Peter Julius M. Estacio

Grouped with Nino de Mesa and Osh Ong

ENGG 27.01

October 10, 2025

Progress Report 3 for Project 2: Hybrid Root-Finding Approach

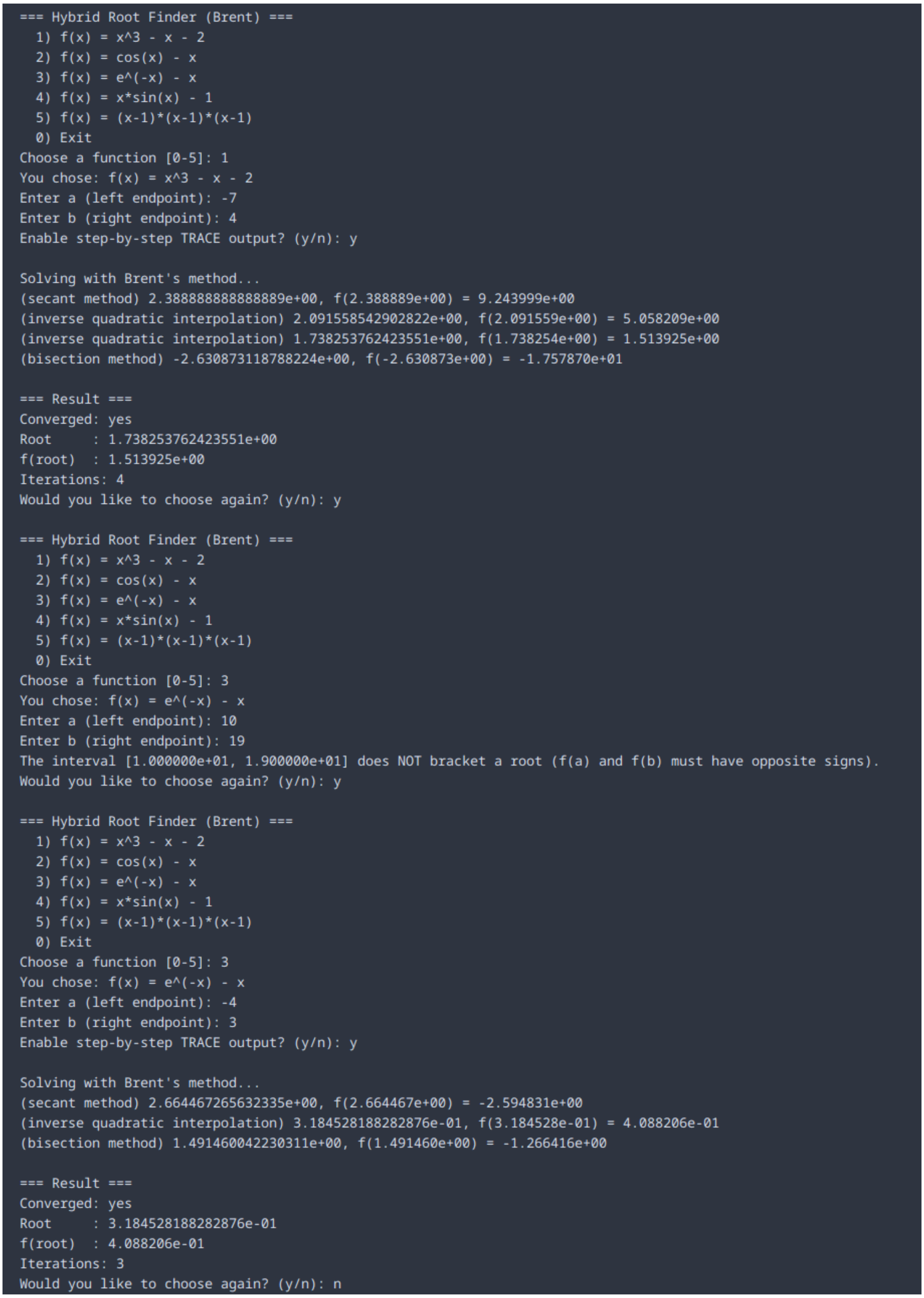
Recent progress for this report includes using header implementation for the program, separating the code into the main.cpp file and the header brent_root.h file. This was done in order to compartmentalize the program and ensure the file security. The header file mainly defines functions and structures used in the program, while the implementation itself is programmed within the main file.

Final tweaks were also made to the program itself in order to achieve the specified requirements. For example, the program's console output was formatted, allowing for a more organized user interaction. In the implementation itself, the program was also made to account for small denominator values, reducing errors in the output. Lastly, checks were added to account for non-finite values, adding to the stability of the program. These changes were tested and verified, and the program was observed to comply with specifications and display correct outputs.

Code Locator:

Code Items	File name	Line numbers
Routine/Function implementing the hybrid root-finding approach	main.cpp	53 – 197
Routine for the open method used	main.cpp	Inverse Quadratic Interpolation: 90 - 98 Secant: 99 - 108
Routine for the bracketing method used	main.cpp	117 – 136
Code used to decide between open and bracketing methods	main.cpp	110 – 115

A sample output of the program is shown below:



```
=== Hybrid Root Finder (Brent) ===
  1) f(x) = x^3 - x - 2
  2) f(x) = cos(x) - x
  3) f(x) = e^(-x) - x
  4) f(x) = x*sin(x) - 1
  5) f(x) = (x-1)*(x-1)*(x-1)
  0) Exit
Choose a function [0-5]: 1
You chose: f(x) = x^3 - x - 2
Enter a (left endpoint): -7
Enter b (right endpoint): 4
Enable step-by-step TRACE output? (y/n): y

Solving with Brent's method...
(secant method) 2.388888888888889e+00, f(2.388889e+00) = 9.243999e+00
(inverse quadratic interpolation) 2.091558542902822e+00, f(2.091559e+00) = 5.058209e+00
(inverse quadratic interpolation) 1.738253762423551e+00, f(1.738254e+00) = 1.513925e+00
(bisection method) -2.630873118788224e+00, f(-2.630873e+00) = -1.757870e+01

=== Result ===
Converged: yes
Root     : 1.738253762423551e+00
f(root)  : 1.513925e+00
Iterations: 4
Would you like to choose again? (y/n): y

=== Hybrid Root Finder (Brent) ===
  1) f(x) = x^3 - x - 2
  2) f(x) = cos(x) - x
  3) f(x) = e^(-x) - x
  4) f(x) = x*sin(x) - 1
  5) f(x) = (x-1)*(x-1)*(x-1)
  0) Exit
Choose a function [0-5]: 3
You chose: f(x) = e^(-x) - x
Enter a (left endpoint): 10
Enter b (right endpoint): 19
The interval [1.000000e+01, 1.900000e+01] does NOT bracket a root (f(a) and f(b) must have opposite signs).
Would you like to choose again? (y/n): y

=== Hybrid Root Finder (Brent) ===
  1) f(x) = x^3 - x - 2
  2) f(x) = cos(x) - x
  3) f(x) = e^(-x) - x
  4) f(x) = x*sin(x) - 1
  5) f(x) = (x-1)*(x-1)*(x-1)
  0) Exit
Choose a function [0-5]: 3
You chose: f(x) = e^(-x) - x
Enter a (left endpoint): -4
Enter b (right endpoint): 3
Enable step-by-step TRACE output? (y/n): y

Solving with Brent's method...
(secant method) 2.664467265632335e+00, f(2.664467e+00) = -2.594831e+00
(inverse quadratic interpolation) 3.184528188282876e-01, f(3.184528e-01) = 4.088206e-01
(bisection method) 1.491460042230311e+00, f(1.491460e+00) = -1.266416e+00

=== Result ===
Converged: yes
Root     : 3.184528188282876e-01
f(root)  : 4.088206e-01
Iterations: 3
Would you like to choose again? (y/n): n
```