Computational Physics Weekly Assessments
Week 1 Finite difference methods

In this assessment, you are set a small number of tasks to perform with regards to numerical differentiation. You will produce a single file to be submitted electronically via DUO before the assessment deadline. To assist you with this assessment the concept of functions is briefly revisited, an example of plotting data is given AND SOME SAMPLE CODE IS GIVEN AT THE END!!

Functions

A function is how a computer program encapsulates functionality and may be considered as the programming equivalent of a mathematical function – something is passed in, operated on and returned. For example, the code below defines a function called '`f`' that returns the second power of whatever numeric value is passed to it:

```
>>> def f(x):
     return x**2

>>> print f(10)
100
>>> print f(0.5)
0.25
>>>
```

A function can be applied to either a single number or an entire numpy array of numbers. For example we can create a numpy array of values between two numbers and then apply the function to all those values as follows:

```
>>> import numpy
>>> xs = numpy.arange(start=0, stop=2, step=0.2)
>>> print xs
[ 0.   0.2  0.4  0.6  0.8  1.   1.2  1.4  1.6  1.8]
>>> ys = f(xs)
>>> print ys
[ 0.    0.04  0.16  0.36  0.64  1.    1.44  1.96  2.56  3.24]
```

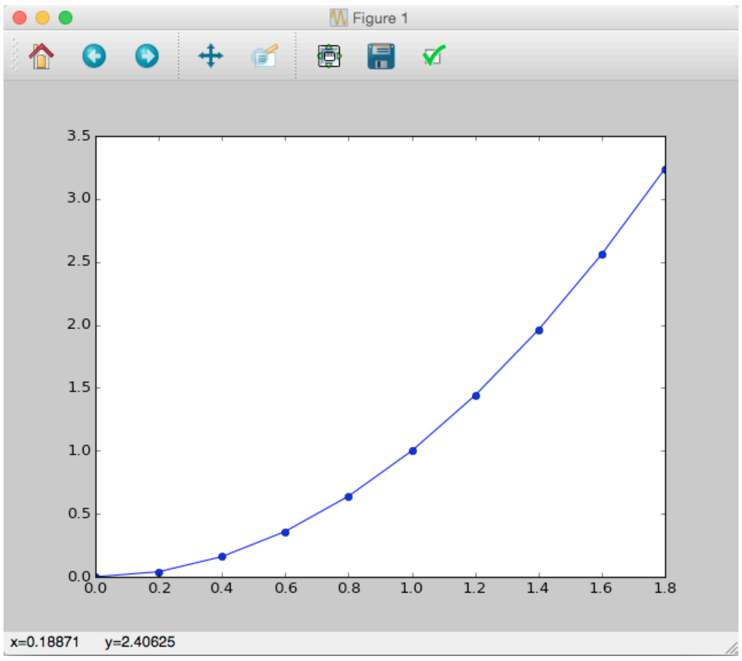
Note how the name '`xs`' has been used here instead of 'x' etc. This is a conscious choice to denote plurality, made for the purpose of clarity as a variable named 'x' is often assumed to store just a single number or ordinate.

Plotting data

Having generated a series of x-values and evaluated our function for all of them, it would be instructive to plot this data. ‘matplotlib’ is a plotting environment for Python installed on the CIS Windows system and included in the “Enthought Canopy” version of Python which is the recommended choice for use on personal machines. This can be done as follows:

```
>>> import matplotlib.pyplot as pyplot
>>> pyplot.figure()
<matplotlib.figure.Figure instance at 0x01883710>
>>> pyplot.plot(xs, ys, marker='o')
[<matplotlib.lines.Line2D instance at 0x01914148>]
>>> pyplot.show()
```

The final command results in the following figure being displayed:



However, as scientists, we recognise that a graph without context such as axes labels is devoid of meaning. The use of a few more commands results in a well-labelled graph as would be expected in, for example, a laboratory write-up or a weekly assessment submission.

Note the use of the US English spelling of colour, ‘color’ *by matplotlib. The inconsistent use of the spelling of ‘colour’ and ‘color’ between different packages is purely done to confuse you, the programmer.*

```
>>> pyplot.figure()
>>> pyplot.plot(xs, ys, color='red', label='f(x)')
>>> pyplot.xlabel('x')
>>> pyplot.ylabel('f(x)')
>>> pyplot.legend()
>>> pyplot.show()
```

Weekly assessment tasks:

1. Create a module, named 'cp_1.py'
2. Implement a function called 'f' that operates on a number or an array of numbers equivalent to the mathematical function: $f(x) = \cos(x)$
3. Create a function 'g_bdm(x, dx)' that uses the backwards difference method to estimate the derivative of f(x) with respect to x, using a step size of dx.
4. Using your function from (3) plot the **error** in the derivative of f(x) with respect to x, as calculated for several values of dx. The accuracy of the method depends upon the value of dx chosen, and you should plot three curves covering the cases where dx is too small, too large and well chosen.
5. Answer some questions about numerical differentiation.

Hints and details:

With regards to (2) above:

- This is similar to the code snippet shown earlier in this document. To ensure that the function you create works with arrays of numbers as well as individual numbers, you must use `numpy.cos(x)` in your function, not `math.cos(x)`.

With regards to (3) above:

- See the lecture notes on DUO for the backwards difference methods

With regards to (4) above

- Evaluate the functions at 100 evenly spaced points between -2*pi and 2*pi

```
xs = numpy.linspace(-2*numpy.pi, 2*numpy.pi, 100)
ys = f(xs)
pyplot.plot(xs, ys)
```

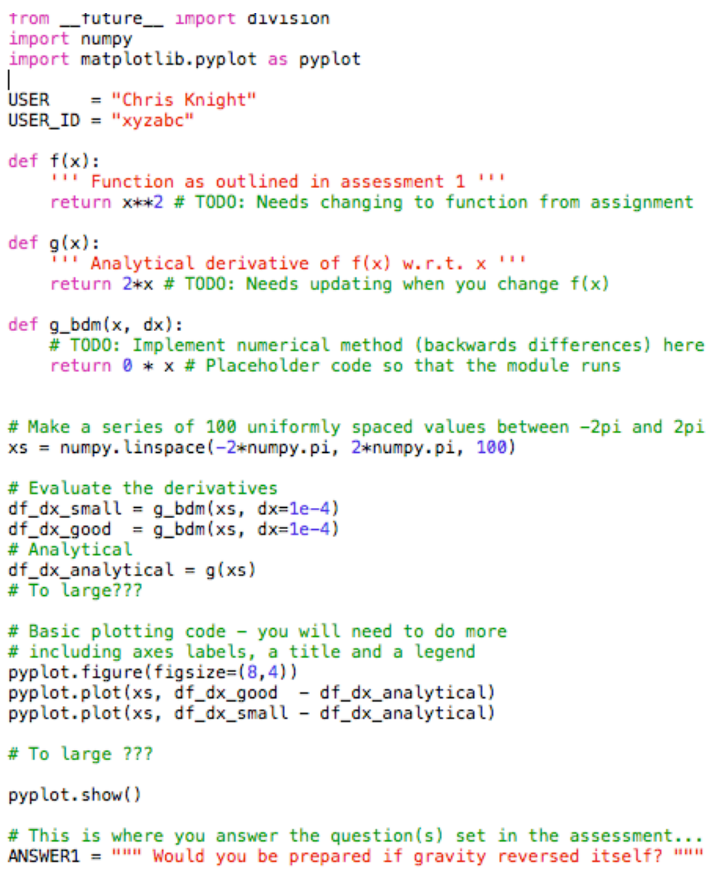
- To show the error, plot the difference of your derivate and the analytical derivative, that is `g_fdm(x, dx) - g(x)`, with the later being your function for the analytic derivative of the function.
- Use your program to experiment with different values of dx. Hint: Why can't you use an infinitely small value of dx?
- Recall the use of exponent notation for assigning values of dx. For example "dx=2e-15" is how we tell Python that $dx = 2\times10^{-15}$. Find values of dx where the method loses accuracy because its too big, and where it is too small.
- Don't forget to give each curve a label and to display a legend. Give your plot a title with `pyplot.title("???")`.
- Start your figure with the command '`pyplot.figure(figsize=(8,4))`' – this will ensure that your code produces a figure of the correct size and shape when electronically marked.

With regards to (5) above

- The question is "With reference to your figure, describe why the numerical method looses accuracy in the case where dx is too small and the case where dx is too large"
- Use your code to help find the answer! Change dx to different values and see what happens – compare the numerical and analytical solutions visually on your graph
- Answer the question in no more than about 40 words. Place the answer as a string variable in the global scope off your code (i.e. not in a function). Call the variable ANSWER1 – see the example below.

In general

- Place the code for parts (4) and (5) in the global scope after the function definitions.
- Place two variables at the start of the code,
    - USER="your name"
    - USER_ID = "your CIS login"
- Your module must run in order to be awarded any marks.
- Your solution should contain no more than about 30 lines of code and perhaps 12 lines of comments. Excessively long submissions will loose marks in the "code quality" section.
- A partial example is illustrated below, using the function $f(x) = x^2$ -obviously you need to expand on the code.



```python
from __future__ import division
import numpy
import matplotlib.pyplot as pyplot

USER    = "Chris Knight"
USER_ID = "xyzabc"

def f(x):
    ''' Function as outlined in assessment 1 '''
    return x**2 # TODO: Needs changing to function from assignment

def g(x):
    ''' Analytical derivative of f(x) w.r.t. x '''
    return 2*x # TODO: Needs updating when you change f(x)

def g_bdm(x, dx):
    # TODO: Implement numerical method (backwards differences) here
    return 0 * x # Placeholder code so that the module runs


# Make a series of 100 uniformly spaced values between -2pi and 2pi
xs = numpy.linspace(-2*numpy.pi, 2*numpy.pi, 100)

# Evaluate the derivatives
df_dx_small = g_bdm(xs, dx=1e-4)
df_dx_good  = g_bdm(xs, dx=1e-4)
# Analytical
df_dx_analytical = g(xs)
# To large???

# Basic plotting code - you will need to do more
# including axes labels, a title and a legend
pyplot.figure(figsize=(8,4))
pyplot.plot(xs, df_dx_good  - df_dx_analytical)
pyplot.plot(xs, df_dx_small - df_dx_analytical)

# To large ???

pyplot.show()

# This is where you answer the question(s) set in the assessment...
ANSWER1 = """ Would you be prepared if gravity reversed itself? """
```