PHYS30009: Intro to Computational Physics — Exercise 1

The deadline for this exercise is **Monday 17th October 2022** at 12:30 p.m. Your program (*.py) files should be uploaded into Blackboard at the appropriate point in the *PHYS_20032_30009_2022_TB-4: Introduction to Computational Physics (PHYS20032 + PHYS30009) 2022* course.

S. Hanna

Objectives of the exercise

This starter exercise is designed to be straightforward. The purpose is to make sure that you have fulfilled the following requirements needed for the rest of the course:

- Become familiar with programming in Python
- Understand how to enter and run simple Python programs (scripts) using the Spyder environment
- Understand the origins of numerical errors in computer programs
- Be able to produce output from a program and send data to the computer screen

In this first exercise, you will be required to submit a single program that address all of the points below from both problems. No report will be required.

Problem 1 (60% of marks): Series expansion of $\arctan(x)$

It is often necessary to calculate a logarithmic or trigonometric function in a computer program. All computer languages have built-in functions for this, which rely on series expansions. Here you will test the convergence of such an expansion by writing a Python function to evaluate the arc-tangent function, using the following Taylor series:

$$\arctan x = \sum_{n=0}^{N} \frac{(-1)^n}{2n+1} x^{2n+1} \quad \text{for} \quad |x| \leq 1 \qquad (1)$$

where $N \to \infty$. For $|x| > 1$, you can use the following relationships which will enable you to calculate $\arctan x$ for any real value of x:

$$\arctan(x) = \begin{cases} \frac{\pi}{2} - \arctan\left(\frac{1}{x}\right) & x > 0 \\ -\frac{\pi}{2} - \arctan\left(\frac{1}{x}\right) & x < 0 \end{cases} \qquad (2)$$

Note that you will need to combine the logic from Eqs. (1) and (2) in order to compute the correct value of the arctangent for each range of x values.
You should begin by copying the piece of code on the right of the page into a Python file – you will find it on Blackboard under Ex1. This is a simple menu program that allows the user to choose which part of the exercise to try. You should adapt this template to your needs.
The code works by continually looping and asking the user to enter a value, “a” to try part (a), “b” to try part (b), etc. and “q” to quit. Make sure you can run this program and that you understand how it works. Then, proceed as follows.

```
# put your "import" statements here

def MyArcTan(x,N):
    answer = 0 # initial value
    # put your function definition here and
        make sure you return the correct answer
    # use a loop to increment your answer N
        times
    answer = x # final value - correct this
    return answer

MyInput = '0'
while MyInput != 'q':
    MyInput = input('Enter a choice, "a", "b",
        "c" or "q" to quit: ')
    print('You entered the choice: ',MyInput)
    if MyInput == 'a':
        print('You have chosen part (a)')
        #
        # sample code for part (a); needs to be
            improved to get better marks
        #
        Input_x = input('Enter a value for x (
            floating point number): ')
        x = float(Input_x)
        Input_N = input('Enter a value for N (
            positive integer): ')
        N = int(Input_N)
        print('The answer is: ',MyArcTan(x,N))
    elif MyInput == 'b':
        print('You have chosen part (b)')
        #
        # put your code for part (b) here
        #
    elif MyInput != 'q':
        print('This is not a valid choice')
print('You have chosen to finish - goodbye.')
```

(a) In this first part, you are going to write a function to evaluate Eqs. (1) and (2). Begin by completing the missing function `MyArcTan()` in the template, by providing Python code to evaluate Eq. (1). Note:

- You will need to use a loop to accumulate your answer over N terms.
- Your function should accept x and N as arguments, and return the resulting value of the summed series. You can test your function by selecting option "a" in your test menu.
- Ensure your function works for $|x| \leq 1$.
- The code is set so that it asks the user to enter values of x and N, and prints $\arctan(x)$ to the screen. Think of ways you might improve on this.
- Check that your function produces a good approximation to $\arctan(x)$ for known points, e.g. $\arctan(1/\sqrt{3}) = \pi/6$ and $\arctan(1) = \pi/4$.
- Finally, extend your function to work for all x using Eq. (2). You will need a conditional statement in your function to choose the appropriate formula for each value of x.

(b) In this second part you are going to tabulate the values of `arctan(x)` for $-2 \leq x \leq 2$.

- For menu item "b", create a loop, allowing the variable x to vary in the range $-2 \leq x \leq 2$.
- Ask the user to enter a value of N and check it is valid (positive integer).
- For each x value, print the value returned by `MyArcTan()`, as well as the result from the built-in $\arctan()$ function in Python (either from the "math" or "numpy" modules), *and the difference between the two*.
- Test your program for different values of N and note how the accuracy varies with N, up to about $N = 20$, particularly close to $x = 1$. Make sure you print your output in a neatly formatted table with headings at the top of each column.

(c) Create a new menu item "c" to evaluate π to 7 significant figures, using the fact that $\arctan(1) = \pi/4$. You should use your `MyArcTan()` function for this. You will need to experiment with values of N in order to obtain this level of accuracy. Your code should print the result and compare it with the "correct" value of π.

(d) Your probably found that your code for part "c" runs quite slowly, making it impractical to use this method to evaluate π to higher precision. However, it is possible to obtain π to greater precision using the following unlikely looking identity:

$$\frac{\pi}{4} = \arctan\left(\frac{1}{2}\right) + \arctan\left(\frac{1}{5}\right) + \arctan\left(\frac{1}{8}\right) \tag{3}$$

In this final section of your program, create another menu item, and use Eq. (3) and your `MyArcTan()` function to evaluate π to 12 decimal places. In your program, set the smallest value of N you can find that achieves this. Again, your code should print the result and compare it with the "correct" value of π.

Problem 2 (40% of marks): Root-finding of a polynomial

While the roots of quadratic equations may be found by formula, higher-order polynomial equations cannot generally be solved in this way, and require some form of numerical technique. In Problem 2, you will test the convergence of the Newton-Raphson method for root finding. The Newton-Raphson approach relies on expanding the function $f(x)$ as a Taylor series, truncating at the first derivative, and rearranging to give:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{4}$$

Usually, each application of Eq. (4) yields an improved guess at the root, but we need to repeat the process iteratively until we have the required accuracy.

(e) Write a program to find the roots of the polynomial:

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 = 0$$

using the Newton-Raphson method. The 5 parameters, a_0 to a_4 should be entered by the user. The program should also request an initial guess from the user, x_0, and the accuracy required, δ, and should print the result. Your program should end when:

$$|x_{i+1} - x_i| \leq \delta$$

or when some maximum number of iterations (e.g. 100) is exceeded. You can either write your own function for this, or use the SciPy function `scipy.optimize.newton()` mentioned in lectures.

Merge this with your menu code from Problem 1 (this is because we require a single program to be uploaded to Blackboard).

(f) Now modify your code to add a loop that cycles over a range of initial values, x_0. You will need to specify the minimum and maximum values of x_0, as well as an increment. For each value of x_0, you should print the value of the root found, together with the number of iterations needed to obtain 6 significant figures.

(g) Test the operation of your program by verifying the roots of:

$$f(x) = x^4 + x^3 - 12x^2 - 2x + 10 = 0$$

to be:

$$\begin{aligned} -3.82305015 \\ -1.00000000 \\ 0.89259640 \\ 2.93045375 \end{aligned}$$

Can you identify initial guesses that fail to converge on a solution or that require excessively large numbers of iterations?

Submitting your work

No report is required for this exercise. Therefore you should submit just the final versions of your Python program (which should contain solutions to both problems with a single menu). Please note the following points:

- Blackboard plagiarism checker (Turnitin) won't accept a file ending ".py" so please rename your file with a ".txt" extension, or use the "cut-and-paste" submission option (if available).[1]
- Please also give your programs sensible distinguishing names, including your name or userid e.g. "my_userid_ex1.txt" or "myname_ex1.txt".

If you have any problems submitting your work, please contact Dr. Hanna (s.hanna@bristol.ac.uk) or ask a demonstrator.

[1] Strictly speaking, Turnitin can be instructed to accept files with the ".py" extension. However, it will only apply the automatic plagiarism checking if the ".txt' extension is used.