International Journal of Scientific Research in Computer Science and Engineering
Research Paper
Volume-5, Issue-5, pp.57-62, October (2017)
E-ISSN: 2320-7639

Solution of Differential Equations by Parallel Processing and Analysis of Performance Improvement

S.Dubey[1*], S. Prajapat[2], R. Verma[3], R. Jhaggar[4]

[1*]International Institute of Professional Studies, Devi Ahilya University, Indore, India
[2]International Institute of Professional Studies, Devi Ahilya University, Indore, India
[3]International Institute of Professional Studies, Devi Ahilya University, Indore, India
[4]International Institute of Professional Studies, Devi Ahilya University, Indore, India

[*]*Corresponding Author: shubhamdubey1312@gmail.com*

Available online at: www.isroset.org

Received 04th Jun 2017, Revised 20th Jun 2017, Accepted 12th Aug 2017, Online 30th Oct 2017

***Abstract*—** When solving the problems based on more than two variables most of the time relative changes on the variables need to be calculated. The degree of dependency among variables and number of variable quantities make solution finding more complex. It is very obvious to get the help of differential equations. When we deal with the functions frequently changing by nature, so it is highly likely to find local maxima, local minima, slope of tangent and value of the function at given point. Finding solution is not a single process but a set of processes where some binary operations, repetitions of the operations, checking the consistency of function within the interval etc take place. Here it becomes very important that the time taken to execute the problem of differential equation must be least. Thus the needs of pipelining, parallel processing, speculative computation etc become crucial factor. So requirement of solution of differential equations become very necessary. The degree of equation, complexity of equation, no of variables etc are the parameters those play very important roles during the solution process. When this task is assigned to serial processors the time taken to solve a problem is greater than the time taken by the parallel processors to do the same task. Here in this research we are calculating the time taken by serial processors and by parallel processors too. Although entry of vector processors and array processors improved the performance very much but since performance is the subject to frequently improvement. So putting that context in mind our study has tried to minimize the time taken for solving the differential equation by dividing a big problem in small ones in such a way so that dependency for data among processors i.e. communication will be very less. Means research has tried to improve the ratio of computation over communication ratio.

***Keywords*—** Differential equation, Parallel Processing, Core, SIMD, Euler's Method, maxNumCompThreads , Execution time.

I. INTRODUCTION

Differential equations always play a vital role in the field of mathematics and calculus. In order to calculate the rate of change in one quantity with respect to the other one, it's really obvious need of differential equations. As differential equation used in many fields like physics, chemistry and biology, so solution of differential equation opted efficiently and quickly as well. For complex differential equation the sequential approach of solving differential equation takes long time. Finding solution is not a single process but a set of processes where some binary operations, repetitions of the operations, checking the consistency of function within the interval etc take place. Here it becomes very important that the time taken to execute the problem of differential equation must be least. Thus a vital need of Pipelining, Performance improvement approaches, Parallel Processing etc are demanded. There are many places where differential equation used in the critical situation for example weather forecasting, treatment of diseases like analysis of heart beat, infection evolution, cancer etc. so there must be quick output from differential equation required which is acquired by the parallel approach to solve the differential equation faster than sequential approach. In order to solve ordinary differential equation, the Euler method is used which is a first order numerical procedure and it takes the equation with a given initial value [1][2]. It is the simplest Runge-Kutta method and most basic method for numerical integration of ordinary differential equations [3]. This Euler approach is the simplest method which gets the initial position of the physical quantity and function [4]. When dealing with fractional differential equation the approach of solution for leas computation will become very crucial [1]. These initial positions must have a finite value and initial point. Our study is based on Euler method to solve the differential equation with given initial point, starting value, ending value, step value and function.

Although Array and vector processors were capable to gain through put much more than the sequential processors. Where more than one, operation can be executed parallel.
In Weather forecasting, calculation of speed of storm and with the speed calculation of the arrival time /collision time of it to a city etc need to be done, in order to minimize the harm. If these types of complex calculations are executed with the sequential approach then it may take days, weeks and months. So there is need of parallel approach to solve the differential equation that gives output within a time limit. Section II holds the brief about the study, behind our study and section III where methodology by which implementation has been done, is discussed. Section IV is the place where the results and table figures are taken place. Last and section of Conclusion is Section V.

II. Related Work

In [5] Authors has worked with using Adomian decomposition method (ADM) and Modified decomposition method (MDM) in order to find solution of systems of nonlinear partial differential equations .The Discussed ADM decomposition methods has been applied to reconstruct first and second order initial value problems, which opts solution by first and second transformation based approach. The results shows both the methods were used in methodology were very easy and effective to provide the solution. Nonlinear systems of simultaneous differential equations were introduced and were solved by ADM and MDM methods.

Author V. Rajaraman has discussed few methods to solve differential equations in Computer Oriented Numerical Methods. FORTRAN and C implementations of the algorithms have been mentioned in the literature. Euler's method, Runge- Kutta Method, Predictor corrector method was discussed in brief [6].

This Study [7] emphasized that the error on solving differential equation can be reduced a bit more if reduced step size will be used. Research titled as "Study of Numerical Analysis – Differential Equation " has concluded that if linear differential equation has been used the method gives no error. The proof of this thing has been supported as the 2nd derivative of line will be zero [7].

In order to performance improvement we can't ignore the SIMD and MIMD base computers those are already capable enough to solve a huge problem within few instants. The discovery of SIMD systems brings revolution in the field of calculations and computer science as well [8] .

Although there are different language to work with but some certain features of R attract the users to work with it. R is not only a graphical language but t language of some readymade functions those can be used during execution directly and dynamically [9].

III. Methodology

A. Terminology

- Euler's Method: This method is used to solve the ordinary differential equation with an initial value. It is the first order numerical procedure. It gets five parameters. This method is the iterative method in which, for each step it puts the current value of position and step value in the function and store its results in the array. After reaching through each step value of it.
- euler (t_0, t_d, s_v, f, e_v)

1. t_0 = Initial Point.
2. t_d = Steps to reach at final point.
3. s_v = Starting value for differential equation.
4. f = Function to evaluate differential equation.
5. ev = Ending value for differential equation.

- maxNumCompThreads(N) : This is the MATLAB predefined method which set the maximum number of computational threads to N. It divides the task into equivalent number of threads. If we do not pass parameter to this function it automatically set computational threads on the basis of default setting in MATLAB. We can pass parameters on the basis of how many cores available in our system in which we run that function. This function is used to divide our task into number of parts and execute individual part with number of cores so that it speedup performance.
- feval() : This function executes the function which is specified in the argument in the Automation server and attach to handle h that is the first argument of this function.
- eval(): This eval() function take string as parameter which the valid arithmetic expression and return result after evaluating the expression
- axes(): This function draws an axes graphics object in the current figure which is specified in the function arguments.
- set(): This set function sets up the named properties to the property name specified in the argument list identified by handle h.
- get(): It returns the value of the property name that is specified in the argument of this function which is identified by the handle h.

- tic(): This is timing function which starts the timer to calculate the time required by our program.
- toc(): This function stops the timer which is started by the toc function and returns the total time required to execute the function.

B. *Action Summary at GUI and Computation*

User has to give five values in the fields of GUI. for using this five fields get function is used which return the value entered by user and store it into the corresponding variables and this variables are used as global variable so that it can be used outside the function, each fields are converting into the double value which are in the form of string data type except function field. Then two approaches are used one is sequential approach which solve the differential equation by using one core for execution and return results and the second one is parallel approach which solve the differential equation by using two cores for execution and it will also return results. Figure 1 gives brief about the user interface.

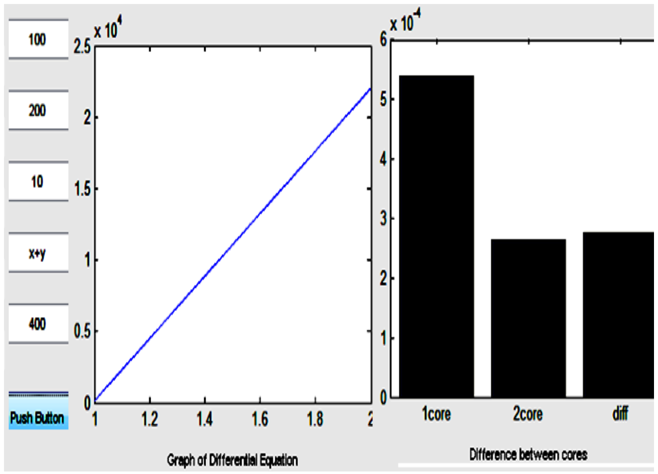

Figure 1. GUI for execute a given function

C. *Sequential approach for solving differential equation:*

After getting input from user *maxNumCompThreads(1)* function will be called. It sets the computational threads to 1. Later on it call the *Euler's* method to solve the differential equation. In this method *tic* function is used. That starts timer for calculating time required in our program to execute then create steps from initial value to end value with the given step value and assign it in another variable called it as stepping array which stores array of values. After this i.e. by assigning the value of initial point to another value, call it as position array which contain array of values of positions among each steps. Then it iterates the loop from one to length of stepping array. After every iterations a function call has been made using below parameters.

1. Each elements of stepping array per iteration.
2. Each elements of position array per iteration.
3. Function which is entered by user.

This function call is used under the *feval* function which handles the calling and execution of function in the automation server. The server returns output from the function in the cell array. In this function nested function call takes place. It calls another function *eval* which is used to evaluate the function that is the expression in the form of string data type and it returns results after executing given expression. This *eval* function's output is the output of the function which is called in the iteration. Then this result is multiplied with the stepping parameter and adding it with the current element of position array. After the iteration, *toc* function has been used. That stops the timer and returns the total time required from starting to end of function execution and this time stored in another variable which is the output of this *Euler's* function with the stepping array and the position array.

D. *Parallel approach for solving differential equation:*

Again this *Euler's* function executes after the setting number of computational threads = 2 with the help of *maxNumCompThreads (2)* function. *Euler's* method gives output of time required to execute this function with the help of two threads along with position array and stepping array. The output of time required by both the *Euler's* function has been stored in another array and subtraction of these output has been calculated to get difference of the time required to solve the given problem by one thread and the two threads. After execution of both the approaches the analysis over both the approaches has been done on the basis of performance improvement with the help of graphical representation.

By using *bar* function that plots the bar graph for graphical representation of performance improvement. There are few graphs taken using data from tables. It has been done by using the *set* function which takes the three arguments.

1. gca which is used to handle the bar graph in the current axes.
2. 'XTickLabel' which sets the names of bar in the graph at x axis.
3. Array of names which associates with each bar in the graph.

It creates the axes graphics object pointing to the specified axes which is given in the axes as parameter. So it will plot the bar graph to the axes1. After this *plot* function plot the graph of solution of differential equation which takes different position of the objects as a position array variable. Then again *axes* function, used to plot the graph in the axes2.

IV. Results and Discussion

A. *Analysis on the basis of varying ending value by taking function = x + y*

Following values are identical for this analysis

Initial point = 10, Starting value = 10, Step Value = 10 and function = x + y here table 1 holds data about the tome difference between cores of the system.

Table1. Time difference between cores with varying ending value

S. No.	Ending Value	Time difference in second
1	100	0.0002
2	250	0.0002
3	500	0.0003
4	750	0.0005
5	1000	0.0014
6	1500	0.0017
7	2000	0.0031
8	5000	0.0322
9	10000	0.0394
10	100000	0.496

Figure 2 shows the time difference nature between cores of the processor, to execute the differential equation, by shifting the ending values.

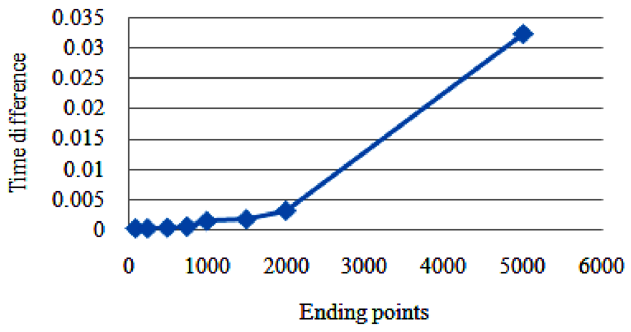

Figure 2. Execution Time difference between with variable edining value

B. Analysis on the basis of varying step value by taking function = x + y

Values are identical for this analysis: Initial point = 10, Starting value = 10, Ending Value = 100, function = x + y Table 2 shows the difference between times taken for both the cores on varying steps. Here step is variable quantity thus all changes were reflected over steps, but are counted with respect to other entities also.

Table 2. Time difference between cores with varying step value

S. No.	Step Value	Time Difference
1	100	0.1904 sec
2	50	0.1556 sec
3	25	0.1349 sec
4	20	0.0029 sec
5	10	0.0014 sec
6	5	0.0010 sec
7	1	0.0050 sec

Here figure 3. Shows graphical representation of time difference between the cores when we are varying step values.

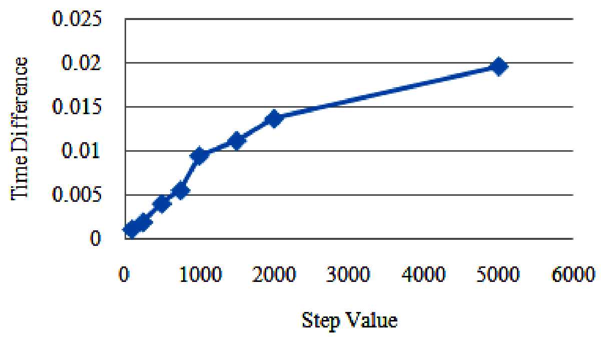

Figure 3. Execution Time difference between with variable step value

C. Analysis on the basis of varying ending value by taking function = x ^ y

Given values are identical for this analysis: Initial point = 10, Starting value = 10, Step Value = 10 ,function = x ^ y

Table 3. Time difference between cores with varying ending value

S.No.	Ending Value	Time difference
1	100	0.0012 sec
2	250	0.0020 sec
3	500	0.0041 sec
4	750	0.0056 sec
5	1000	0.0095 sec
6	1500	0.0112 sec
7	2000	0.0137 sec
8	5000	0.0196 sec
9	10000	0.0250 sec
10	100000	0.0413 sec

Figure 4 is the graphical representation of nature of difference observed in table 3.

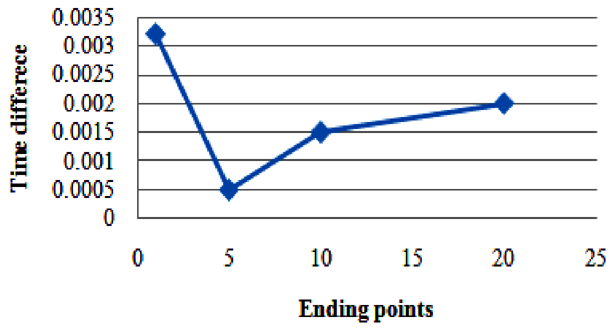

Figure 4. Execution Time difference between with variable ending value

D. Analysis on the basis of varying step value by taking function = x ^ y

For analysis values are monotonic: Initial point=10, Starting value = 10, Ending Value = 100, function = x ^ y. Values of difference opted from analysis are mentioned in table 4. Since Power function makes major difference thus we have taken it as a benchmark function

Table 4. Time difference between cores with varying step value

S. No.	Step Value	Time Difference
1	100	0.2188 sec
2	50	0.1345 sec
3	25	0.0608 sec
4	20	0.0020 sec
5	10	0.0015 sec
6	5	0.0005 sec
7	1	0.0032 sec

Nature of the difference is shown in figure 5against table 4.

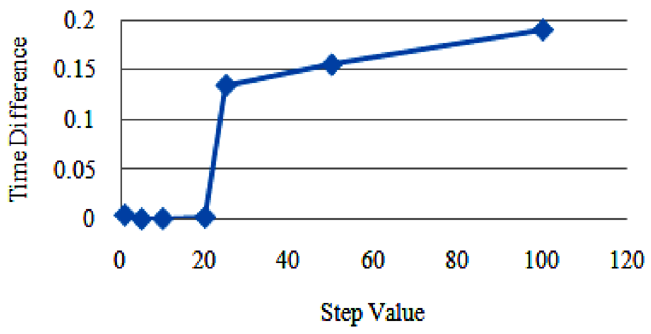

Figure 5. Execution Time difference between with variable step value

E. Analysis of communication over computation

Taking 2 threads for solving differential equation with varying difference between the ending value and the starting value, some results were gained those are in table 5. Set initial value to 10, Step Value to 10 and function to x + y.

Table 5. Difference in time taken by cores when 2 threads are used on varying difference between starting and ending value

S. No.	Difference in ending and starting Values	Time Difference
1	90	0.0003 sec
2	240	0.0004 sec
3	490	0.0006 sec
4	740	0.0019 sec
5	990	0.0036 sec
6	1490	0.0090 sec
7	1990	0.0144 sec
8	4090	0.0205 sec
9	9990	0.0241 sec
10	99990	0.0293 sec
11	999990	-0.0713 sec

On taking 16 threads for solving differential equation with varying difference between the ending value and the starting value table 6 has been gained. Set initial value to 10, Step Value to 10 and function to x+ y.

Table 6. Difference in time taken by cores when 16 threads are used on varying difference between starting and ending value

S. No.	Difference in ending and starting Values	Time difference
1	90	0.0001 sec
2	240	0.0003 sec
3	490	0.0006 sec
4	740	0.0019 sec
5	990	0.0036 sec
6	1490	0.0090 sec
7	1990	0.0144 sec
8	4090	0.0205 sec
9	9990	0.0241 sec
10	99990	0.0293 sec
11	999990	-0.0713 sec

Figure 6 give details about the impact of allocation so many threads over limited threads allocation.

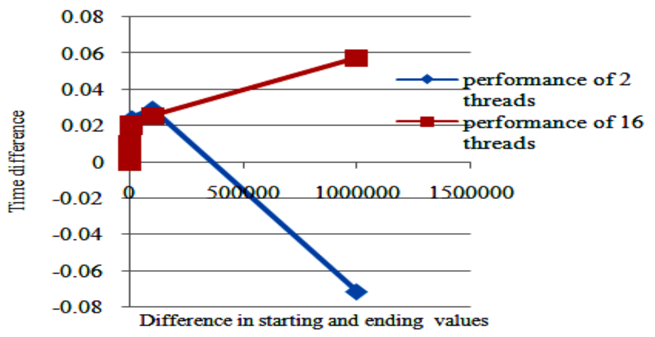

Figure 6. Impact of communication and computation ration in performance

V. CONCLUS ION AND FUTURESCOPE

The aim of study was analysis of sequential and parallel processing based solving approaches with varying ending vales and step values. Co conclusion of the paper is for same number of cores number of threads will proportional to performance. As increasing the number of cores with the number of threads it will give high performance than the two cores. After getting tables of result section some certain conclusions have been opted those are:

- If difference between starting value and ending value is large high performance improvement has been gained due to a better ratio of communication over computation.
- On the basis of different functions give different performance improvement. The time taken to solve the differential equation by single core is 1.3 seconds and by two cores it is 1.2 seconds, for same differential equation. This paper concludes that the performance improvement by solving the differential equation by two cores is 1.08 times.
- It is also concluded that.

As well as future work extension is to be concern it's really scope of extension since performance improvement is a subject of increment. Thus for minimizing communication and increasing computation work can be done. Then prediction based equations can be targeted for future work.

REFERENCES

[1] A. Ma, *"On Improving Euler Methods for Initial Value Problems"*, Scholar Research Library, *Nigeria,* ISSN 0975-508X, pp. 369-379, 2010.

[2] A. Ochoche, "*Improving the Improved Modified Euler Method for Better Performance on Autonomous Initial Value Problems* ", Leonardo Journal of Sciences, Nigeria, ISSN 1583-0233, Issue 12, pp. 57-66, 2008.

[3] S. P. Mondal, S. Roy and B. Das,*"Numerical Solution of First Order Linear Differential Equations in Fuzzy Environment by Runge-Kutta-Fehlberg Method and Its Application"*, International Journal of Differential Equations, Vol. 2016, Article ID 8150497, 2016.

[4] M. A. Akanbi, *"A Third Order Euler Method for Numerical Solution of Ordinary Differential Equation"*, ARPN Journal of Engineering and Applied Sciences, Nigeria, ISSN 1819-6608, Vol. 5, Issue 8, pp. 42-49, 2010.

[5] S. Fadugba, B. Ogunrinde, T. Okunlola, *"Euler's Method for Solving Initial Value Problems in Ordinary Differential Equations"*, The Pacific Journal of Science and Technology, Nigeria, Vol. 13, Issue 2, pp. 152-158, 2012.

[6] V. Rajaraman, "*Computer Oriented Numrical Methods"*, Prentice – Hall of India PHI Publications , Edition 3, pp 164-183, 2004 .

[7] R. Jaiswal, A. A. Pathan ,"*Study of Numerical Analysis – Differential Equation*", International Journal of Advanced Research in Computer Science and Software Engineering,Volume 5, Issue 10, pp. 709- 713,2015.

[8] A. Rajput, B. Ishwarkar, S.Kadhao, "*Parallel Processing Unit with MIMD Architecture*", International Journal of Advanced Research in Computer Science and Software Engineering,Volume 4, Issue 4, pp. 1055-1060, 2014..

[9] K. Soetaert, T. Petzoldt and R. W. Setzer, *"Solving Differential Equations in R"*, The R Journal, ISSN 2073-4859, Vol. 2, Issue. 2, pp. 5-15, 2010.

[10] R. S. Kareem, *"Numerical Methods for Fractional Differential Equations"*, International Journal of Computer Science and Network Security, **Iraq**, VOL. 14, Issue. 1, pp. 42-45, 2014.

[11] S. Dubey, R. Jhaggar, R. Verma, D. Gaur, *"Encryption and Decryption of Data by Genetic Algorithm"*, International Journal of Scientific Research in Computer Science and Engineering, Vol.5, Issue.3, pp.47-52, 2017.

[12] S.Dubey, R. Jhaggar, N. Jhariya, A. Thakur, *"System for Providing News Associated with Location"*, International Journal of Scientific Research in Computer Science and Engineering, Vol.5, Issue.3, pp.25-29, 2017.

[13] S. Dubey, R. Jhaggar, R. Verma, D. Gaur, *"Encryption and Decryption of Data by Genetic Algorithm"*, International Journal of Scientific Research in Computer Science and Engineering, Vol.5, Issue.3, pp.47-52, 2017.

Authors Profile

Dr. Shaligram Prajapat has been working in academics since past 2 decades. Before joining Devi Ahilya University, he has worked for Pioneer Institute, Shri Gujarati Samaj Indore, Govt.Holkar ScienceCollege, Indore. He has served these organizations in capacity of Reader /Associate Professor, Sr. Lecturer, Asst. Professor/ Lecturer. In research career, with Ph.D. (2012 - 2016) from Maulana Azad National Institute of Technology (M.A.N.I.T.) (An Institute of national importance) in Computer Applications from Bhopal India (2016) and Master of Philosophy (Computer Science) (2012) from Devi Ahilya University Indore.He got appreciation for research work with highest google scholar citation in DAVV on 26 January 2017.He has also been awarded as a young investigator award by IRNet in 2012.

Mr. S Dubey pursed Higher School from Govt. Boys Higher Secondary School, Rewa, India In 2011. He is currently pursuing Master of Technology from International Institute of Professional Studies, Devi Ahilya University, Indore. He is a life member of IAENG & IAENG computer society since 2015. He has worked on various projects. His main research work focuses on Parallel Processing, Cryptography Algorithms, Web Apllications and Privacy in network, Data Mining. He has 3 years of teaching experience.