

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.

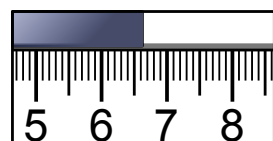
Suppose you use a digital multimeter to measure the current in a circuit, and the readout is stable (i.e., not fluctuating). Then you should report a result like this:



$$= (0.320 \pm 0.005) \text{ A}$$


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... 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:




$$= (6.60 \pm 0.05) \text{ cm}$$

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... A, which is a range of ± 0.015 A. Since we use one significant figure for errors, the result is reported like this:



$$= (0.33 \pm 0.02) \text{ A}$$

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:



$$= (0.35 \pm 0.05) \text{ A}$$

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, \dots, X_N . Then

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

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 :

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

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

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

For products and fractions,

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

For powers, exponentials and logarithms,

$$\begin{aligned} Z = X^n & \quad \leftrightarrow \quad \frac{\Delta Z}{|Z|} = |n| \frac{\Delta X}{|X|} \\ Z = \ln(X) & \quad \leftrightarrow \quad \frac{\Delta Z}{|Z|} = \frac{\Delta X}{|X|} \\ Z = \log_{10}(X) & \quad \leftrightarrow \quad \frac{\Delta Z}{|Z|} = \frac{1}{\ln(10)} \frac{\Delta X}{|X|} \\ Z = \exp(\alpha X) & \quad \leftrightarrow \quad \frac{\Delta Z}{|Z|} = |\alpha| \Delta X. \end{aligned}$$

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:

$$\begin{aligned} \Delta T &= |T| \sqrt{\left(\frac{\Delta m}{m}\right)^2 + \left(\frac{\Delta(v^2)}{v^2}\right)^2} \\ &= \frac{1}{2}mv^2 \sqrt{\left(\frac{\Delta m}{m}\right)^2 + 4\left(\frac{\Delta v}{v}\right)^2}. \end{aligned}$$

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

$$Y = aX + b.$$

The goal is to determine a and b , which are called **estimators**.

Curve fitting, and the propagation of error into estimators, is a complicated subject in applied statistics. In this course, you only need to know how to use 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. If X_i is error-free and Y_i has error $\pm\Delta Y_i$, a standard weight assignment is

$$w_i = \begin{cases} (\Delta Y_i)^{-1} & \text{(Python convention)} \\ (\Delta Y_i)^{-2} & \text{(Matlab convention)}. \end{cases}$$

(The two cases are only a matter of definition; see below.)

If both X_i and Y_i have errors, there is no standard way to assign the weights. Some texts recommend letting the variable with more serious error be the Y variable, so that 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

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

We can fit our data to the relation

$$T^2 = aL + b,$$

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

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

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

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

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 `polyfit` function, from the `numpy` module.

In this code sample, `polyfit` is called with five inputs: the x data, the y data, the degree of the fitting polynomial (1), an array of weights (the inverse of the standard errors), and `cov=True` (which says to include estimator errors in the output).

The function returns two outputs, named `est` and `covar` in this code sample. The first output is an array of estimators, in decreasing polynomial degree (in this case, a followed by b). The second output is a “covariance matrix”, whose diagonal terms are the estimator variances (i.e., the squares of the standard errors).

```

from scipy import *

## 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])

Tsqr = T**2          # Estimates for T^2
Tsqr_error = 2*T*dT   # Errors for T^2, from error propagation rules

## Fit L vs Tsqr with a first-order polynomial
degree = 1
weights = 1/Tsqr_error
est, covar = polyfit(L, Tsqr, degree, w=weights, cov=True)

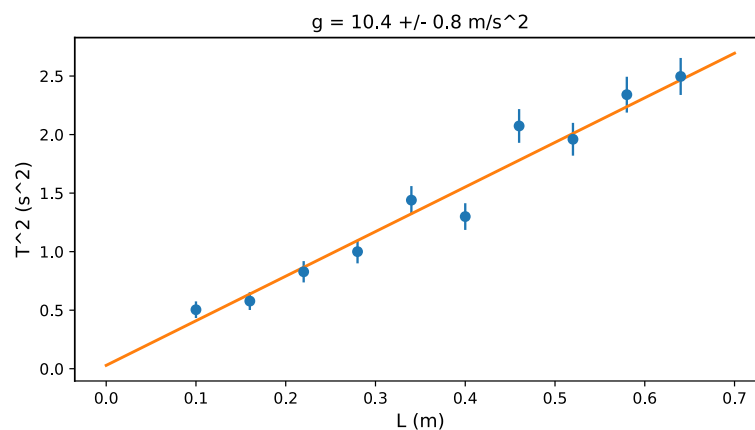
## Estimators and their errors:
a = est[0]            # Estimate of a
b = est[1]            # Estimate of b
da = sqrt(covar[0,0]) # Standard error of a
db = sqrt(covar[1,1]) # Standard error of b

## From a, calculate the estimate of g
g = 4*pi*pi/a
dg = (g/a)*da

## Plot the data points, with error bars
import matplotlib.pyplot as plt
plt.errorbar(L, Tsqr, yerr=Tsqr_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, a*L2+b)
## 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](#). Note that in Matlab, each weight is specified as the inverse square of the standard error. Also, the estimators are listed in order of *increasing* polynomial degree (i.e., b followed by a).

```
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

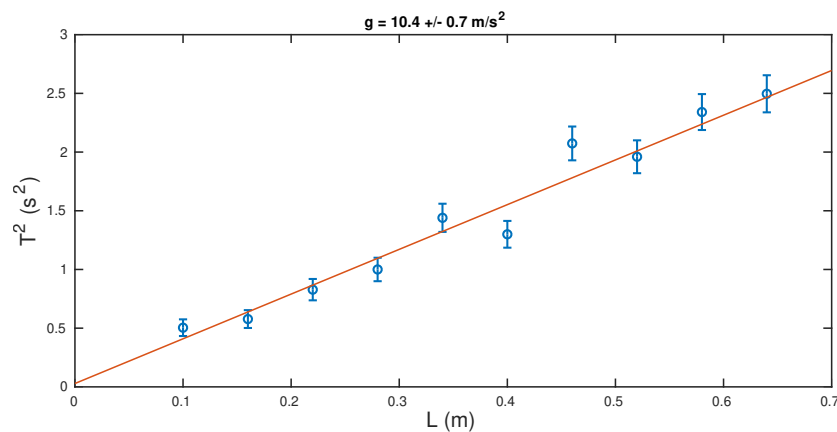
%% Fit L and Tsq with weighted linear least squares
wt = 1 ./ (Tsq_error.^2); % Vector of weights
[est, dev, stats] = glmfit(L, Tsq, 'normal', 'weights', wt);

%% Estimators and their errors:
a = est(2);           % Estimate of a
b = est(1);           % Estimate of b
da = stats.se(2);     % Standard error of a
db = stats.se(1);     % Standard error of b

%% Compute the estimate of g
g = 4*pi*pi/a;
dg = (g/a)*da;

%% 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, a*L2+b); hold off;
```

The resulting plot looks like this:



References

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