Introduction to scientific computing

August 2018

In this assignment, we focus on coding practices more common in science, like numerical methods, calculating statistics, and vectorization. While `numpy` or `scipy` may have functions for many of these calculations, we ask you to create your own solution for practice and to develop your programming skills.

Exercise 1

Finite differences can be used to estimate the derivative (rate of change) of some data. Consider a sequence y_i. A first order finite difference is defined as,

$$\Delta y_i = y_{i+1} - y_i$$

We can use this to estimate the derivative as,

$$y'(x_i) \approx \frac{y_{i+1} - y_i}{h}$$

where h is the step size $(x_{i+1} - x_i)$.

Create a program that uses a for loop to estimate the rate of change of y defined as,

```
import numpy
h = .1
x = numpy.arange(10)*h
y = numpy.cos(x)
```

Hint: the final array will have one less element than y.

Exercise 2

The **numpy** module makes it easier and faster to do calculations on large arrays of data, using array arithmetic. Instead of using a for loop or list comprehension, mathematical calculations can be applied to each component of an array. For example, `array([1,2,3,4])**2` returns `array([1,4,9,16])`. Not only is it a faster computation, it's also much less typing! Use array arithmetic to estimate the derivative of y as defined in Question 1. *Hint: "slicing" the array as `y[:-1]` can give you the first N - 1 elements.*

Exercise 3

Consider swinging a sling in a horizontal circle overhead. The tension in the rope should satisfy

$$F_T = m\sqrt{g^2 + \left(\frac{v^2}{r}\right)^2}$$

Calculate the tension, and propogate the uncertainty using the following measurements. Assume that g is a constant with a value of 9.81 m/s^2.

Quantity	Symbol	Value	Uncertainty
mass	m	1.42	± 0.1kg
velocity	v	35.0	± 0.9m/s
radius	r	120	± 2.0cm

Print the result in Newtons. *Note: There may be uncertainty packages in Python that propagate the uncertainty efficiently. For this exercise you should write the uncertainty propagation yourself.*

Exercise 4

Often experimental data is stored in tables of observations. Numpy provides a function `numpy.loadtxt()` which reads data from a file into an array. See the documentation for details, like how to skip the first couple lines.

The file `sample-correlation.txt` has two sets of measurements that might be correlated. Make a script that loads data from the file, calculates the Pearson correlation coefficient (`numpy.corrcoef()`), and prints the correlation coefficient between the two variables. *'corrcoef' returns more information than you need for this question, select only the element that corresponds to the correlation coefficient.*

Exercise 5

A quick but poor way of calculating an integral is with a Riemann sum. Our sample code tries to calculate the left Riemann sum,

$$\int_a^b f(x)dx \approx \sum_{i=0}^{N-1} f(x_i)\Delta x,$$

of $cos(x)$ on $[0, \pi/2]$, but has errors. Find and correct the mistakes. *Note: N is the number of intervals, and $x_i = i\delta x$.*

```
import numpy as np
a, b = 0, np.pi/2
N = 50 # Number of intervals
dx = (a-b)/49
x = np.arange(a, b, N)
f = np.cos(x)
riemann_sum = np.sum(f * dx)
print(riemann_sum)
```

Exercise 6

Write a function, `average()` which takes an array, and returns the average of the values in the array. Test it by calculting the average of `numpy.arange(-5,5)`

Exercise 7

Create another function, `stdev()` which takes an array, and returns the mean and *sample* standard deviation of values in the array. That is,

```
mean_x , stdev_x = stdev(x)
```

You can reuse `average()` from the last question. Test the function again using `numpy.arange(10)`.

Exercise 8

Linear regression is simple enough that there is a formula for the estimated parameters. If the data satisfies

$$y = a + bx,$$

then the best parameters are,

$$\hat{b} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2},$$

and

$$\hat{a} = \overline{y} - \hat{b}\overline{x}$$

Write a script that loads the data in `sample-resistor.txt`, and estimates the parameters (resistance and voltage offset) using the above formulae and $\mathbf{V = IR}$.