
Investigation Of Root Finding Methods: The secant , Inverse Quadratic Interpolation and Brents Methods

Daniel P

November 1, 2025

1 INTRODUCTION

Root finding is a crucial area within numerical analysis with applications in key industries, such as quantitative finance, where it is used to find the implied volatility of pricing models where there is no closed-form solution. We will investigate two root-finding methods: the secant method and inverse quadratic interpolation (IQI). We will first examine how they can be derived from Inverse Lagrange interpolation. Then, we will implement these algorithms on the same root-finding problem and compare their efficiency.

Each method uses inverse Lagrange interpolation, where they fit an interpolating polynomial of degree $n - 1$ to n points, fitting x in terms of $f(x)$ to estimate the value of x for $f(x) = 0$ the secant method uses a polynomial of degree 1, and IQI uses degree 2. They iterate, utilising a series of the roots of these interpolating polynomials to approximate the root of the function, getting closer to the root at each iteration (assuming convergence). Both methods are also 'open' or 'non-bracketing'. That is, they do not require the root to lie between points. Due to this, there is no certainty of convergence as they can diverge or oscillate.

We successfully implemented both methods on our root-finding problem. We found that

the rate at which IQI converges to the root is faster than the secant method; however, the computational cost for IQI is greater, and both methods incur issues when estimating roots of functions that are 'near flat' close to the root.

2 THEORY AND METHODS

2.1 INVERSE LAGRANGE INTERPOLATION

Inverse Lagrange interpolation is a method of estimating the output of a function by fitting an interpolating polynomial of degree $n - 1$ to a set of n known points of a function $(x_1, f_1), (x_2, f_2) \dots (x_n, f_n)$, where $f_i = f(x_i)$. It will be fitting x in terms of $f(x)$, meaning it will take the form shown by Equations (1) & (2) [1].

$$P(f) = \sum_{i=0}^n x_i \cdot L_i(f) \quad (1)$$

Where $L_i(f)$ is a Lagrange basis polynomial [2] in terms of the function value shown by Equation (2)

$$L_i(f) = \prod_{\substack{j=0 \\ j \neq i}}^n \frac{f - f_j}{f_i - f_j} \quad (2)$$

This allows us to estimate the corresponding x value from a given set of function points and a given value of $f(x)$. This is important to root finding, as we can take known function points, use inverse Lagrange interpolation to fit a polynomial between these and find the root of the interpolating polynomial that is where $f(x) = 0$, estimating the root of the function. This is the heart of the following methods, so it is essential to understand this completely.

2.2 SECANT METHOD

This is a linear inverse Lagrange interpolation [3]. That is, we have a set of 2 known points (x_i, f_i) & (x_{i-1}, f_{i-1}) where $f_i = f(x_i)$ from section 2.1 with 2 points we can use inverse Lagrange interpolation to fit a 1-degree interpolating polynomial to estimate the root of a function. So plugging our points into Equations (1) & (2) we get

$$P_1(f) = x_{i-1} \left(\frac{f - f_i}{f_{i-1} - f_i} \right) + x_i \left(\frac{f - f_{i-1}}{f_i - f_{i-1}} \right) \quad (3)$$

Setting $f = 0$ estimates the root of the function, and this estimate will be our x_{i+1} for the next iteration.

$$P_1(0) = x_{i-1} \left(\frac{0 - f_i}{f_{i-1} - f_i} \right) + x_i \left(\frac{0 - f_i}{f_i - f_{i-1}} \right) \quad (4)$$

We can now simplify Equation (4) to the following.

$$x_{i+1} = \frac{f_i x_{i-1} - f_{i-1} x_i}{f_i - f_{i-1}} \quad (5)$$

Below is the rearranged form we iterate with.

$$x_{i+1} = x_i - f_i \left(\frac{x_i - x_{i-1}}{f_i - f_{i-1}} \right) \quad (6)$$

We have now derived the secant method from inverse Lagrange interpolation into an iterable form. We can take two initial 'guesses' of the root (x_0, f_0) & (x_1, f_1) , use Equation (6) to find the root of the line created from these points, $(x_2, 0)$, estimating function root, then use this (x_2, f_2) as our next estimate and carry out iterations of this until we meet our stopping criterion which will be discussed in section 2.4, this all assumes that our series of roots is converging as of course there are cases where they do not converge.

2.3 INVERSE QUADRATIC INTERPOLATION

For IQI, we consider three points (x_i, f_i) , (x_{i-1}, f_{i-1}) & (x_{i-2}, f_{i-2}) where $f_i = f(x_i)$, we derive IQI from Inverse Lagrange Interpolation like secant except using 3 points. Therefore, our interpolating polynomial is a quadratic [4]. So, using Equations (1) & (2) and plugging in our points, we get the following

$$\begin{aligned} P_2(f) &= x_{i-2} \frac{(f - f_{i-1})(f - f_i)}{(f_{i-2} - f_{i-1})(f_{i-2} - f_i)} \\ &+ x_{i-1} \frac{(f - f_{i-2})(f - f_i)}{(f_{i-1} - f_{i-2})(f_{i-1} - f_i)} \\ &+ x_i \frac{(f - f_{i-2})(f - f_{i-1})}{(f_i - f_{i-2})(f_i - f_{i-1})} \end{aligned} \quad (7)$$

Setting $f = 0$ to find the root of this interpolating polynomial to approximate our function root gives us our next approximation x_{i+1} , shown in Equation (8).

$$\begin{aligned} x_{i+1} = & x_{i-2} \left(\frac{f_{i-1}f_i}{(f_{i-2} - f_{i-1})(f_{i-2} - f_i)} \right) \\ & + x_{i-1} \left(\frac{f_{i-2}f_i}{(f_{i-1} - f_{i-2})(f_{i-1} - f_i)} \right) \\ & + x_i \left(\frac{f_{i-1}f_{i-2}}{(f_i - f_{i-1})(f_i - f_{i-2})} \right) \end{aligned} \quad (8)$$

As described in the secant method, we now have this in the form ready to iterate, but this time, we use three points at each iteration.

2.4 IMPLEMENTATION OF OUR ALGORITHMS

Now that we have our secant method and IQI in forms ready to iterate, we start iteration by plugging our initial guesses into our equations. This is two points for the secant method $(x_1, f(x_1))$ & $(x_0, f(x_0))$ and the three points for IQI $(x_2, f(x_2))$, $(x_1, f(x_1))$ & $(x_0, f(x_0))$ plugging into these Equations (6) & (8) gives us the first approximation, we continue iterating in this fashion using each new approximation to find the next approximation until we meet our stopping criterion. Our stopping criterion for our algorithms is that the absolute difference between successive approximations is less than a given tolerance ϵ . Hence, we continue iteration until Equation (9) holds true, with the final value being our last approximation. We took ϵ as 10^{-11} .

$$|x_i - x_{i+1}| < \epsilon \quad (9)$$

This criterion works because as we iterate if it is converging to the root, the difference between successive approximations should be shrinking therefore when the difference between consecutive approximations is 10^{-11} , we say we have slowed down enough in our opinion to have reached convergence.

For comparison we will use both methods on the same root-finding problem. We will find the root of the Function shown by Equation (10) and Figure 1. This was taken as an example from Burden and Faires [5].

$$f(x) = \cos(x) - x \quad (10)$$

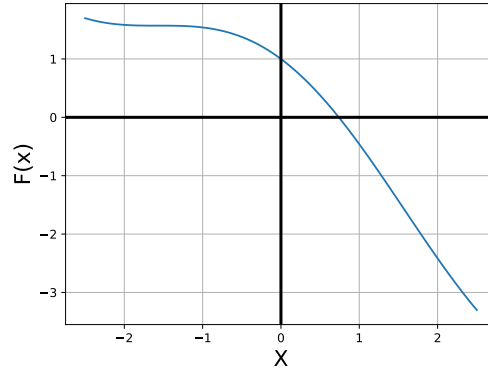


Figure 1: Plot of the function shown by Equation (10) for $x \in [-2.5, 2.5]$.

2.5 ORDER OF CONVERGENCE

Once we have applied our algorithm and found a root, we use the order of convergence denoted α to provide a descriptive statistic for how quickly a sequence of approximations converges, that is, how fast the error is shrinking and can be calculated retrospectively, allowing us to compare methods. Once we have run our algorithm and we have our final approximation to the root x , we can calculate the error at each approximation E_i using the following Equation (11).

$$x_i - x = E_i \quad (11)$$

We can then plot the absolute error for each approximation x_i against the absolute error at each approximation x_{i-1} and use the following logic to find the order of convergence α .

$$\lim_{i \rightarrow \infty} \frac{|E_i|}{|E_{i-1}|^\alpha} = c \quad (12)$$

As we are plotting absolute error for x_i on the y-axis and absolute error at x_{i-1} on the x-axis, we can write this as the following

$$\frac{Y}{X^\alpha} = c \quad (13)$$

Taking the Log of these, we get the following

$$\log(Y) = \log(c) + \alpha \log(X) \quad (14)$$

We plotted an LSE of our log plot with a polynomial of degree 1 and took this gradient as our estimate of order of convergence. We will fit this line twice for each method, once including our initial guesses and once not including them, to see what difference they could make.

3 RESULTS AND DISCUSSION

We plotted the results from both methods with their lines of best fits in Figure 2 as detailed in section 2.5 giving us Figure 2.

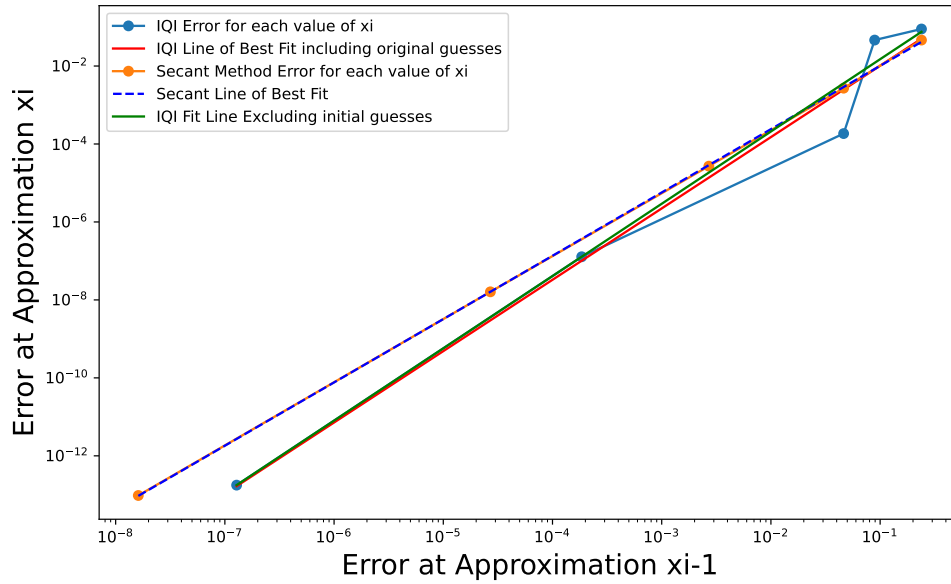


Figure 2: This shows the approximation x_i against the absolute error at approximation x_{i-1} for IQI in light blue and the secant method in Orange. With the LSE for the log plot of IQI, including the errors of our initial guesses with the solid red line and not including our guesses with a green. We plotted our LSE for our secant method with the dashed dark blue line, including our initial guess. We did not find it necessary to plot without initials as they would look identical.

3.1 SECANT METHOD RESULTS

For the secant method, we took our $x_0 = 0.5$ & $x_1 = \frac{\pi}{4}$. We plug these into Equation (10) to give our values of $f(x_0) = 0.377582$ & $f(x_1) = -0.078292$ giving us our initial points and begin iteration. Using Equation (6), we continue this iteration until we meet our stopping criterion, as discussed in section 2.5.

We have the root returned in 5 iterations within the tolerance of 10^{-11} , as 0.7390851. We verified that this has been implemented correctly, as our intermediate results are the same as the example in Burden and Faires [5]. We then plotted the absolute error described in section 2.5, giving us our orange line in Figure 2. This shows how the error decreases at each iteration until it meets our tolerance in 5 iterations.

Our LSE, including initial guesses shown in Figure 2 gives our $\alpha = 1.6241477$, not plotted, but we calculated the gradient excluding our initial guesses, giving us a $\alpha = 1.6175062$. The known order of convergence for the secant method is approximately 1.618 [6], So we are in the correct range for our order convergence, although this does not hold for every function.

3.2 INVERSE QUADRATIC INTERPOLATION RESULTS

Our three initial guesses were $x_0 = 0.5, x_1 = 0.65$ & $x_2 = \frac{\pi}{4}$ plugging these values into Equation (10) to give our values of $f(x_i)$, then we iterated with Equation (8) following the methodology outlined in section 2.4.

We have the root returned in 4 iterations within a tolerance of 10^{-11} as 0.7390851, the same as the secant method. Our plot of absolute error, as described in section 2.5, is shown in Figure 2. This is a light blue solid line. This shows again that the absolute error decreases at each new approximation. with our initial guesses included $\alpha = 1.830351$ and without $\alpha = 1.854327$. If we compare these to the known order of convergence, approximately 1.84 [4], although not absolute proof, this helps verify the correct implementation somewhat.

3.3 COMPARISON

From Figure 2, we see immediately that the convergence speed is quicker for IQI as the difference between successive approximations is larger, and IQI reached our stopping criterion in one less iteration. We can also see this quantitatively by comparing the α IQI had $\alpha = 1.854327$ excluding the initial guesses. In contrast, the secant method had $\alpha = 1.6175062$, telling us that IQI converged faster, meaning it shrunk the error faster.

The difference in the convergence rate is because we are using a higher-order interpolating polynomial for our IQI. Referring back to our derivations of the secant and IQI,

both use inverse Lagrange interpolation, but IQI uses a polynomial degree 2. A higher-order polynomial can more accurately capture the behaviour of our curved function, giving us a more accurate approximation of the root at each iteration, thus minimising the absolute error of Equation (11) better at each iteration than secant, resulting in a greater rate of convergence.

However, they both have issues and using the secant method over IQI may be beneficial. For example, take the function shown by Equation (15) and Figure 3.

$$f(x) = (x - 1)^3 \quad (15)$$

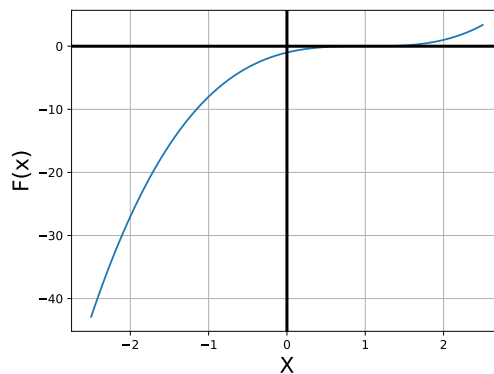


Figure 3: This is a plot of Equation (15) for $x \in [-2.5, 2.5]$

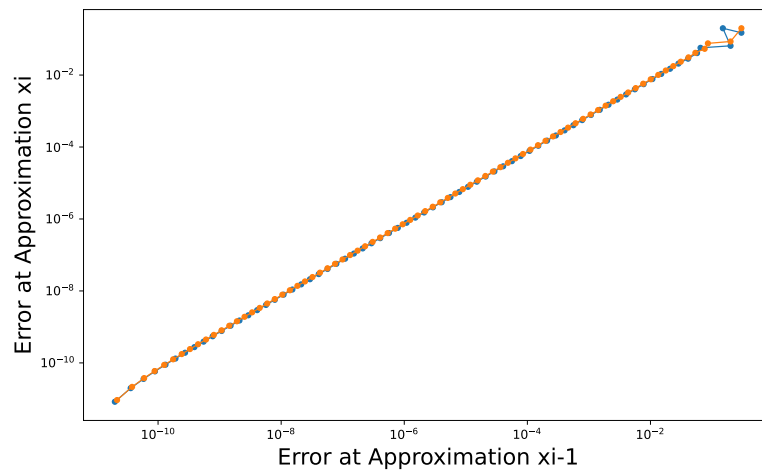


Figure 4: The secant method Applied to Equation (15) with initial guesses 0.7 and 1.2. and IQI applied with initial guesses 0.7, 0.85 and 1.2.

Figure 4 shows that we applied both methods and found the root as 1. secant took 79 iterations with $\alpha = 1.002622$ and IQI 68 iterations with $\alpha = 1.003114$, so both have slowed down substantially. Looking at Figure 3, we see this function becomes near flat close to $f(x) = 0$. Therefore, when using IQI, the second-degree polynomial that we are finding the root to get x_{i+1} is much more 'flat' within the interpolating range; thus, we are losing the majority of the benefit that IQI has over the secant method, and both methods rely on slopes to make 'jumps' so as there is no slope close to zero successive approximations are close together making it a slow process.

Although the number of iterations taken for the secant method is 11 greater to converge than the IQI, Looking at Equations (6) & (9) using IQI has 20 operations to perform per iteration In comparison for secant we complete a total of 5 operations (this is excluding operations needed to evaluate f_i each time). Therefore, we would have to complete approximately 1360 operations to find the root using IQI. While using approximately 395 operations for the secant method. We then calculated the average time to compute each method for Equation (15) by running it 100,000 times and took the average time to compute in seconds IQI had an average compute time of 7.334×10^{-5} while secant had 3.881×10^{-5} So, the secant method is more computationally efficient.

4 CONCLUSIONS AND FURTHER WORK

4.1 CONCLUSIONS

In this study, we derived the secant method and IQI from inverse Lagrange interpolation and implemented both algorithms effectively on a root-finding problem. Although close, IQI converges faster to the root than the secant method under favourable functions. This leads us to believe IQI is the more favourable method to implement on a root-finding problem, assuming we have reasonable initial guesses. The reason behind the faster convergence of IQI is its use of quadratic interpolating polynomials rather than linear interpolating polynomials, which allow for better approximations of the root, assuming a 'well-behaved' function. However, We discovered that the secant method might be preferable as it would be less computationally taxing and, therefore, faster to compute, which may be a more favourable quality, depending on where it is implemented.

However, we also discovered that both these methods have issues when implemented on functions that are close to flat near the root. Both slow down to an approximately equal order of convergence of 1, and we can run into divisions by zero if the values of f_i are too close to compute.

4.2 BRENT'S METHOD

IQI and secant have issues, such as running into divisions by zero, slowing down, and not guaranteeing convergence. To combat this, it is best to use a hybrid root-finding method like Brent's method. Brent's combines bracketing, bisection, secant, and IQI [7]. It performs them within the algorithm shown by the flow chart in Figure 5 [8] [9], which guarantees it finds the root, avoids divisions by zero, and deals with poor functions better.

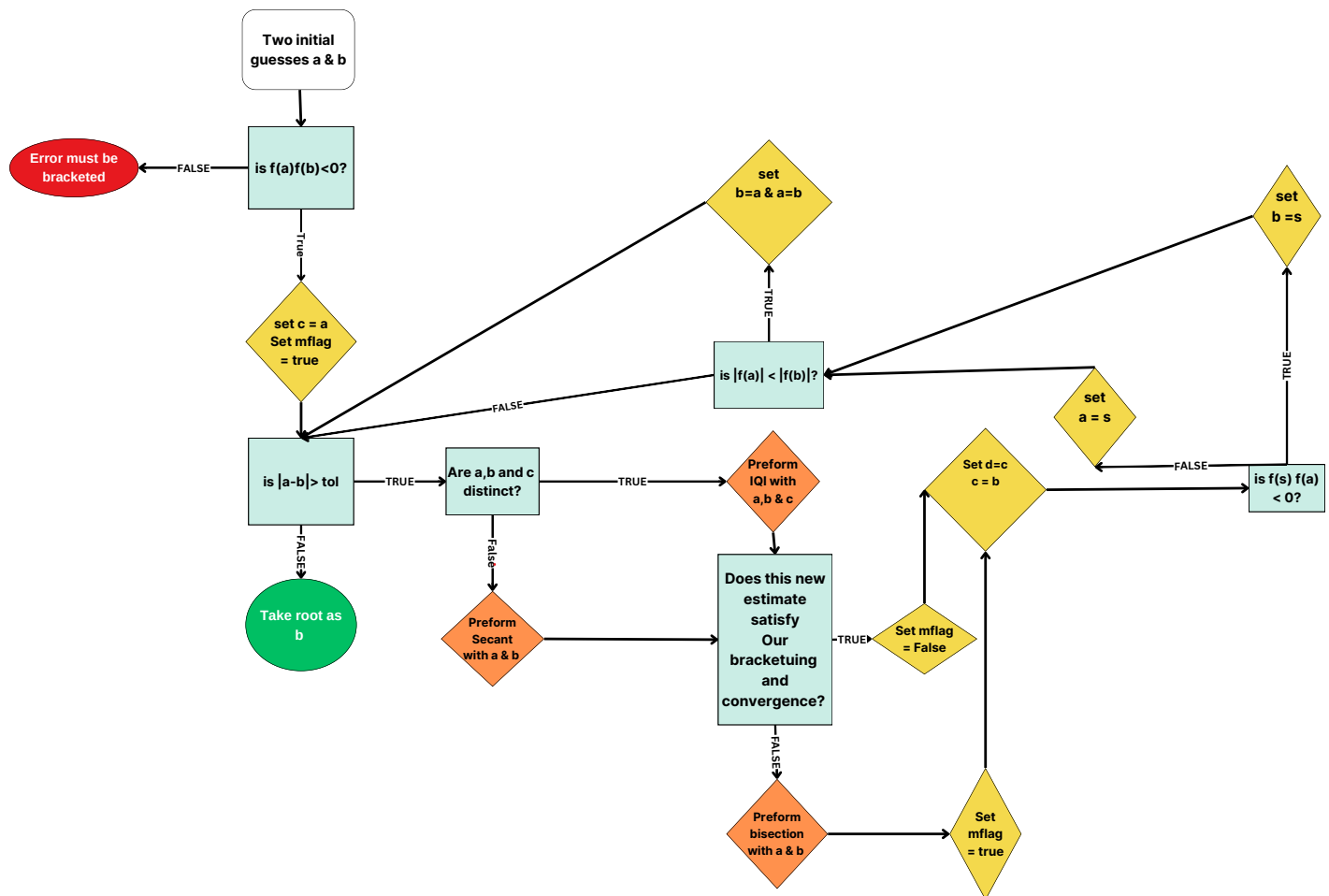


Figure 5: Flow diagram of Brent's method algorithm guarantees convergence from two initial bracketed guesses using bisection while taking advantage of the speed of secant and IQI when applicable. With s being the latest approximation for each iteration. It uses the set of rules shown in Figure 6 to determine if bisection should be used to maintain bracketing and rate of convergence, and it iterates until $|a - b| < \epsilon$ returning our root as our last b value.

This algorithm Figure 5 uses the logic shown in Figure 6 [8] [9] to decide whether the new estimate at each iteration s satisfies the bracketing and convergence criterion. The first two pieces of logic ensure that our new guess s is bracketed; if not, it performs bisection, and the remaining logic ensures they converge effectively. If not, it performs bisection. This guarantees that we converge and converge effectively.

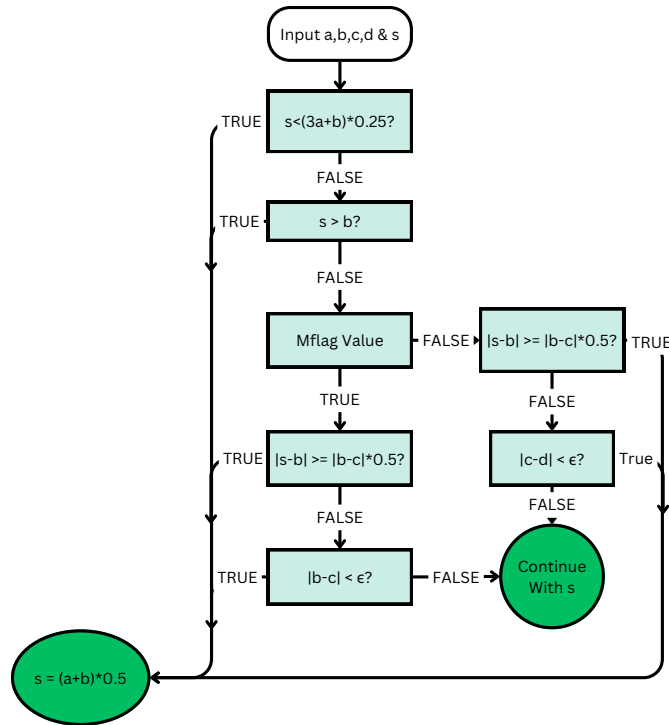


Figure 6: Flow diagram of Brent's methods logic for bracketing and convergence criterion. These rules ensure the approximations are bracketed and attempt to maximise the convergence rate by reporting to bisection.

We implemented Brent's on Equation (15) with $x_0 = 0.7$ and $x_1 = 1.2$. The error plot is shown in Figure 7. This converged to 1 in 44 iterations, significantly fewer than IQL & secant, which took 68 & 79, respectively. However, $\alpha = 1.0007$, similar to IQL and Secant, this is due to the sporadic nature of Brent's error plot. The LSE we use to estimate α doesn't effectively capture the size of jumps between points as it minimises the sum of squares distance to the line from each point; therefore, the mean of the LSE significantly shifts for the outlier jumps in Brent's method which I believe doesn't

truly capture how much the error is reduced by. However, it is the best estimate of the rate at which the error shrinks quantitatively apart from the number of iterations. An investigation of interest would be to investigate the R-Square of the estimates of α to verify if the LSE is a 'good estimator'.

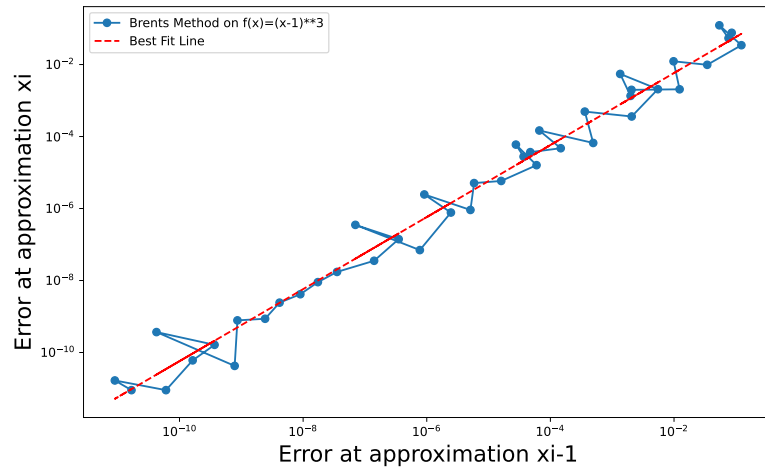


Figure 7: Brents applied method on Equation (15) with initial guesses 0.7 and 1.2. excluding the initial guesses on the plot.

The average time to compute Brent's method was 1.16×10^{-4} seconds, an order of magnitude greater than for IQI and secant. This is the trade-off we must allow for guaranteed convergence while minimising our iterations.

REFERENCES

- [1] Dr. Anita Pal - Numerical Analysis Chapter 2 Section 6 interpolation https://epgp.inflibnet.ac.in/epgpdata/uploads/epgp_content/S000025MS/P001476/M014250/ET/1456308981E-textofChapter2Module6.pdf (accessed 03/11/2024)
- [2] Wolfram Mathworld - Lagrange Interpolating Polynomial <https://mathworld.wolfram.com/LagrangeInterpolatingPolynomial.html> (accessed 28/11/2024)
- [3] Wikipedia - secant Method - https://en.wikipedia.org/wiki/secant_method (accessed 03/11/2024)
- [4] Wikipedia - Inverse Quadratic Interpolation - https://en.wikipedia.org/wiki/Inverse_quadratic_interpolation (accessed 03/11/2024)
- [5] *RL Burden, J D Faires, and A M Burden. Numerical analysis. Cengage Learning 2016, library shelf Mark: QA297 BURD* (accessed 03/11/2024)
- [6] libre Texts Mathematics - 2.4 order of convergence - [https://math.libretexts.org/Bookshelves/Applied_Mathematics/Numerical_Methods_\(Chasnov\)](https://math.libretexts.org/Bookshelves/Applied_Mathematics/Numerical_Methods_(Chasnov)) (accessed 03/11/2024)
- [7] Brian Heinold - An Intuitive Guide to Numerical Methods - https://www.brianheinold.net/notes/An_Intuitive_Guide_to_Numerical_Methods_Heinold.pdf (accessed 19/11/2024)
- [8] Zhengqiu Zhang - An Improvement to the Brent's Method - <https://www.cscjournals.org/manuscript/Journals/IJEA/Volume2/Issue1/IJEA-7.pdf> (accessed 19/11/2024)
- [9] Nick Ryan - Root-Finding Algorithms Tutorial in Python - <https://nickcdryan.com/2017/09/13/root-finding-algorithms-in-python-line-search-bisection-secant-newton-raphson-boydens-inverse-quadratic-interpolation-brents/> (accessed 19/11/2024)