Division of Physics & Applied Physics
PH2198 Physics Laboratory IIA
Error Analysis

1   Measurement error

The result of a single measurement should be reported in the format

$$(\text{estimate}) \pm (\text{measurement error}).$$

The **estimate** is the reading from the device. The **measurement error** specifies the range where the true value may lie. By convention, the measurement error has one significant figure, and the estimate has same precision as the measurement error.
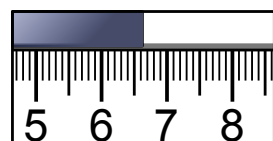
Suppose you use a digital multimeter to measure the current in a circuit, and the readout is stable (i.e., not fluctuating). Then you should report a result like this:

 $= (0.320 \pm 0.005)$ A

That's because, according to the readout, the value may be between 0.315 A (which gets rounded up to 0.32 A) and $0.324999\ldots$ A (which gets rounded down). Thus, the measurement error is ± 0.005 A. Note that the estimate is reported as 0.320 A, not 0.32 A, to ensure that it has the same precision as the error [1].

When using a device with hatch marks, such as a ruler or analog oscilloscope display, the measurement error is determined by the smallest markings. For example, if the smallest markings on a ruler have 1 mm spacing, the measurement error is ± 0.5 mm, so a reading should be reported like this:

 $= (6.60 \pm 0.05)$ cm

In more complicated situations, you must exercise your judgment. For instance, suppose you have a digital multimeter reading that is not stable: the last digit changes constantly, so that the reading fluctuates between 0.32, 0.33, and 0.34 A. The value is between 0.315 A and $0.344999\ldots$ A, which is a range of ± 0.015 A. Since we use one significant figure for errors, the result is reported like this:

 $= (0.33 \pm 0.02)$ A

Alternatively, suppose the last digit is changing so fast that you can't make out what values it takes. Then you can report the result like this:

 $= (0.35 \pm 0.05)$ A

Measurement uncertainties can also come from other aspects of an experiment. Suppose you use a ruler to measure the distance to an object, but the object wobbles by ± 2 mm, larger than the 1 mm hatch marks of the ruler. In that case, you should report a measurement error of ± 2 mm, not ± 0.05 mm.

2   Sampling Error (Repeated Measurements)

In some experiments, the system being measured is intrinsically random. For example, radioactive decay occurs randomly, so counting the number of decay events yields a slightly different result from interval to interval, *no matter how precise your apparatus is*. Often, we find a **mean** value by taking many measurements and averaging them. The result is reported in the format

$$\text{(estimated mean)} \pm \text{(standard error of the mean)}.$$

The quantity after the $\pm$ refers to **sampling error**, which is *not* the same as measurement error. Sampling error arises because you only took a finite number of samples, so your estimate of the mean is not completely certain.

In this type of scenario, measurement error is usually ignored, as the sampling error caused by the system's randomness is larger than the measurement error. (In the opposite case, where measurement error is larger, there'd be no point doing repeated measurements. And if the measurement error is about as large as the system's randomness, you're probably doing social science, so good luck with that.)

Suppose you take N measurements, and the results are $X_1, X_2, \ldots, X_N$. Then

$$\text{estimated mean} \equiv \bar{X} = \frac{X_1 + X_2 + \cdots + X_N}{N}.$$

You can compute $\bar{X}$ using the `mean` function in Python or Matlab. Also,

$$\text{standard error of the mean} = \frac{\sigma}{\sqrt{N}},$$

where σ is the **standard deviation of the sample**, defined as

$$\sigma = \sqrt{\frac{\sum_{n=1}^{N}(X_n - \bar{X})^2}{N-1}}.$$

You can compute σ using the `std` function in Python or Matlab.

3   Error Estimates for Sums, Products, and Powers

When analyzing experiments, you often need to take sums, products, and powers of measured data. For instance, you might measure the voltage V and current I across a circuit, and use them to find the resistance $R = V/I$. Errors in V and I will turn into errors in R. This is called **error propagation**.

The basic rules of error propagation are easy to summarize (if you want the detailed derivations, refer to Ref. [1]). Consider two independent quantities $X \pm \Delta X$ and $Y \pm \Delta Y$. We call $\pm\Delta X$ and $\pm\Delta Y$ the **standard errors** for X and Y, regardless of whether they originate from measurement error or sampling error. Now, suppose we derive $Z \pm \Delta Z$ by addition or subtraction of X and Y:

$$\begin{cases} Z = X + Y \text{ or} \\ Z = X - Y \end{cases} \quad \leftrightarrow \quad \Delta Z = \sqrt{(\Delta X)^2 + (\Delta Y)^2}.$$

More generally, for any two (error-free) constants α and β,

$$Z = \alpha X + \beta Y \qquad \leftrightarrow \qquad \Delta Z = \sqrt{\alpha^2(\Delta X)^2 + \beta^2(\Delta Y)^2}.$$

For products and fractions,

$$\begin{cases} Z = \alpha XY \text{ or} \\ Z = \alpha X/Y \end{cases} \qquad \leftrightarrow \qquad \frac{\Delta Z}{|Z|} = \sqrt{\left(\frac{\Delta X}{X}\right)^2 + \left(\frac{\Delta Y}{Y}\right)^2}.$$

For powers, exponentials and logarithms,

$$Z = X^n \qquad \leftrightarrow \qquad \frac{\Delta Z}{|Z|} = |n|\frac{\Delta X}{|X|}$$

$$Z = \ln(X) \qquad \leftrightarrow \qquad \Delta Z = \frac{\Delta X}{|X|}$$

$$Z = \log_{10}(X) \qquad \leftrightarrow \qquad \Delta Z = \frac{1}{\ln(10)}\frac{\Delta X}{|X|}$$

$$Z = \exp(\alpha X) \qquad \leftrightarrow \qquad \frac{\Delta Z}{|Z|} = |\alpha|\,\Delta X.$$

More complicated formulas can usually be handled by combining these rules. For example, suppose you seek an object's kinetic energy $T = \frac{1}{2}mv^2$, using data $m \pm \Delta m$ and $v \pm \Delta v$. First, find the error in the quantity v^2 using the rule for powers:

$$\frac{\Delta(v^2)}{|v^2|} = 2\frac{\Delta v}{|v|}.$$

Then, plug this into the rule for products:

$$\Delta T = |T|\sqrt{\left(\frac{\Delta m}{m}\right)^2 + \left(\frac{\Delta(v^2)}{v^2}\right)^2}$$

$$= \frac{1}{2}mv^2\sqrt{\left(\frac{\Delta m}{m}\right)^2 + 4\left(\frac{\Delta v}{v}\right)^2}.$$

4 Curve Fitting

Many experiments require you to perform **curve fitting**. You have data points

$$\begin{array}{cc} X_1 & Y_1 \\ X_2 & Y_2 \\ \vdots & \vdots \\ X_N & Y_N \end{array}$$

which may be either direct measurement results, or derived from measurements. You are then required to fit these data to a relation such as

$$Y = b_0 + b_1 X.$$

The goal is to determine b_0 and b_1, which are called **estimators**.

Curve fitting, and the propagation of error into estimators, is a complicated subject in applied statistics. For most experiments in this course, you only need a simple curve fitting method known as **weighted linear least squares regression**.

In this method, each data point (X_i, Y_i) is assigned a relative weight w_i. Data points with smaller errors are treated as more importance (higher "weight") during the fit. A standard weight assignment is

$$w_i = (\Delta Y_i)^{-2}.$$

Note that this ignores the errors in X. There is no standard weighting procedure that accounts for errors in both X and Y. So it is best to let the variable with more "serious" error be the Y variable, so its error is used for weighting.

As an example, suppose we measure the period of a pendulum, T, with varying arm length L. Theoretically, T and L are related to the gravitational constant g by

$$T = 2\pi \sqrt{\frac{L}{g}} \quad \Rightarrow \quad T^2 = \frac{4\pi^2}{g} L.$$

We can fit our data to the relation

$$T^2 = b_0 + b_1 L,$$

and use b_1 to estimate $4\pi^2/g$. Hence, we can find $g = 4\pi^2/b_1$, and its error Δg:

$$\frac{\Delta g}{g} = \frac{\Delta b_1}{b_1} \quad \Rightarrow \quad \Delta g = \frac{4\pi^2 \Delta b_1}{b_1^2}.$$

Suppose all our measurements of T have the same measurement error $\pm\Delta T$. From the error propagation rule for powers (Section 3),

$$\frac{\Delta(T^2)}{T^2} = 2\frac{\Delta T}{T} \quad \Rightarrow \quad \Delta(T^2) = 2\,T\,\Delta T.$$

Thus, the errors in T^2 are not uniform: data points with large T have more error.

4.1   Sample Python program

A sample Python program for weighted linear least squares curve fitting is shown below. The fitting is done by the `curve_fit` function, from the `scipy.optimize` module.

In this program, `curve_fit` is called with four inputs: the model function, the x data, the y data, and the standard errors of the y data. The first input is a function that you must define, telling `curve_fit` what kind of curve to fit to (in this case, a first-order polynomial). The last three inputs are arrays of numbers.

The `curve_fit` function returns two outputs. The first output is an array of estimators (i.e., the additional inputs that, when passed to the model function, give the best fit to the data). The second output is a "covariance matrix", whose diagonal terms are the estimator variances (i.e., taking their square roots yields the standard errors of the estimators).

```
from scipy import *
from scipy.optimize import curve_fit

## Input data (generated by simulation with random noise).
L  = array([0.10, 0.16, 0.22, 0.28, 0.34, 0.40, 0.46, 0.52, 0.58, 0.64])
T  = array([0.71, 0.76, 0.91, 1.00, 1.20, 1.14, 1.44, 1.40, 1.53, 1.58])
dT = array([0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05])

Tsq = T**2              # Estimates for T^2
Tsq_error = 2*T*dT      # Errors for T^2, from error propagation rules

## Define a function, named f, that returns an order-1 polynomial.
def f(x, b0, b1): return b0 + b1*x

## Use f to fit the L vs Tsq data
est, covar = curve_fit(f, L, Tsq, sigma=Tsq_error)

## Estimators and their errors:
b0  = est[0]                    # Estimate of intercept
b1  = est[1]                    # Estimate of slop
db0 = sqrt(covar[0,0])          # Standard error of intercept
db1 = sqrt(covar[1,1])          # Standard error of slope


g  = 4*pi*pi/b1                 # Estimate for g (computed from b1)
dg = (g/b1)*db1                 # Estimate for standard error of g

## Plot the data points, with error bars
import matplotlib.pyplot as plt
plt.errorbar(L, Tsq, yerr=Tsq_error, fmt='o')
plt.xlabel('L (m)', fontsize=12)
plt.ylabel('T^2 (s^2)', fontsize=12)
## Include the fitted curve in the plot
L2 = linspace(0, 0.7, 100)
plt.plot(L2, b0+b1*L2)
## State fitted value of g in the figure title
t = "g = " + "{:.1f}".format(g) + " +/- " + "{:.1f}".format(dg) + " m/s^2"
plt.title(t, fontsize=12)
plt.show()
```
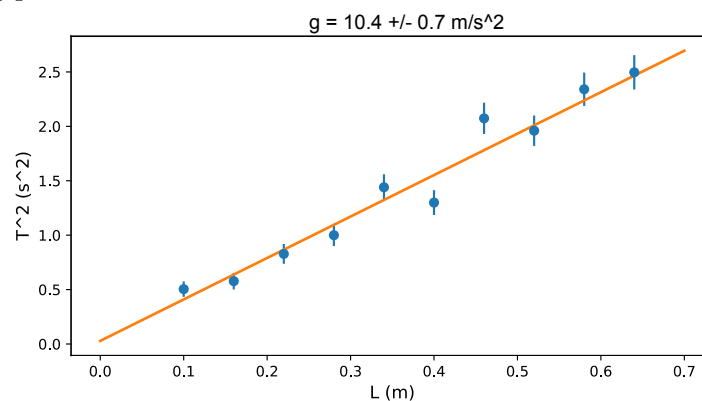
The resulting plot looks like this:

4.2 Sample Matlab program

Here is a sample Matlab program for weighted linear least squares curve fitting. It uses the `glmfit` function from Matlab's Statistics Toolbox.

```matlab
L  = [0.10, 0.16, 0.22, 0.28, 0.34, 0.40, 0.46, 0.52, 0.58, 0.64];
T  = [0.71, 0.76, 0.91, 1.00, 1.20, 1.14, 1.44, 1.40, 1.53, 1.58];
dT = [0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05];

Tsq = T.^2;              % Estimates for T^2
Tsq_error = 2*T.*dT;     % Errors for T^2, from error propagation rules

%% Fit L and Tsq with weighted linear least squares
wt = 1 ./ (Tsq_error.^2);    % Vector of weights
[est, dev, stats] = glmfit(L, Tsq, 'normal', 'weights', wt);

%% Estimators and their errors:
b0  = est(1);                 % Estimate of intercept
b1  = est(2);                 % Estimate of slope
db0 = stats.se(1);            % Standard error of intercept
db1 = stats.se(2);            % Standard error of slope

g  = 4*pi*pi/b1;              % Estimate for g (computed from b1)
dg = (g/b1)*db1;              % Estimate for standard error of g

%% Plot the data points and fitted curve
errorbar(L, Tsq, Tsq_error, "o");
xlabel("L (m)");
ylabel("T^2 (s^2)");
%% State the fitted value of g in the figure
t = "g = " + num2str(g,3) + " +/- " + num2str(dg,1) + " m/s^2";
title(t);
%% Plot the fitted curve
L2 = linspace(0, 0.7, 100);
hold on; plot(L2, b0 + b1*L2); hold off;
```
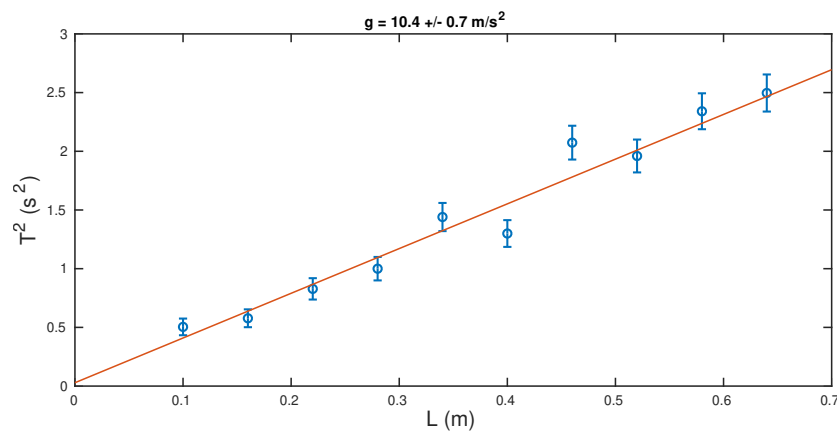
The resulting plot looks like this:

4.3   Curve fitting with Origin

The procedure for curve fitting with Origin is summarized below. For more details, consult the Origin online documentation.

1. Populate a worksheet with columns for (i) X, (ii) Y, and (iii) ΔY. You can enter the numbers manually, or import them from an external file.

2. Select the menu item `Plot → Symbol → Scatter`. Click on the appropriate boxes to specify (i) which column to use for the horizontal (X) variable, (ii) which column to use for the vertical (Y) variable, and (iii) which column to use for the error ΔY.

3. Select the menu item `Analysis → Fitting → Linear Fit`. In the dialog box, check that the right fitting options are entered. Under `Fit Control → Errors as Weight`, ensure that the `Instrumental` option is chosen. This corresponds to the standard weighting scheme from Section 4. Once ready, perform the fit.

4. The fitted curve will be inserted automatically into the graph. By default, a `Fit Results` table is also automatically added; you can click-and-drag to move or resize this table, and double-click to edit its contents.

   By default, Origin makes the table too small to be legible, so you ought to increase the size. Also, you should omit any statistical quantities listed in this table that are not relevant to your data analysis (e.g., the Pearson's r). Finally, you still need to convert the estimators into whatever quantities you are ultimately interested in (such as g).

   The resulting plot looks like this:
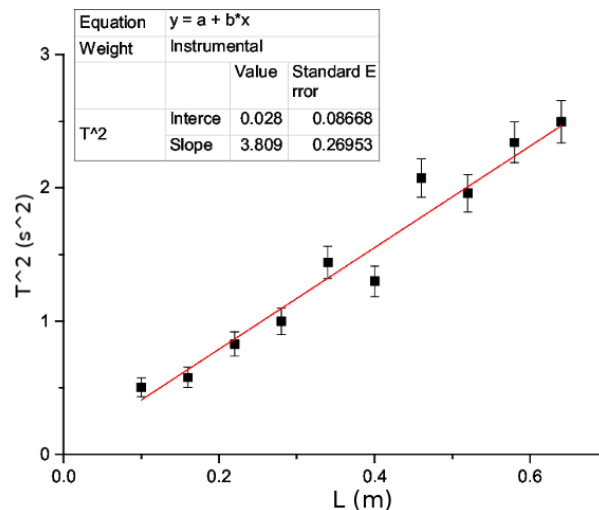
References

[1] I. G. Hughes and T. P. A. Hase, *Measurements and their uncertainties* (Oxford University Press, 2010).