Contents

List of Figures

List of Tables

0.0.1 Conclusion

These are our conclusions based on the tables and the plots:

0.0.1.1 Iterations

0.0.1.1.1 Bisection, False Position, HybridBF As you see in the plot below the HybridBF method demonstrates superior performance compared to both the bisection and false position methods in terms of the number of iterations required.

As we see here in P_{21} the false position method have much more number of iterations than both hybrid and bisection methods which will lead to more CPU time as we will see in the next section.

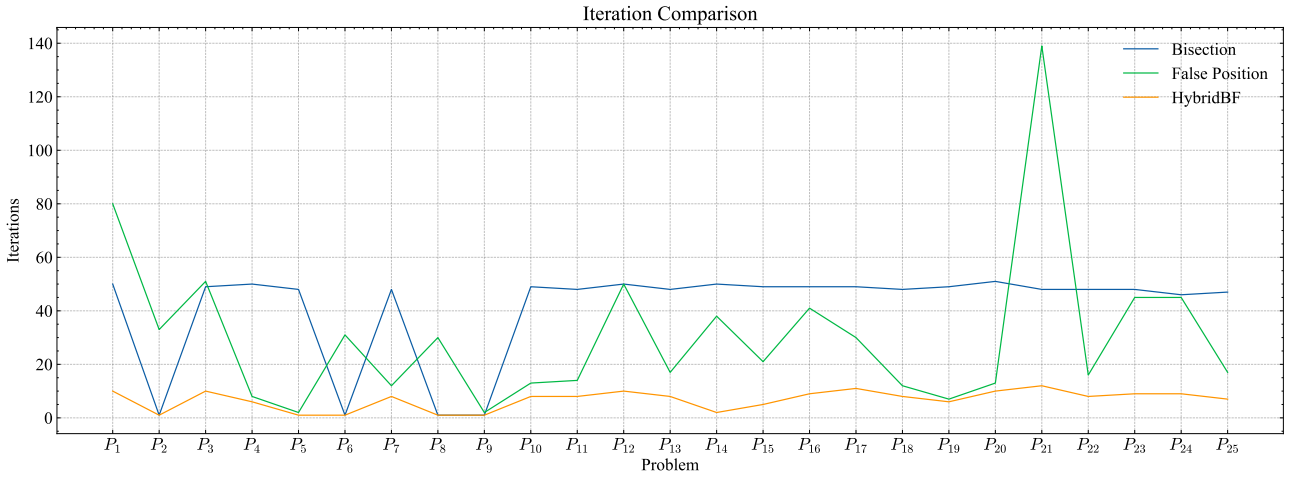

Figure 1: Iterations Comparison Bisection, False Position, HybridBF

0.0.1.1.2 Secant, False Position, HybridSF The HybridSF method here didn't have the same performance improvement as in the previous case. The secant method is the fastest in terms of iterations, followed by the hybrid method then false position, However There are 7 problems where the HybridSF method is faster than the secant method with 1 or 2 iterations, these are P_4, P_5, P_7, P_9, P_{18}, P_{19}, P_{20}.

You can also notice that the graph of the secant method is not continuous on P_{21} since secant method is not guaranteed to converge, so it wasn't able to find the root in this problem.

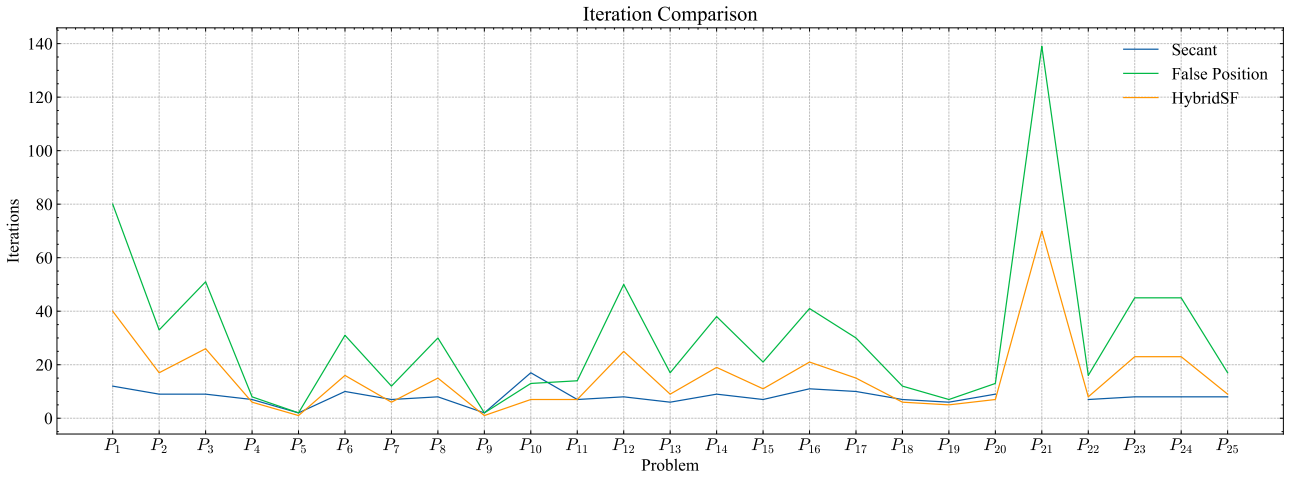

Figure 2: Iterations Comparison Secant, False Position, HybridSF

0.0.1.2 CPU Time

0.0.1.2.1 Bisection, False Position, HybridBF As a result of having less number of iterations the hybridBF method shows significant improvement over the bisection method in terms of CPU time, with a ratio of 21:4. This translates to approximately 84% for the hybrid method and 16% for the bisection method.

It also shows an improvement over the false position method in terms of CPU time, with a ratio of 19:6. This translates to approximately 76% for the hybrid method and 24% for the false position method.

As a general trend, the hybrid method is faster than both the bisection and false position methods when it comes to finding the approximate root.

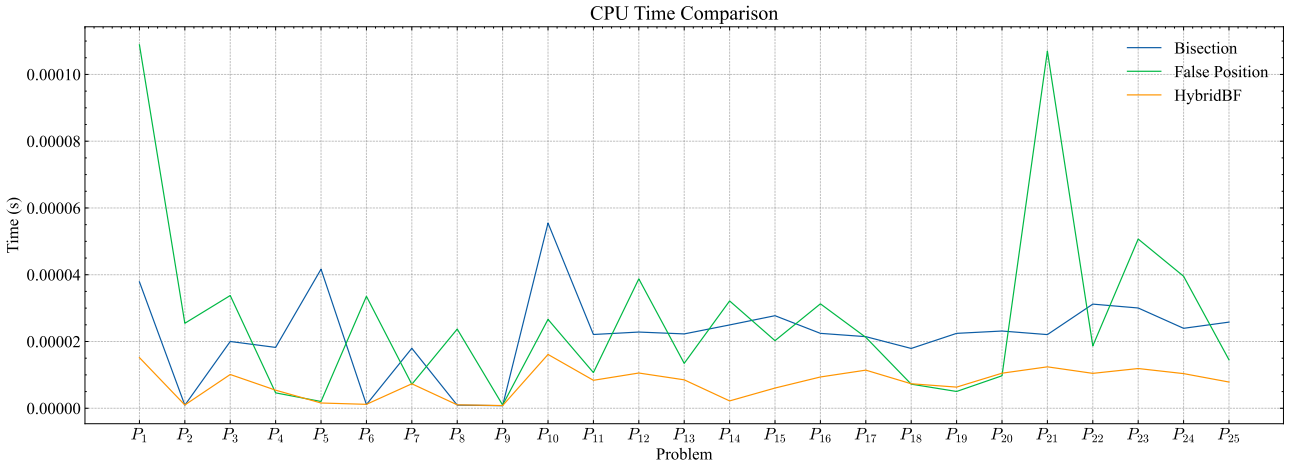

Figure 3: CPU Time Comparison Bisection, False Position, HybridBF

0.0.1.2.2 Secant, False Position, HybridSF As a result of secant method having less number of iterations than both HybridSF and false position and much more simple implementation than HybridSF, it has less CPU time. The secant method is the fastest in terms of CPU time, followed by the hybrid method then false position.

This happens in all problems except only two problems which are P_5, P_9 where the HybridSF method is *slightly* faster than the secant method.

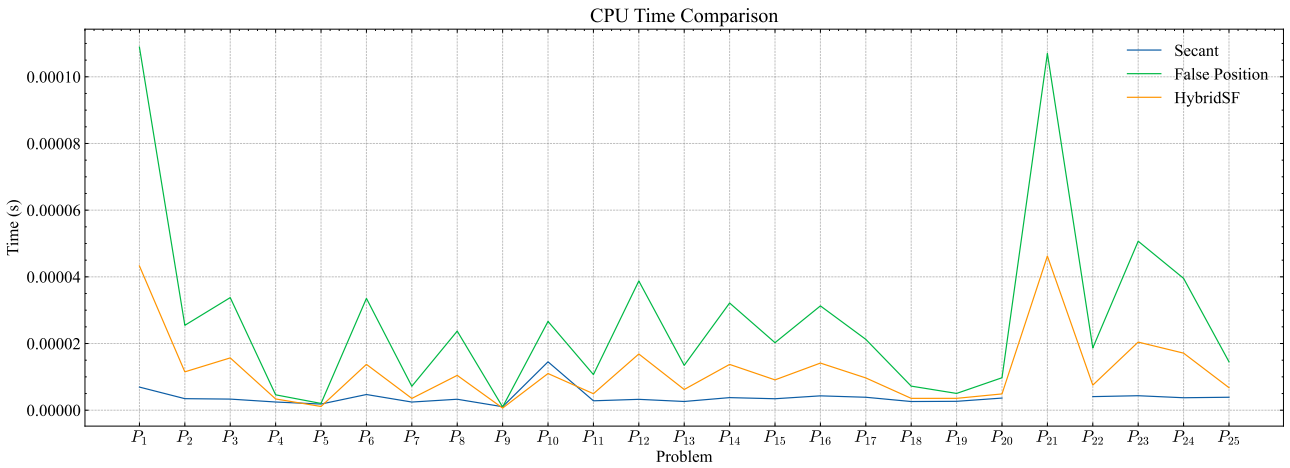

Figure 4: CPU Time Comparison Secant, False Position, HybridSF

0.0.1.3 Function Value

0.0.1.3.1 Bisection, False Position, HybridBF The hybrid method outperforms both the bisection and false position methods in terms of function value, with smaller absolute values that are closer to zero.

This happens in all problems except only one problem P_{24} in which the bisection method had the nearest value to zero.

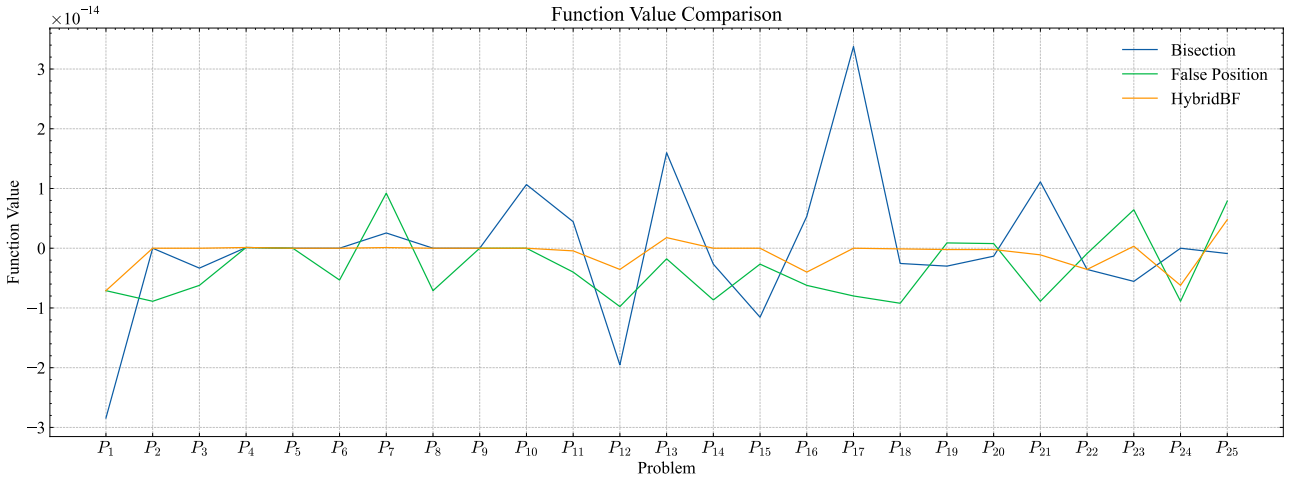

Figure 5: Function Value Comparison Bisection, False Position, HybridBF

0.0.1.3.2 Secant, False Position, HybridSF Again the secant method shows an improvement over both the false position and HybridSF methods in terms of function value, with smaller absolute values that are closer to zero.

This happens in all problems except two P_4 in which both HybridSF and false position methods had the nearest value to zero, and P_{15} in which the HybridSF had a value closer to zero.

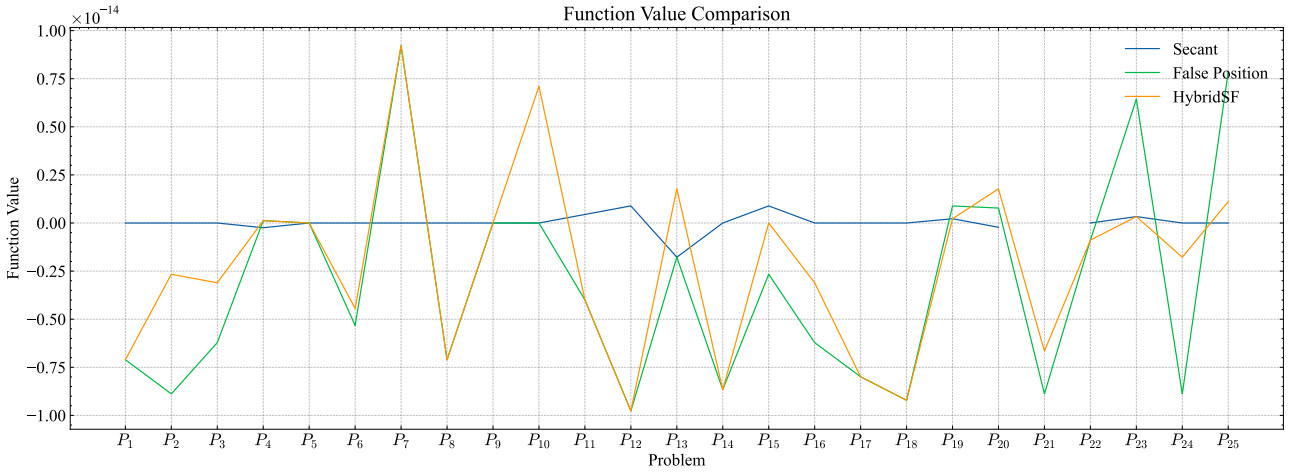

Figure 6: Function Value Comparison Secant, False Position, HybridSF

0.0.1.4 Secant vs HybridBF

When comparing the results of the fastest two algorithms which are the secant and HybridBF methods, we found the following:

0.0.1.4.1 Iterations There is no winner here, the secant method is faster in some problems and the HybridBF method is faster in others but there is a slight advantage for the HybridSF method over the secant method.

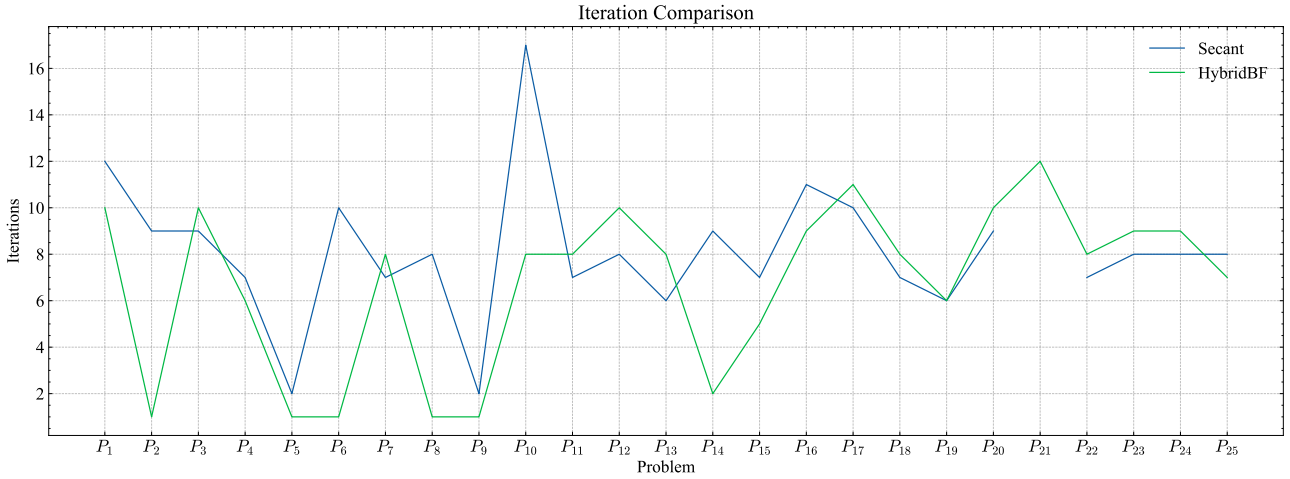

Figure 7: Iterations Comparison Secant, HybridBF

0.0.1.4.2 CPU Time When comparing the CPU time of both algorithms we found that the secant method is faster in all problems except for 6 problems which are P_2, P_5, P_6, P_8, P_9, P_{14}.

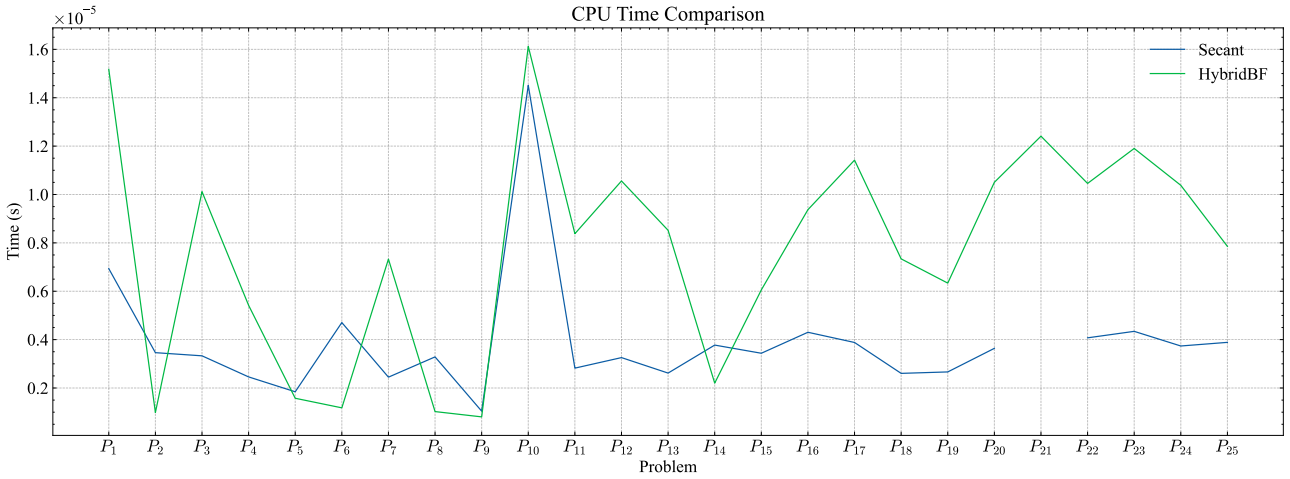

Figure 8: CPU Time Comparison Secant, HybridBF

0.0.1.4.3 Function Value When comparing function values with absolute values closer to zero, the results of secant method were better than the HybridBF method in nearly all problems except for P_{15}.

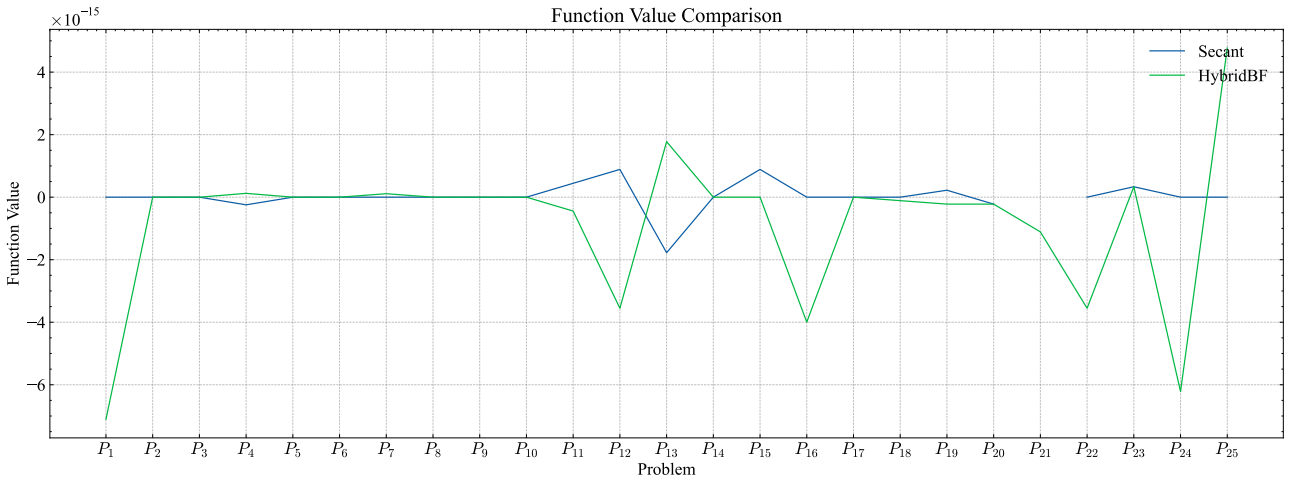

Figure 9: Function Value Comparison Secant, HybridBF

Used Machine Specifications

The results were taken on a machine with the following specifications:

- Operating System: Linux Mint 21.3 Cinnamon
- Processor: Intel© Core™ i5-8300H CPU @ 2.30GHz × 4
- Memory: 16GB
- Python Version: 3.11.6