QExpy: A python package for undergraduate laboratories

Ryan Martin, Queen’s University
12 June 2018
CAP 2018, Dalhousie University
Outline

• Motivation
• Error propagation in QExpy
• Plotting in QExpy
• Using QExpy
Motivation 1: Is it that useful for students to “manually” propagate errors?

• What do we want students learning in the lab, in terms of data analysis?
  – Is it useful for them to get very good at error propagation and taking partial derivatives?
  – Do you propagate errors “à la” undergrad in your own research?
• We had a lot of complaints about students “not enjoying” the labs, as they find it very time consuming to work on lab reports and analyse data out of the lab

→ Let’s take the tedious error propagation out of the labs and have the students think about the physics and the data instead!

From error propagation:
\[ \frac{e}{m} = (1.75 +/- 0.48) \times 10^{11} \text{ C/kg} \]
From Monte Carlo, using the mean and standard deviation
\[ \frac{e}{m} = (1.85 +/- 0.54) \times 10^{11} \text{ C/kg} \]

Using Monte Carlo method with mode and 68% confidence:
\[ \frac{e}{m} = (1.55 +/- 0.55) \times 10^{11} \text{ C/kg with 71.74% confidence} \]
Motivation 2: Have a “professor approved” computing package for error propagation

- We teach an error analysis and statistics course to 2nd year students in which we encourage them to use computers for calculations and teach them some python programming.
- After the first year teaching it, one of the students had developed a few python function for propagating errors which he shared with others.
- Decided to hire that student over a summer to develop a more polished and “professor approved” version as a python package.

→ QExpy: an open source python package that anyone can install
Error propagation in QExpy

- Designed to be easy to use and compatible with Jupyter Notebooks.
- Define variables of type “Measurement”
- Work with those variables, errors are propagated automatically.
- Can use most functions (e.g. trig, sqrt).
- Significant figures can be handled easily.
- Units handled somewhat (still in dev).
- Can interface with everything else that you can do in python, e.g. read files of data, etc.

In [3]:
```python
import qexpy as q

#Define two measurements, x and y
x = q.Measurement(5,1)  # 5 +/- 1
y = q.Measurement(10,0.2)  # 10 +/- 0.2

A quantity that depends on these
z = (x+y)/(x-y)

#Choose sig figs to show:
q.set_sigfigs(2)
print("z = ",z)
```

```
z = -3.00 +/- 0.80
```
Error propagation: implementation

• By default, errors are propagated using the "derivative method" (first order, exact derivatives):

\[
\sigma_F^2 = \left( \frac{\partial F}{\partial x} \sigma_x \right)^2 + \left( \frac{\partial F}{\partial y} \sigma_y \right)^2 + 2 \frac{\partial F}{\partial x} \frac{\partial F}{\partial y} \sigma_{xy}
\]

In the statistics class, they learn about this formula, its limitations, covariance, etc.

• QExpy also implements “Min-Max” and Monte-Carlo errors; second year students are using MC error analysis by the end of the year!

• Arrays of measurements (mean and std, error-weighted mean), numpy “under the hood”

• Visualize error contributions (still in dev)
In [11]:
# Define a plot object
fig1 = q.MakePlot(xdata = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
ydata = [0.3, 1.6, 4.5, 4.1, 4.6, 3.6, 6.1, 7.9, 8.7, 9.8],
yerr = 1.5,
xname = 'length', xunits='m',
yname = 'force', yunits='N',
data_name = 'xydata')

# Fit to a linear function
results = fig1.fit("linear")
# Add residuals sub plot
fig1.add_residuals()
fig1.show()

------------- Fit results ---------------
Fit of xydata to linear
Fit parameters:
xydata_linear_fit0_fitpars_intercept = -0.12 +/- 0.69,
xydata_linear_fit0_fitpars_slope = 0.95 +/- 0.11

Correlation matrix:
[[ 1.   -0.886]
 [ -0.886  1.  ]]

chi2/ndof = 3.62/7
------------- End fit results ---------------
Plotting and fitting

• Plotting through Bokeh (interactive) or matplotlib, can access backend and fine tune the look of the plots, defaults are reasonable in most cases.

• Fitting for polynomials and Gaussian included, users can also provide their own custom functions.

• Error bands are correct (MC errors that include correlations between parameters) → can use to plot any function where there are errors and correlations between parameters and it will draw the error bands!

```python
# define our model with 2 parameters:
def model(x, *pars):
    return pars[0] * np.sin(pars[1] * x)

# fit the model - we must provide a guess for the parameters
fig4.fit(model, parguess=[1,1], fitcolor="darkgoldenrod")
fig4.add_residuals()
fig4.show()
```
Use of QExpy at Queen’s

- Students using it in all years, engineering and A&S, through Jupyter Notebooks.
- In the error analysis and statistics class, students are not allowed to use QExpy; they are instead encouraged to use python to program their own error analysis routines.
- In the labs, they can use QExpy, since they should, in principle, understand what it is doing behind the scenes.
- Have used in our first year calculus-based physics course, as part of a way to introduce students to programming early on. Reasonably successful (this coming year, more python programming intro!).

![Jupyter Notebook example](image-url)
Me too!

• Install (requires python3):
  
  pip install qexpy

• Documentation (Google: qexpy read the docs):
  

• Examples (Google: qexpy github):
  
  https://github.com/Queens-Physics/qexpy/tree/master/examples/jupyter

• Contribute? (yes, please!)
  
  https://github.com/Queens-Physics/qexpy
Summary

• QExpy is an open source python package for data analysis in undergraduate physics labs (error propagation and plotting).
• QExpy was primarily developed by and for physics students.
• It is helping our students to think more about the physics and the data instead of being bogged down in error propagation.
• Serving as an intro to programming in python to our first year students.
• We have a supporting statistics course to ensure that students have a foundation in data analysis.

Acknowledgements:
• Department of Physics, Engineering Physics & Astronomy at Queen’s for their tremendous support
• A.B. McLean for the original inspiration and valuable discussions
• Undergraduates: Connor Kapahi, Turner Garrow for their great work developing the package!