IOSR Journal of Mathematics (IOSR-JM)
e-ISSN: 2278-5728, p-ISSN: 2319-765X. Volume 11, Issue 4 Ver. II (Jul - Aug. 2015), PP 32-36
www.iosrjournals.org

Parallel Hybrid Algorithm of Bisection and Newton-Raphson Methods to Find Non-Linear Equations Roots

Khalid Ali Hussein[1],Abed Ali H. Altaee[2],Haider K. Hoomod[3]
1,3(Department of computer/Al-Mustansiriyah University/ Baghdad –Iraq)
2(Department of Mathematics/ Al-Mustansiriyah University/ Baghdad –Iraq)

Abstract: *In this paper a new parallel Hybrid algorithm is introduced which is based on the Bisection algorithm and Newton-Raphson algorithm. The proposed Hybrid algorithms helps in finding real roots of single non-linear equations in less number of iterative operations and reduce the time of solving. These methods have been applied in parallel environment. A description of the algorithms details and comparison between them is included in this work.*

Keyword: *Parallel Numerical Algorithm, bisection Method, Newton-Raphson Method, HybridAlgorithm, Parallel Hybrid Algorithm, parallel finding roots.*

I. Introduction

The topics of parallelism areinterested of many applications in Mathematics and Computer sciences,Physics, Chemistry, Engineering branches, etc.Most of the algorithms are sequential, that is, they specify a sequence of steps in which each step consists of a single operation. These algorithms are well suited to today's computers, which basically perform operations in sequential fashion. Although the speed at which sequential computers operate has been improving at an exponential rate from any years, the improvement is now coming at greaterand greater cost. As a consequence, researchers have sought more cost-effective improvements [4].As more computers have incorporated some form of parallelism, the emphasisin algorithm design has shifted from sequential algorithms to parallel algorithms, that is, algorithms in which multiple operations are performed simultaneously [5].

Maedersubmitted a diagram of parallel Newton-Raphson method without explain the algorithm.They got the results by simulating processes running in parallel duringexecution of a serial program on a conventional SISD VAX 11-780 computer. Parallel process execution times were measured using operating system utilities which were separated checked for reliability and consistency.In the case of single-root methods, implementations of the parallel Newton-Raphson and secant methods, a two processor algorithm was used.[1] So,Ioana adept the serial algorithm of Bisection method at parallel execution with 3 processors, she consider the real function f(x) has only one zero in the interval [a,b] ,by means of three sequences a_n, b_n and c_n, every processor generating one of them[2]. In this paper, we introduce a new way to parallel Bisection method, and we proposed a new parallel hybrid algorithm to finding roots of real function. It is organized as follows: section 2 establishment to root-finding, Bisection, Newton-Raphson methods and Hybrid method in sequentially algorithms; section 3a new approach to convert sequentially numerical algorithm to parallel numerical algorithm of Bisection, Newton-raphson and Hybrid methods ; section 4 results and discussions.

II. Root-finding Algorithm

The root-finding problem is one of the most important computational problems. It rises in a wide variety of practical applications in mathematics, physics, chemistry, biosciences, engineering, etc. As a matter of fact, determination of any unknown appearing implicitly in scientific or engineering formulas gives rise to a root-finding problem.A root-finding algorithm is a numerical method to finding a value x such that f(x)=0, for given function f. Numerical root-finding methods use iteration, producing a sequence of numbers that hopefully converge towards a limit (the so-called "fixed point") which is a root. The first values of this series are initial guesses. The method computes subsequent values based on the old ones and the function f.

2.1. Bisection Method:

The Bisection (Binary Search) method which is based on theIntermediate value Theorem (IVT).The bisection method in mathematics is a root-finding method that repeatedly bisects an interval and then selects a subinterval in which a root must lie for further processing. It is a very simple and robust method, but it is also relatively slow. Because of this, it is often used to obtain a rough approximation to a solution which is then used as a starting point for more rapidly converging methods. The method is also called the interval halving method, the binary search method, or the dichotomy method [6][7][8]. If f(x) is continuous on [a,b] with f(a) and f(b) of opposite sign .The main idea of bisection method that[9]:

Let c= (a+b) / 2 (midpoint of [a,b]), compute f(c)
If f (c) =0 , then c is a root
If f(a) . f(c) < 0 , a root exists in [a,c]
Else the root exists in [c,b] .

2.2. Newton – Raphson Method:

In numerical analysis ,Newton-Raphson method is a very popular numerical method used for finding successively better approximations to the zeroes of a real-valued function f (x) = 0 .[9]

$X_{n+1} = x_n - [f (x_n) / df(x_n)]$ (1)

This method is distinguished from the methods by the fact that it requires the evaluation of both the function f(x) and the derivative f ' (x) , at arbitrary points x.

2.3HybridAlgorithm [3]

This algorithm is a new approach to compute the roots of nonlinear equations f(x)=0, by propose hybrid algorithm between the Bisection algorithm and Newton-Raphson algorithm. It's take a first approximation by apply two times the Bisection method and complete a correct approximation by use the Newton-Raphson method.

Given f ,df , a , b , and $\delta=10^{-6}$ (tolerance)
Step 1 : for i=1 to 2
Step 2 : $x_i = (a+b) / 2$
Step 3 : If f (x_i) = 0 or f (x_i) < δ , then step 10
Step 4 : If f (a) * f (x_i) < 0 , then b= x_i
Step 5 : Else a = x_i
Step 6 :end for
Step 7 : x = x_i - [f (x_i) / df(x_i)]
Step 8 : If f (x) < δ , then go to step 10
Step 9 : x_i = x , and go to step 7
Step 10 : stop iteration .

III. Parallel Algorithms

In this section we introduce a new approach to convert the sequential numerical algorithms to parallel numerical algorithms

.

3.1 Parallel Newton-Raphson Algorithm

Meaider [1] proposed Parallel Newton-Raphson Algorithm by a diagram without explains it's, we put the parallel algorithm as follows:

Given f ,df, a[k] ,b[k] and δ (tolerance) .
Step 1 : do in parallel for four roots
Step 2 : i=1
Step 3 : x_0 = (a[k] + b[k]) / 2
Step 4 : compute f (x_o) and df(x_o) in parallel
Step 5 : x = x_0 - [f (x_o) / df(x_o)]
Step 6 : If f(x_o) | < δ , then go to step 9
Step 7 : x_0 = x
Step 8 : i = i + 1 , go back to step 3
Step 9 : stop iteration .

3.2 parallel Bisection Method:

We proposed a new approach to convert the Bisection algorithm to parallelism by distributed k^{th} intervals of roots over k^{th} cores, also we making parallelism to find the images of this intervalsends.

Given f , a[k] , b[k] , and δ (tolerance)
Step 1 : 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 (a[k]) * f (c) < 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 .

3.3 Parallel Hybrid algorithm

In the hybrid algorithm we have two approximations, we convert itto parallelism by distributed the first approximation in k^{th} intervals for k^{th} cores with finding images of intervals ends in parallel , also the results comes from first parallel algorithm enter in the second parallel approximation by finding f(x) &df(x) in k^{th} intervals in parallelism .

Given f ,df , a[k] , b[k] and δ (tolerance)
Step 1 : do in parallel for four roots
Step 2 : for i=1 to 2
Step 3 : x_i = (a[k] + b[k]) / 2
Step 4 : compute f(x_i) and f(a[k]) in parallel
Step 5 : If f (x_i) = 0 or f (x_i) < δ , then step 15
Step 6 : If f (a) * f (x_i) < 0 , then b[k] = x_i
Step 7 : Else a[k] = x_i
Step 8 : end for
Step 9 : for i = 1 to n
Step 10 : do in parallel compute f (x_i) and df(x_i)
Step 11 : x = x_i - f (x_i) / df(x_i)
Step 12 : If f (x_i) < δ , then go to step 15
Step 13 : x_i = x
Step 14 : end for
Step 15 : stop iteration .

IV. Results and Discussions

We implement the parallel numerical algorithms on Dell i7 core Intel (4 cores)computer using Matlab ver. 7.12.0.635 (R2011a) 32-bit. The case study is real function (have real roots) as a form quartic function $f(x)= a x^4 + b x^3 + c x^2 + d x + e$,we take as for example $f(x)= x^4 +3x^3 – 15x^2 -2^x + 9$,with tolerance $\delta=10^{-6}$, where i= number of iterative, a & b are initials values and c= (a + b) / 2.

Tables (1), (2),and (3) show the results of implement the above three parallel algorithms. From these results, the parallel Hybrid method is better than parallel bisection method with respect to number of iterative and elapsed time, but the result between parallel Hybrid and parallel Newton-Raphson is convergent.

The serial Bisection method needs $\log_2$ [(b-a)/Ɛ] function evaluations, additions and multiplications to enclose the zero in an interval of length Ɛ [2]. The number of operations in parallel bisection method is $\frac{1}{m}\log_2$ [(b-a)/Ɛ] where, (m) represent who many cores we used to the parallel system. The serial Newton-Raphson method needs $\log_2\Omega$, where Ω is the number of iterativeoperations needs to reach f(x) <δ (tolerance). So the number of operations of Hybrid is needs $\log_2$ [(b-a) / Ɛ] + $\log_2\Omega$, thus the number of operations to parallel Hybrid method needs to $\frac{1}{m}$ {$\log_2$[(b-a) / Ɛ] + $\log_2$ Ω} where m represent who many cores we used to the parallel system.

Table (1) Results of Parallel Bisection 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						
time is0.166101 seconds ; Tolerance = 10^{-6}								

Table(2) Result of Parallel Newton-Raphson Algorithm .

	Interval =[-6,-4]				Interval =[-2,0]			
i	x_o	$f(x_o)$	$df(x_o)$	X	x_o	$f(x_o)$	$df(x_o)$	x
1	-5.000000	-106.0000	-127.000000	-5.834646	-1.000000	-6.000000	33.000000	-0.818182
2	-5.834646	73.066818	-315.090292	-5.602754	-0.818182	-0.599959	26.379414	-0.795438
3	-5.602754	7.102661	-254.900600	-5.574890	-0.795438	-0.009493	25.544485	-0.795067
4	-5.574890	0.095019	-248.095517	-5.574507	-0.795067	-0.000003	25.530836	-0.795067
5	-5.574507	0.000018	-248.002609	-5.574506	-0.795067	-0.000000	25.530832	-0.795067
6	-5.574506	0.000000	-248.0026091	-5.574506				

	Interval =[0,2]				Interval =[2,4]			
i	x_o	$f(x_o)$	$df(x_o)$	X	x_o	$f(x_o)$	$df(x_o)$	x
1	1.000000	-4.000000	-19.000000	0.789474	3.000000	30.000000	97.000000	2.690722
2	0.789474	-0.063351	-18.106575	0.785975	2.690722	5.878503	60.361313	2.593333
3	0.785975	-0.000051	-18.077273	0.785972	2.593333	0.486799	50.492960	2.583692
4	0.785972	-0.000000	-18.077250	0.785972	2.583692	0.004514	49.557809	2.583601
5					2.583601	0.000000	49.549009	2.583601

Time is 0.107681 seconds ; Tolerance = 10^{-6}

Table (3) Result of Parallel Hybrid Algorithm.

Interval =[-6,-4]

	First Approximate By Use Bisection Algorithm					Second Approximate By Use N-R Algorithm			
i	C	f(c)	a	b		C	x	f(c)	df(c)
1	-5.00000	-6.00000	-4.00000	-106.00000	1	-5.500000	-5.577362	-17.812500	-230.250000
2	-5.50000	-6.00000	-5.00000	-17.812500	2	-5.577362	-5.574510	0.709056	-248.695592
					3	-5.574510	-5.574506	0.000987	-248.003557
					4	-5.574506	-5.574506	0.000000	-248.002591

Interval =[-2,0]

	First approximate by use parallel bisection algorithm					Second approximate by use parallel N-R algorithm			
i	C	A	b	f(c)		C	x	f(c)	df(c)
1	-1.000000	-2.000000	0.000000	-6.00000	1	-0.500000	-0.902542	5.937500	14.750000
2	-0.500000	-1.000000	0.000000	5.93750	2	-0.902542	-0.802236	-2.955698	29.466734
					3	-0.802236	-0.795103	-0.183987	25.794107
					4	-0.795103	-0.795067	-0.000934	25.532176
					5	-0.795067	-0.795067	-0.000000	25.530832

Interval =[0,2]

	First approximate by use parallel bisection algorithm					Second approximate by use parallel N-R algorithm			
i	C	A	b	f(c)		C	x	f(c)	df(c)
1	1.000000	0.000000	2.000000	-4.00000	1	0.500000	0.828947	4.687500	-14.250000
2	0.500000	0.000000	1.000000	4.68750	2	0.828947	0.786342	-0.784178	-18.405580
					3	0.786342	0.785972	-0.006687	-18.080369
					4	0.785972	0.785972	-0.000001	-18.077250

Interval=[2,4]

	First approximate by use parallel bisection algorithm					Second approximate by use parallel N-R algorithm			
i	C	A	b	f(c)		C	x	f(c)	df(c)
1	2.000000	1.000000	3.000000	-15.0000	1	2.500000	2.591317	-3.812500	41.750000
2	2.500000	2.000000	3.000000	-3.81250	2	2.591317	2.583658	0.385218	50.296824
					3	2.583658	2.583601	0.002845	49.554556
					4	2.583601	2.583601	0.000000	49.549009

Elapsed time is 0.118284 seconds.

References

[1]. 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.

[2]. IoananChiorean, Parallel Numerical Methods for Solving Nonlinear Equations, StudiaUniy. "BABES-BOLYAI", Mathematica, Vol.XLVI, NO.4, 53-59, Dec. 2001.

[3]. Khalid A. Hussein, Abed Ali H. Altaee and Haider K. Hoomod,A new Approach to Find Roots of Nonlinear Equations by Hybrid Algorithm to Bisection and newton-Raphson Algorithms, Iraqi journal for Information Technology , Vol.7th No.1, 2015.

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

[5]. Guy E. Blellochand Bruce M. Maggs , Parallel Algorithms , Carnegie Mellon University, ACM Computing Surveys, CRC Press, Vol.28 ,No.1 , 1996.

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

[7]. R.L. Burden; J.D. Faires , NumericalAnalysis (3rd ed.), PWS Publishers, Boston, MA, 1985.

[8]. http://siber.cankaya.edu.tr/NumericalComputations/ceng375/node32.html .

[9]. J.H.Mathews and K.D.Fink , Numerical Methods Using Matlab , Fourth Edition , Pearson prentice Hall , 2004.

[10]. Soram R. , and others , On The Rate of Convergences of Newton-Raphson Method , the international journal of engineering and sciences (IJES) , vol.2, Issue 11 , pp. 5-12, October 2013.