Scientific Computing: Lecture 19

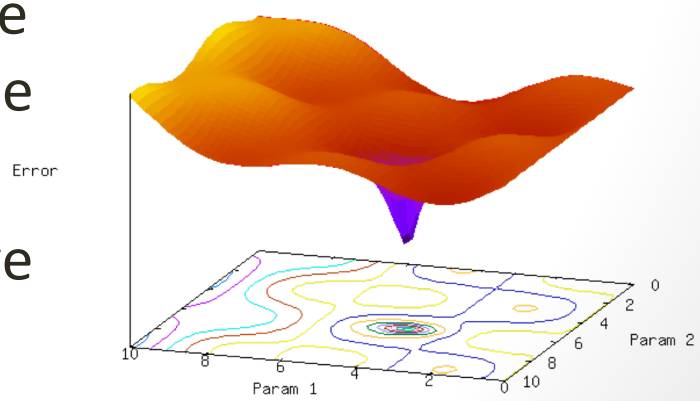
- Non-linear Regression
  - General ideas and warnings: Levenberg-Marquardt method
  - Python tools and examples.
- 'Quality of Fit' for non-linear regression
  - General thoughts
  - Generating estimates from curve_fit output
- Some Real World Examples

CLASS NOTES

- Levenburg-Marquardt material posted soon.
- Decision on Optional Components
- HW07 due tonight
- Be thinking about proposals for the Final Project!

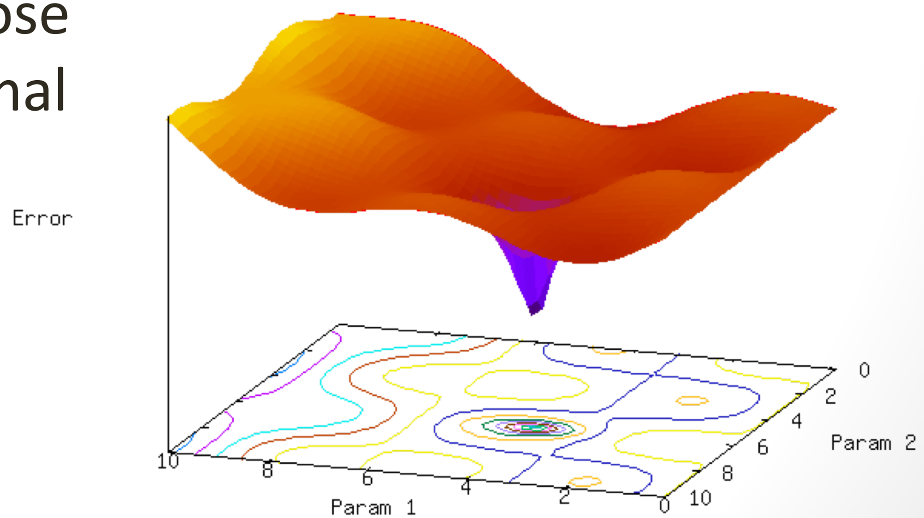
General Ideas

- Most widely used algorithm used for non-linear fits is called the Levenberg-Marquardt method. It is iterative.
- Idea is compute <u>gradient</u> of error surface at starting point in parameter space (initial guess of parameter values). Then take a "step" (by adjusting parameters) "down hill" in direction of steepest descent.
- Compute gradient at new location and repeat.
- Once gradients are small, assume you are near minimum and shape is parabolic (like linear case).
- Compute minimum directly as we did with linear case.



Warnings about non-linear fits

- Your error surface is no longer monotonic about the global minimum – meaning there are LOCAL minima.
- Before settling on a final fit, try different starting points. You MAY end up in a lower minimum!
- Do everything you can to start the process with parameter values as close as possible to the optimal values.



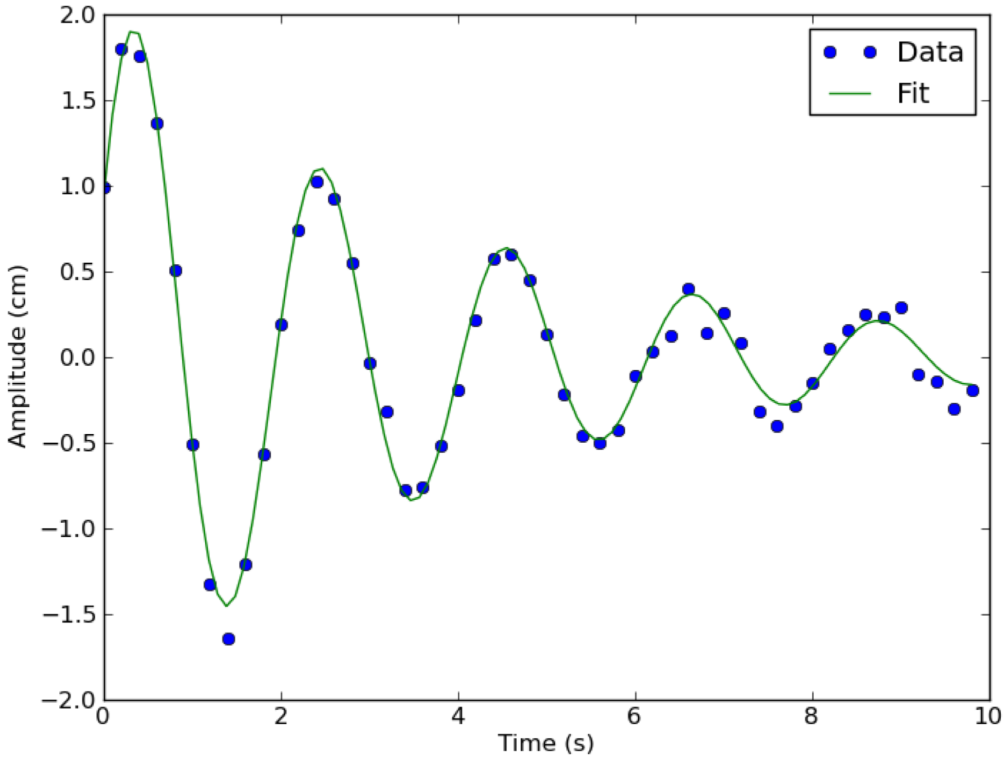
- Pay attention to the parameter values – do they make physical sense??
- Proceed with caution!

Python tools for non-linear fitting

- There are several approaches, some easier to use and some are more robust.
- curve_fit in scipy.optimize
  - use: fit=curve_fit (funct,xdata,ydata,p0=params0)
  - comments: fairly convenient and generally robust – almost always will converge.
  - returns: tuple of fitted parameters, variance-covariance (VC) matrix – more on this later.
- curve_fit() is a ‘wrapper’ function for scipy.optimize.functions.leastsq. Using leastsq() directly will provide MUCH more detail about the statistics of the fit.
- However, we can compute a ROUGH estimate of the quality of the fit from the curve_fit() output…

Example: Damped sine

- Model is a damped sine function: $y = Ae^{-\tau t}\sin(\omega t + \phi)$
- Four free parameters – need at least 12 data points to fit.



'Quality of Fit' metrics

- Linear fit has a correlation coefficient (r^2)
- Can compute a similar quantity with nonlinear fits as a ratio of the sum of squares of residuals (SSR) to total sum of squares (SST) :

$$SST = \sum_i [y_i - \bar{y}]^2$$

$$SSR = R^2 = \sum_{i=1}^{N} [y_i - F(x_i; \mathrm{params})]^2$$

- Then, r^2 can be computed by:

$$\boxed{r^2 = \sqrt{1 - \frac{SSR}{SST}}}$$

Confidence in parameters

- So how accurate are the fitted parameters?
- That is a complicated question. In an ideal world, you would run lots of fits, adjusting the data within error bars, and compute a standard deviation of the variance in the resulting parameters for each fit.
- A simpler (and less accurate) method is to multiply the diagonal elements of the variance-covariance (VC) matrix by the square root of the reduced sum of squares (or reduced chi-square).
- VC matrix returned by curve_fit is a nxn matrix for n free parameters. Diagonals give variance of each parameter, and off diagonals give covariance between variables – 'How much does a change 'A' effect the final value of B?')

Confidence in parameters

- If I have three free parameters (a,b,c):

$$\mathrm{cov} = \begin{pmatrix} aa & ab & ac \\ ab & bb & ab \\ ac & bc & cc \end{pmatrix}$$

- For N data points and m parameters, the reduced SSR is

$$\chi^2 = \frac{SSR}{N-m}$$

- Then an approximate error of the fitted parameters are:

$$\delta a = aa\sqrt{\chi^2} \quad ; \quad \delta b = bb\sqrt{\chi^2} \quad ; \quad \delta c = cc\sqrt{\chi^2}$$

Power of a Good Model

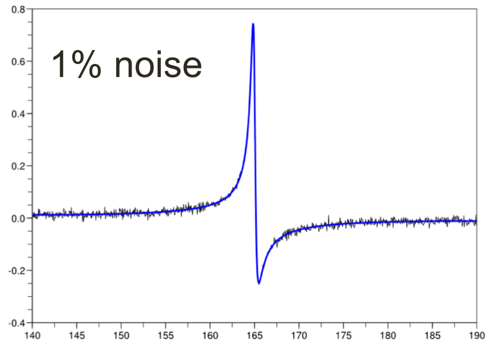
I gave a talk at a meeting several years ago in a Signal Processing and Noise session part of which showed the power of having a good model.

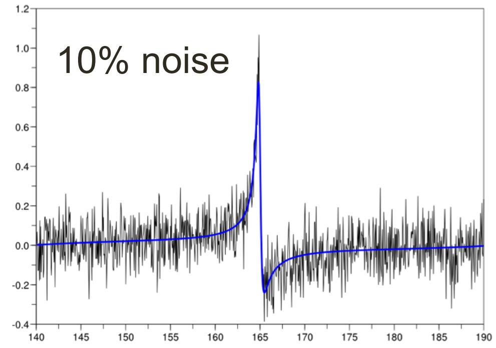
- Mechanical resonances follow a Lorentzian line shape.

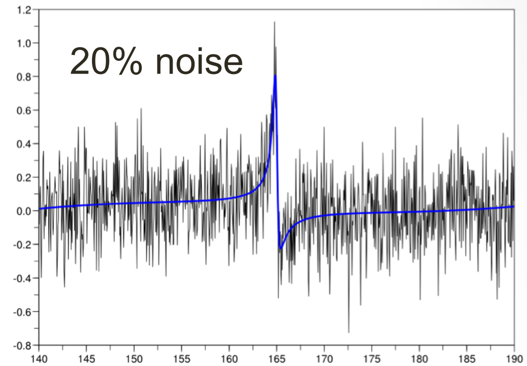
$$A(f) = \frac{\frac{f}{f_0}\cos(\phi) + (1 - \frac{f}{f_0} Q\sin(\phi))}{\left(\frac{f}{f_0}\right)^2 + \left(1 - \left(\frac{f}{f_0}\right)^2\right)^2 Q^2} + a_0 + a_1 f + a_2 f^2$$

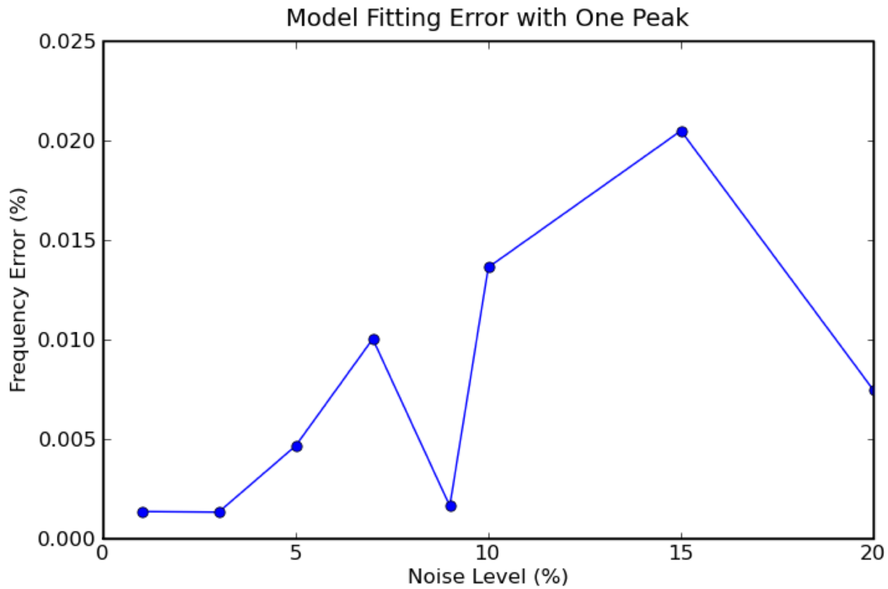
- Fit resonance data with model to determine fo and Q.
- Manually pick (click) best guess for fo as starting point (Python script).
- Test effect of fo on noise level by generating synthetic data with Gaussian noise of $x\%$ of peak amplitude.

Noise and frequency error: 1 peak



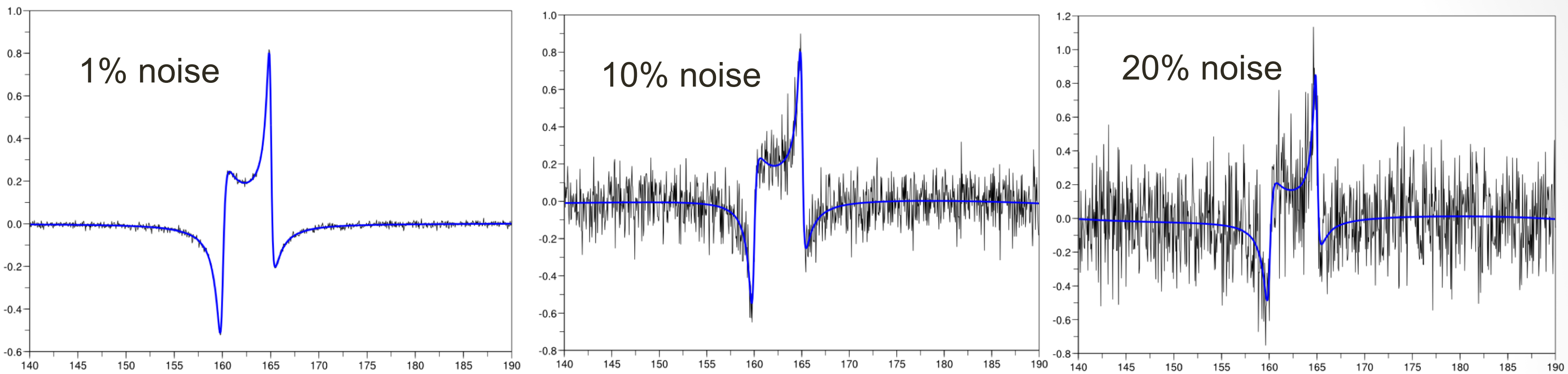


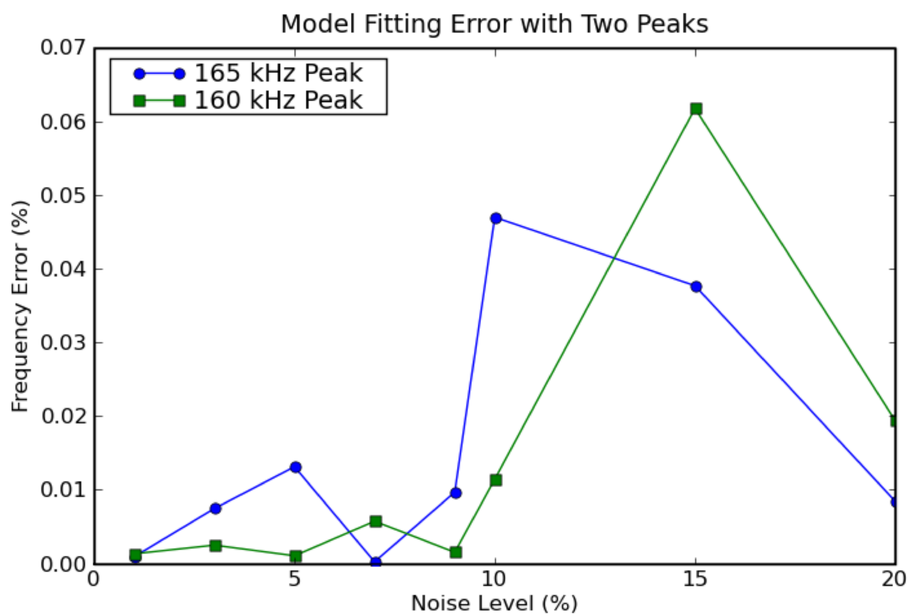




J.R. Gladden, Dept. of Physics, Univ. of Mississippi

Noise and frequency error: 2 peaks





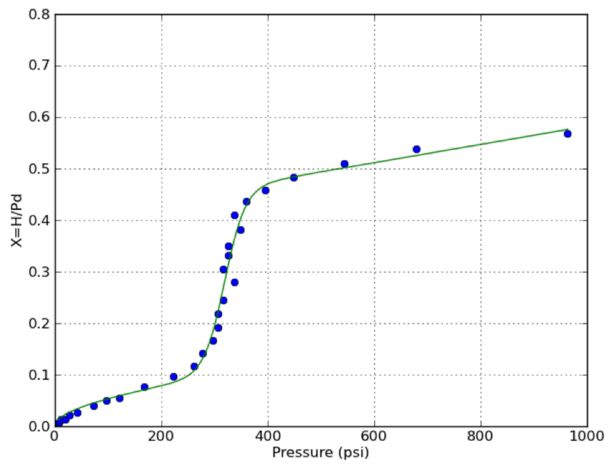
J.R. Gladden, Dept. of Physics, Univ. of Mississippi

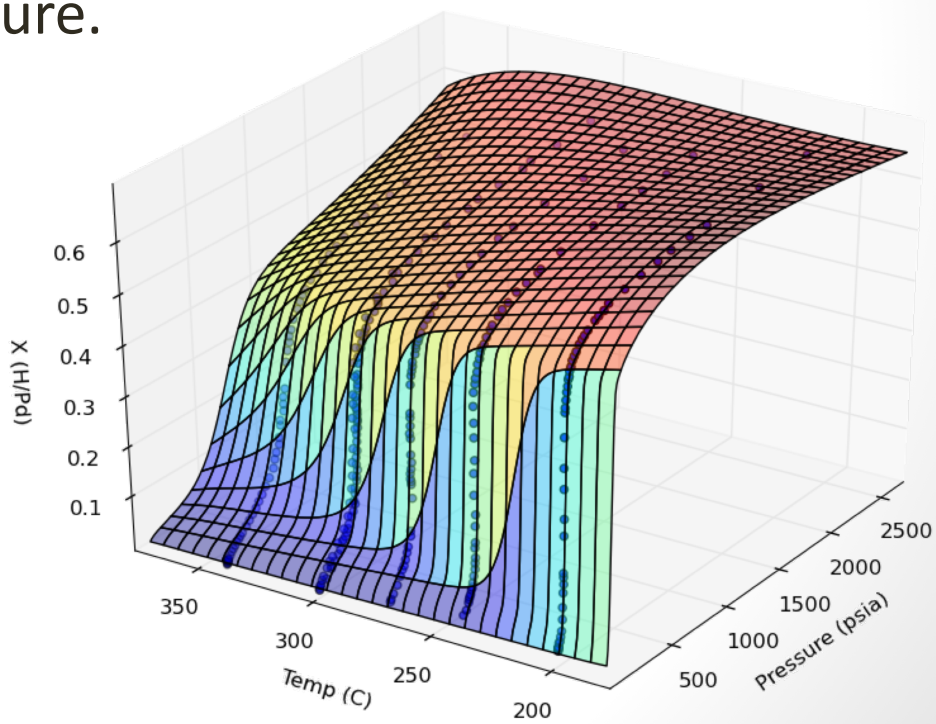
Real World Example

- We developed a automated fitting program to extract the center frequency of a mechanical resonance peak as the temperature is changed over a BROAD range with around 60 temperature points.
- Features:
  - A single peak is extracted from a resonance spectrum.
  - That peak is fit with a Lorentzian for the lowest temperature and center frequency and Q are stored.
  - Repeat until peak for all temperatures are fit.
  - Plot $f(T)$ and $1/Q(T)$ and output to a text file.

Fitting Surfaces

- It is also possible to fit data to surfaces (2 independent variables).
- Below was part of a research project fitting hydrogen content in metallic hydrides as function of temperature and pressure.
- Based on modified sigmoidals





J.R. Gladden, Dept. of Physics, Univ. of Mississippi

Exercise: Gaussian Data Fit

- Some statistical data is obtained which seems to exhibit a Gaussian distribution.
- The model is then:

$$f(x) = A \exp\left[\frac{-(x-x_0)^2}{2\sigma^2}\right]$$

  where A is the amplitude, x_0 is the expected value and σ^2 is the variance.
- Generate some synthetic data with variable random noise.
- Use curve_fit() to obtain optimal values for these parameters. Plot data and fit.
- If time, also compute the correlation coefficient for the fit and uncertainties in the parameters.