Assignment 4

Due Date: March 25, 8:59pm

Please see the guidelines at https://www.cs.toronto.edu/~lczhang/338/homework.html

What to Hand In

Please hand in 2 files:

- Python file containing all your code, named `csc338_a4.py`. If you are using Jupyter Notebook to complete the work, your notebook can be exported as a .py file (File -> Download As -> Python). Your code will be auto-graded using Python 3.6, so please make sure that your code runs. There will be a 20% penalty if you need a remark due to small issues that render your code untestable.
- PDF file named `csc338_a4_written.pdf` containing your solutions to the written parts of the assignment. Your solutions can be hand-written, but must be legible. Graders may deduct marks for illegible or poorly presented solutions.

Submit the assignment on **MarkUs** by 9pm on the due date. See the syllabus for the course policy regarding late assignments. All assignments must be done individually.

Q1. Interval Bisection [10 pt]

Part (a) [5 pt]

Write a function `bisect` that returns a list of intervals where the root of the function `f(x)` lies. Each interval should be half the size of the previous, and should be obtained using the interval bisection method.

Part (b) [2 pt]

In the `bisect` function, to obtain the midpoint between `a` and `b`, it is best to write

```
mid = a + (b - a) / 2
```

rather than

```
mid = (a + b) / 2
```

Why is the first computation better than the second? Provide an explanation in your PDF writeup.

Part (c) [3 pt]

Use the interval bisection method to find the root of the function

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

accurate to 8 significant decimal digits. The root of $f(x)$ is between -1 and 4.

Show your work in your python file, and store the root you find in the variable `bisection_root`.

Q2. Fixed-Point Iteration [20 pt]

Part (a) [4 pt]

Write a function `fixed_point` to find the fixed-point of a function `f` by repeated application of `f`. The function should return a list of values `[x, f(x), f(f(x)), ...]`.

Part (b) [8 pt]

To find a root of the equation

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

we can consider fixed-point problems involving the following different functions :

1. $g_1(x) = \frac{x^2+2}{3}$
2. $g_2(x) = \sqrt{3x-2}$
3. $g_3(x) = 3 - \frac{2}{x}$
4. $g_4(x) = \frac{x^2-2}{2x-3}$

Analyze the convergence properties of each of the corresponding fixed-point iteration schemes for the root x = 2 by analyzing $g_i'(x)$. Do you expect the fix-point iteration to diverge or converge? What is the rate of convergence?

This question should be done by hand. Submit your solution in your PDF writeup.

Part (c) [8 pt]

Confirm your analysis in Part (b) by using the `fixed_point` function to generate the fixed-point iterations. Verify the convergence of g_i, or lack thereof. Choose the starting value of x to be $x_0 = 3$.

Analyze the numerical results (outputs of the `fixed_point`) function and approximate the **observed** convergence rate.

Submit your solution in your PDF writeup.

Q3. Newton's Method [25 pt]

Part (a) [5 pt]

Write a function `newton` to find a root of `f(x)` using Newton's Method. This Python function should take as argument both the mathematical function `f` and its derivative `df`, and return a list of successively better estimates of a root of `f` obtained from applying Newton's method.

Part (b) [2 pt]

Use your function from part (a) to solve for a root of

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

Start with $x_0 = 3$, and stop when the root is accurate to at least 8 significant decimal digits.

Show your work in your python file, and store the root you find in the variable `newton_root`.

Part (c) [6 pt]

Consider the following non-linear equations $h_i(x) = 0$.

1. $h_1(x) = x^3 - 5x^2 + 8x - 4$
2. $h_2(x) = xcos(20x) - x$
3. $h_3(x) = e^{-2x} + e^x - x - 4$

Write out the statement for updating the iterate x_k using Newton's method for solving each of the equations $h_i(x) = 0$.

Include your solution in your PDF writeup.

Part (d) [3 pt]

Plot each of the three functions, showing the roots of each function (if any). Include your plots in your PDF writeup.

Part (e) [3 pt]

Use the `newton` function to try and solve $h_i(x) = 0$, for `n = 100` iterations, starting with `x = 1.5`.

Save the return values of calls to the function `newton` to the variables `newton_h1`, `newton_h2`, `newton_h3`,

Part (f) [6 pt]

Explain why Newton's method either does not converge or converges slowly for each of the functions h_1, h_2 and h_3.

Include your explanations in your PDF writeup.

Q4. Secant Method [17 pt]

Part (a) [6 pt]

Write a function `secant` to find a root of `f(x)` using the secant method. The function should return a list of successively better estimates of a root of `f` obtained from applying the secant method.

Part (b) [3 pt]

Use the `secant` function to find a root of $f(x) = x^3 + x^2 + x - 4$, accurate up to 8 significant decimal digits.

Show your work in your Python file. Save the result in the variable `secant_root`.

Part (c) [4 pt]

Show that the iterative method

$$x_{k+1} = \frac{x_{k-1}f(x_k) - x_k f(x_{k-1})}{f(x_k) - f(x_{k-1})}$$

is mathematically equivalent to the secant method for solving a scalar nonlinear equation $f(x) = 0$.

Include your solution in your PDF writeup.

Part (d) [4 pt]

When implemented in finite-precision floating-point arithmetic, what advantages or disadvantages does the formula given in part (c) have compared with the formula for the secant method given in lecture?

This is the formula given in lecture:

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}$$

Include your solution in your PDF writeup.

Q5. Conditioning [8 pt]

Consider the problem of finding the roots of the functions $f_1, \dots\ f_4$. What is the condition number of each problem?

Save the condition numbers in the variables `cond_f_1` to `cond_f_4`. If a root is not provided, you can find it using any method you wish, including by hand. Your condition numbers should be accurate up to 3 significant decimal digits.

1. $f_1(x) = x^3 + x^2 + x - 4$
2. $f_2(x) = x^3 - 5x^2 + 8x - 4$, at $x = 2$
3. $f_3(x) = xcos(20x) - x$, at $x = 0$
4. $f_4(x) = x^2 - 4sin(x)$, at $x = 0$

If the condition number is infinite, store a very large number or a non-numerical value, but do not use the value `None`, so I know that you attempted the problem.