CSC338. Homework 1

Due Date: Wednesday January 15, 9pm

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 `hw1.py`.
- PDF file named `hw1_written.pdf` containing your solutions to the written parts of the assignment. Your solution can be hand-written, but must be legible. Graders may deduct marks for illegible or poorly presented solutions.

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 renders your code untestable.

Make sure to remove or comment out all matplotlib or other expensive code before submitting your code!

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.

```
import math
import numpy as np
```

Question 1.

For parts (a) and (b), consider the problem of evaluating the function $g(x) = x^2 + x - 4$. Suppose there is a small error, h in the value of x.

Part (a) – 6pt

What is the absolute error and relative error in computing $g(x)$? Implement function and `g_abs_err(x, h)` and `g_rel_err(x, h)` to compute those quantities.

```
def g_abs_err(x, h):
    """Returns the absolute error of computing `g` at `x` if `x` is
    perturbed by a small value `h`.
    """
    return None

def g_rel_err(x, h):
    """Returns the relative error of computing `g` at `x` if `x` is
    perturbed by a small value `h`.
    """
    return None
```

Part (b) – 3pt

Estimate the condition number for the problem. Simplify this answer. For what values of x is this problem well-conditioned? Include your solution in your PDF file.

Part (c) – 2pt

For parts (c) and (d), consider the problem of finding a root of the function $g(x) = x^2 + x + c$ by using the Quadratic Formula and taking the positive squareroot. Suppose there is a small error, h in the value of c.

For what values of c is this problem well-posed? Include your solution in your PDF file.

Part (d) – 6pt

What is the absolute error and relative error in computing $g(x)$? Implement function and `g_root_abs_err(c, h)` and `g_root_rel_err(c, h)` to compute those quantities.

```
def g_root_abs_err(c, h):
    """Returns the absolute error of finding the (most) positive root of `g` when
    `c` is perturbed by a small value `h`.
    """
    return None

def g_root_rel_err(c, h):
    """Returns the relative error of finding the (most) positive root of `g` when
    `c` is perturbed by a small value `h`.
    """
    return None
```

Question 2.

Consider the function, which is also implemented below. You can run the `plot_f` python function to see what the graph of f looks like.

$f(x) = \frac{x - sin(x)}{x^3}$

```
def f(x):
    return (x - math.sin(x)) / math.pow(x, 3)

def plot_f():
    import matplotlib.pyplot as plt
    xs = [x for x in np.arange(-3.0, 3.0, 0.05) if abs(x) > 0.05]
    ys = [f(x) for x in xs]
    plt.plot(xs, ys, 'bo')

# plot_f() # please comment this out before submitting, or your code might be untestable
```

Part (a) – 2pt

What is $f(0.00000001)$? Save the results in the variable `q2_est`.

Given that f is continuous except at $x = 0$, what should $f(0.00000001)$ be? Save the results in the variable `q2_true`.

```
q2_est = None
q2_true = None
```

Part (b) – 2pt

Why does the Python statement compute such inaccurate values of $f(0.00000001)$? Include your answer in your PDF File.

Part (c) – 5pt

Define a Python function `f2(x)` that uses a different algorithm to compute more accurate values of $f(x)$ for $0 < x \leq \frac{\pi}{2}$. More specifically, the relative error should be no more than 1% for those values of x.

```
def f2(x):
    return None
```

Question 3. – 4pt

The sine function is given by the infinite series

$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + ...$

Compute the absolute forward and backwards error if we approximate the sine function by the first two terms of the series for elements of the array `xs` below.

Save your solution in the array `q3_forward` and `q3_backward`, so that `q3_forward[i]` and `q3_backward[i]` are the absolute forward and backward errors corresponding to the input `xs[i]`.

You may find the functions `math.sin` and `math.asin` helpful. You may assume that the true value of $\sin(x)$ can be computed using the function `math.sin`.

Update (Jan 9th): If the backward error does not exist, please enter "DNE".

```
xs = [0.1, 0.5, 1.0, 3.0]

q3_forward  = [None, None, None, None]
q3_backward = [None, None, None, None]

# math.sin(0.2)
# math.asin(0.2) # arcsin
```