Paper ID OT 03

ALGORITHM FOR NON-LINEAR EQUATIONS

Pradip Jamkar[1], Manjusha Tomar[2], V.M. Kale[3]
1-3[Department of Basic Engineering, Indira College of Engineering & Management, Pune]

Abstract:
The study is aimed at comparing the rate of performance, the rate of convergence of Bisection and Trisection method. These method have been applied in parallel environment. It was then concluded that Trisection method is most effective and gives good result.
Keywords: Parallel Numerical Algorithm, Bisection method, Trisection Method.

INTRODUCTION

The most basic problem in Numerical Analysis (methods) is the root-finding problem. For a given function f(x), the process of finding the root involves finding the value of x for which f(x) = 0. If the function equals zero, x is the root of the function. A root of the equation f(x) = 0 is also called a zero of the function f(x). f(x) may be algebraic, trigonometric or transcendental function. The determination of any unknown appearing implicitly in scientific or engineering formulas gives rise to root finding problem [1]. The common root finding methods include Bisection, Newton- Raphson, False position, Secant methods [4] etc. Different methods converge to the root of different rates. That is, some methods are faster in converging the root than others. The rate of convergence could be linear, quadratic or otherwise. The higher the order the faster the method converges [2].

METHOD

Bisection method is a simple root finding method. It requires prior knowledge of the value of a continuous function at the endpoints of a given interval where the root exist. The function value at the endpoints must have different signs (i.e.). Using the Intermediate Value Theorem, bisection method iterates and reduces the interval into halves, disregarding the where no sign change occurs. It will iterate until it converges to the root or the level of desired accuracy is reached.

ROOT FINDING ALGORITHM

Consider a transcendental equation f (x) = 0 which has a zero in the interval [a,b] and f (a) * f (b) < 0. Bisection scheme computes the zero, say c, by repeatedly halving the interval [a,b]. That is, starting with c = (a+b) / 2. The interval [a,b] is replaced either with [c,b] or with [a, c] depending on the sign of f (a) * f (c). This process is continued until the zero is obtained. Since the zero is obtained numerically the value of c may not exactly match with all the decimal places of the analytical solution of f (x) = 0 in the interval [a,b]. Hence any one of the following mechanisms can be used to stop the bisection iterations:

- C1. Fixing a priori the total number of bisection iterations N i.e., the length of the interval or the maximum error after N iterations in this case is less than $| b-a | / 2^N$.
- C2. By testing the condition $| c_i - c_{i-1}|$ (where i are the iteration number) less than some tolerance limit, say epsilon, fixed a priori.
- C3. By testing the condition $| f (c_i) |$ less than some tolerance limit alpha again fixed a priori.

PARALLEL ALGORITHM

Parallel Bisection Method is a method to convert the bisection algorithm to parallelism by distributed k^{th} intervals of roots over k^{th} cores.
Given f, a[k], b[k], and δ(tolerance)
Step1. Do in parallel for four roots
Step 2. i=1
Step 3: c= [a (k) + b (k)]/2
Step 4: compute f(c) and f [a (k)] in parallel
Step 5: If f(c) =0 or f(c) < δ, then step 9
Step 6: If f(c)*f [a (k)] < 0 , then b (k) = c
Step 7: Else a (k) = c
Step 8: i = i+1, go back to step 3.
Step 9: stop iteration.
Parallel Trisection Method:

Given f, a[k], b[k], and δ(tolerance)
Step1. Do in parallel for four roots
Step 2. i=1
Step 3: c= [2a (k) + b (k)]/3
Step 4: compute f(c) and f [a (k)] in parallel
Step 5: If f(c) =0 or f(c) < δ, then step 9
Step 6: If f(c)*f [a (k)] < 0 , then b (k) = c
Step 7: Else a (k) = c
Step 8: i = i+1, go back to step 3.
Step 9: stop iteration.

RESULT AND DISCUSSION

We implement the parallel numerical algorithm using MATLAB (R2015a). The case study is a real function have a real roots as a form of quartic function $f(x) = x^4+3x^3-15x^2-2x+9$ with tolerance $\delta = 10^{-6}$, where i=number of iterations, a and b are initial values and c= (2a+b)/3. The result analysis by comparing Bisection and Trisection methods we have seen that Trisection method gives good and effective results by taking minimum elapsed time and less iterations.

Table (1) Results of Bisection Parallel Algorithm

	Interval =[-6,-4]		Interval=[-2,0]		Interval =[0,2]		Interval =[2,4]	
I	C	f(c)	C	f(c)	c	f(c)	c	f(c)
1	-5.000000	-106.000000	-1.000000	-6.000000	1.000000	-4.000000	3.000000	30.000000
2	-5.500000	-17.812500	-0.500000	5.937500	0.500000	4.687500	2.500000	-3.812500
3	-5.750000	47.363281	-0.750000	1.113281	0.750000	0.644531	2.750000	9.644531
4	-5.625000	12.834229	-0.875000	-2.157959	0.875000	-1.638428	2.625000	2.135010
5	-5.562500	-2.960190	-0.812500	-0.450668	0.812500	-0.482407	2.562500	-1.024155
6	-5.593750	4.817491	-0.781250	0.349244	0.781250	0.085267	2.593750	0.507859
7	-5.578125	0.898990	-0.796875	-0.046229	0.796875	-0.197589	2.578125	-0.269887
8	-5.570313	-1.037988	-0.789063	0.152628	0.789063	-0.055907	2.585938	0.116032
9	-5.574219	-0.071349	-0.792969	0.053480	0.785156	0.014745	2.582031	-0.077663
10	-5.576172	0.413357	-0.794922	0.003695	0.787109	-0.020565	2.583984	0.019000
11	-5.575195	0.170888	-0.795898	-0.021250	0.786133	-0.002906	2.583008	-0.029378
12	-5.574707	0.049741	-0.795410	-0.008773	0.785645	0.005920	2.583496	-0.005200
13	-5.574463	-0.010812	-0.795166	-0.002538	0.785889	0.001507	2.583740	0.006897
14	-5.574585	0.019463	-0.795044	0.000579	0.786011	-0.000699	2.583618	0.000848
15	-5.574524	0.004325	-0.795105	-0.000979	0.785950	0.000404	2.583557	-0.002176
16	-5.574493	-0.003243	-0.795074	-0.000200	0.785980	-0.000148	2.583588	-0.000664
17	-5.574509	0.000541	-0.795059	0.000190	0.785965	0.000128	2.583603	0.000092
18	-5.574501	-0.001351	-0.795067	-0.000005	0.785973	-0.000010	2.583595	-0.000286
19	-5.574505	-0.000405	-0.795063	0.000092	0.785969	0.000059	2.583599	-0.000097
20	-5.574507	0.000068	-0.795065	0.000043	0.785971	0.000025	2.583601	-0.000003
21	-5.574506	-0.000169	-0.795066	0.000019	0.785972	0.000008	2.583602	0.000044
22	-5.574506	-0.000050	-0.795066	0.000007	0.785972	-0.000001	2.583601	0.000021
23	-5.574507	0.000009	-0.795066	0.000001	0.785972	0.000003	2.583601	0.000009
24	-5.574506	-0.000021			0.785972	0.000001	2.583601	0.000003
25	-5.574506	-0.000006					2.583601	0.000000
26	-5.574506	0.000001						
27	-5.574506	-0.000002						
28	-5.574506	-0.000000						

Table (2) Results of Trisection Parallel Algorithm

	Interval [-6,-4]		Interval [-2,0]		Interval [0,2]		Interval [2,4]	
1	C	f(C)	C	f(C)	C	f(C)	C	f(C)
2	-5.333333333	-53.02469136	-1.333333333	-18.95061728	0.666666667	2.086419753	2.666666667	4.456790123
3	-5.777777778	55.58680079	-0.888888889	-2.556774882	1.111111111	-6.101356501	2.222222222	-12.21018137
4	-5.62962963	14.04244121	-0.592592593	4.416717942	0.814814815	-0.524760039	2.37037037	-8.496399412
5	-5.530864198	-10.59400494	-0.790123457	0.125754549	0.716049383	1.241302096	2.469135802	-5.058594614
6	-5.596707819	5.565978396	-0.855967078	-1.622889924	0.748971193	0.6627874	2.534979424	-2.29649096
7	-5.574759945	0.062866287	-0.834019204	-1.022340713	0.770919067	0.271139589	2.578875171	-0.233085545
8	-5.560128029	-3.540878415	-0.819387289	-0.631785281	0.785550983	0.007611111	2.608139003	1.245111684
9	-5.56988264	-1.144134799	-0.809632678	-0.375778825	0.795305594	-0.169087269	2.588629782	0.250391532
10	-5.573134177	-0.34010794	-0.803129604	-0.207048203	0.78880252	-0.051200651	2.582126708	-0.072947542
11	-5.574218022	-0.071529675	-0.798794222	-0.095423724	0.786634829	-0.01198294	2.5842944	0.034377682
12	-5.574579304	0.018059718	-0.795903967	-0.021390842	0.785912265	0.001080862	2.582849272	-0.037222883
13	-5.574458877	-0.011806931	-0.79397713	0.02779398	0.78615312	-0.003273249	2.583330981	-0.013378443
14	-5.574539162	0.008103777	-0.795261688	-0.004980751	0.78599255	-0.000370454	2.583652121	0.002530301
15	-5.5745124	0.001466701	-0.794833502	0.005950893	0.785939027	0.000597096	2.583438028	-0.008076635
16	-5.574494559	-0.00295792	-0.795118959	-0.001336121	0.785956868	0.000274582	2.583509392	-0.004541482
17	-5.574506453	-8.18E-06	-0.795023807	0.001093216	0.785968762	5.96E-05	2.583556968	-0.002184439
18	-5.574510418	0.000975073	-0.795087242	-0.000526305	0.785976691	-8.38E-05	2.583588686	-0.000612956
19	-5.574509096	0.000647321	-0.795066097	1.36E-05	0.785971405	1.18E-05	2.583609831	0.000434753
20	-5.574508215	0.00042882	-0.795080193	-0.000346351	0.785973167	-2.01E-05	2.583595734	-0.000263725
21	-5.574507628	0.000283153	-0.795075495	-0.000226383	0.785971992	1.17E-06	2.583600433	-3.09E-05
22	-5.574507236	0.000186041	-0.795072362	-0.000146404	0.785972384	-5.91E-06	2.583603566	0.000124316
23			-0.795070274	-9.31E-05			2.583601477	2.08E-05
24			-0.795068881	-5.75E-05			2.583600781	-1.37E-05
25			-0.795067953	-3.38E-05				
26			-0.795067334	-1.80E-05				
27			-0.795066922	-7.51E-06				

REFERENCES

[1] Biswa Nath Datta(2012), lecture notes on numerical solution of root finding problems. www.math.niu.edu/dattab.

[2] Charles A and W. W. Cooper (2008), Non-linear power of adjacent extreme points methods in linear programming, Econometrica Vol.25 (1) pp. 132-153.

[3] A.J. Maeder and S.A. Wynton, Some parallel for polynomial root finding, Journal of computation and applied mathematics 18, 71-81, North Holland, 1987.

[4] Srivastava, R.B. and Srivastava, S. (2011), Comparison of Numerical Rate of Convergence of Bisection, Newton and Secant Methods. Journal of Chemical, Biological and Physical Sciences, Vol.2 (1) pp 472-479

[5] Guy E.Blellochand Bruce M.Maggs, Parallel Algorithms, Carnegie Mellon University, ACM Computing Surveys (CSUR), 1996.

[6] R.L.Burden, J.D. Faires, Numerical Analysis, 9th Edition, Brooks/Cole, 2011.

[7] R.L.Burden, J.D. Faires, Numerical Analysis, 3rd Edition, PWS Publishers, Boston, MA, 1985.