Curve Fitting and Confidence Intervals

Assoc. Prof. John Quinn

Recap

The Propagation of Errors Formula

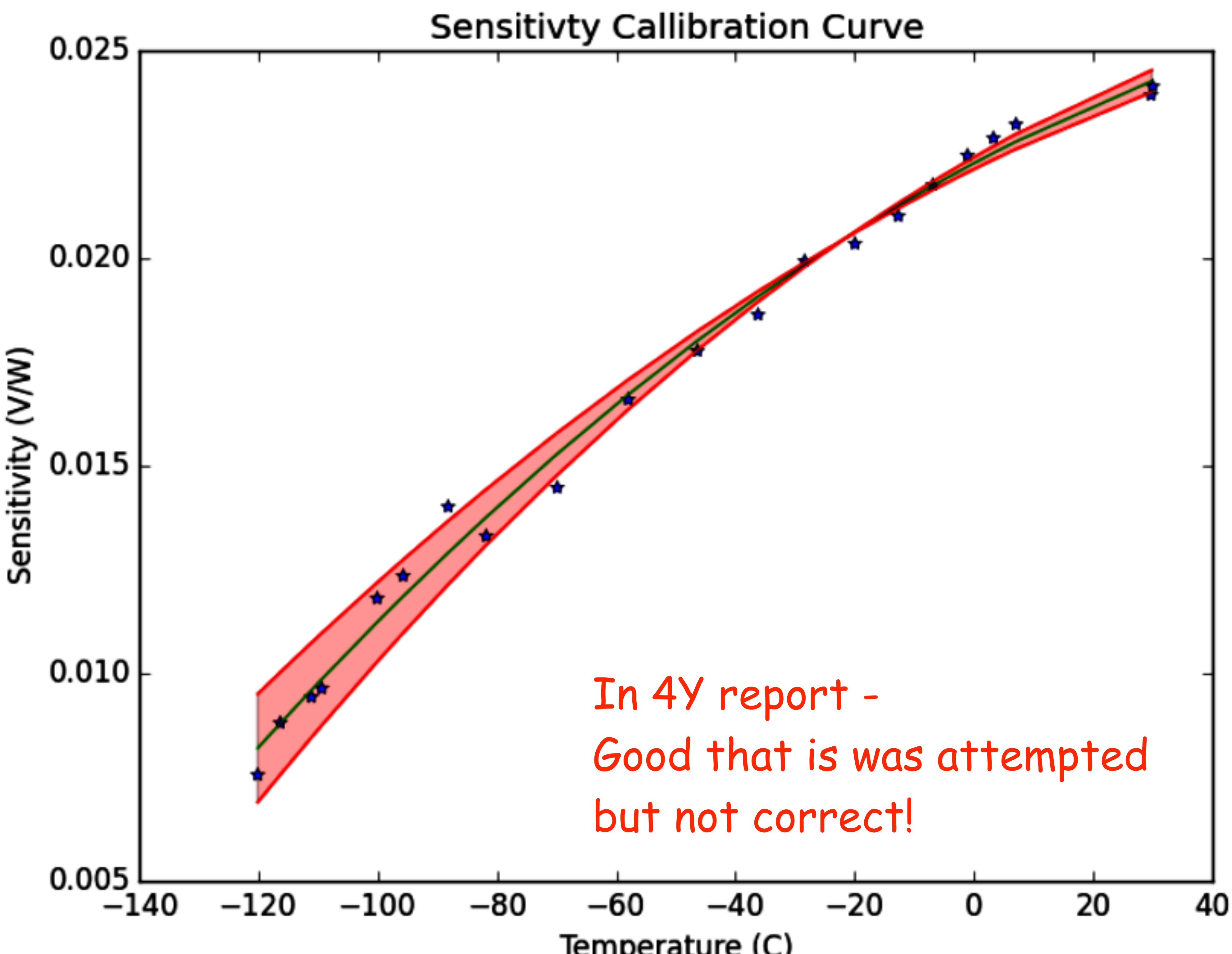
$$\sigma_u^2 = \left(\frac{\partial f}{\partial x}\right)^2 \sigma_x^2 + \left(\frac{\partial f}{\partial y}\right)^2 \sigma_y^2 + \ldots + 2\left(\frac{\partial f}{\partial x}\right)\left(\frac{\partial f}{\partial y}\right)\underset{\substack{\uparrow \\ \text{Covariance}}}{\sigma_{xy}^2} + \ldots$$

In the case of the measurements being uncorrelated:

$$\sigma_u^2 = \left(\frac{\partial f}{\partial x}\right)^2 \sigma_x^2 + \left(\frac{\partial f}{\partial y}\right)^2 \sigma_y^2 + \ldots + 2\xcancel{\left(\frac{\partial f}{\partial x}\right)\left(\frac{\partial f}{\partial y}\right)}\sigma_{xy}^2 + \ldots$$

Recap

- Motivation:

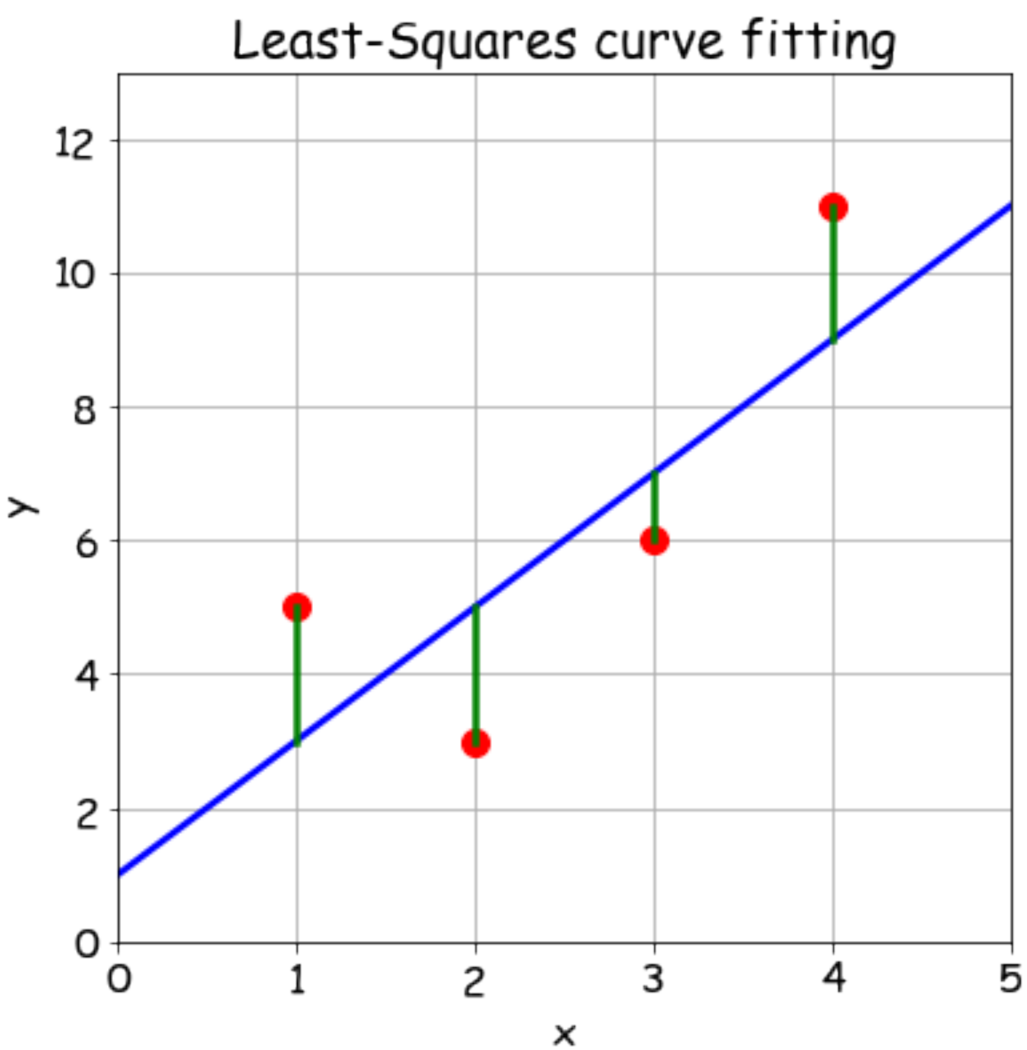


Curve Fitting/Regression

- Curve fitting is the process of finding the parameters of some function so that the function gives the "best agreement" with experimental data.
- For linear functions this can be done analytically using some criterion.
- For non-linear functions the optimisation in most cases must be done numerically.
- However, now there are numerical software packages which greatly simplify the task
- Here, we will focus exclusively on the numerical approaches and not the analytic solutions.

Method of Least Squares

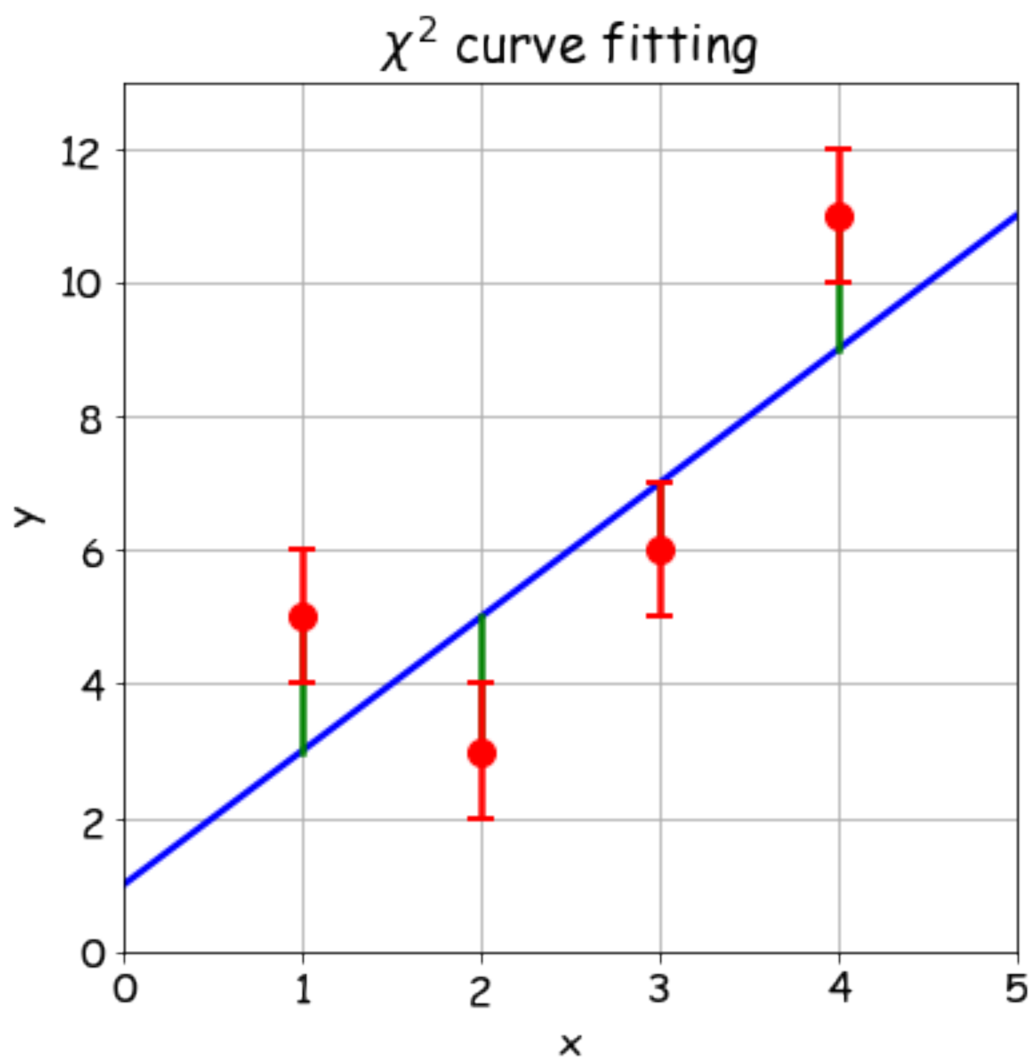
- The Method of Least Squares:
  - If we do not have (or ignore) error bars on measured data points, then we can apply the method of least squares:
    - find the parameters of the fit function which minimise the sum of the squares of the vertical differences between each point and the function value at that point.



$$S(pars) = \sum_{i=1}^{N} \left[y_i - f(x_i, pars) \right]^2$$

$\chi 2$ minimisation

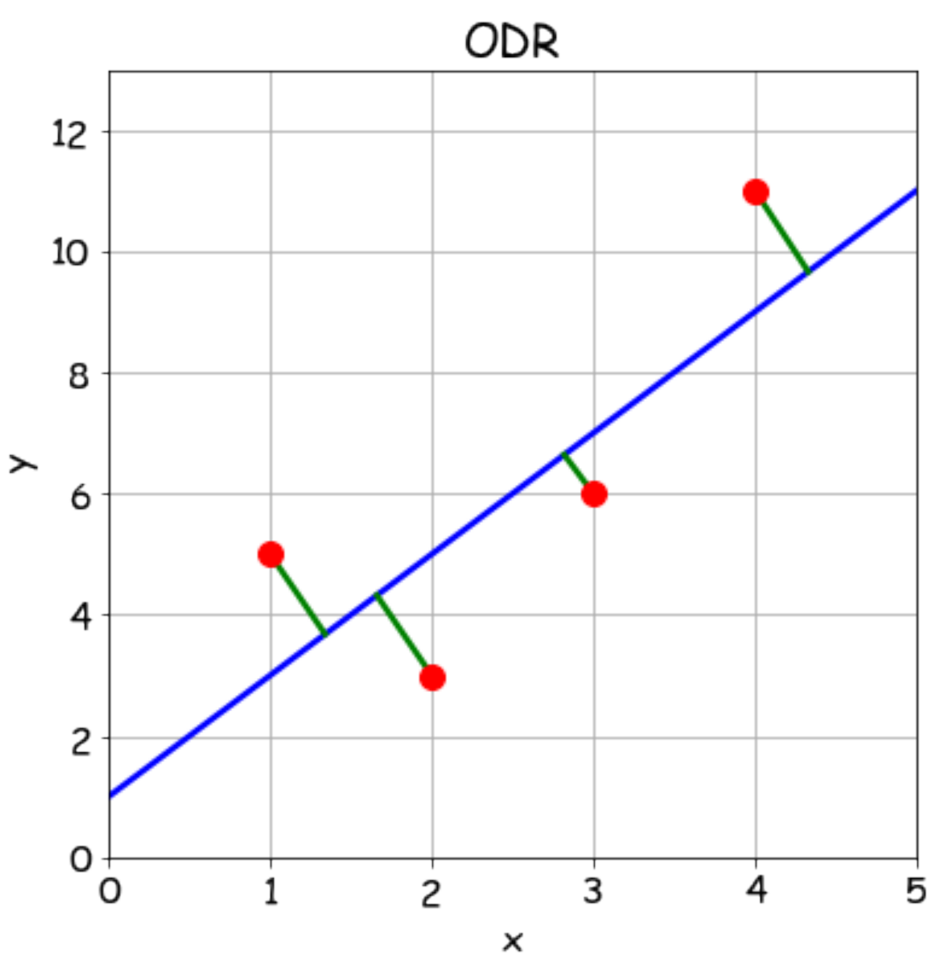
- $\chi 2$ Minimisation (note: often confusingly also called "least squares" as well since it is so similar):
  - If we have error bars on the dependent variables (y) then we can weight each point by its error.
  - We divide the 'vertical distance' each point is from the fitted curve by its error, in effect we are calculating the number of standard errors each point is from the curve.
  - The optimum parameters for the function are the ones which minimise the sum of the standard errors squared (this is also know as the $\chi 2$ sum)

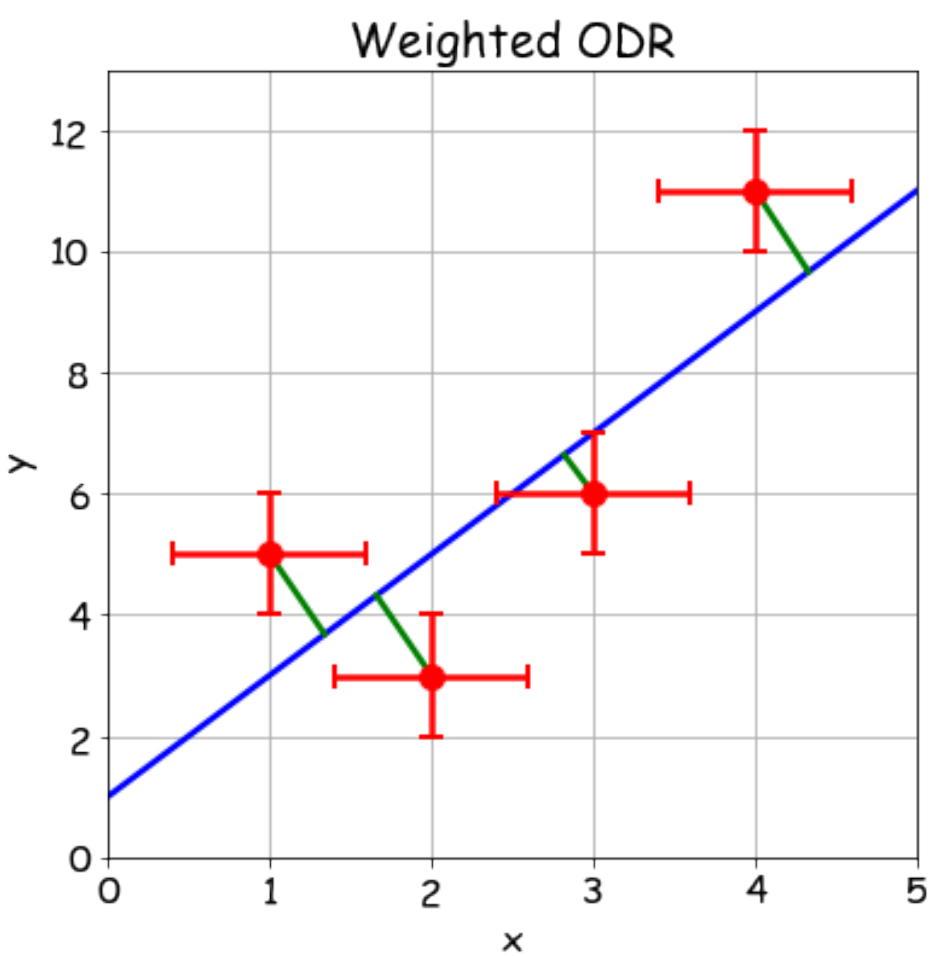


$$S(pars) = \chi^2(pars) = \sum_{i=1}^{N} \left[\frac{y_i - f(x_i, pars)}{\sigma_i} \right]^2$$

Orthogonal Distance Regression

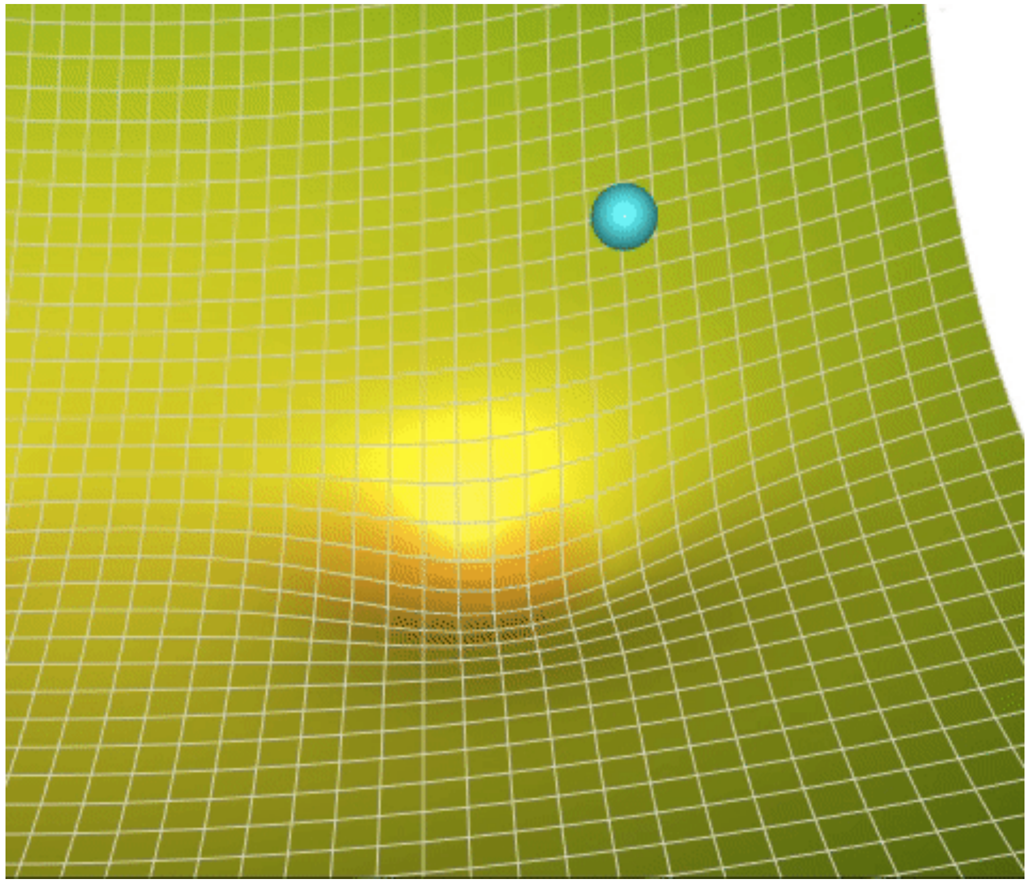
- Advanced Topic!: for information
- Orthogonal Distance Regression (ODR) minimised the orthogonal distance to the curve.
- If we have errors in both x and y then there is a method known as Weighted Orthogonal Distance Regression (weighted ODR).
- ODRPACK is a FORTRAN-77 library for performing ODR with possibly non-linear fitting functions.
- Scipy has an interface to ODRPACK: https://docs.scipy.org/doc/scipy/reference/odr.html





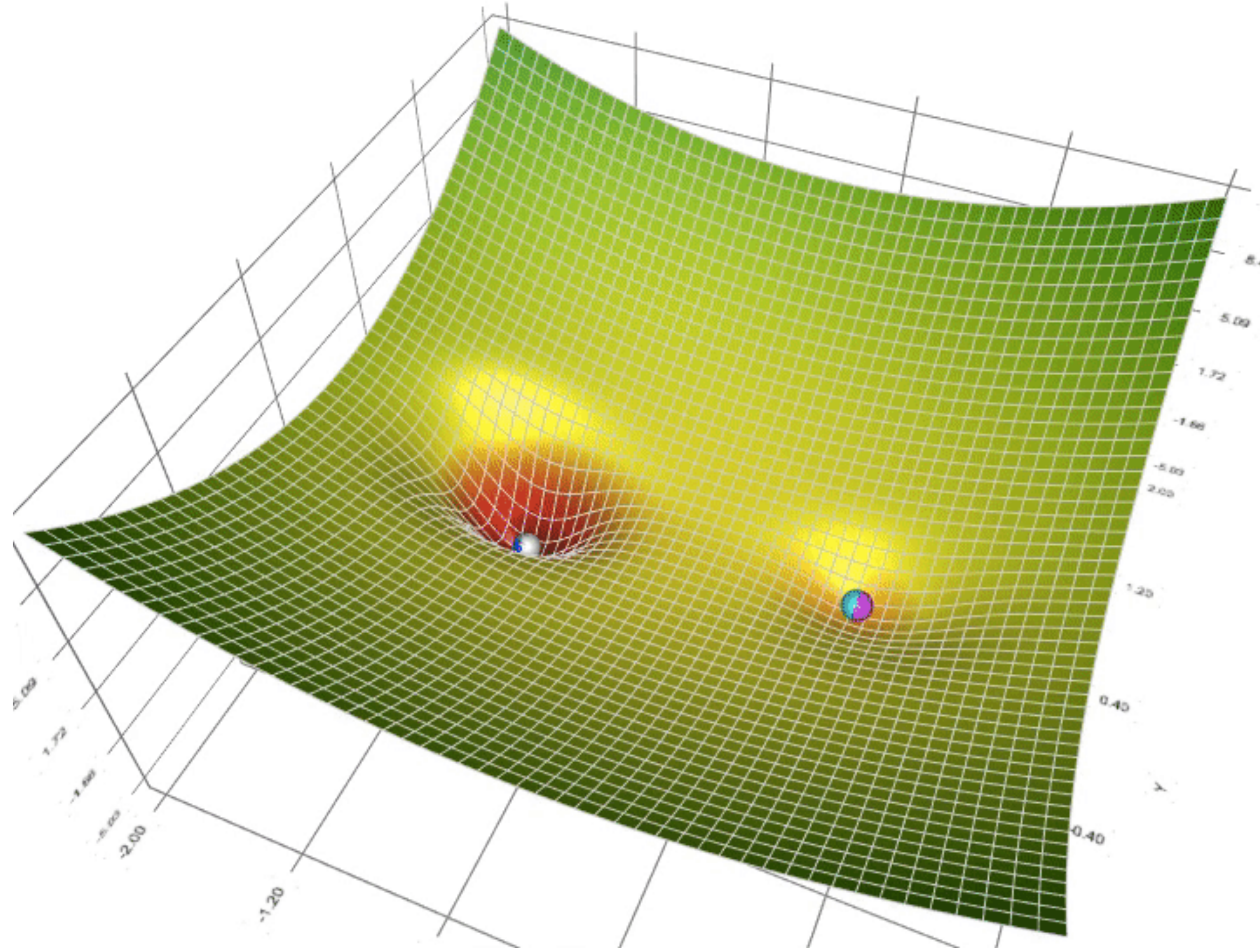
Levenberg-Marquardt Minimisation

- An algorithm must be used to find the parameters which minimises the $\chi 2$ sum (or LSQ/WODR)
- For functions which are non-linear this is not trivial.
- One of the most widely used and trusted algorithm is the Levenberg-Marquardt algorithm:
  - combination of gradient-descent and Gauss-Newton methods to search multi-dimensional space looking for minimum



https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c

Levenberg-Marquardt Minimisation



https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c

- can get stuck in a local minimum
- may not converge if initial parameters far away from minimum
- initial parameters somewhat close to the optimum parameters are desired (i.e. plot your curve with initial parameters over data first and adjust parameters before running method)

scipy.optimize.curve_fit() uses LM method

https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html

Arguments:

f,	function to be fit (callable in form: f(x,*pars)
xdata,	array of x data points
ydata,	array of y data points
p0=None,	initial guesses at function parameters (default: 1s)
sigma=None,	array of uncertainties or covariance matrices (def: all 1s)
absolute_sigma=False,	True or False: True if sigma has real errors for False if relative weights (in that case errors on best-fit pars are adjusted to give a reduced chi-square of 1)
check_finite=True,	Check for infinities and nans in input array
bounds=- inf, inf,	array of 2-tuples of lower and upper parameter bounds
method=None,	method to use (default is 'lm' for unconstrained)
jac=None,	method to calculate the Jacobian/Gradient vector
**kwargs	

Returns:

popt,	values of the parameters which minimised chisq sum
pcov	variance-covariance matrix of the fitted parameters.

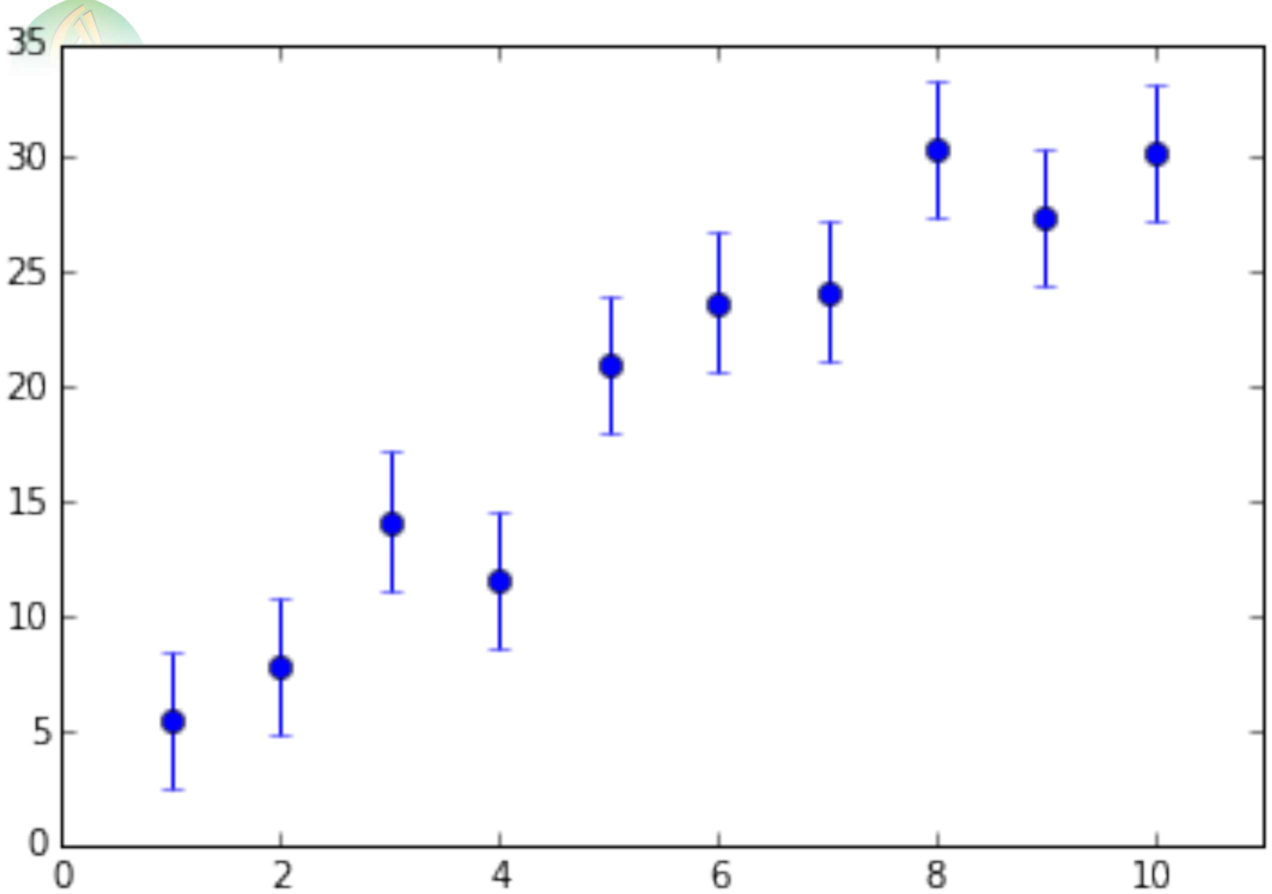
Curve Fitting

- In general we often assume that data are not correlated and neglect the covariance terms (probably good to check!)
- When we fit a theoretical model (function) to data there are uncertainties on (acceptable range of) the best-fit parameters.
- The uncertainty on any value calculated using the function with best-fit parameters is simply standard propagation of errors.
- However, the parameters of the model are probably quite correlated.
- For example, consider fitting a line of the form $y=mx+c$.
  - The function parameters m and c are highly correlated (a change in the slope m (increase) will cause a change in the intercept c (decrease)).
  - The fit function is $f(x,m,c)$
    - with uncertainties on the parameters this becomes: $f(x,\ m\pm\sigma_m,\ c\pm\sigma_c)$ so there is an uncertainty on the value $f(x)$ due to the variances and covariances of the fit parameters.

Curve Fitting

- Fitting software routines (such as SciPy's `curve_fit()`) generally return the full variance-covariance matrix:
  - The errors on the fitted function parameters are the square root of the diagonal terms,
  - while the off-diagonal elements are the covariances.
  - If you are using the returned fitted parameters in the fit function to estimate some quantity and its error, or a confidence region, then you must include the covariance terms.

Example: fitting a straight line to some data



```
def f(x,m,c):
    return m*x+c

popt,pcov=curve_fit(f,x,y,sigma=e,
                    absolute_sigma=True)
```

```
popt=[ 2.43346158 4.60848392]

pcov=[ [ 0.10909091  -0.6       ]
       [-0.6          4.19999991] ]
```

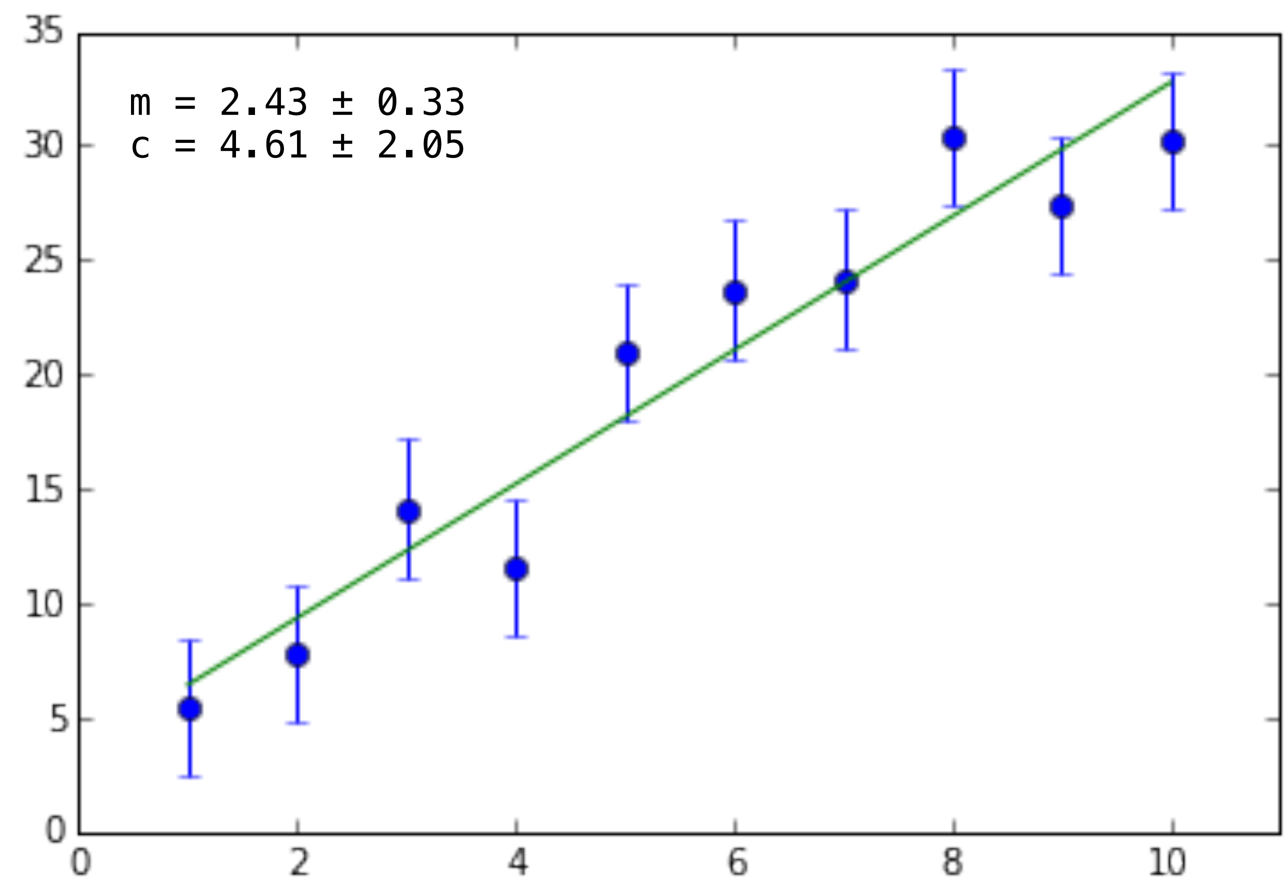
```
m=popt[0]
σm=np.sqrt(pcov[0][0])

c=popt[1]
σc=np.sqrt(pcov[1][1])

σmcsq=pcov[0][1]

print(f"{m:.2f} ± {σm:.2f}")
print(f"{c:.2f} ± {σc:.2f}")

2.43 ± 0.33
4.61 ± 2.05
```



Example: fitting a straight line to some data

```
popt=[ 2.43  4.61]
pcov=[[ 0.12 -0.60]
      [-0.60  4.20 ]]
```

$$\sigma_u^2 = \left(\frac{\partial f}{\partial x}\right)^2 \sigma_x^2 + \left(\frac{\partial f}{\partial y}\right)^2 \sigma_y^2 + 2\left(\frac{\partial f}{\partial x}\right)\left(\frac{\partial f}{\partial y}\right)\sigma_{xy}^2$$

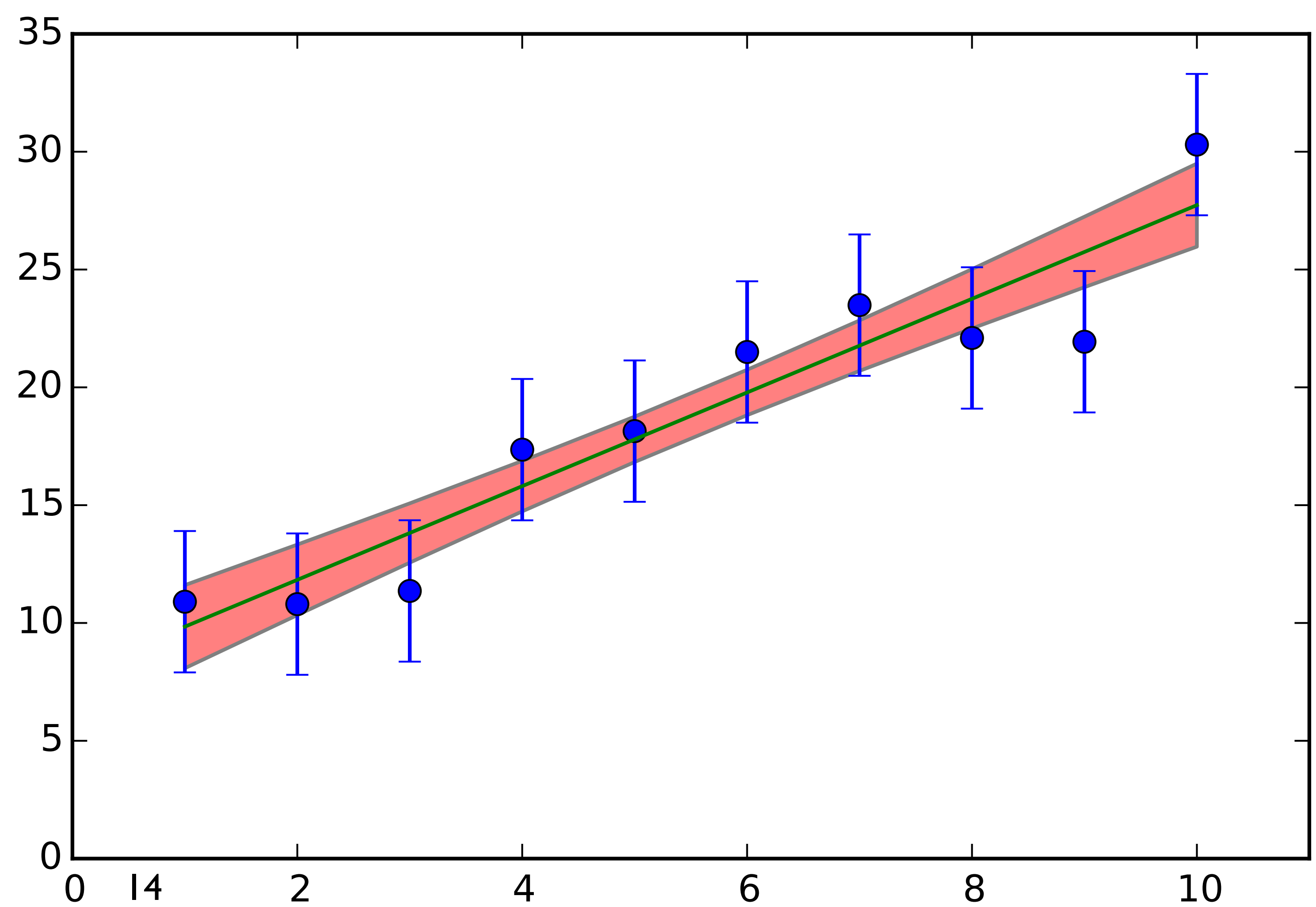
- Say for a given x, we want to use our best fit (optimum) parameters to get a corresponding value of y and its uncertainty using the function and its best-fit parameters.
- i.e., $f(x) = f(x, m_{opt} \pm \sigma_{m,opt}, c_{opt} \pm \sigma_{c,opt})$, so, for example, $f(6)=?\pm?$
- Use propagation of errors using the full covariance terms:

$$y = mx + c$$

$$\frac{\partial y}{\partial m} = x \qquad \frac{\partial y}{\partial c} = 1$$

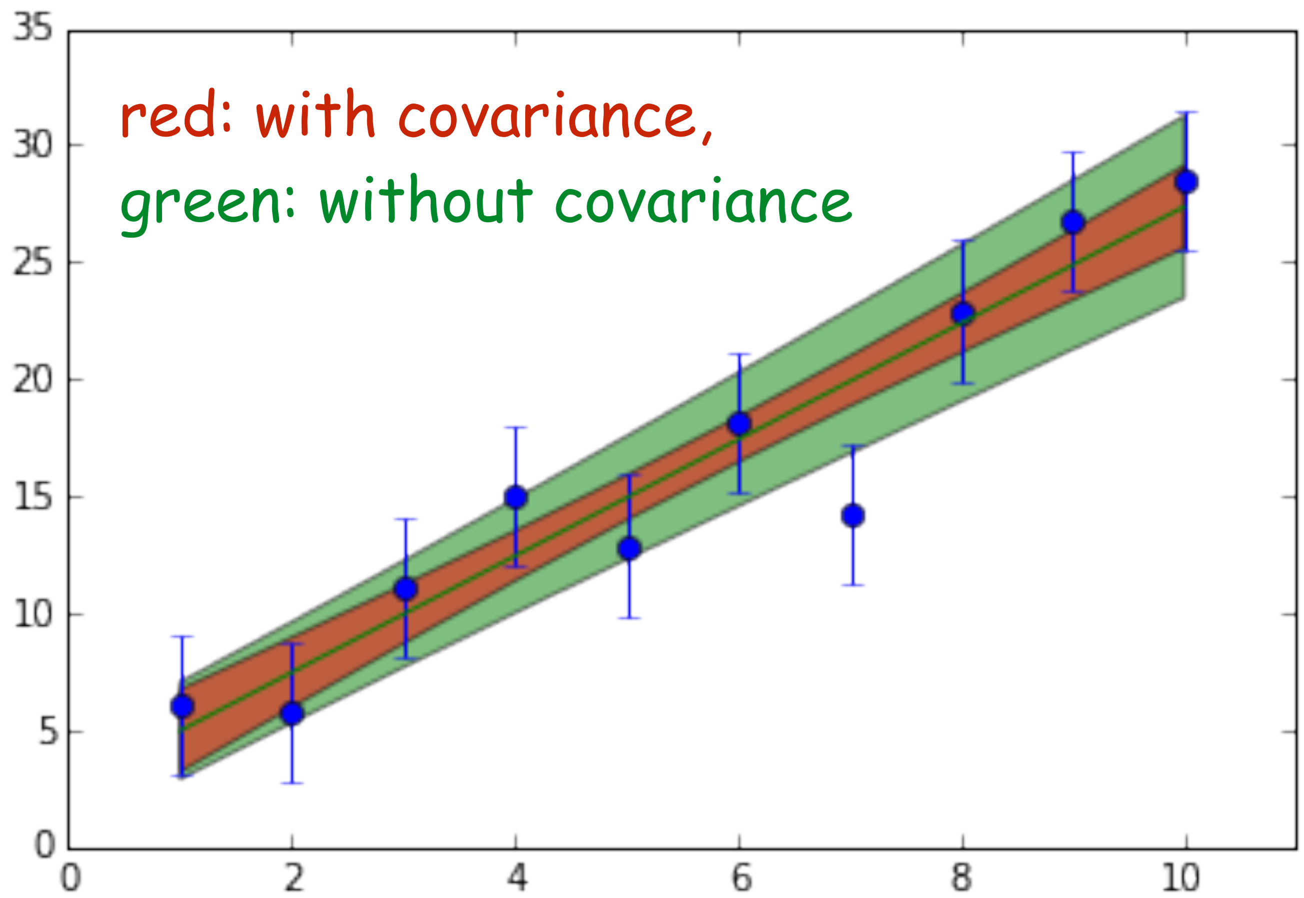
$$\sigma_y = \sqrt{x^2\sigma_m^2 + \sigma_c^2 + 2x\sigma_{mc}^2}$$

e.g. At x=6, y=21.06 ± 0.93

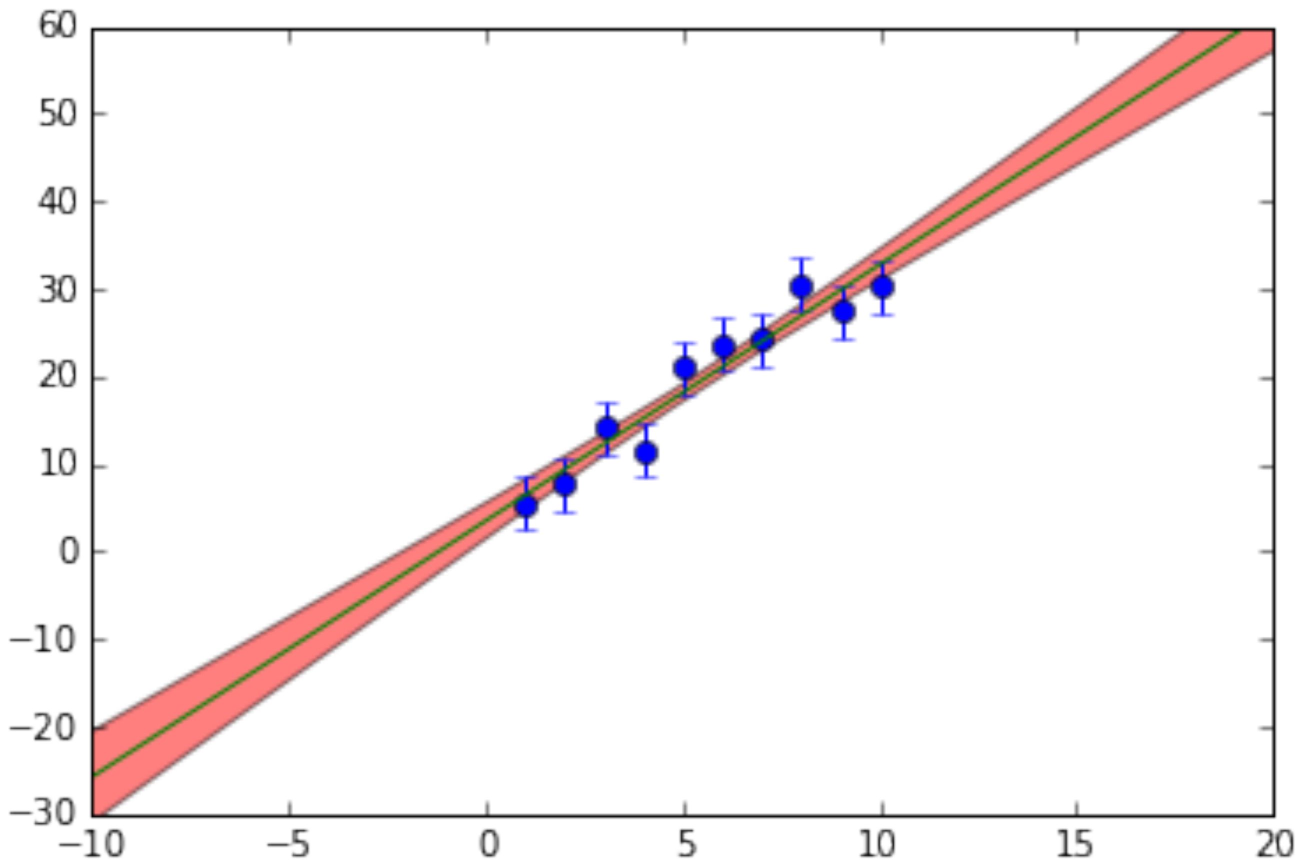


Example: fitting a straight line to some data

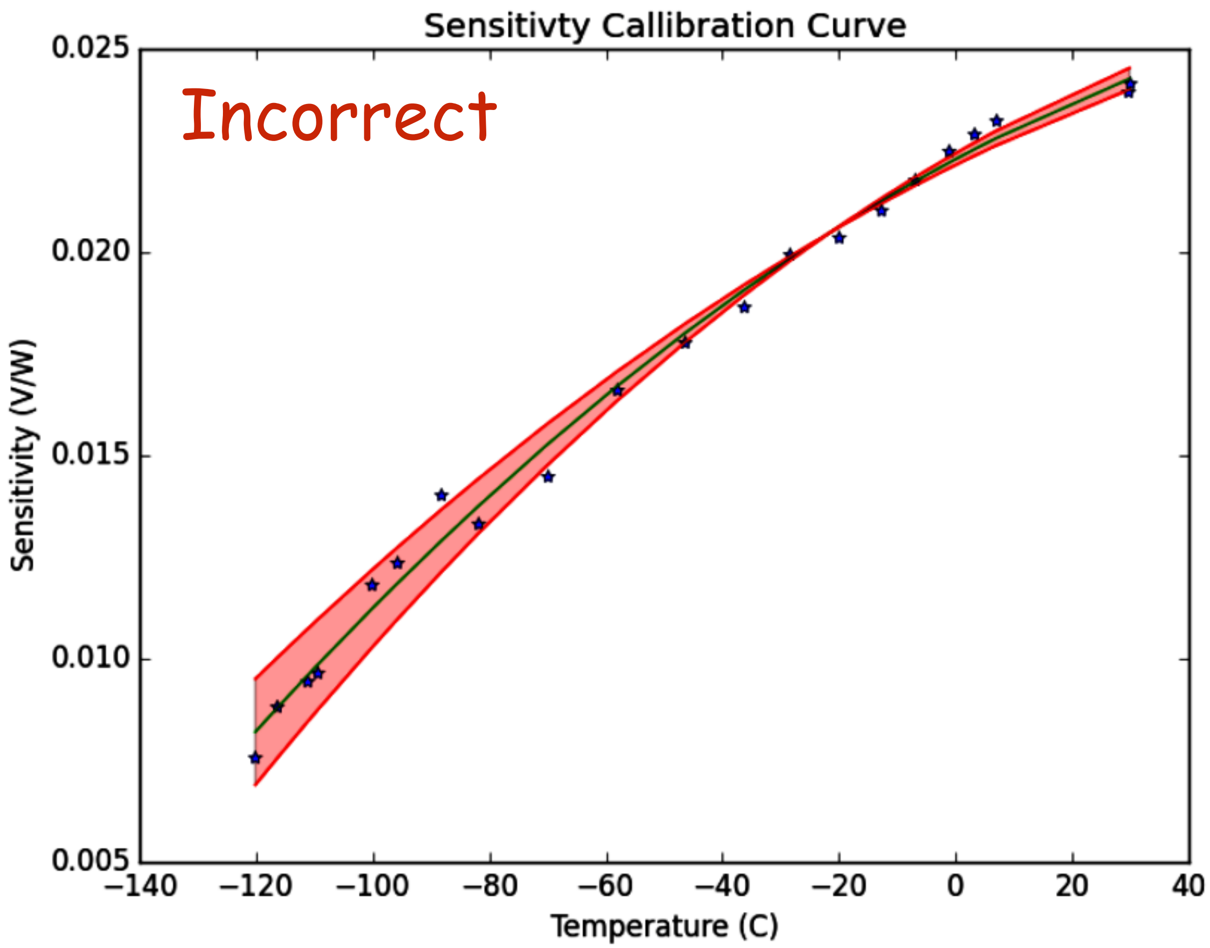
With and without covariance

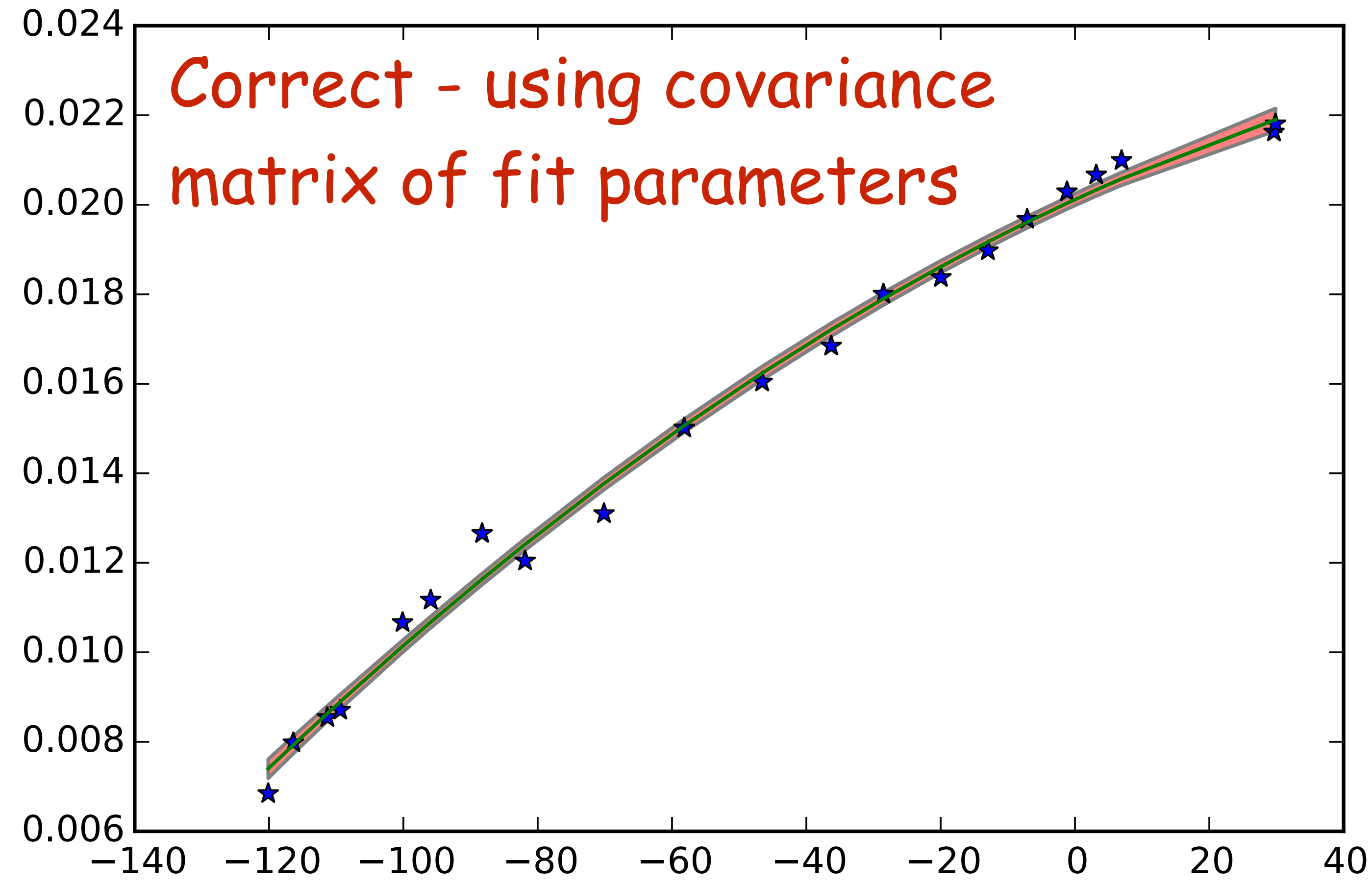


Extended Range



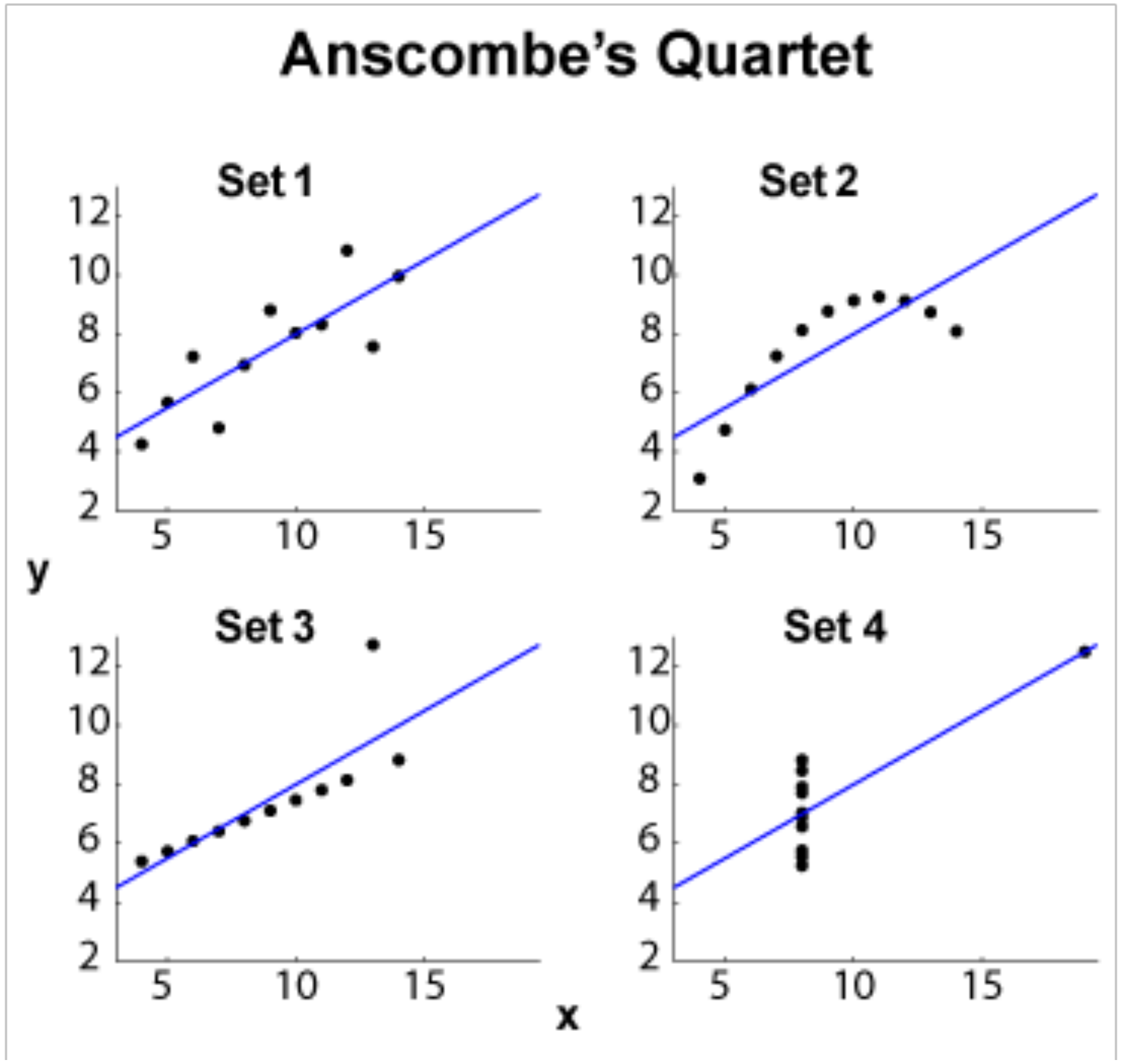
Revisit sensitivity curve from third slide





Anscombe's Quartet

- Anscombe's Quartet is a group of four data sets that provide a useful caution against blindly applying statistical methods to data.
- Each data set consists of ten x- and y-values such that the mean and variance of x and y, the correlation coefficient, regression line, and error of fit using the line are the same. But as you can see, they are clearly quite different data .



- Always plot and visually inspect your data and best-fit curve!

http://neurochannels.blogspot.ie/2008/07/anscombes-quartet.html

Conclusions

- Numerical routines such as scipy.optimize.curve_fit() can be used to find the optimum parameter for a function to best-describe a data set.
- If you want true errors on the best-fit parameters then you must include errors on the data points (and tell curve_fit() to use them)
  - if errors on the data points are not used then, or just used as relative weights, the errors on the fit parameters returned as so that the reduced $\chi 2$ is ≈1.
- scipy.optimize.curve_fit() returns the optimum parameters and the full covariance matrix, giving errors on the parameters and correlations between them.
- If the fit function with best-fit parameters is used to calculate a value, or several to display a confidence interval, then the co-variance terms must be used to propagate the error on the parameters through the function.
- it is critical to give starting values for the fit somewhat close to the optimum to ensure convergence.
- you should always plot and visually inspect data and fitted function