28.4: The QExpy python package for experimental physics

QExpy is a Python module that was developed with students from Queen's University to handle all aspects of undergraduate physics laboratories. In this section, we look at how to use QExpy to propagate uncertainties and to plot experimental data.

28.4.1: Propagating uncertainties

In Chapter 2, we saw how to use the "derivative method" to propagate the uncertainty from measurements into the uncertainty in a value that depended on those measurements. In *Example 2.3.2*, we propagated the uncertainties $x = (3.00 \pm 0.01)\text{m}$ and $t = (0.76 \pm 0.15)\text{s}$ to the quantity $k = \frac{t}{\sqrt{x}}$. We show below how easily this can be done with QExpy:

python code 28.4.1

QExpy to propagate uncertainties

```
#First, we load the QExpy module
import qexpy as q
#Now define our measurements with uncertainties:
t = q.Measurement(0.76, 0.15) # 0.76 +/- 0.15
x = q.Measurement(3, 0.1) # 3 +/- 0.1
#Now define k, which depends on t and x:
k = t/q.sqrt(x) #use the QExpy version of sqrt() since x is of type Measurement
#Print the result:
print(k)
```

Output

```
0.44 +/- 0.09
```

which is the result that we obtained when manually applying the derivative method. Note that we used the square root function from the QExpy module, as it "knows" how to take the square root of a value with uncertainty (a "Measurement" in the language of QExpy).

We also saw that when we had repeated measurements of the same quantity (Section 2.3), one could define a central value and uncertainty for that quantity by using the mean and standard deviations of the measurements. QExpy can easily take a set of measurements (an array of values) and convert them into a single quantity (a "Measurement") with a central value and uncertainty that correspond to the mean and standard deviation of the set of measurements:

python code 28.4.2

QExpy to calculate mean and standard deviation

```
#First, we load the QExpy module
import qexpy as q
#We define $t$ as an array of values (note the square brackets):
t = q.Measurement([1.01, 0.76, 0.64, 0.73, 0.66])
#Choose the number of significant figures to print:
q.set_figs(2)
print("t=",t)
```

Output

```
t = 0.76 +/- 0.15
```

By using QExpy, we do not need to tediously calculate the mean and standard deviation, as we had in *Example 2.3.1*.

28.4.2: Plotting experimental data with uncertainties

In Chapter 2 we had presented the data in *Table A4.4.1* which corresponded to our measurements of how long it took (t) for an object to drop a certain distance, x. We had also introduced Chloe's Theory of gravity that predicted that the data should be described by the following model:

$$t = k\sqrt{x}$$

where k was an undetermined constant of proportionality.

x[m]	t[s]	$\sqrt{x}[\mathrm{m}^{\frac{1}{2}}]$	$k[\mathrm{sm}^{-\frac{1}{2}}]$
1.00	0.33	1.00	0.33
2.00	0.74	1.41	0.52
3.00	0.67	1.73	0.39
4.00	1.07	2.00	0.54
5.00	1.10	2.24	0.49

Table A4.4.1*: Measurements of the drop times, t, for a bowling ball to fall different distances, x. We have also computed $\sqrt{x}$ and the corresponding value of k.*

The easiest way to visualize and analyze those data is to plot them. In particular, if we plot (graph) t versus $\sqrt{x}$, we expect that the points will fall on a straight line that goes through zero, with a slope of k (if the data are described by Chloe's Theory). We can use QExpy to graph the data as well as determine ("fit") for the slope of the line that best describes the data, since we expect that the slope will correspond to the value of k. When plotting data and fitting them to a line (or other function), it is important to make sure that the values have at least an uncertainty in the quantity that is being plotted on the y axis. In this case, we have assumed that all of the measurements of time have an uncertainty of 0.15s and that the measurements of the distance have no (or negligible) uncertainties. The python code below shows how to use QExpy to plot and fit the data to a straight line.

python code 28.4.3

Using QExPy to plot and fit linear data

```
#First, we load the QExpy module:
import qexpy as q

#Use matplotlib as the plot engine (try using 'bokeh' instead of 'mpl')
q.plot_engine = 'mpl'

#Set the number of significant figures to 2:
q.set_sigfigs(2)

#Then we enter the data:
#start with the values for the square root of height:
sqx = [1., 1.41, 1.73, 2., 2.24]

#and then, the corresponding times:
t = [0.33, 0.74, 0.67, 1.07, 1.1]

#Let us attribute an uncertainty of 0.15 to each measured values of t:
terr = 0.15

#We now make the plot. First, we create the plot object with the data
#Note that x and y refer to the x and y axes
fig = q.MakePlot(xdata = sqx, xname = "sqrt(distance)[m^0.5]", ydata = t,
yerr = terr, yname = "time [s]", data_name = "My data")
```

```
23
24 #Ask QExpy to also determine the line of best fit.
25 fig.lit("linear")
26
27 #Then, we show it:
28 fig.show()
```

Output

```
01 ----------Fit results----------
02 Fit of My data to linear
03 Fit parameters:
04 My data_linear_fit0_fitpars_intercept = -0.24 +/- 0.22,
05 My data_linear_fit0_fitpars_slope = 0.61 +/- 0.13
06
07 Correlation matrix:
08 [[ 1.     -0.968]
09  [-0.968   1.   ]]
10  chi/2ndof = 2.04/2
11  ----------End fit results----------
```

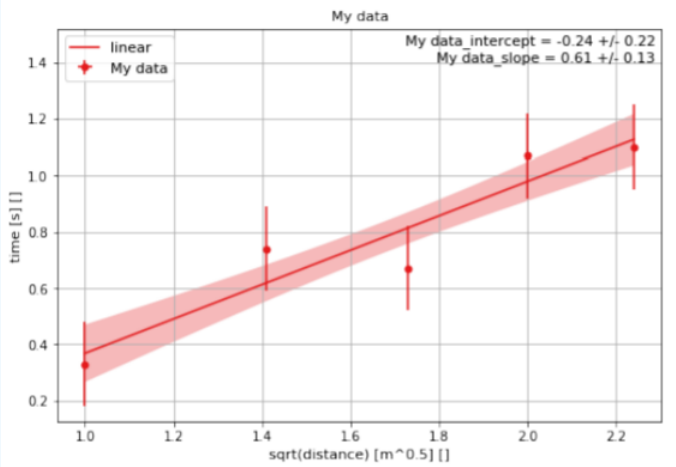

Figure A4.4.1: *QExpy plot of t versus $\sqrt{x}$ and line of best fit.*

The plot in Figure A4.4.1 shows that the data points are consistent with falling on a straight line, when their error bars are taken into account. We've also asked QExpy to show us the line of best fit to the data, represented by the line with the shaded area. When we asked for the line of best fit, QExpy not only drew the line, but also gave us the values and uncertainties for the slope and the intercept of the line. The shaded area around the line corresponds to other possible lines that one would obtain using different values of the slope and intercept within their corresponding uncertainties. The output also provides a line that tells us that `chi2/ndof = 2.04/2` ; although you do not need to understand the details, this is a measure of how well the data are described by the line of best fit. Generally, the fit is assumed to be "good" if this ratio is close to 1 (the ratio is called "the reduced chi-squared"). The "correlation matrix" tells us how the best fit value of the slope is linked to the best fit value of the intercept, which you do not need to worry about here.

Since we expect the slope of the data to be k, this provides us a method to determine k from the data as $(0.61 \pm 0.13)\mathrm{sm}^{-\frac{1}{2}}$. **Performing a linear fit of the data is the best way to determine a constant of proportionality between the measurements**. Finally, we expect the intercept to be equal to zero according to our model. The best fit line from QExpy has an intercept of (-0.24 ± 0.22)s, which is slightly below, but consistent, with zero. From these data, we would conclude that the measurements are consistent with Chloe's Theory.

- **28.4: The QExpy python package for experimental physics** by Ryan D. Martin, Emma Neary, Joshua Rinaldo, and Olivia Woodman is licensed CC BY-SA 4.0. Original source: https://github.com/OSTP/PhysicsArtofModelling/blob/master/README.md.