(Computer) Matrix Methods for Propagation of Errors

John Quinn

Recap: Propagation of Errors

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

The Propagation of Errors Formula

Function of two variables: u=f(x,y):

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

Function of three variables: u=f(x,y,z):

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

Recap: Errors on value from best-fit function

- Propagate errors (and covariances) on best-fit values through the function parameters to calculate the uncertainty on f(x, pars).

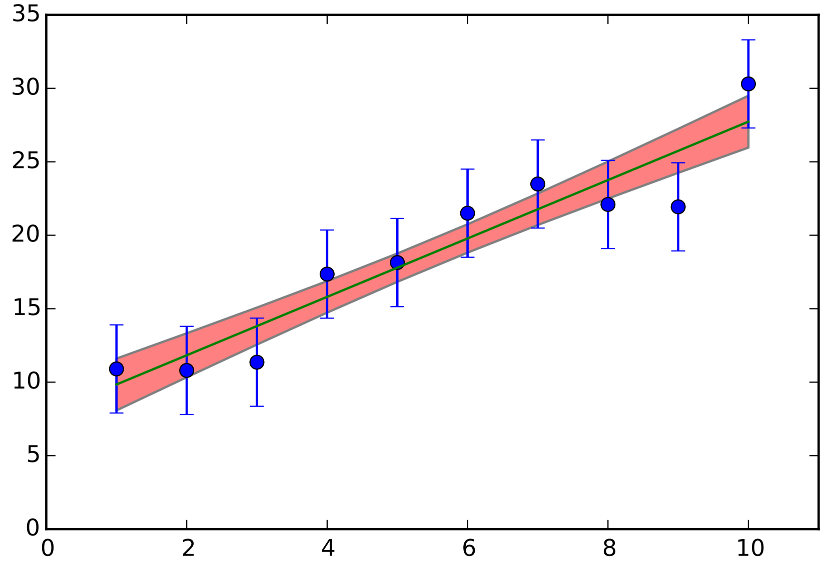
- e.g. linear:

$$y = f(x, m, c) = mx + c$$

best-fit m and c obtained from fit.

$$y \pm \sigma_y = f(x, m, c) \pm \sigma_f(x, m, \sigma_m, c, \sigma_c, \sigma_{mc}^2)$$

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

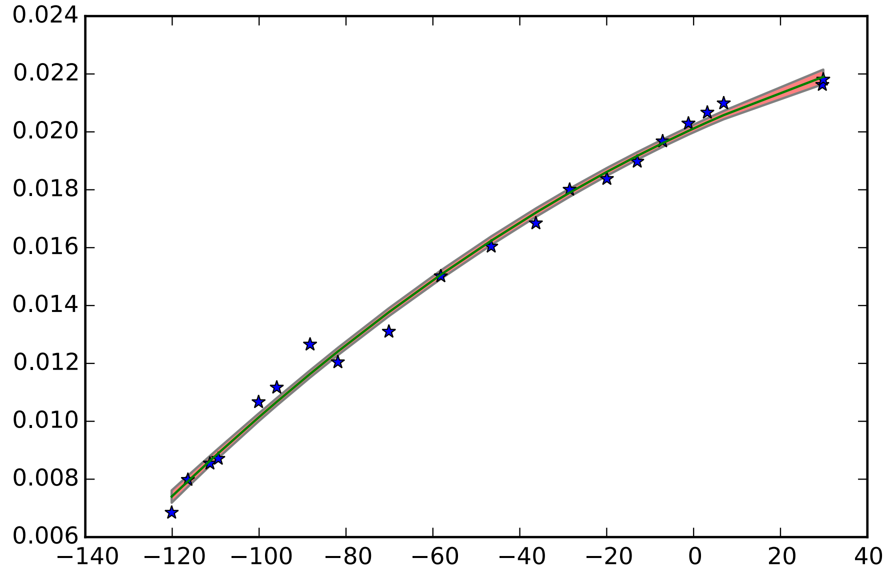


- e.g. quadratic:

$$y = f(x, a, b, c) = ax^2 + bx + c$$

best-fit a, b and c obtained from fit.

$$y \pm \sigma_y = f(x, a, b, c) \pm \sigma_f(x, a, \sigma_a, b, \sigma_b, c, \sigma_c, \sigma_{ab}^2, \sigma_{ac}^2, \sigma_{bc}^2)$$



Recap: Vector Notation

$$f(x+\delta x, y+\delta y) \approx f(x,y) + \frac{\partial f}{\partial x}\delta x + \frac{\partial f}{\partial y}\delta y$$

- For multiple variables (using vector notation):

$$f(\mathbf{x}+\delta\mathbf{x}) \approx f(\mathbf{x}) + \mathbf{g}(\mathbf{x})^T \delta\mathbf{x} + \ldots$$

f(**x**) is scalar function of several variables (passed as a vector).

where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \mathbf{g}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \qquad \delta\mathbf{x} = \begin{bmatrix} \delta x_1 \\ \delta x_2 \\ \vdots \\ \delta x_n \end{bmatrix}$$

$\mathbf{g}(\mathbf{x})$ is the Gradient vector (also called the Jacobian)

Matrix Notation

- Software packages usually return the covariance matrix of the form:

$$\mathbf{C} = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \sigma_{13}^2 & \dots & \sigma_{1n}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 & \sigma_{23}^2 & \dots & \sigma_{2n}^2 \\ \sigma_{31}^2 & \sigma_{32}^2 & \sigma_{33}^2 & \dots & \sigma_{3n}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_{m1}^2 & \sigma_{m2}^2 & \sigma_{m3}^2 & \dots & \sigma_{mn}^2 \end{bmatrix}$$

- Writing out (and coding) the full propagation of errors formula can get unwieldy.

- However, the full propagation of errors formula including all of the covariance terms can conveniently be written in matrix notation:

$$\sigma_u^2 = \mathbf{g}(\mathbf{x})^T \mathbf{C}\, \mathbf{g}(\mathbf{x})$$

recall that $\mathbf{g}(\mathbf{x})$ is the gradient/Jacobian vector

Statistical Error Propagation

Joel Tellinghuisen[†]

Department of Chemistry, Vanderbilt University, Nashville, Tennessee 37235

Received: September 26, 2000; In Final Form: February 1, 2001

The simple but often neglected equation for the propagation of statistical errors in functions of correlated variables is tested on a number of linear and nonlinear functions of parameters from linear and nonlinear least-squares (LS) fits, through Monte Carlo calculations on 10^4–4×10^5 equivalent data sets. The test examples include polynomial and exponential representations and a band analysis model. For linear functions of linear LS parameters, the error propagation equation is exact. Nonlinear parameters and functions yield nonnormal distributions, but their dispersion is still well predicted by the propagation-of-error equation. Often the error computation can be bypassed by a redefinition of the least-squares model to include the quantity of interest as an adjustable parameter, in which case its variance is returned directly in the variance-covariance matrix. This approach is shown formally to be equivalent to the error propagation method.

Introduction

Perhaps one of the "best-kept secrets" in experimental physical science is the simple matrix expression for error propagation[1–3]

$$\sigma_f^2 = \mathbf{g}^{\mathrm{T}}\,\mathbf{V}\,\mathbf{g} \qquad (1)$$

in which σ_f^2 represents the variance in some function f of a set of parameters $\boldsymbol{\beta}$, whose variance-covariance matrix is $\mathbf{V}$, with the ith element in the vector $\mathbf{g}$ being $\partial f/\partial \beta_i$. To be sure, undergraduate chemistry and physics students are drilled in the form of this equation that applies for *uncorrelated* variables:

$$\sigma_f^2 = \sum \left(\frac{\partial f}{\partial \beta_i}\right)^2 \sigma_{\beta_i}^{\;2} \qquad (2)$$

uted, with variances (the diagonal elements of **V**) known exactly at the outset:[1–3,7]

$$\mathbf{V} = \mathbf{A}^{-1} \qquad (3)$$

where **A** is the matrix of the normal equations. Accordingly, linear functions of such parameters are unbiased and normal, with variances σ_f^2 known exactly. On the other hand, nonlinear parameters and nonlinear functions of linear parameters are not normally distributed and in fact are usually biased.[7] Nevertheless, for the cases examined here, this nonnormality seldom translates into a serious deficiency in the predictions of eq 1 and its "normal" interpretation for establishing confidence limits. Indeed, the 10% "rule of thumb" suggested for nonlinear LS parameters[7] seems also to apply to functions of such parameters: If the relative standard error σ_f/f is $\leq 1/10$, confidence

Check!

Function of two variables: u=f(x,y): $\sigma_u^2 = \left(\frac{\partial f}{\partial x}\right)^2 \sigma_x^2 + \left(\frac{\partial f}{\partial y}\right)^2 \sigma_y^2 + 2\left(\frac{\partial f}{\partial x}\right)\left(\frac{\partial f}{\partial y}\right)\sigma_{xy}^2$

Check using matrix calculations: $\mathbf{g}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$ $\mathbf{C} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 \\ \sigma_{xy}^2 & \sigma_y^2 \end{bmatrix}$

$$\mathbf{g}(\mathbf{x})^T \mathbf{C}\, \mathbf{g}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix} \begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 \\ \sigma_{yx}^2 & \sigma_y^2 \end{bmatrix} \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial f}{\partial x}\sigma_x^2 + \frac{\partial f}{\partial y}\sigma_{xy}^2 & \frac{\partial f}{\partial x}\sigma_{xy}^2 + \frac{\partial f}{\partial y}\sigma_y^2 \end{bmatrix} \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

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

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

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

How does this help?

- How does this help matters?
  - ans:
    - software packages such as numpy will to matrix multiplication for us.
    - easy to incorporate all covariance terms

Error on Area using Matrix Approach in Python

$$\sigma_u^2 = \mathbf{g}(\mathbf{x})^T \mathbf{C}\, \mathbf{g}(\mathbf{x})$$

$$A = w \times l$$

$$\frac{\partial A}{\partial w} = l$$

$$\frac{\partial A}{\partial l} = w$$

Python>=3.5: Matrix Multiplication of 2D arrays with '@':

Steps:

1. Make a function which returns the gradient (column) vector (e.g. area calculation)

```
def gr(w,l):
    dAdw=l
    dAdl=w
    return np.array( [ [dAdw], [dAdl] ] )
```

2. If we have the covariance matrix, C, for the measurements of width (w) (mean width = mw) and length (l) (mean length = ml) then the error on the area ($A = mw \times ml$) is:

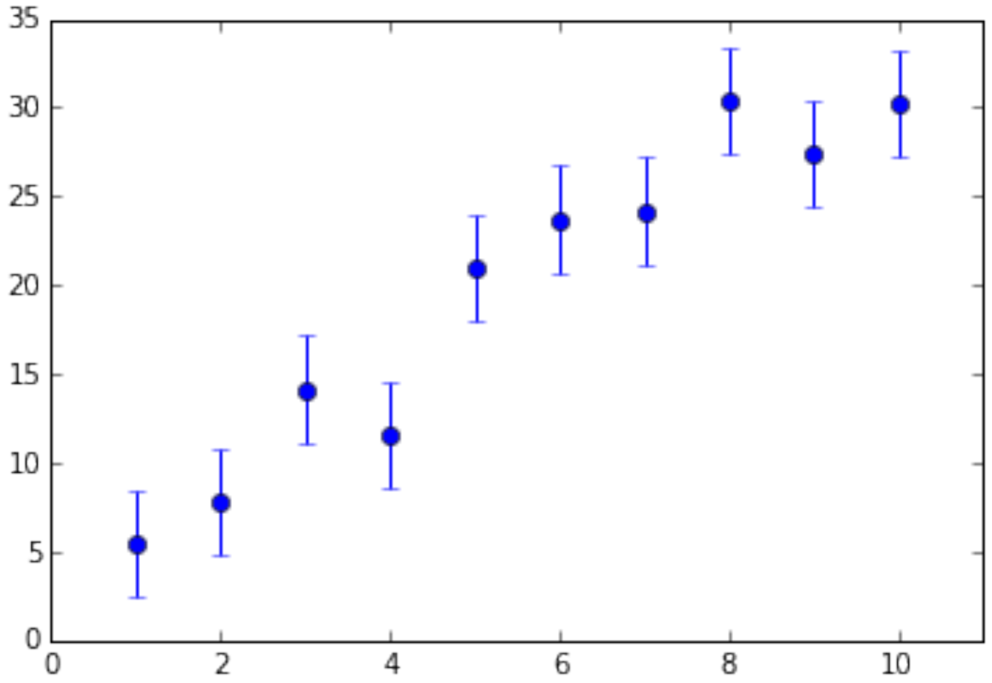
```
σA=np.sqrt( float(gr(mw,ml).T @ C @ gr(mw,ml)) )
```

The Matrix Method gives the exact same results as:

$$\sigma_A = \sqrt{\left(\frac{\partial A}{\partial w}\right)^2 \sigma_w^2 + \left(\frac{\partial A}{\partial l}\right)^2 \sigma_l^2 + 2\left(\frac{\partial A}{\partial w}\right)\left(\frac{\partial A}{\partial l}\right)\sigma_{wl}^2}$$

and is much easier to scale to 3 or more parameters!

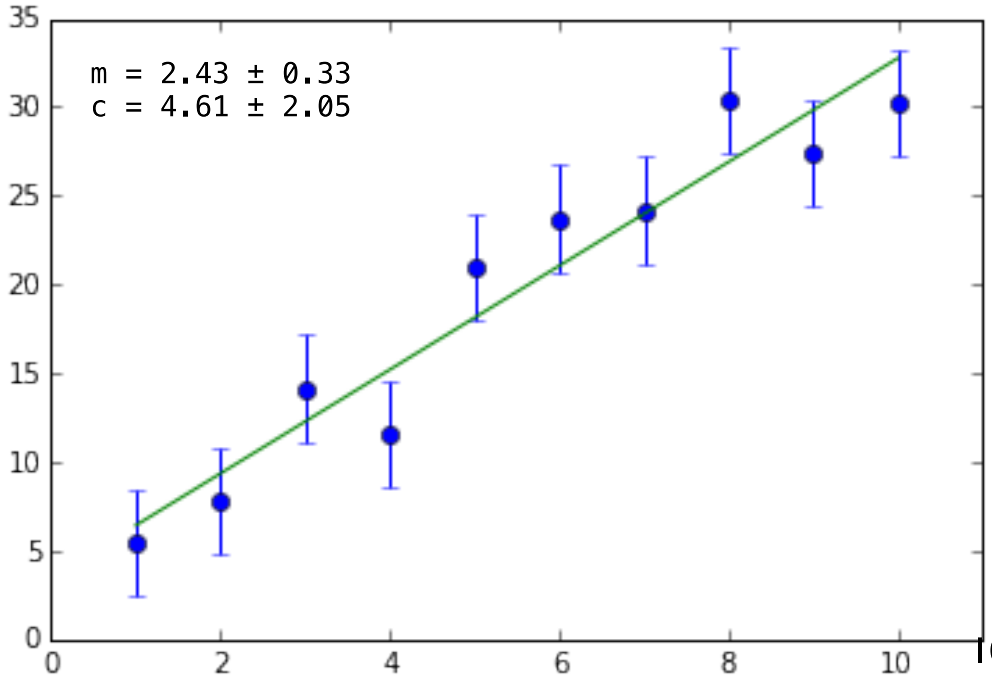
Revisit Example: fitting a straight line to data



```
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 fit to get a value of y and its uncertainty.
- i.e., $y \pm \sigma_y = f(x, m, c) \pm \sigma_f(x, m, \sigma_m, c, \sigma_c, \sigma_{mc}^2)$, so, for example, f(6)=?±?
- 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 ± 0.96

$$\sigma_y^2 = \mathbf{g(x)}^{\mathbf{T}}\,\mathbf{C}\,\mathbf{g(x)}$$

```
def gr(x,m,c):
    dfdm=x
    dfdc=1
    return np.array( [ [dfdm], [dfdc] ] )
```

e.g., what is error on f(6)?

```
np.sqrt( float(gr(6,m,c).T @ pcov @ gr(6,m,c)) )

0.96
```

Using the Matrix Approach for arrays

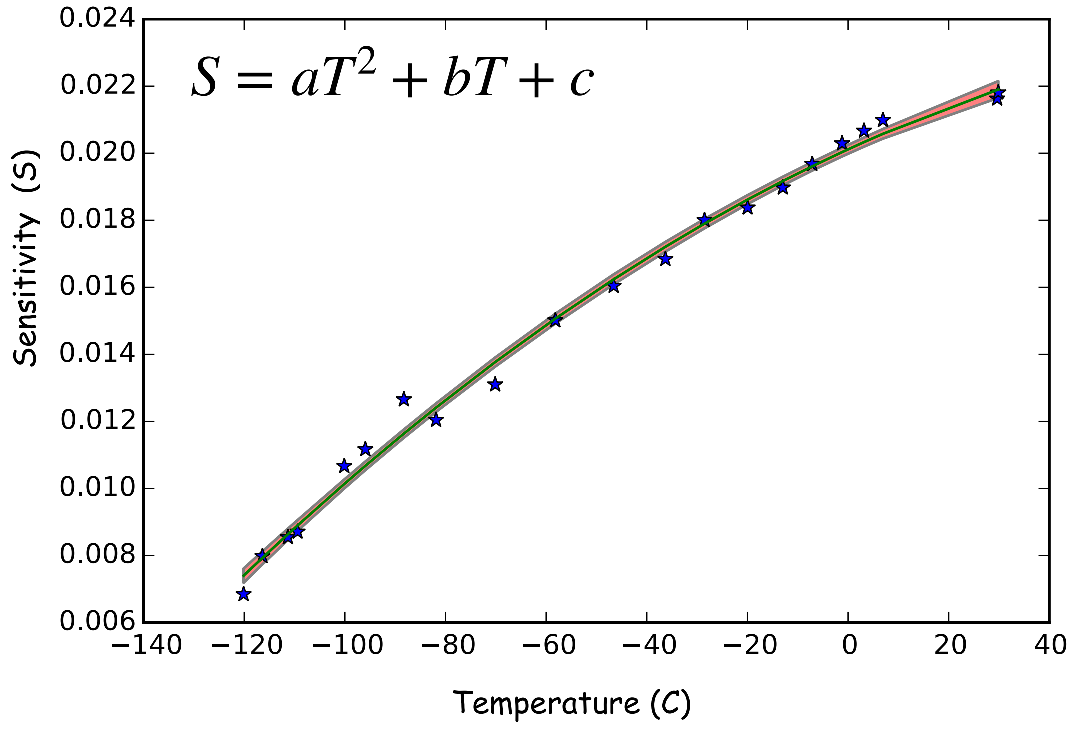
```
np.sqrt( float(gr(6,m,c).T @ pcov @ gr(6,m,c) )

0.96
```

- The above will not work vectorised on numpy arrays of data for x! (needs cleverer coding!).
- Instead do element by element (e.g. list comprehension or loop)
- To do many values (i.e. x is a numpy array) using list comprehension:

```
    s2=np.array([float(gr(z,m,c).T @ pcov @ gr(z,m,c)) for z in x])
```

- or use a loop!

Revisit sensitivity curve



```
# T = Temperatures (already loaded)
# S = Sensitivites (already loaded)


def F(T,a,b,c):
    return a*T**2 + b*T + c


popt,pcov=curve_fit(F,T,S)


plt.plot(T,S,"*")
plt.plot(T,F(T,*popt))
```

```
def gr(T,a,b,c):
    dfda=T**2
    dfdb=T
    dfdc=1
    return np.array([[dfda],[dfdb],[dfdc]])

es=[np.sqrt(float(gr(Ti,*popt).T @ pcov @ gr(Ti,*popt))) for Ti in T]

upper=F(T,*popt)+es
lower=F(T,*popt)-es

plt.fill_between(T,upper,lower,facecolor='red',alpha=0.5);
```

Revisit sensitivity curve

Equivalent:

```
np.sqrt(float(gr(Ti,*popt).T @ pcov @ gr(Ti,*popt)))
```

$$\sigma_S = \sqrt{\left(\frac{\partial S}{\partial a}\right)^2 \sigma_a^2 + \left(\frac{\partial S}{\partial b}\right)^2 \sigma_b^2 + \left(\frac{\partial S}{\partial c}\right)^2 \sigma_c^2 + 2\left(\frac{\partial S}{\partial a}\right)\left(\frac{\partial S}{\partial b}\right)\sigma_{ab}^2 + 2\left(\frac{\partial S}{\partial a}\right)\left(\frac{\partial S}{\partial c}\right)\sigma_{ac}^2 + 2\left(\frac{\partial S}{\partial b}\right)\left(\frac{\partial S}{\partial c}\right)\sigma_{bc}^2}$$

Conclusions

- Computer packages such as np.cov() and scipy.optimize.curve_fit() return the full variance-covariance matrix.
- The matrix method for propagating errors:

$$\sigma_u^2 = \mathbf{g}(\mathbf{x})^T \mathbf{C}\, \mathbf{g}(\mathbf{x})$$

  is completely equivalent to the standard propagation of errors formula but much more convenient for propagating error including the full covariance terms on a computer.

- Even though we generally neglect covariances of measurements, if we want to estimate the uncertainty on a value calculated with a function which has uncertainties on its parameters (e.g. if those parameters were derived from fitting the function to data) then we must include the covariance terms as it is likely there is strong correlations between the parameters.