

AERO-222: Introduction to Aerospace Computation, Spring 2023
Homework #1, Due Date: Thursday, February 9, 2023

Show all work and justify your answers!

Instructions

- *This homework contains both handwritten and coding problems and shall be submitted according to the following guidelines.*
- *Hardcopy:*
 - *Due on CANVAS at 11:59 PM on the day of the deadline.*
 - *Shall include screenshots of any hand-written work.*
 - *For coding problems, the hardcopy shall include any relevant derivations and emphasize the final results (i.e. boxed, highlighted, etc.).*
 - *Shall be submitted as a single file according to the provided template with the following naming scheme: “LastnameHW#.pdf”*
 - *If preferable, you can put all of your work into a single Jupyter notebook (.ipynb) with photos of your hand-written work as well. Markdown allows for images.*
- *Coding Submission:*
 - *Due on CANVAS at 11:59 PM on the day of the deadline.*
 - *Shall be submitted as a single file according to the provided template with the following naming scheme: “LastnameHW#.py” or “LastNameHW#.ipynb”.*
 - *The script shall print out all outputs asked for in the problem.*
- *Late submissions will be accepted with a 10 point deduction per day late.*

-
- 1. Root-finding Algorithms (Coding Problem) (15 pts)** Calculate the roots (to an absolute error $\varepsilon_x < 1e - 08$) of

$$f(x) = 3x^2 \sin x - x \cos x + 4 = 0$$

Use the following 3 methods:

- Bisection method
- Secant method
- Regula-Falsi method

These methods require a set of starting points/bounds. Use $x_1 = -2$ and $x_2 = +2$, which will meet the initial requirements for all 3 methods. For each method, calculate and plot $f(x_n)$ as a function of the iteration number n . Which method converges fastest?

Solution:

The Secant Method converges the fastest.

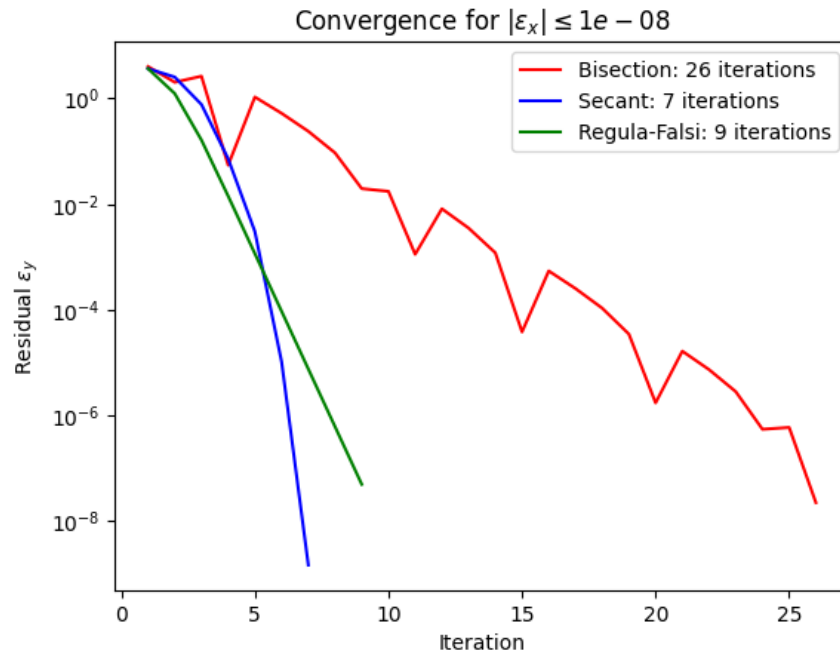


Figure 1: The solution should look very similar to this, all methods should converge to zero

2. Round-off Error (Coding Problem) (10 pts) Write a script to:

- (a) Evaluate the cubic polynomial, $f(x)$, at $x = 1.32$, using default machine precision and again using only three significant digits at each arithmetic operation. Calculate the absolute and relative error of the final result:

$$f(x) = 3.15x^3 - 2.11x^2 - 4.01x + 10.33$$

- (b) Repeat part (a) but do it with *nested multiplication*. Compare errors. Which takes fewer operations? The nested form is:

$$f(x) = [(3.15x - 2.11)x - 4.01]x + 10.33$$

Solution:

- (a) With default machine precision:

$$3.15 \times 1.32^3 - 2.11 \times 1.32^2 - 4.01 \times 1.32 + 10.33 = 8.82120625$$

With rounding to 3 significant digits in between operations (powers are treated as 1 operation):

$$3.15 \times 1.32^3 + 2.11 \times 1.32^2 - 4.01 \times 1.32 + 10.33 = 8.822$$

$$\text{Absolute Error} = 7.9375 \times 10^{-4}$$

$$\text{Relative Error} = 8.9982 \times 10^{-5}$$

$$\text{Number of Steps} = 9$$

(b) With default machine precision:

$$[(3.15x - 2.11)x - 4.01]x + 10.33 = 8.82120625$$

With rounding to 3 significant digits in between operations:

$$[(3.15x - 2.11)x - 4.01]x + 10.33 = 8.822$$

$$\text{Absolute Error} = 7.9375 \times 10^{-4}$$

$$\text{Relative Error} = 8.9982 \times 10^{-5}$$

$$\text{Number of Steps} = 6$$

In this case, the errors are the same because both results round to the same value for 3 significant digits. The first approach is in general more accurate but requires more operations. If any very small values show different results (smaller than 10^{-10}), its due to machine/python round-off errors.

3. Truncation Error (Coding Problem) (20 pts) Evaluate $f(x) = e^{-5}$ to five digits of precision using the following two approaches:

$$(a) \quad e^{-5} = \sum_{k=0}^{10} \frac{(-5)^k}{k!} = \sum_{k=0}^{10} \frac{(-1)^k 5^k}{k!}$$

$$(b) \quad e^{-5} = \frac{1}{e^5} \approx \frac{1}{\sum_{k=0}^{10} \frac{(5)^k}{k!}}$$

Note that the true value to five digits of precision is 1.8316×10^{-2} . Which formula gives more accurate results and why? Plot the error of each approach as a function of iteration number.

Solution:

$$(a) \quad 1 - 5 + 12.5 - 20.83 + 26.04 - 26.04 + 21.70 - 15.50 + 9.69 - 5.38 + 2.69 = 8.6404 \times 10^{-1}$$

$$(b) \quad 1/(1 + 5 + 12.5 + 20.83 + 26.04 + 26.04 + 21.70 + 15.50 + 9.69 + 5.38 + 2.69) = 6.8315 \times 10^{-3}$$

The actual value to 4 digits of precision is 6.7379×10^{-3} . The second method is more accurate because it does not oscillate about the value, but instead always stays smaller than 1 and larger than 0. Thus, the truncation error of the second method is lower for the same number of steps.

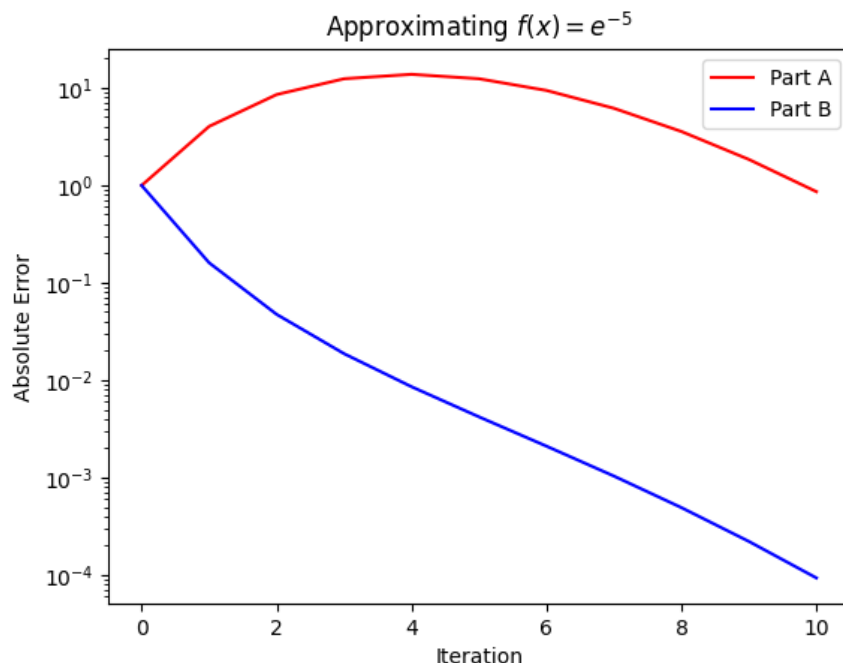


Figure 2: Method 2 is more accurate

4. **Taylor Series (15 pts)** Expand the function, $f(x) = x^4 + \sin x$, by Taylor series up to degree 3 for the following cases:

- (a) centered at $x_0 = 4$ and evaluated at $x_1 = x_0 + 0.2$;
- (b) centered at $x_0 = 3$ and evaluated at $x_1 = x_0 - 0.7$.

Solution:

First write the general form of the Taylor polynomial for clarity:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)(x - x_0)^2}{2!} + \frac{f'''(x_0)(x - x_0)^3}{3!}$$

Then, insert the given equation:

$$f(x) \approx x_0^4 + \sin x_0 + (4x_0^3 + \cos x_0)(x - x_0) + \frac{(12x_0^2 - \sin x_0)(x - x_0)^2}{2!} + \frac{(24x_0 - \cos x_0)(x - x_0)^3}{3!}$$

Insert given values:

- a. centered $x_0 = 4$ and evaluated $x_0 + 0.2 = 4.2$

$$f(x) \approx 4^4 + \sin 4 + (4(4)^3 + \cos 4)(0.2) + \frac{(12(4)^2 - \sin 4)(0.2)^2}{2!} + \frac{(24(4) - \cos 4)(0.2)^3}{3!}$$

answer: 310.296

b. centered $x_0 = 3$ and evaluated $x_0 - 0.7 = 2.3$

$$f(x) \approx 3^4 + \sin 3 + (4(3)^3 + \cos 3)(-0.7) + \frac{(12(3)^2 - \sin 3)(-0.7)^2}{2!} + \frac{(24(3) - \cos 3)(-0.7)^3}{3!}$$

answer: 28.4869

5. Variables and Computer Precision (10 pts) Answer the following questions:

- (a) Which can store a larger number, a signed int or unsigned int?
- (b) Which uses more memory, a float or double? Which is more precise?
- (c) If I'm trying to establish if two integers are equal, what's the easiest way to compare their values in code? Write the statement that would achieve this?
- (d) If I'm trying to establish if two real numbers (double or float) are equal, what's the "correct" way to compare their values in code? Write the statement that would achieve this.
- (e) What is Python default machine precision?

Solution:

- (a) unsigned int
- (b) double, double
- (c) `a == b`
- (d) `|a - b| < ϵ`
- (e) Double (64 bits or I will also accept 53 bit if they are talking about the mantissa),
9223372036854775807, (`sys.maxsize`) 1.7976931348623157e+308 (`sys.float_info.max`)
[10e+4931 with numpy]

6. Base Conversion (10 pts) Show all steps:

- (a) Convert 1482 from decimal to binary.
- (b) Convert 0.1 from decimal to binary using 1 byte (8 bits).
- (c) Does adding zeros to the back (ones side) of a base 10 integer change the value?
What about a base 2 (binary) integer?
- (d) Now consider the value to be a decimal. Does anything change?

Solution:

- (a) 10111001010
- (b) 0.00011001
- (c) Yes, yes
- (d) No (the number is unchanged)