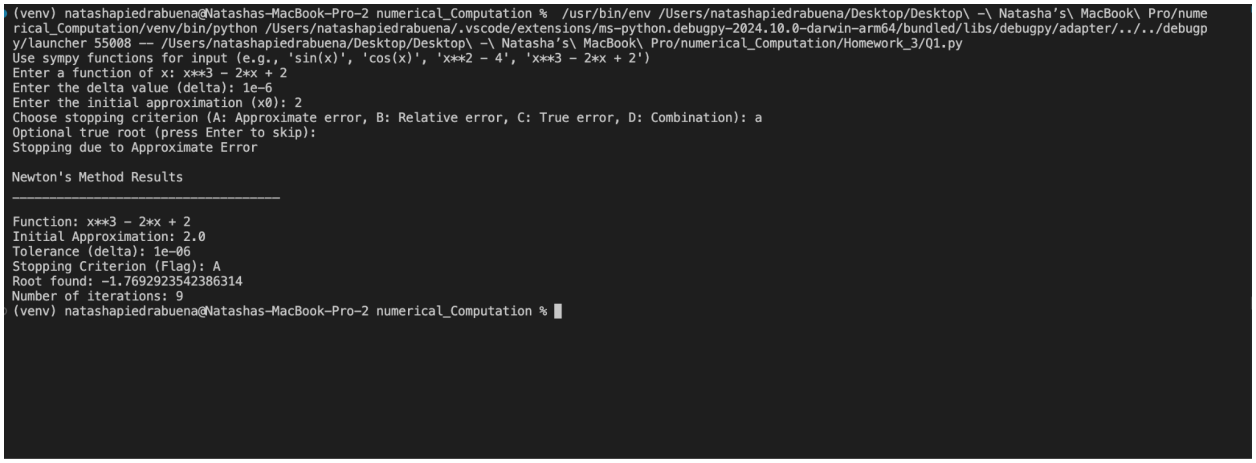
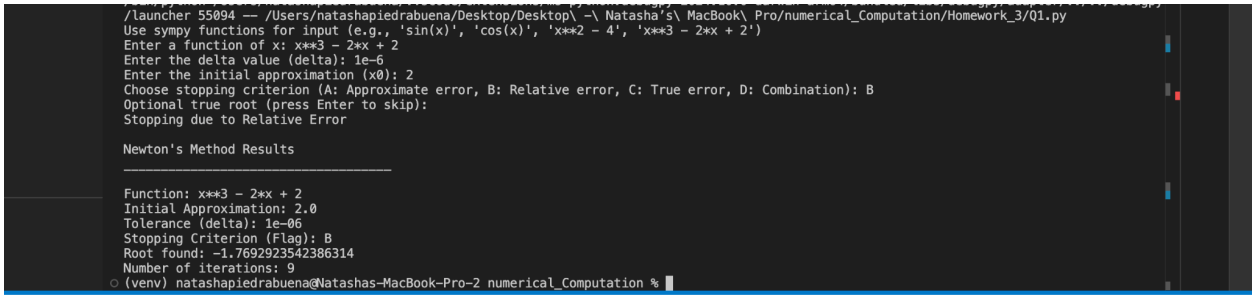
Natasha Piedrabuena
Numerical Computation
Newton's Method

In this report, Newton's Method, a widely used root-finding algorithm, was implemented to solve nonlinear equations. This method involves an iterative process that refines an initial guess for the root by using the function's derivative. The user is able to specify stopping criteria based on four different options: [Absolute Approximate Error, Relative Approximate Error, True Absolute Error, Combination of Absolute and True Absolute Error]. The aim of this report is to present the results from applying Newton's Method to a specific nonlinear equation, compare the performance of the different stopping criteria, and evaluate the effectiveness of AI-generated code versus manually written code. This will provide insights into how code generated by AI differs from code written by a human. In this project I utilized python and the function parameters (initial approximation, tolerance, derivative, function, and stopping criteria flag. The output should return to the main function with the root and number of iterations.

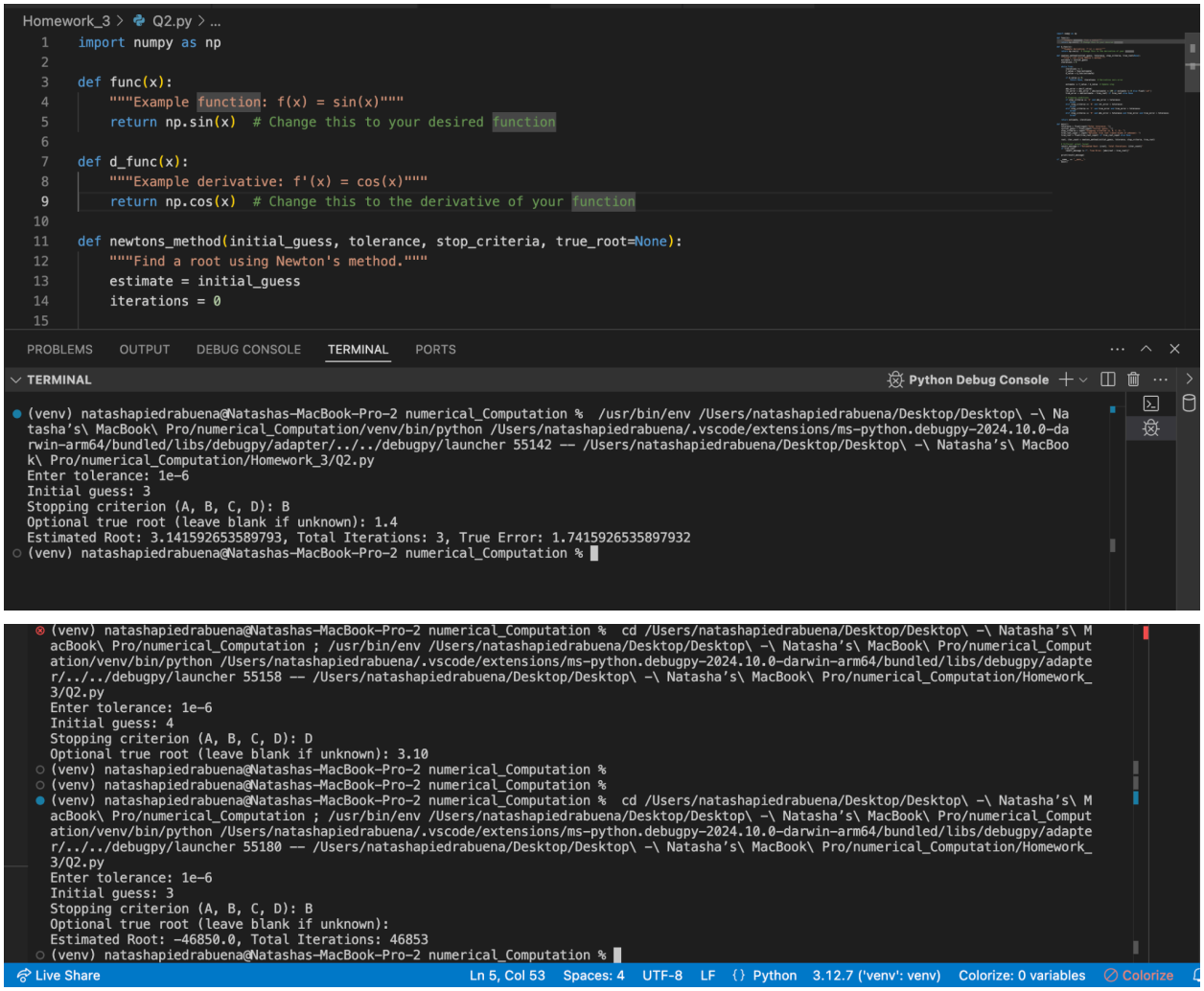
Q1) Newton's Method
Newton's Method iteratively refines the root estimate by using the formula:
To start off, the user provides the function, and the derivative is then calculated using sympy library, the tolerance (delta), an initial guess x0, and optionally the true root (if available). The algorithm then proceeds by iterating the Newton's method formula, adjusting the approximation of the root at each step until the stopping criterion is satisfied.





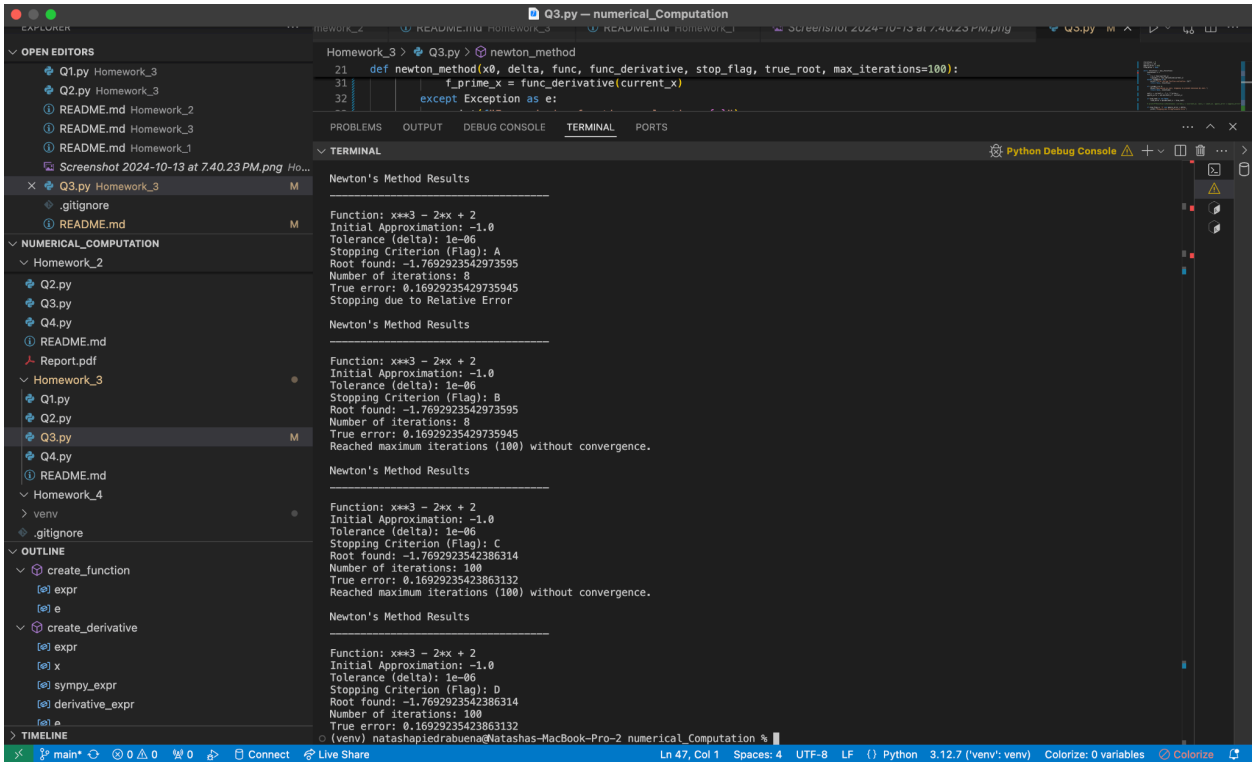
Q2) For this question, I asked chatgpt to solve the problem from Q1. To build a newton method that follows all the criteria. To start off the implementation seems fine except the issue with user

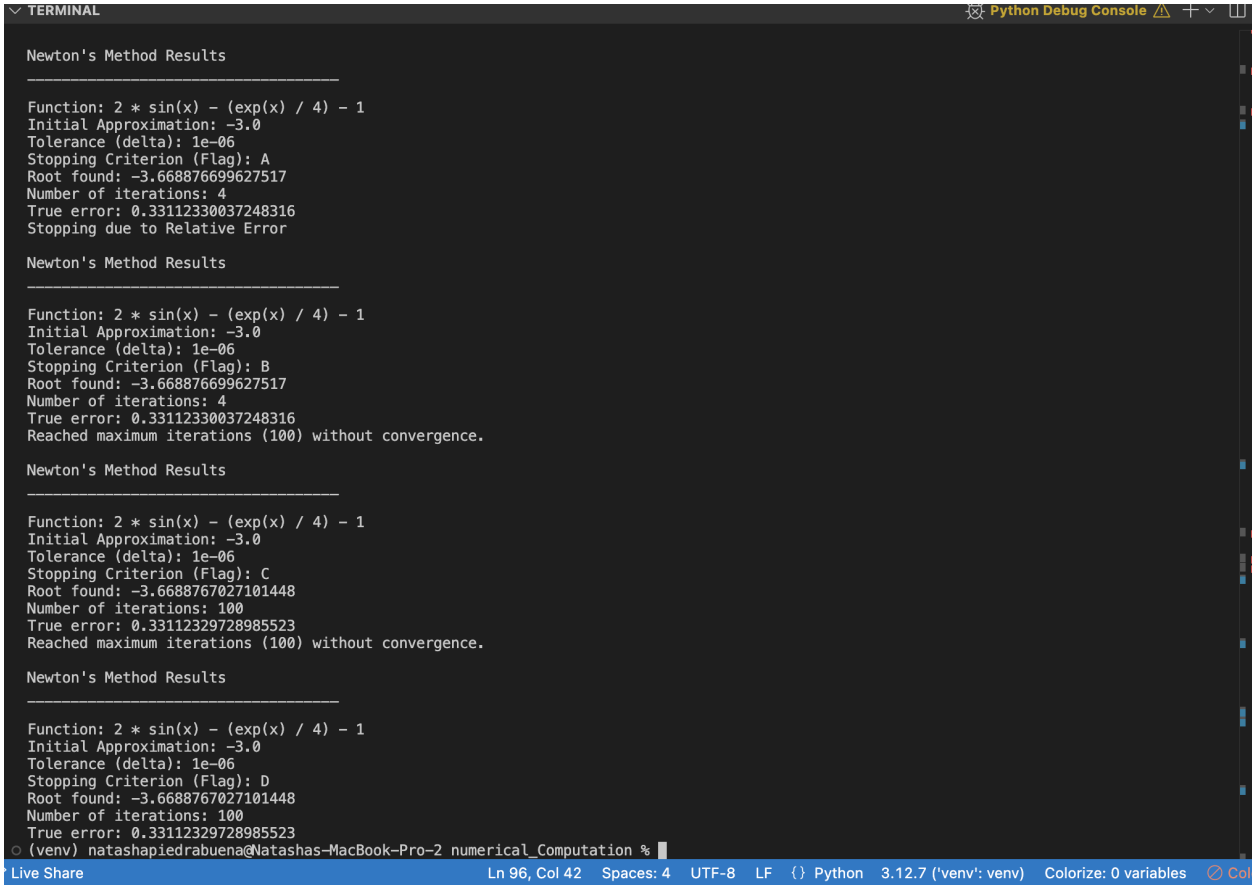
input. Then after further testing, I realized only stop criteria A,B actually worked while C and D didn't due to issues with implementation of true root. This led to the program to loop while never stopping and I had to end the program after a minute.



In this sense I believe that is needed to improve is remove the true root = None in parameter and also adjust the logic of the true root in the newton function.

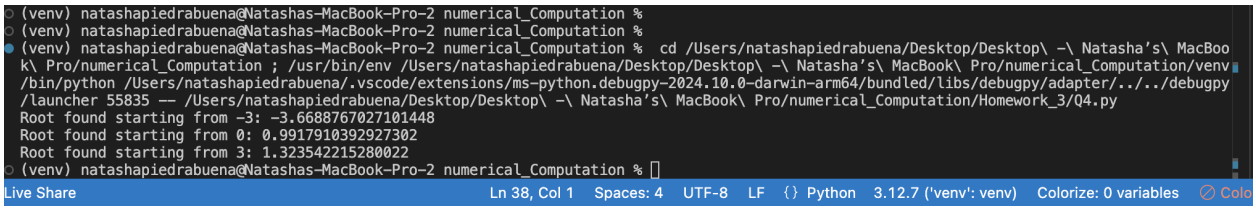
Q3) For question 3, I had to adjust my code to accept a function like 2 * sin(x) - (exp(x) /4) - 1 without a sympy math domain error, adding exception handling in various parts of the code to catch these issues and also making sure the true root is properly implemented. I also had to make a for loop with a list that contains all the stopping criteria.


Q3.py — numerical_Computation
Homework_3 > Q3.py > newton_method
def newton_method(x0, delta, func, func_derivative, stop_flag, true_root, max_iterations=100):
f_prime_x = func_derivative(current_x)
except Exception as e:
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
TERMINAL
Python Debug Console
Newton's Method Results
Function: x**3 - 2*x + 2
Initial Approximation: -1.0
Tolerance (delta): 1e-06
Stopping Criterion (Flag): A
Root found: -1.7692923542973595
Number of iterations: 8
True error: 0.16929235429735945
Stopping due to Relative Error
Newton's Method Results
Function: x**3 - 2*x + 2
Initial Approximation: -1.0
Tolerance (delta): 1e-06
Stopping Criterion (Flag): B
Root found: -1.7692923542973595
Number of iterations: 8
True error: 0.16929235429735945
Reached maximum iterations (100) without convergence.
Newton's Method Results
Function: x**3 - 2*x + 2
Initial Approximation: -1.0
Tolerance (delta): 1e-06
Stopping Criterion (Flag): C
Root found: -1.7692923542386314
Number of iterations: 100
True error: 0.16929235423863132
Reached maximum iterations (100) without convergence.
Newton's Method Results
Function: x**3 - 2*x + 2
Initial Approximation: -1.0
Tolerance (delta): 1e-06
Stopping Criterion (Flag): D
Root found: -1.7692923542386314
Number of iterations: 100
True error: 0.16929235423863132
(venv) natashapiedrabuena@Natashas-MacBook-Pro-2 numerical_Computation %
OPEN EDITORS
Q1.py Homework_3
Q2.py Homework_3
README.md Homework_2
README.md Homework_3
README.md Homework_1
Screenshot 2024-10-13 at 7.40.23 PM.png
Q3.py Homework_3
.gitignore
README.md
NUMERICAL_COMPUTATION
Homework_2
Q2.py
Q3.py
Q4.py
README.md
Report.pdf
Homework_3
Q1.py
Q2.py
Q3.py
Q4.py
README.md
Homework_4
venv
.gitignore
OUTLINE
create_function
expr
e
create_derivative
expr
x
sympy_expr
derivative_expr
TIMELINE
main*
Connect
Live Share
Ln 47, Col 1
Spaces: 4
UTF-8
LF
Python
3.12.7 ('venv': venv)
Colorize: 0 variables
Colorize


TERMINAL
Python Debug Console
Newton's Method Results
Function: 2 * sin(x) - (exp(x) / 4) - 1
Initial Approximation: -3.0
Tolerance (delta): 1e-06
Stopping Criterion (Flag): A
Root found: -3.668876699627517
Number of iterations: 4
True error: 0.33112330037248316
Stopping due to Relative Error
Newton's Method Results
Function: 2 * sin(x) - (exp(x) / 4) - 1
Initial Approximation: -3.0
Tolerance (delta): 1e-06
Stopping Criterion (Flag): B
Root found: -3.668876699627517
Number of iterations: 4
True error: 0.33112330037248316
Reached maximum iterations (100) without convergence.
Newton's Method Results
Function: 2 * sin(x) - (exp(x) / 4) - 1
Initial Approximation: -3.0
Tolerance (delta): 1e-06
Stopping Criterion (Flag): C
Root found: -3.6688767027101448
Number of iterations: 100
True error: 0.33112329728985523
Reached maximum iterations (100) without convergence.
Newton's Method Results
Function: 2 * sin(x) - (exp(x) / 4) - 1
Initial Approximation: -3.0
Tolerance (delta): 1e-06
Stopping Criterion (Flag): D
Root found: -3.6688767027101448
Number of iterations: 100
True error: 0.33112329728985523
(venv) natashapiedrabuena@Natashas-MacBook-Pro-2 numerical_Computation %
Live Share
Ln 96, Col 42
Spaces: 4
UTF-8
LF
Python
3.12.7 ('venv': venv)
Colorize: 0 variables

Q4)

So I asked Chatgpt to implement Q3 and to be able to stop in all the criteria however in this output it can be seen that a root was found but it didn't perform all the criteria and its not easy to adjust to various points.The roots seem to differ from my own code due to the fact that chat gpt put the wrong initial guesses -3, 0, 3.



Results

Overall, the choice of stopping criterion depends on the trade-off between computational cost and desired accuracy. If precision is paramount, Flag A and B is the most appropriate since it takes less iterations compared to C and D. When I utilized chatgpt to generate code for the newton method I can tell the difference in the code right away. The AI-generated code was mostly correct, meeting the problem's requirements, but exhibited several differences compared to my manually implemented code.

In conclusion, Newton's method provided effective means of solving the non-linear equation within the specified intervals. The choice of stopping criterion significantly impacts the trade-off between convergence speed and accuracy. The AI-generated code performed well but required some manual refinement to match the flexibility and robustness of the manually implemented solution.