A Better Root Finding Method using False Position and Inverse Quadratic Interpolation Methods

Satya Sai Prakash Vakkalagadda[1]

[1]*Student of B. Tech, Computer Science and Engineering, GITAM (Deemed to be University), Visakhapatnam, India*

---***---

Abstract - *A root-finding algorithm is for finding roots of continuous functions. A root of a function f is a number x such that f(x) = 0. As roots of a function are not exactly computable or expressible in closed form, root-finding algorithms provide approximations to roots, expressed either as floating point numbers or as small isolating intervals. The efficiency of such algorithms depends on the number of iterations they take to reach the root of the functions. There is always a need to develop a better root finding algorithm whose efficiency is better than the known algorithms. A better version similar to the structure of Brent- Dekker that involves False Position, Inverse Quadratic Interpolation and Bisection Methods is discussed here. Instead of using the conditions prescribed by Brent in his algorithm, we consider only one condition where the solution c generated in an iteration needs to be in the interval (a, b) else Bisection is used. This modified version takes a smaller number of iterations in almost all the cases thereby taking less execution time than Brent-Dekker method.*

Key Words: Secant Method, Root-Finding Algorithms, Brent-Dekker Method, Inverse Quadratic Interpolation, Bisection Method, False Postion Method.

1. INTRODUCTION

A function f(x), when continuous contain root(s), maybe real root(s), which is called as root(s) of the function or zero(s) of the function. Such zero of a function can be found out using root finding algorithms. One such method is Brent-Dekker method. This method uses a combination of Secant, Inverse Quadratic Interpolation and Bisection Methods. This method is added as fzero in Python libraries. Secant method is a linear interpolation method. Although it is a relatively fast when compared to other methods, it converges slowly in short steps than directly reaching to zero of the function. Sometimes, secant may diverge instead of converging to zero when a wrong initial interval is given. Combining it with IQI seems to make the whole method fast converging and never diverging but this method can still be developed in terms of its convergence and execution time.

Sometimes, a method is more suited to a sequential approach than a parallelized approach. Here, we are discussing about a method that is obtained from Brent-Dekker but involves False Position Method, IQI Method and Bisection Method. Using False Postion in place of Secant Method makes the fzero more efficient and fast converging. Although, Secant is faster than False Position, when combined with IQI, False Position reaches zero of the function faster than the Secant. Thus, giving us a method that converges in at least half the time taken by Brent-Dekker. We state that this modified method could be more efficient than Brent-Dekker root finding algorithm.

1.1 Bisection Method

The Bisection Method is also called as Interval Halving Method, Binary search method or Dichotomy method. It is based on Bolzano's theorem for continuous functions.

Bolzano's Theorem: If a function f(x) is continuous on an interval [a, b] and f(a)*f(b)<0, then a value c ϵ (a, b) exist for which f(c)=0.

f(a)*f(b)<0 means that they both have opposite signs. The x value for which the plot is crossing the x-axis is the root of the equation f(x)=0. The closeness of this root to the real root depends on the tolerance we set for the algorithm. Bisection is easy to implement and very robust. But this method is relatively slow.

For a given function, f(x), Bisection Method works as follows:

1. Choose a and b for which f(a)*f(b)<0.
2. A midpoint c is calculated as the average of a and b, c=(a+b)/2.
3. Evaluate f(c).
4. If f(c)=0, we found the root. Return it. Else, step 5.
5. If f(c)*f(a)>0 replace a with c.
7. If f(c)*f(b)>0 replace b with c.
8. Goto step 2 or end if the MAX iterations are reached or tolerance is satisfied [1].

1.2 Secant Method

Secant method is considered most effective in finding the root of a non-linear equation. This falls under open bracket type. This method uses two initial guesses and finds the root of a function through interpolation approach. At successive iteration, two of the most recent guesses are used to find the next approximation. Although this method is faster than many other methods, its convergence is not always guaranteed.

The Secant Method works as follows:

1. Get initial guess values a, b and tolerance e.
2. evaluate f(a), f(b).
3. c= b−(f(b)∗(b−a))/(f(b)−f(a)).
4. Goto step 2 or end if f(c)=0 or |c-b| < e [2].

1.3 Inverse Quadratic Interpolation

In this method let us say there are three points xn−2, xn−1, and xn as initial values. The function will be evaluated at each of these points resulting in yn−2 = f(xn−2), yn−1 = f(xn−1), and yn = f(xn), respectively. Assuming that f has an inverse quadratic function g, then xn−2 = g(yn−2), xn−1 = g(yn−1), and xn = g(yn), and so on. This process is done by computing a parabola that goes through these three given points and taking the intersection of the parabola with the x-axis as the new root estimate. Fitting the points with a parabola in y, we have:

$$f^{-1}(y) = \frac{(y-f_{n-1})(y-f_n)}{(f_{n-2}-f_{n-1})(f_{n-2}-f_n)}x_{n-2} + \frac{(y-f_{n-2})(y-f_n)}{(f_{n-1}-f_{n-2})(f_{n-1}-f_n)}x_{n-1} + \frac{(y-f_{n-2})(y-f_{n-1})}{(f_n-f_{n-2})(f_n-f_{n-1})}x_n.$$

Figure 1 Lagrange Polynomial

This form is also called a Lagrange polynomial.

Setting y = 0, the new root estimate, xn+1, will be computed as follows:

$$f^{-1}(y) = \frac{(y-f_{n-1})(y-f_n)}{(f_{n-2}-f_{n-1})(f_{n-2}-f_n)}x_{n-2} + \frac{(y-f_{n-2})(y-f_n)}{(f_{n-1}-f_{n-2})(f_{n-1}-f_n)}x_{n-1} + \frac{(y-f_{n-2})(y-f_{n-1})}{(f_n-f_{n-2})(f_n-f_{n-1})}x_n.$$

Figure 2 Modified Polynomial with y = 0

However, it cannot compute complex roots because it will always cross the x-axis. Also, if the three initial guess values chosen are very far from the root, then this method will not converge [3].

1.4 Brent-Dekker Method

This is a hybrid method that combines bisection, secant and IQI methods with some additional features that make it completely robust and usually very efficient. It is an enclosure method that begins with an initial interval across which the function f changes sign and, as the iterations proceed, determines a sequence of nested intervals that share this property and decrease in length. If f is continuous on the initial interval, then each of the decreasing intervals determined by the method contains a solution, and the limit of the iterates is a solution.

The following data provides a rough outline of how the method works. The method builds upon an earlier method of T. J. Dekker and is the basis of MATLAB's fzero routine. At each iteration, Brent-Dekker's method first tries a step of the secant method or IQI. If this step is unsatisfactory, which usually means too long, too short, or too close to an endpoint of the current interval, then the step reverts to a bisection step. This reverting step guards the method from making huge number of steps with very less increase in the root value. IQI is efficient and finds the root if it exists. But, IQI is time consuming and the conditions for the selection of bisection method increases the number of iterations required than necessary. Although Brent-Dekker method is efficient and robust, there always exists a better method [4].

2. A Better Root-Finding Method (Modified Version of Brent-Dekker)

This Modified version of Brent-Dekker uses False Position and IQI with Bisection method. The efficiency and utility of this method is discussed under Results and Discussion section. False Position is a technique to find zeros of a function. It is similar to Secant method but uses a smaller number of function evaluations than Secant method. It takes a smaller number of steps to converge to root of the equation than Secant method. It converges faster than secant method. Combining it with IQI and reducing the number of conditions as in actual Brent-Dekker method to just one condition for selecting IQI (select IQI only if the last two found roots have a difference of less than half the tolerance) makes this method more efficient and robust than Brent-Dekker method.

Here is the pseudo code of this method:

```
1.  Read initial interval [a,b]
2.  Read tolerance 2t;
3.  if f(a) < f(b): swap(a,b)
4.  c:=a, fc:=fa
5.  repeat
6.  if fa! = fc && fb!=fc && (b-a)<t*b: \\IQI
7.  s = ((fb * fa) / ((fc - fb) * (fc - fa)) * c) + ((fc * fa) / ((fb
    - fc)*(fb - fa))*b)+((fc * fb) / ((fa - fc) * (fa - fb)) * a)
8.  else: //False Position
9.  s=a+fa*(a-b)/(fb-fa)
10. if s is not in [a,b]: //Bisection
11. s=(a+b)/2, fs:=f(s), c:=b
12. if fs*fb:
13. a:=b, fa:=fb
14. else:
15. fa:=fa/2, b:=s, fb:=fs
16. until abs(b-a)
17. b is the required approximate solution
```

This method provided a great reduction in the number of steps when compared to the original Brent-Dekker method on a variety of polynomial functions.

Table -1: Test cases, interval used and root value in the interval using matlab fzero method.

Test cases and values obtained using matlab fzero		
Function	Root Value	Interval Used
0.5e^x - 5x + 2	3.401795	(1,4)
−2x^4 + 2x^3 − 16x^2 − 60x + 100	1.240787	(0,4)
((e^x)*cos(x))-(x*sin(x))	1.225393	(0,3)
x^5−5x+3	0.618033	(0,1)
x3−0.926x2+0.0371x+0.043	0.618033	(0,0.8)
(-9+(sqrt(99+(2*x)-(x*x)))+cos(2*x))	1.211283	(-2,6)
sin(cosh(x))	1.811526	(0,2)
exp(-exp(-x))-x	0.567143	(0,1)

The below table provides a view point on how much execution time the modified better root-finding method and the actual fzero method have taken to get to the result on a standard i5 8th Gen Processor with 8 GB RAM.

Table -2: Modified brent-dekker version vs actual results from Table-1

Modified Brent-Dekker vs fzero method		
	Execution Time (micro sec)	
Function	Modified Better Version	Fzero (Brent-Dekker Method)
0.5e^x - 5x + 2	37	57
−2x^4 + 2x^3 − 16x^2 − 60x + 100	24	204
((e^x)*cos(x))-(x*sin(x))	124	340
x^5−5x+3	43	170
x3−0.926x2+0.0371x+0.043	38	86
(-9+(sqrt(99+(2*x)-(x*x)))+cos(2*x))	93	172
sin(cosh(x))	140	196
exp(-exp(-x))-x	39	50

The graph showing the above details of execution times for different test cases is as follows:

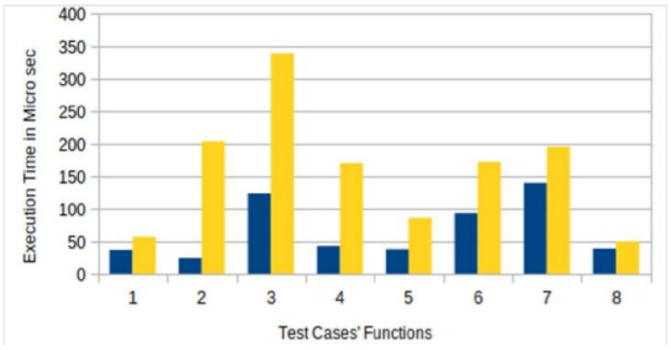

Chart -1: Modified Better Root Finding Method vs Brent-Dekker Method Execution Times for Test Cases in Table-2

As we can see from the above graph, the modified better root-finding algorithm is taking less execution time than the actual brent-dekker for almost every test case presented. Thus, we can confirm that this is a better sequential root finding technique using False Position, IQI and Bisection methods with robustness and efficiency more than Brent-Dekker method.

3. CONCLUSION

In this paper, we have seen how Bisection, Secant and Inverse Quadratic Interpolation methods work and why they might be inefficient in certain ways. We have also seen the working of Brent-Dekker based fzero method and tested it for root values on some polynomial functions. We have also seen how the modified better root finding algorithm works and the time it takes to attain the roots in a much less time and with much better efficiency. One cannot predict which algorithm is always better but it is safe to say that the method presented in this paper is almost always efficient in finding the roots than other methods discussed.

REFERENCES

[1] https://x-engineer.org/undergraduate-engineering/advanced-mathematics/numerical-methods/the-bisection-method-for-root-finding/

[2] I. K. Argyros, S. K. Khattri, “On the Secant Method”, Journal of Complexity, 2013.

[3] Mark James B. Magnaye, “The Inverse Quadratic Interpolation Method for Finding the Root(s) of a Function”, University of Batangas Graduate School.

[4] Richard P. Brent, “Algorithms for Minimization Without Derivatives”, Prentice-Hall Inc., Englewood Cliffs, New Jersy, 1973.