Math 151A HW 1

This is your first homework. It is due on Wednesday, Jan 22nd 11:59pm. For **problems 7**, you must finish them according to the requirement in the syllabus.For the other problems, please do not include them in the submission, but please do all the problems. For problems labeled **Computational exercise**, you must write code for them.

1. Using a calculator of your choice, determine both the absolute and relative errors in the approximation of p^* by p. Please report all errors with 5 digits of accuracy.

   (a) $p^* = \sqrt{34}$, $p = 5.8$

$$e_{\text{abs}} \approx 3.0952 \times 10^{-2}, \quad e_{\text{rel}} \approx 5.3082 \times 10^{-3}$$

   (b) $p^* = 9!$, $p = \sqrt{18\pi}\left(\frac{9}{e}\right)^9$

$$e_{\text{abs}} \approx 3343.1, \quad e_{\text{rel}} \approx 9.2128 \times 10^{-3}$$

2. Find the third order Taylor polynomial $P_3(x)$ (using the term up to x^3) for the function $f(x) = \sqrt{x+1}$ about $x_0 = 0$. Approximate $\sqrt{0.5}$, $\sqrt{0.75}$, $\sqrt{1.25}$, and $\sqrt{1.5}$ using $P_3(x)$, and compute the absolute errors with 3 digits of accuracy.

$$P_3(x) = 1 + \frac{1}{2}x - \frac{1}{8}x^2 + \frac{1}{16}x^3$$

   The absolute errors $|f(x) - P_3(x)|$ for $x = -1/2$, $x = -1/4$, $x = 1/4$, and $x = 1/2$ are:

$$3.83 \times 10^{-3},\ 1.86 \times 10^{-4},\ 1.30 \times 10^{-4},\ 1.82 \times 10^{-3}, \text{ respectively.}$$

3. Use the IVT to show that there is a solution to the equation

$$g(x) = x\cos(x) - 2x^2 + 3x - 1 = 0$$

   on the interval $[0.2, 0.3]$.

$$g(0.2) = -0.2840 < 0 \quad \text{and} \quad g(0.3) = 0.0066 > 0$$

   Since $g(x)$ is continuous on $[0.2, 0.3]$, the IVT guarantees a root in this interval.

4. Let $f(x) = 1.01e^{4x} - 4.62e^{3x} - 3.11e^{2x} + 12.2e^{x} - 1.99$.

(a) Use three-digit rounding arithmetic and the fact that $e^{1.53} = 4.62$ to evaluate $f(1.53)$ **directly** using only addition, subtraction, and multiplication. (Hint: $e^{2x} = e^x \cdot e^x$.) Please ensure that you do rounding after each operation.

First, $4.62 \times 4.62 = 21.3$, $4.62 \times 21.3 = 98.4$, $4.62 \times 98.4 = 455$.

$$\begin{aligned} & 1.01 \times 4.62^4 - 4.62 \times 4.62^3 - 3.11 \times 4.62^2 + 12.2 \times 4.62 - 1.99 \\ = {} & 1.01 \times 455 - 4.62 \times 98.4 - 3.11 \times 21.3 + 12.2 \times 4.62 - 1.99 \\ = {} & 460 - 455 - 66.2 + 56.4 - 1.99 = -6.79 \end{aligned}$$

(b) Rewrite $f(x)$ in nested form, as done in class.

$$f(x) = (((1.01e^x - 4.62)e^x - 3.11)e^x + 12.2)e^x - 1.99$$

(c) Redo the calculation in part (a) using the nested form from part (b).

$$\begin{aligned} & (((1.01 \times 4.62 - 4.62) \times 4.62 - 3.11) \times 4.62 + 12.2) \times 4.62 - 1.99 \\ = {} & ((0.05 \times 4.62 - 3.11) \times 4.62 + 12.2) \times 4.62 - 1.99 \\ = {} & (-2.88 \times 4.62 + 12.2) \times 4.62 - 1.99 = -1.1 \times 4.62 - 1.99 = -7.07 \end{aligned}$$

(d) Compare the results in parts (a) and (c) to the true three-digit value $f(1.53) = -7.61$. Which method is more accurate? Report the absolute or relative error.

Absolute error in (a):

$$|-6.79 - (-7.61)| = 0.82$$

Absolute error in (c):

$$|-7.07 - (-7.61)| = 0.54$$

The nested method (c) is more accurate.

5. (a) Show that the sequence $p_n = \left(\frac{1}{10}\right)^n$ converges linearly to $p = 0$.

Firstly, we can easily verify that $p_n \to 0$. By definition, compute with $\alpha = 1$:

$$\lim_{n\to\infty} \frac{|p_{n+1} - 0|}{|p_n - 0|} = \frac{1}{10} =: \lambda \in (0, \infty)$$

(b) Show that the sequence $p_n = 10^{-2^n}$ converges quadratically to $p = 0$.

Firstly, we can easily verify that $pn \to 0$. Using the definition of quadratic convergence with $\alpha = 2$:

$$\lim_{n\to\infty} \frac{|p_{n+1} - 0|}{|p_n - 0|^2} = \frac{10^{-2^{n+1}}}{(10^{-2^n})^2} = 1 =: \lambda \in (0, \infty)$$

6. Suppose a numerical method at the k-th step produces an approximation x_k for the root of some continuous function f in the interval $[a_k, b_k]$. Find the root (the zero) of the line that passes through the points $(a_k, f(a_k))$ and $(b_k, f(b_k))$. (Hint: the equation of the line passing through the two given points can be found with 'point-slope' form from precalculus.) (Note: the solution to this problem will be useful in problem [7].)

   The line is
   $$\frac{y - f(b_k)}{x - b_k} = \frac{f(a_k) - f(b_k)}{a_k - b_k}.$$
   Set $y = 0$, we have
   $$x = b_k - \frac{f(b_k)(a_k - b_k)}{f(a_k) - f(b_k)}.$$
   or
   $$x = a_k - \frac{f(a_k)(b_k - a_k)}{f(b_k) - f(a_k)}.$$

7. **Computational Exercise**

   Numerical analysts often need to learn new computational techniques and adapt existing programs to implement these methods for comparison. This task typically involves understanding the underlying concept of a new technique.

   In this exercise, you will follow this process to implement an improved variant of the bisection method called the Regula Falsi (False Position) method. Unlike the Bisection Method, which takes the approximate root p_k as the midpoint of the interval $[a_k, b_k]$, the Regula Falsi method approximates p_k as the zero of the line passing through the points $(a_k, f(a_k))$ and $(b_k, f(b_k))$.

   (a) Compare the Regula Falsi method to the Bisection Method by using both to find the root of $(x-1)(x-2)(x-3)$ with a starting interval of $[1.75, 2.95]$. Stop the iteration when the residual is less than 10^{-6}.

      i. How many iterations does each method take?

      ii. What is the approximate root for each method?

      The results for the Bisection Method are:

      $$\text{Approximate root of } (x-1)(x-2)(x-3) \text{ is } 2.0000007629$$

      Iterations = 19

      In contrast, the Regula Falsi method produces:

      $$\text{Approximate root of } (x-1)(x-2)(x-3) \text{ is } 1.9999999998$$

      Iterations = 5

   (b) Repeat the comparison for the function $x^6 - x - 1$ using initial intervals of $[1.0, 1.2]$ and $[1.0, 2.0]$. Stop the iteration when the residual is less than 10^{-6}.

      i. How many iterations does each method take?

      ii. What is the approximate root for each method?

      iii. Is the Regula Falsi method always better than the Bisection Method?

      For the function $f(x) = x^6 - x - 1$, the Bisection Method produces:

      $[1.0, 1.2]$:

      $$\text{Approximate root is } 1.1347240448$$

      Iterations = 19

      $[1.0, 2.0]$:

      $$\text{Approximate root is } 1.1347241401$$

Iterations = 21

The Regula Falsi method performs better with the smaller interval $[1.0, 1.2]$:

$$\text{Approximate root is } 1.1347241059$$

Iterations = 8

However, with the larger interval $[1.0, 2.0]$:

$$\text{Approximate root is } 1.1347240531$$

Iterations = 92

The Regula Falsi method is more efficient when the initial interval is close to the root but can perform worse than the Bisection Method when the interval is too large.

```
import numpy as np

# default precision float64

def f(x):
    return (x-1)*(x-2)*(x-3)

def bisection(f, a, b, tol):
    # f(a) and f(b) must have opposite signs
    # tol is the tolerance
    if f(a)*f(b) > 0:
        raise ValueError("f(a) and f(b) must have opposite signs")
    num_iter = 0
    while True:
        c = (a+b)/2
        num_iter += 1
        if np.abs(f(c)) <= tol:
            # return the root and the number of iterations
            return c, num_iter
        if f(a)*f(c) < 0:
            b = c
        else:
            a = c

def RegulaFalsi(f, a, b, tol):
    # f(a) and f(b) must have opposite signs
    # tol is the tolerance
    if f(a)*f(b) > 0:
        raise ValueError("f(a) and f(b) must have opposite signs")
    num_iter = 0
    while True:
        c = (a*f(b) - b*f(a)) / (f(b) - f(a))
        num_iter += 1
        if np.abs(f(c)) <= tol:
            # return the root and the number of iterations
            return c, num_iter
        if f(a)*f(c) < 0:
            b = c
        else:
            a = c
```

```
print(bisection(f, 1.75, 2.95, 1e-6))
print(RegulaFalsi(f, 1.75, 2.95, 1e-6))

def f2(x):
    return x**6 - x - 1

print(bisection(f2, 1.0, 1.2, 1e-6))
print(bisection(f2, 1.0, 2.0, 1e-6))
print(RegulaFalsi(f2, 1.0, 1.2, 1e-6))
print(RegulaFalsi(f2, 1.0, 2.0, 1e-6))
```