2*np.pi*nu*t+phi)
    lhs=myvec.T@Ninv@myvec
    myerrs[i]=np.sqrt(1/lhs)
    lhs2=myvec.T@Ninv2@myvec
    myerrs2[i]=np.sqrt(1/lhs2)

plt.clf()
plt.plot(myerrs2/myerrs) #plot the ratio of the bad to correct errors
plt.show()
plt.savefig('error_ratio.png')
```

Right: ratio of error bars assuming noise is white vs. accounting for 60 Hz. You really want to know if you have correlated noise!



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): $F(k)=\sum f(x)\exp(-2\pi ikx/N)$. I suggest you commit this to memory.
- IDFT: $f(x)=1/N \sum F(k)\exp(2\pi ikx/N)$
- Note: x,k integers, go from 0 to N-1.
- Also note: $\sum\exp(-2\pi ikx/N)=0$ 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

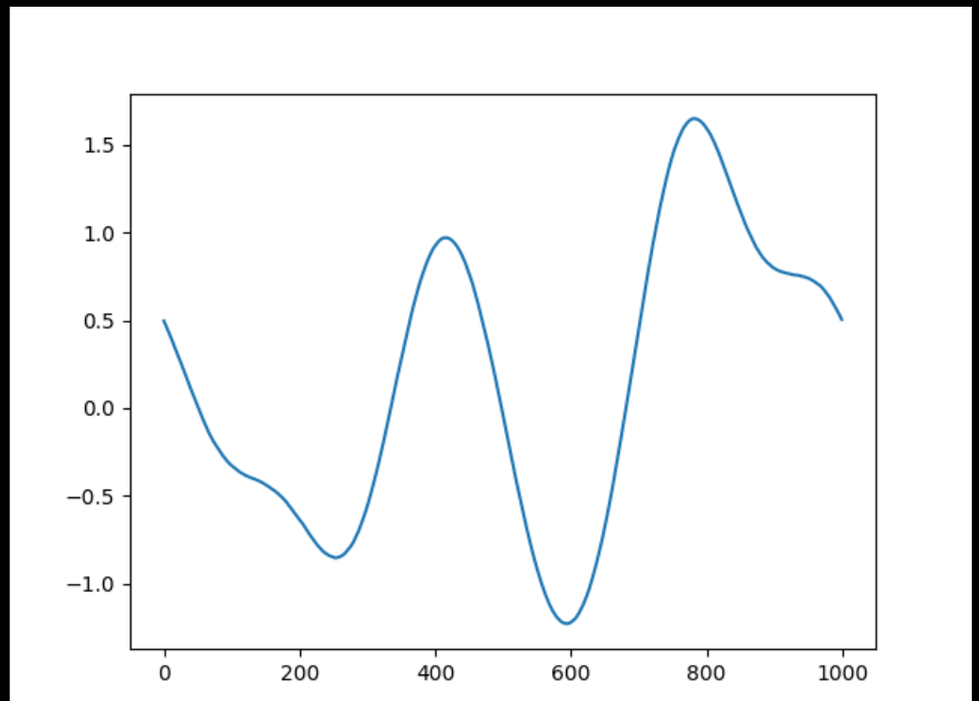
- Say we have noise where $\langle f(x)f(x+dx)\rangle = g(dx)$ (not of x)
- Fourier space: $\langle F(k)F^*(k')\rangle = \langle \sum f(x)\exp(-2\pi ikx/N) \sum f(x')\exp(2\pi ik'x'/N)$
- Can swap x' for x+dx, since sum is over all points, can sum over dx:
- $\langle F(k)F^*(k')\rangle = \langle \sum f(x)\exp(-2\pi ikx/N) \sum f(x+dx)\exp(2\pi ik'(x+dx)/N)\rangle$
- Reorder sum over x then dx: $\langle \sum \exp(2\pi ik'dx/N) \sum f(x)f(x+dx)\exp(2\pi ix(k'-k)/N\rangle$
- Now $\langle f(x)f(x+dx)\rangle = g(dx)$ (by assumption), can come out. $\sum \exp(2\pi ik'dx/N) g(dx) \sum \exp(2\pi ix(k'-k)/N)$
- Interior goes to N for k'=k, left with $N \sum g(dx)\exp(2\pi ikdx/N)\delta_{kk'}$, 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)
myps=np.fft.rfft(mycorr)

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



- 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 $\langle F(k)^2 \rangle$ = DFT(g(dx)) where $g(dx)=\langle f(x)f(x+dx) \rangle$

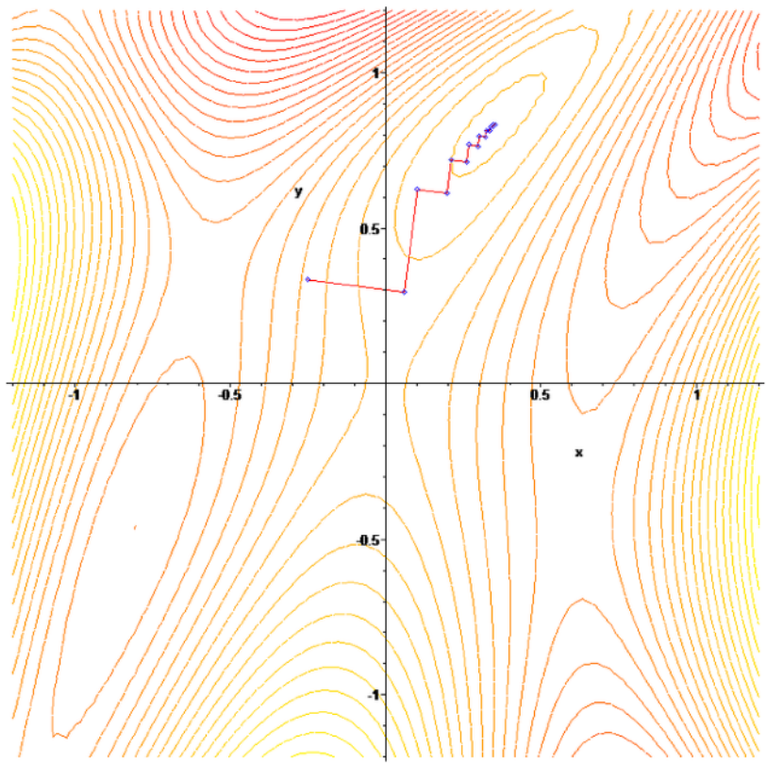
Nonlinear Fitting

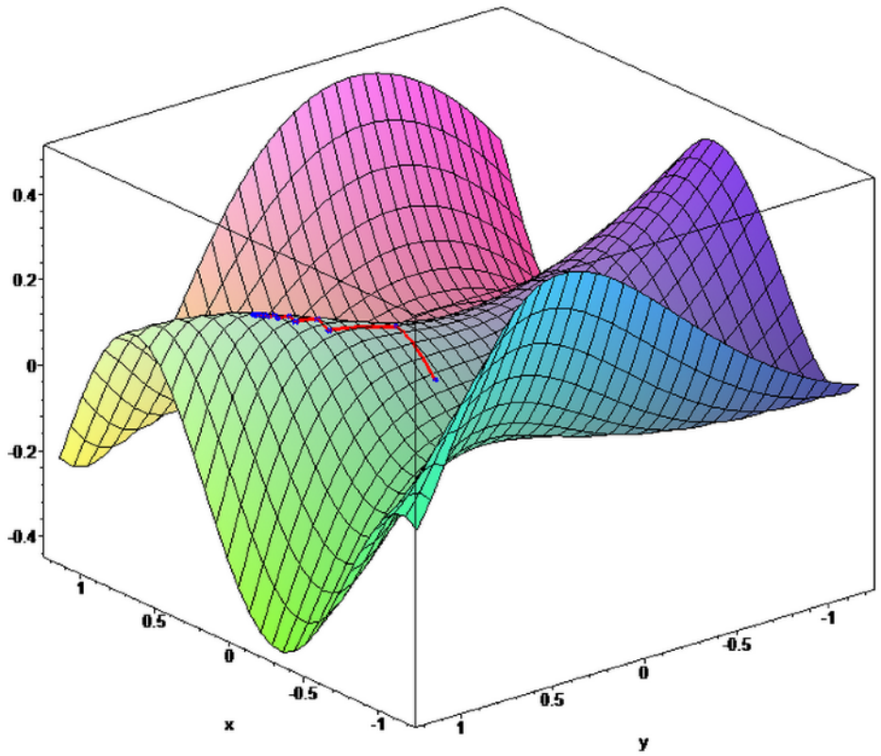
- Sometimes data depend non-linearly on model parameters
- Examples are Gaussian and Lorentzian $(a/(b+(x-c)^2)$
- Often significantly more complicated - cannot reason about global behaviour from local properties. May be multiple local minima
- Many methods reduce to how to efficiently find the "nearest" 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 $F(x,y) = \sin\left(\frac{1}{2}x^2 - \frac{1}{4}y^2 + 3\right)\cos(2x + 1 - e^y)$.

e gradient descent algorithm in action. (1: contour)





From wikipedia. Zigagging is inefficient.

Worked Example

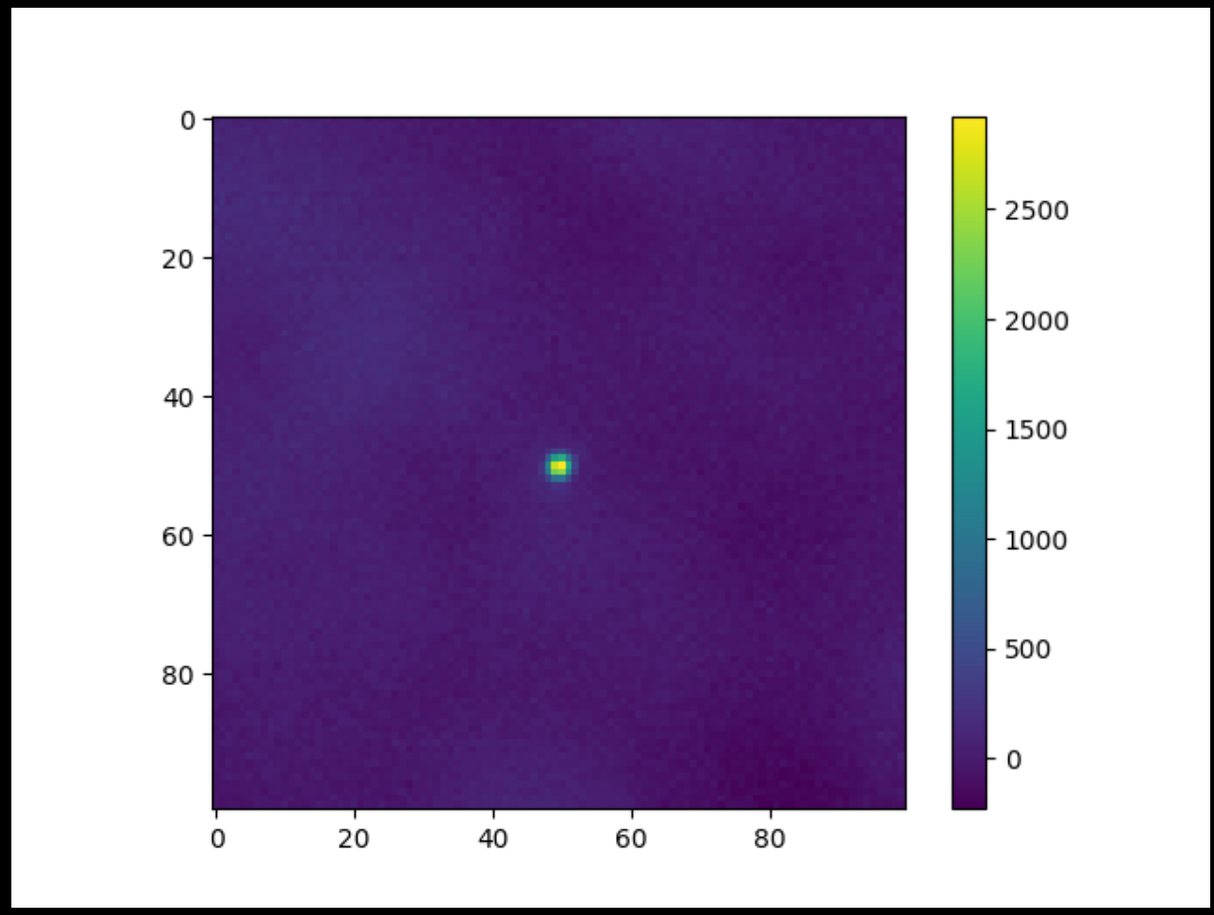
- What is the best-fit mean and error for a set of uncorrelated gaussian variables with same mean but individual errors?
- A=? Show that $A^T N^{-1} A = \sum(\sigma_i^{-2})$, $A^T N^{-1} d = \sum d_i/\sigma_i^2$.
- Define weights $w_i = \sigma_i^{-2}$. Then $m = \sum w_i d_i / \sum w_i$. Variance of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

Worked Example 2

- Let's assume that N is constant and diagonal, and we have a single paramers.
- 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) = σ/sqrt($\sum m_i^2$). If there are ~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: <d>=Am. Nonlinear: <d>=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.

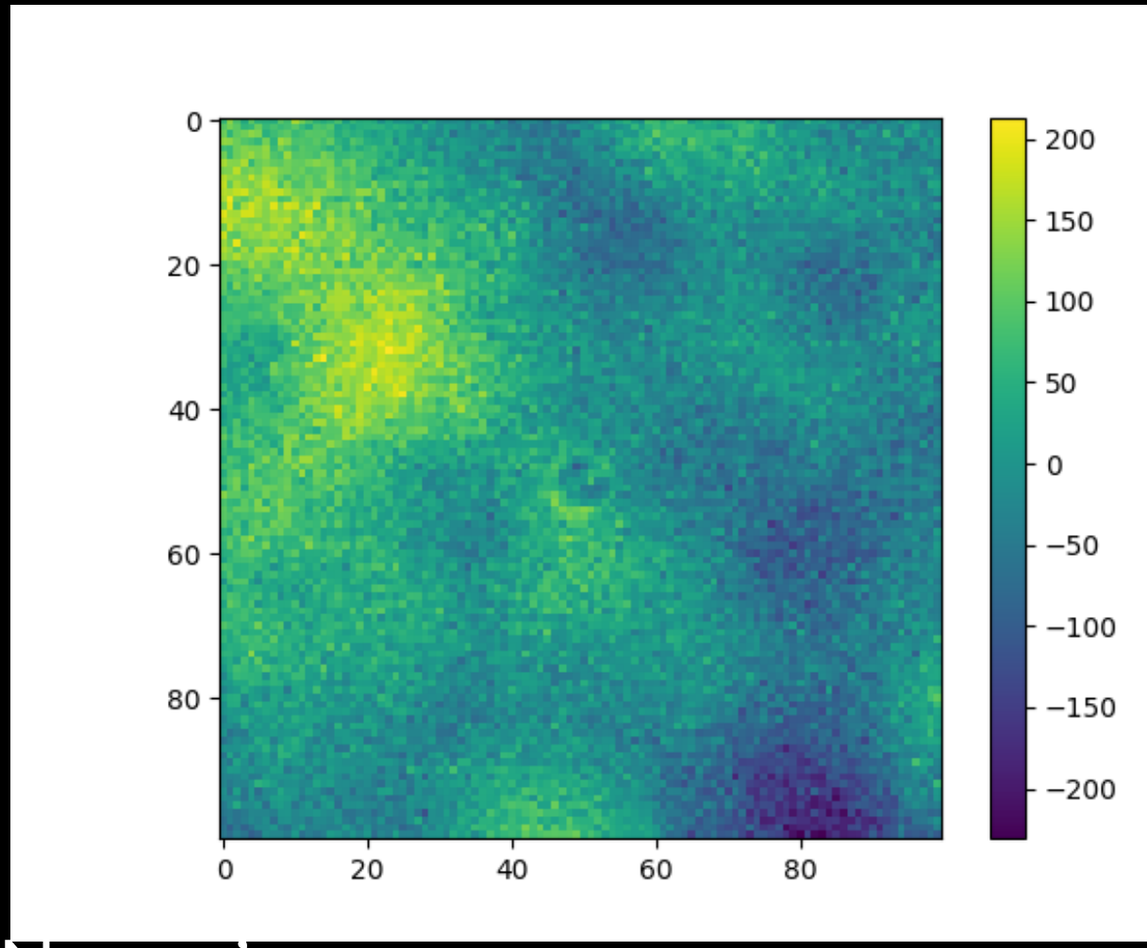
- Call $\partial A/\partial m$ A_m. Call $d-A(m_0)$ r. Then $\nabla\chi^2\approx -A_m^T N^{-1}(r-A_m\delta m)$

- 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 << 1 (why?).

ACT Map Example

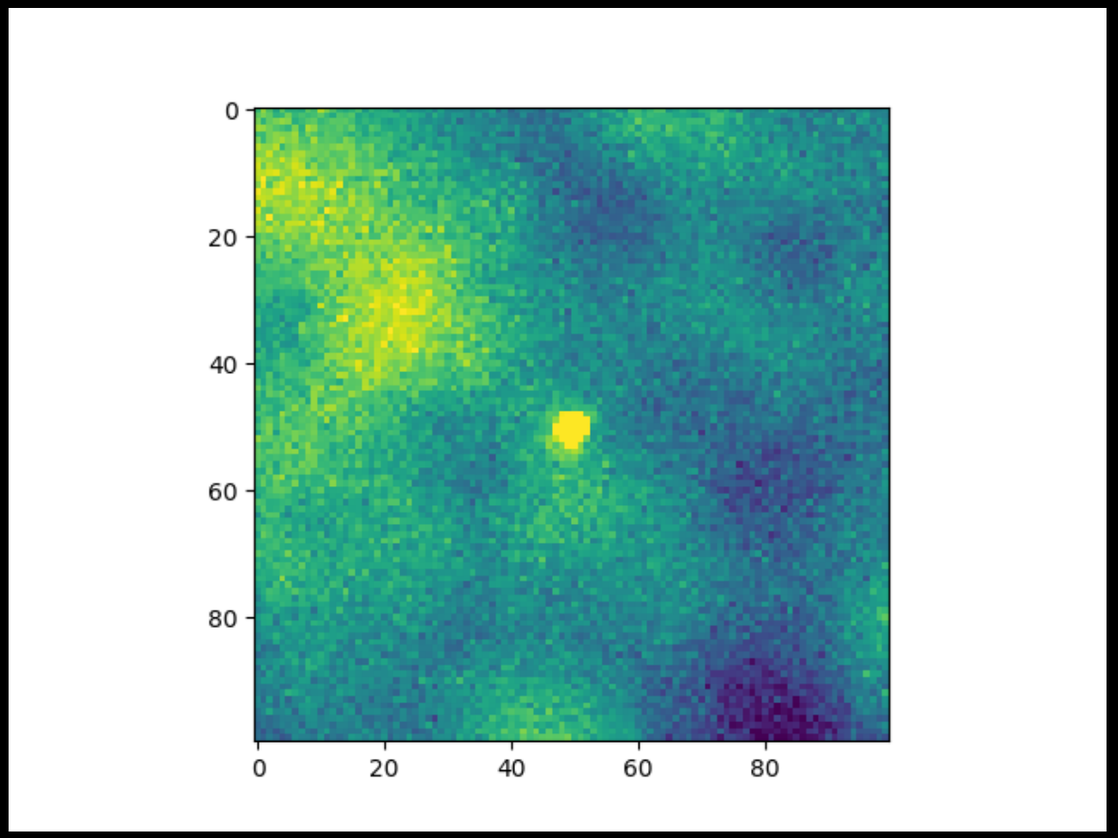


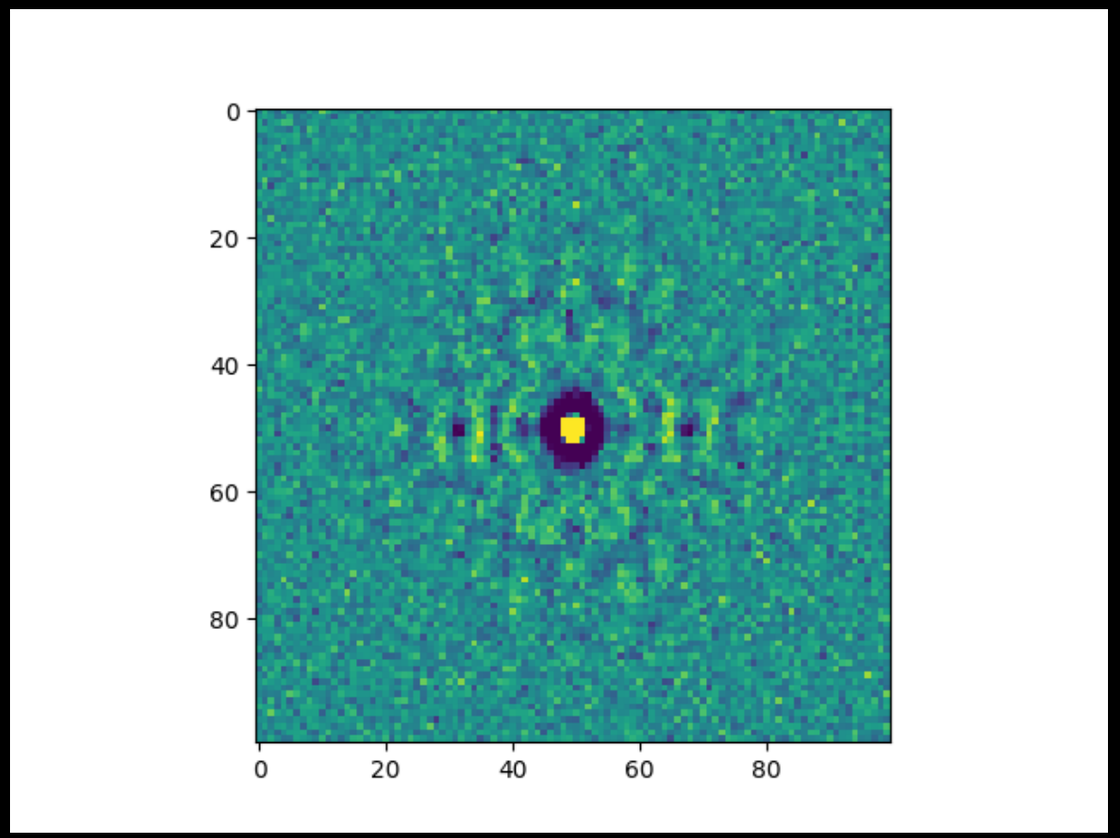
- Look at fit_act_flux_newton.py
- This implements numerical derivatives w/ Newton's method to fit a Gaussian (including sigma, dx, dy) to the ACT data.
- How should we estimate the noise, and hence the parameter uncertainties?
- Think about how we would do this accounting for the correlated noise?

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 $N^{-1}d$






```
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 0.009947632452080320 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.