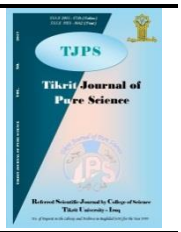
Tikrit Journal of Pure Science 23 (6) 2018

ISSN: 1813 – 1662 (Print)
E-ISSN: 2415 – 1726 (On Line)

TJPS

TIKRIT JOURNAL OF PURE SCIENCE

Journal Homepage: http://main.tu-jo.com/ojs/index.php/TJPS/index



IMPROVING PARALLEL SOLUTIONS FOR METHOD OF LINES TO 1-D HEAT EQUATION BY USING FIVE POINT FINITE DIFFERENE

Qays Younis Mahmmod[1] , Akram S. Mohammed[2] , Zeyas M. Abdullah[2]

[1] *Ministry of Education of Iraq , apartment of Nineveh representation of Erbil*

[2] *Mathematics Department, College of Computer Science and Mathematics, Tikrit University, Tikrit, Iraq*

ARTICLE INFO.

Article history:

-Received: 13 / 11 / 2017

-Accepted: 10 / 1 / 2018

-Available online: / / 2018

Keywords: parallel solutions, method of lines, Difference Methods, *5FDM*, Runge – Kutta 4order, Runge – Kutta 5 order.

Corresponding Author:

Name: Akram S. Mohammed

E-mail: akr_te@yahoo.com

Tel:

Abstract

The aim of the study is to explain the numerical solutions of the one dimensional (1-D) of heat by using method of lines (MOLs). In the (MOLs) the derivative is firstly transformed to equivalent 5 point central finite differences methods (FDM) that is also transformed to the ordinary differential equations (ODEs). The produced (ODEs) systems are solved by the well-known techniques method of ODEs such as the 4th Runge - Kutta method and Runge - Kutta Fehlberg. And since of the conversion of the second derivative to the equivalent of the 5 points FDM which led to an increase in the size of the system equations ODEs, and thus increased we have improved the performance of these (MOLs) techniques by introduce parallel processing to speed up the solution of the produced ODE systems. The developed parallel technique, are suitable for running on MIMD (Multiple Instruction Stream, Multiple Data Stream) computers.

1. Introduction

The method of lines is a general technique for solution partial differential equations (PDEs) by typically using finite difference relationships for the spatial derivatives and ordinary differential equations for the time derivative.

Many physical, chemical and engineering problems, mathematically, can be modeled in the form of system of partial differential equations or system of ordinary differential equations.

Parabolic PDEs describe practically useful phenomena such as transport-chemistry problems of the advection-diffusion-reaction type and problem of this type play an important role in the modeling of pollution of the atmosphere, ground water and surface water. Qais Younis Mahmmod;[15]; thesis Submitted by " Improving Parallel Numerical Solutions Of Partial Differential equations".Fatmah M. Alabdali, Huda Omar Bakodah [3] ; she's study "A New Modification of the Method of Lines for First Order Hyperbolic PDEs". Norma Alias Norma Alias& Noriza Satam, Roziha Darwis, Norhafiza Hamzah; [13] ; "Some Parallel Numerical Methods in Solving Partial Differential Equations".Nur Izzati Che Jawias& Fudiah Ismail& Mohamed Sulieman and Azmi Jaffar;[14];" Fourth Order Four-stage Diagonally Implicit Runge-Kutta Method For Linear Ordinary Differential Equations. M. Javidi [10]; "THE MOL SOLUTION FOR THE ONE-DIMENSIONAL HEAT EQUATION SUBJECT TO NONLOCAL CONDITIONS". Karline Soetaert & Filip Meysman [5];" Solving partial differential equations, using R package ReacTran;. M.A. Rehman M. S. A. Taj and M. M. Butt [11]; "fifth-order numerical methods for heat equation subject to boundary integral specification" . Louise Olsen-Kettle [8]; "Numerical solution of partial differential equations". Samir Hamdi_, William E. Schiesser y and Graham W. Gri_thsz[17]; "Method of Lines". Randall J. Leveque [16];"Finite Difference Methods Ordinary and Partial Differential Equations Steady-State and Time-Dependent Problem".. Sang-Bae kim [18];" parallel numerical methods for partial differential equations". Malik Shahadat Ali Taj [12];"Higher Order Parallel Splitting Methods For Parabolic Partial Differential Equations". Jeremy Kepner [4];Parallel MATLAB for Multicore and Multinode Computers;[6]; The Solution of Partial Differetial Equations by the Numerical Method of Lines Combined with a Parallel ODE Solver"

2 .Classification of PDEs: [15, 21]

The classification of PDEs is important for the numerical solution you choose.

$$A(x.,y)U_{xx}+2B(x,y)U_{xy}+C(x,y)U_{yy}=F(x,y)$$

A. Elliptic: $AC>B^2$, for example Laplace's equation as: $U_{xx}+U_{yy}=0$

$$A=C=1, \qquad B=0$$

B. Hyperbolic: $AC<B^2$, for example the 1-D wave equation as: $U_{xx}=\frac{1}{c^2}U_{tt}$

$$A=1 \ , C=-1/c^2, \qquad B=0$$

C. Parabolic: $AC=B^2$ for example, the heat or diffusion equation as: $U_t=\beta U_{xx}$

$$A=1 \ , \quad B=C=0 \cdot$$

3. ELEMENTS OF THE MOL [21,2]

The basic idea of the MOL is to replace the spatial (boundary-value) derivatives in the PDE with algebraic approximations. Once this is done, the spatial derivatives are no longer stated explicitly in terms of the spatial independent variables. Thus, in effect, only the initial-value variable, typically time in a physical problem, remains. In other words, with only one remaining independent variable, we have a system of ODEs that approximate the original PDE. The challenge, then, is to formulate the approximating system of ODEs. Once this is done, we can apply any integration algorithm for initial-value ODEs to compute an approximate numerical solution to the PDE. Thus, one of the salient features of the MOL is the use of existing, and generally well-established, numerical methods for ODEs.

To illustrate this procedure, we consider the MOL solution of Eq. (1). First we need to replace the spatial derivative u_x with an algebraic approximation. In this case we will use a finite difference (FD).

4-Five Point Central Difference method:[14,11,13]

In this research, centered difference methods are used. Hence a fourth order polynomial needs to be fitted through five points. We fit the polynomial through the five points $u_{i-2,j}$, $u_{i-1,j}$, $u_{i,j}$, $u_{i+1,j}$, $u_{i+2,j}$ and . Thus, the formula is given by,

$$\frac{\partial^2 u_{i,j}}{\partial x^2}\approx\frac{-1u_{i-2,j}+16u_{i-1,j}-30u_{i,j}+16u_{i+1,j}-1u_{i+2,j}}{12h^2}$$

This is known as the five-point central difference method.

5-Numerical solution of 1-D heat equation using five point difference method [21,10,5,20,19,7,1,17]

We consider this example: $u_t=\beta^2 u_{xx}$; $0\le x\le 0.02$, $\beta=0.5$, $0\le t$

where the initial conditions are $u(x,0)=10\sin(\pi x)$, and boundary conditions $u(0,t)=0 \quad , u(1,t)=0$

such as the analytical solution of above equation is $u(x,t)=10\sin(\pi x)e^{-0.25\pi^2 t}$

by substitute about $\frac{\partial^2 u}{\partial x^2}=\frac{-1u_{i-2,j}+16u_{i-1,j}-30u_{i,j}+16u_{i+1,j}-1u_{i+2,j}}{12h^2}+o(h^4)$

we have : $\frac{du_i}{dt}=\beta^2\left(\frac{-1u_{i-2}+16u_{i-1}-30u_i+16u_{i+1}-1u_{i+2}}{12h^2}\right)$ ….(2)

if we choose $h=0.00004$; and the values of initial conditions are : $u(x,0)=10\sin(\pi x)$

$u_1=10\sin(\pi(0.00004))\Rightarrow u_1=2.193245422e^{-05}$;

$u_2=10\sin(\pi(0.00008))\Rightarrow u_2=3.386490845e^{-05}$

$u_3=10\sin(\pi(0.00012))\Rightarrow u_2=6.579736267e^{-05}$

⋮ ⋮ ⋮

$u_7=10\sin(\pi(0.00028))\Rightarrow u_7=1.535271796e^{-04}$

Such that $u_0=0;u_7=0$ from initial condition and make $u_{-1}=u_1;u_8=u_6$

From equation (2) we have:

$\frac{du_1}{dt}=0$; $\frac{du_7}{dt}=0$ From boundary conditions $u(0,t)=0 \quad , u(1,t)=0$

From equation (2) we have:

$$\frac{du_2}{dt}=0.25\left(\frac{-1u_{-1}+16u_1-30u_2+16u_3-1u_4}{12h^2}\right)\Rightarrow\frac{du_2}{dt}$$
$$=0.25\left(\frac{-1u_{-1}+16u_1-30u_2+16u_3-1u_4}{12(0.00004)^2}\right)$$
$$\frac{du_3}{dt}=0.25\left(\frac{-1u_1+16u_2-30u_3+16u_4-1u_5}{12h^2}\right)\Rightarrow\frac{du_3}{dt}$$
$$=0.25\left(\frac{-1u_1+16u_2-30u_3+16u_4-1u_5}{12(0.00004)^2}\right)$$
$$\frac{du_4}{dt}=0.25\left(\frac{-1u_2+16u_3-30u_4+16u_5-1u_6}{12h^2}\right)\Rightarrow\frac{du_4}{dt}$$
$$=0.25\left(\frac{-1u_2+16u_3-30u_4+16u_5-1u_6}{12(0.00004)^2}\right)$$
$$\frac{du_5}{dt}=0.25\left(\frac{-1u_3+16u_4-30u_5+16u_6-1u_7}{12h^2}\right)\Rightarrow\frac{du_5}{dt}$$
$$=0.25\left(\frac{-1u_3+16u_4-30u_5+16u_6-1u_7}{12(0.00004)^2}\right)$$
$$\frac{du_6}{dt}=0.25\left(\frac{-1u_4+16u_5-30u_6+16u_7-1u_8}{12h^2}\right)\Rightarrow\frac{du_6}{dt}$$
$$=0.25\left(\frac{-1u_4+16u_5-30u_6+16u_7-1u_8}{12(0.00004)^2}\right)$$

⋮ ⋮ ⋮

$$\frac{du_7}{dt}=0.25\left(\frac{-1u_5+16u_6-30u_7+16u_8-1u_9}{12h^2}\right)\Rightarrow\frac{du_{21}}{dt}$$
$$=0.25\left(\frac{-1u_5+16u_6-30u_7+16u_8-1u_9}{12(0.00004)^2}\right)$$

and from these equations when n=7 as the example we have :

$$\frac{d\vec{u}}{dt}=\begin{pmatrix}u_1'\\u_3'\\u_3'\\u_4'\\u_5'\\u_6'\\u_7'\end{pmatrix}=\begin{pmatrix}-30&16&-1&0&0&0&0\\16&-30&-1&0&0&0&0\\-1&16&-30&16&-1&0&0\\0&-1&16&-30&16&-1&0\\0&0&-1&16&-30&16&-1\\0&0&0&-1&16&-30&16\\0&0&0&0&-1&16&-30\end{pmatrix}\bullet\begin{pmatrix}u_1\\u_2\\u_3\\u_4\\u_5\\u_6\\u_7\end{pmatrix}$$

This MOL approach is sometimes used in practice by first discretizing in space and then applying a software package for systems of ODEs. There are also packages that are specially designed to apply MOL. This approach has the advantage of being relatively easy to apply to a fairly general set of time-dependent PDEs, but the resulting method is often not as efficient as specially designed methods for the PDE this equation we can written as:[15,21,10,5,19,1]

$$\vec{u}' = A\vec{u} + \vec{b} \Rightarrow \vec{u}' = f(u_1,u_2,u_3,u_4,u_5,u_6,u_7) \ldots (3)$$

If we apply an ODE method to discretize the system (3), we will obtain a fully discrete method which produces approximations $u_i^n \approx u_i(t_n)$ at discrete points in time which are exactly the points (x_i,t_n) of the grid.

We solve for $\vec{u}$ using System of First-Order Ordinary Differential Equations (ODEs).

6. System of First-Order Ordinary Differential Equations (ODEs)[19,7,8,16,2,18]

A- Runge–Kutta Methods:

Runge–Kutta methods generate solution estimates with the accuracy of Taylor methods without having to calculate these derivatives. that all one-step methods to solve the IVP

$u' = f(x_i,t_j)$ $,u(x_0) = u_0,$ $a = x_0 \le x \le x_n = b$

Are expressed as: $u_{i+1} = u_i + h\varphi(x_i,t_j)$

Where $\phi(x_i, t_j)$ is an increment function and is essentially a suitable slope over the interval $[x_i, x_{i+1}]$ that is used for extrapolating u_{i+1} from u_i. The order of the Runge–Kutta method is the number of points that are used in $[x_i, x_{i+1}]$ to determine this suitable slope. For example, second-order Runge–Kutta methods use two points in each subinterval to find the representative slope, and so on.

A-1 Classical Runge–Kutta 4 Method:

The classical RK4 method is described by:

$$u_{i+1} = u_i + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4)$$

Where:

$$k_1 = f(x_i,t_j)$$
$$k_2 = f\left(x_i + \frac{1}{2}h, t_j + \frac{1}{2}k_1 h\right)$$
$$k_3 = f\left(x_i + \frac{1}{2}h, t_j + \frac{1}{2}k_2 h\right)$$
$$k_4 = f(x_i + h, t_j + k_3 h)$$

RK_4 methods produce estimates with the accuracy of a fourth-order Taylor method without calculating the derivatives of f(x, t). Instead, four function evaluations per step are performed. The classical RK_4 method is the most commonly used technique for numerical solution of first-order IVPs, as it offers the most acceptable balance of accuracy and computational effort.

The user-defined function RK_4 uses the classical RK_4 method to estimate the solution of an IVP.

And applying the Rk_4 to the eqn.(3) we have the results as the tables(1, 2) when k=0.2:

Table(1) the MOLs by using Runge-Kutta 4 order (Rk4) at k=0.2

T	Numeric u(4) RK4	EXACT u(4) RK4	ABS Error	Numeric u(5) RK4	EXACT u(5) RK4	ABS Error	Numeric u(6) RK4	EXACT u(6)RK4	ABS Error
0	8.660254	10	1.34E+00	10	10	0.00E+00	9.510565	10	4.89E-01
0.2	4.973572	6.10498	1.13E+00	5.914376	6.10498	1.91E-01	5.610682	6.10498	4.94E-01
0.4	2.87859	3.727078	8.48E-01	3.447831	3.727078	2.79E-01	3.284598	3.727078	4.42E-01
0.6	1.67	2.275374	6.05E-01	2.008644	2.275374	2.67E-01	1.924726	2.275374	3.51E-01
0.8	0.969533	1.389111	4.20E-01	1.170178	1.389111	2.19E-01	1.128173	1.389111	2.61E-01
1	0.562992	0.84805	2.85E-01	0.681714	0.84805	1.66E-01	0.661317	0.84805	1.87E-01
1.2	0.326941	0.517733	1.91E-01	0.397148	0.517733	1.21E-01	0.387659	0.517733	1.30E-01
1.4	0.189865	0.316075	1.26E-01	0.231368	0.316075	8.47E-02	0.227243	0.316075	8.88E-02
1.6	0.110262	0.192963	8.27E-02	0.134789	0.192963	5.82E-02	0.133209	0.192963	5.98E-02
1.8	0.064033	0.117804	5.38E-02	0.078524	0.117804	3.93E-02	0.078086	0.117804	3.97E-02

Table(2) the MOLs by using Runge-Kutta 7 order (Rk4) at k=0.2

T	Numeric u(7) RK4	EXACT u(7)RK4	ABS Error	Numeric u(8) RK4	EXACT u(8)RK4	ABS Error	Numeric u(9) RK4	EXACT u(9)RK4	ABS Error
0	10	10	0.00E+00	9.749279	10	2.51E-01	10	10	0.00E+00
0.2	5.977531	6.10498	1.27E-01	5.823917	6.10498	2.81E-01	6.01166	6.10498	9.33E-02
0.4	3.524515	3.727078	2.03E-01	3.445758	3.727078	2.81E-01	3.572725	3.727078	1.54E-01
0.6	2.077419	2.275374	1.98E-01	2.039639	2.275374	2.36E-01	2.122704	2.275374	1.53E-01
0.8	1.224475	1.389111	1.65E-01	1.207499	1.389111	1.82E-01	1.261188	1.389111	1.28E-01
1	0.721732	0.84805	1.26E-01	0.714883	0.84805	1.33E-01	0.749326	0.84805	9.87E-02
1.2	0.425405	0.517733	9.23E-02	0.42324	0.517733	9.45E-02	0.445207	0.517733	7.25E-02
1.4	0.250743	0.316075	6.53E-02	0.250575	0.316075	6.55E-02	0.264517	0.316075	5.16E-02
1.6	0.147794	0.192963	4.52E-02	0.148351	0.192963	4.46E-02	0.157161	0.192963	3.58E-02
1.8	0.087113	0.117804	3.07E-02	0.08783	0.117804	3.00E-02	0.093376	0.117804	2.44E-02

A-2 Runge–Kutta–Fehlberg Method:

One way to estimate the local truncation error for Runge–Kutta methods is to use two RK methods of different order and subtract the results. For cases involving variable step size, the error estimate can be used to decide when the step size needs to be adjusted. Naturally, a drawback of this approach is the number of function evaluations required per step. For example, we consider a common approach that uses a fourth-order and a fifth-order RK. This requires a total of 10 (four for RK_4 and six for RK_5) function evaluations per step. To get around the computational burden, the Runge–Kutta–Fehlberg (RKF) method utilizes an RK5 method that uses the function evaluations provided by its accompanying RK_4method. And it's a form as:

The Runge–Kutta with a fifth-order method form is:

$$u_{i+1} = u_i + h\left(\frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5\right)$$

$$w_{i+1} = u_i + h\left(\frac{16}{135}k_1 + \frac{6656}{12825}k3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6\right)$$

$$k_1 = f(x_i, t_j)$$

$$k_2 = f(x_i + \frac{1}{4}h, t_j + \frac{1}{4}k_1 h)$$

$$k_3 = f(x_i + \frac{3}{8}h, t_j + \frac{3}{32}k_1 h + \frac{9}{32}k_2 h)$$

$$k_4 = f(x_i + \frac{12}{13}h, t_j + \frac{1932}{2197}k_1 h - \frac{7200}{2197}k_2 h + \frac{7296}{2197}k_3 h)$$

$$k_5 = f(x_i + h, t_j + \frac{439}{216}k_1 h - 8k_2 h + \frac{3680}{513}k_3 h - \frac{845}{4104}k_4 h)$$

$$k_6 = f(x_i + \frac{1}{2}h, t_j - \frac{8}{27}k_1 h + 2k_2 h - \frac{3544}{2565}k_3 h + \frac{1859}{4104}k_4 h - \frac{11}{40}k_5 h)$$

We applied the Runge–Kutta–Fehlberg (FRK) Method to the eqn.(3) and we have the results as the tables(3) :

Table(3) the MOLs by using (FRK) at k=0.0001

T	Numeric u(4) RKF	EXACT u(4) RKF	ABS Error	Numeric u(5) RKF	EXACT u(5) RKF	ABSError	Numeric u(6) RKF	EXACT u(6) RKF	ABS Error
0.0001	9.510565	10	4.89E-01	10	10	0.00E+00	9.510565	10	4.89E-01
0.0002	9.487142	9.975356	4.88E-01	9.975436	9.975356	7.98E-05	9.487142	9.975356	4.88E-01
0.0003	9.463729	9.950774	4.87E-01	9.950896	9.950774	1.22E-04	9.463729	9.950774	4.87E-01
0.0004	9.440327	9.926251	4.86E-01	9.926379	9.926251	1.28E-04	9.440327	9.926251	4.86E-01
0.0005	9.416937	9.901789	4.85E-01	9.901888	9.901789	9.85E-05	9.416937	9.901789	4.85E-01
0.0006	9.39356	9.877388	4.84E-01	9.877422	9.877388	3.39E-05	9.39356	9.877388	4.84E-01
0.0007	9.370198	9.853046	4.83E-01	9.852982	9.853046	6.47E-05	9.370198	9.853046	4.83E-01
0.0008	9.346852	9.828765	4.82E-01	9.828568	9.828765	1.97E-04	9.346852	9.828765	4.82E-01
0.0009	9.323523	9.804543	4.81E-01	9.804182	9.804543	3.61E-04	9.323523	9.804543	4.81E-01
0.001	9.300212	9.780381	4.80E-01	9.779824	9.780381	5.57E-04	9.300212	9.780381	4.80E-01
t u	Numeric u(7) RKF	EXACT u(7) RKF	ABS Error	Numeric u(8) RKF	EXACT u(8) RKF	ABS Error	Numeric u(9) RKF	EXACT u(9) RKF	ABS Error
0.0001	10	10	0.00E+00	9.749279	10	2.51E-01	10	10	0.00E+00
0.0002	9.97538	9.975356	2.39E-05	9.725268	9.975356	2.50E-01	9.975363	9.975356	6.55E-06
0.0003	9.950828	9.950774	5.43E-05	9.701323	9.950774	2.49E-01	9.950787	9.950774	1.38E-05
0.0004	9.92634	9.926251	8.92E-05	9.67744	9.926251	2.49E-01	9.926274	9.926251	2.23E-05
0.0005	9.901916	9.901789	1.27E-04	9.653618	9.901789	2.48E-01	9.901822	9.901789	3.23E-05
0.0006	9.877554	9.877388	1.66E-04	9.629854	9.877388	2.48E-01	9.877432	9.877388	4.40E-05
0.0007	9.853251	9.853046	2.05E-04	9.606146	9.853046	2.47E-01	9.853104	9.853046	5.75E-05
0.0008	9.829007	9.828765	2.42E-04	9.582493	9.828765	2.46E-01	9.828838	9.828765	7.26E-05
0.0009	9.804819	9.804543	2.76E-04	9.558892	9.804543	2.46E-01	9.804633	9.804543	8.92E-05
0.001	9.780687	9.780381	3.06E-04	9.535342	9.780381	2.45E-01	9.780488	9.780381	1.07E-04

7- WHY PARALLEL PROCESSING?

Since the method of lines technology is convert the PDE to the system of the ODEs and we results many of the ODES that are need high running time if we used the serial computer, then to solve this equitation and running speed of the many ODEs we used the parallel algorithm by using Matlab in the Laptop which it have Cpu Intel core i5 speed and it have Ram 4 Gb.

8- TYPES OF PARALLELISM: [15]

In 1966, M. J. Flynn proposed a four-way classification of computer systems based on the notions of instruction streams and data streams. Flynn's classification has become standard and is widely used. Flynn coined the abbreviations SISD, SIMD, MISD, and MIMD) for the four classes of computers shown in Fig. 1.11, based on the number of instruction streams (single or multiple) and data streams (single or multiple) [Flyn96]. The SISD class represents ordinary "uniprocessor" machines.

ISSN: 1813 – 1662 (Print)
E-ISSN: 2415 – 1726 (On Line)

Computers in the SIMD class, with several processors directed by instructions issued from a central control unit, are sometimes characterized as "array processors." Machines in the MISD category have not found widespread application, but one can view them as generalized pipelines in which each stage performs a relatively complex operation (as opposed to ordinary pipelines found in modern processors where each stage does a very simple instruction-level operation).

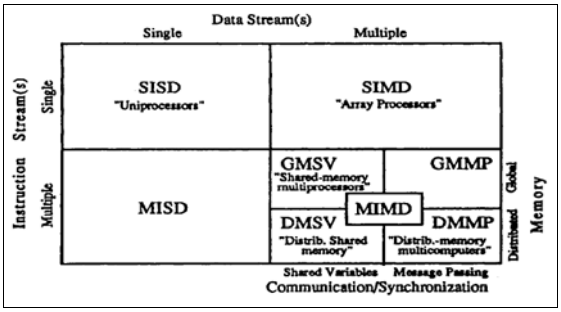


Figure 1 the Flynn-Johnson classification of computer system

9- Parallel Algorithms:[4,12]

A parallel algorithm is one where the tasks could all be performed in parallel at the same time due to their data independence. The DG associated with such an algorithm looks like a wide row of independent tasks. Figure 2 shows an example of a parallel algorithm. A simple example of such a purely parallel algorithm is a web server where each incoming request can be processed independently from other requests. Another simple example of parallel algorithms is multitasking in operating systems where the operating system deals with several applications like a web browser, a word processor, and so on.

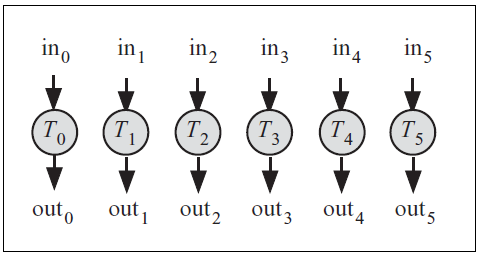
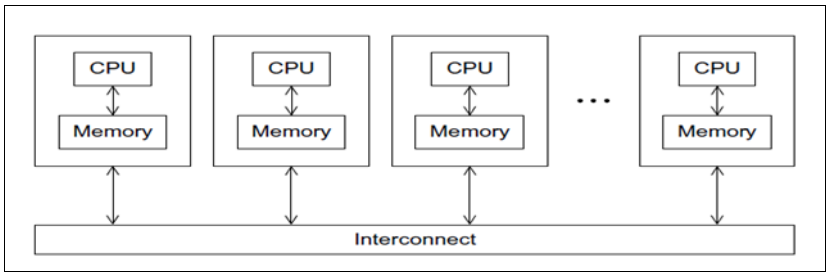


Figure (2) example of parallel algorithm

10- MIMD systems [18,13,9]

Multiple instruction, multiple data, or MIMD, systems support multiple simultaneous instruction streams operating on multiple data streams. Thus, MIMD systems typically consist of a collection of fully independent processing units or cores, each of which has its own control unit and its own ALU. Furthermore, unlike SIMD systems, MIMD systems are usually asynchronous, that is, the processors can operate at their own pace. In many MIMD systems there is no global clock, and there may be no relation between the system times on two different processors. In fact, unless the programmer imposes some synchronization, even if the processors are executing exactly the same sequence of instructions, at any given instant they may be executing different statements.

There are two principal types of MIMD systems: shared-memory systems and distributed-memory systems. In a shared-memory system a collection of autonomous processors is connected to a memory system via an interconnection network, and each processor can access each memory location. In a shared-memory system, the processors usually communicate implicitly by accessing shared data structures. In a distributed-memory system, each processor is paired with its own private memory, and the processor-memory pairs communicate over an interconnection network. So in distributed-memory systems the processors usually communicate explicitly by sending messages or by using special functions that provide access to the memory of another processor. See Figures (3):



Figure(3) MIMD parallel computer design

11- Running Time:

Since the running speed of difference application is the first goal to build for parallel computers, then the important measure to evaluate algorithms are running time which defined a time that the algorithm to take from the first minute at work into the last minute at stopped. And when the processor is started and stopped in the different wait, then the running time equal to the take wait from first processor to yet stopped the finally processor. And to the answer as size N then the running time to have solution at using parallel algorithm is function to N and denote as T(N), and depend on the two point:

- **Counting Steps**

Which is process to calculus the steps or operators that is running by the algorithm in the bed states. And measure the running time in the parallel computers by the number of the unite time called (cycle), while it measure the counting steps by the number of the running steps in the staying kinds from the parallel computers by two measures, it is calculus steps and counting steps which is used to data traffic from the processor to another among the memory shared or the attaching web which is called (routing steps).

- **Steep up (Sp)**

The speed of a program is the time to execute the program while speed up is defined as the time it takes to complete an algorithm with one processor divided by the time it takes to complete the same algorithm with N processors. The formula of speed up for a parallel application is given.

$$speed\ up(sp) = \frac{time(1)}{time(p)}$$

Where Time (1) is execution time for a single processor and Time (p) is execution time using p parallel processors.

- **The Efficiency (e_p):**

The efficiency of a parallel program is a measure of processor utilization. Efficiency is defined as the speed up with processors divided by the number of processors N.

$$efficiency = \frac{speed\ up}{p}$$

Where p is the number of processors.

12- The parallel algorithm to the MOL: [15]

1- First algorithm is level technology for RK4:

we see when at solve the PDE by using the MOL method that they results many of the ODEs which need high running time and to solve this we used first algorithm which is worked as :

1. Cpu_1 : it's used to calculus the initial condition from the equation $u(x_i,0) = 10\sin(\pi x_i)$ for i=1..N and in this example N=9 and the sending to other processors Cpu_4 , Cpu_5 …., CPU_9
2. Cpu_2: it's used to calculus the boundary condition $u(0,t_j) = 0$ for j=1..M and the sending to other processors Cpu_4 , Cpu_5 …., CPU_9
3. Cpu_3: it's used to calculus the boundary condition $u(1,t_j) = 0$ for j=1..M and the sending to other processors Cpu_4 , Cpu_5 …., CPU_9
4. Cpu_4 : it's received the data from the Cpu_1 & Cpu_2 and Cpu_3 to compute the first ODEs U_1 by using Runge-kutta function.
5. Cpu_5 :it's received the data from the Cpu_1 & Cpu_2 and Cpu_3 to compute the first ODEs U_2 by using Runge-kutta function.

⋮ ⋮

6. Cpu_9 :it's received the data from the Cpu_1 & Cpu_2 and Cpu_3 to compute the first ODEs U_9 by using Runge-kutta function.

We can classification the first parallel algorithm as the figure (3):

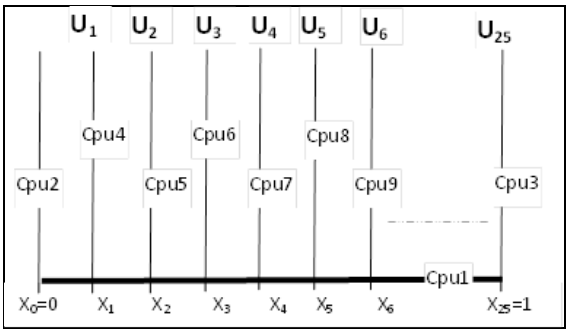


Figure (3) the level algorithm parallel technology to the MOL &RK4

And the running time by using level parallel algorithm of the MOLs by using RK4 are showing in the table (4):

Table (4) the running time of each MOL by using RK4

T	u(4) at time 0.917227609s	u(5) at time 0.931212212s	u(6) at time 0.976410798 s	u(7) at time 1.007195172 s	u(8) at time 1.044946441 s	u(9) at time 1.123303412 s
0	8.660254	10	9.510565	10	9.749279	10
0.2	4.973572	5.914373	5.610679	5.977527	5.823914	6.011656
0.4	2.878591	3.447831	3.284598	3.524515	3.445759	3.572726
0.6	1.67	2.008644	1.924726	2.07742	2.039639	2.122704
0.8	0.969533	1.170178	1.128173	1.224475	1.2075	1.261188
1	0.562992	0.681714	0.661317	0.721733	0.714883	0.749326
1.2	0.326941	0.397148	0.387659	0.425405	0.42324	0.445207
1.4	0.189865	0.231368	0.227243	0.250744	0.250575	0.264517
1.6	0.110262	0.134789	0.133209	0.147794	0.148351	0.157161
1.8	0.064033	0.078524	0.078086	0.087113	0.08783	0.093376

We compute the running time to solve the 1-D equation by MOL and RK4 at using serial Laptop and it's (3.393 s) such that the number of the ODEs are 9 and using 9 processors and we accelerator the solutions about 3.699190873 and the efficient of processors is (0.411021208).

2- Second algorithm is level technology for RK-Fehlberg:

To accelerator the running time to the MOL method which is used Runge-Kutta Fehlberg to solve the ODEs that is results from convert the PDE by MOL we used the level parallel technology as follows:

1. Cpu_1 : it's used to calculus the initial condition from the equation $u(x_i,0)=10\sin(\pi x_i)$ for i=1..N and in this example N=9 and the sending to other processors Cpu_4 , Cpu_5 …., CPU_9
2. Cpu_2: it's used to calculus the boundary condition $u(0,t_j)=0$ for j=1..M and the sending to other processors Cpu_4 , Cpu_5 …., CPU_9
3. Cpu_3: it's used to calculus the boundary condition $u(1,t_j)=0$ for j=1..M and the sending to other processors Cpu_4 , Cpu_5 …., CPU_9
4. Cpu_4 : it's received the data from the Cpu_1 & Cpu_2 and Cpu_3 to compute the first ODEs U_1 by using Runge - kutta Fehlberg function.
5. Cpu_5 :it's received the data from the Cpu_1 & Cpu_2 and Cpu_3 to compute the first ODEs U_2 by using Runge - kutta Fehlberg function.

⋮ ⋮

6. Cpu_9 :it's received the data from the Cpu_1 & Cpu_2 and Cpu_3 to compute the first ODEs U_9 by using Runge-kutta Fehlberg (FRK) function.

The table (5) showing the running time each of the MOL by level parallel algorithm using (FRK):

Table(5) the running time of MOL using level parallel algorithm using (FRK)

t	u(4) at time 0.184769s	ABS error	u(5) at time 0.146398s	ABS error	u(6) at time 0.151273s	ABS error	u(7) at time 0.160539s	ABS error	u(8) at time 0.164062s	ABS error	u(9) at time 0.165782s	ABS error
0	8.660254	1.34E+00	10	0.00E+00	9.510565	4.89E-01	10	0.00E+00	9.749279	2.51E-01	10	0.00E+00
0	8.187449	1.33E+00	9.513892	4.61E-03	9.045264	4.73E-01	9.518599	1.01E-04	9.279325	2.39E-01	9.518743	2.45E-04
0	7.742008	1.32E+00	9.041338	1.88E-02	8.591252	4.69E-01	9.055858	4.32E-03	8.826284	2.34E-01	9.058792	1.39E-03
0.1	7.322107	1.30E+00	8.585237	3.87E-02	8.153165	4.71E-01	8.609603	1.43E-02	8.388945	2.35E-01	8.616753	7.18E-03
0.1	6.92608	1.28E+00	8.147251	6.14E-02	7.733335	4.75E-01	8.180187	2.85E-02	7.968498	2.40E-01	8.19187	1.68E-02
0.1	6.552396	1.26E+00	7.728196	8.52E-02	7.332726	4.81E-01	7.768344	4.51E-02	7.56598	2.47E-01	7.78446	2.90E-02
0.1	6.199651	1.24E+00	7.328314	1.09E-01	6.951492	4.86E-01	7.37454	6.27E-02	7.181842	2.55E-01	7.394859	4.24E-02
0.1	5.866549	1.21E+00	6.947461	1.32E-01	6.589317	4.90E-01	6.998873	8.02E-02	6.816036	2.63E-01	7.023115	5.60E-02
0.2	5.551894	1.19E+00	6.585241	1.53E-01	6.245618	4.93E-01	6.641129	9.71E-02	6.468185	2.70E-01	6.668989	6.93E-02
0.2	5.254578	1.16E+00	6.241098	1.73E-01	5.91967	4.94E-01	6.300873	1.13E-01	6.137707	2.76E-01	6.332031	8.18E-02

we solve the 1-D heat equation in above example by MOL and Runge-Kutta Fehlbreg method it take wait by serial computer (Laptop) is (1.774 s) bout when we used the parallel algorithm by using 9 processors we accelerator the solutions about (2.092746169) and the efficient is (0.232527352).

Conclusion

1. We used the 5 point central (FDM) and Runge-kutta 4th order classes to solve the 1-D heat equation $u_t = \beta^2 u_{xx}$ and we heve the numerical solutions with the little absolute error when using the step time k=0.2
2. We improving the numerical solutions of 1-D heat equation on the above by using MOL and solve the ODE equations by Runge-kutta Fehlberg method and we have results absolute error little than from the Runge-kutta 4 order suh that the using the k=0.0001
3. We accelerator the running time for solving by using MOL since the alot of the number of the ordinary differential equations which leads from the converts the PDEs to ODEs by MOL method and to treat that we used first level parallel algorithm for Runge-kutta 4 order to speed the running time of the solves and we have the speed up about (3.699190873) and the efficient of processors is (0.411021208) such that we using 9 processors.
4. We speed up the MOL equations which is using the Runge-kutta Fehlberg by the second level parallel algorithm to speed up about (2.092746169) and the efficient is (0.232527352) and by using 9 processors.

References

[1] Alkis Gonstantinides & Navid Mostoufi;(1999); Numerical Methods for Chemical Engineers with MATLAB Applications; by Prentice hall PTR ISBN 0-13-013851-7. [8]

[2]Daniel R. Lynch;(2005); NUMERICAL PARTIAL DIFFERENTIAL EQUATIONS FOR ENVIRONMENTAL SCIENTISTS AND ENGINEERS A First Practical Course; 2005 Springer Science + Business Media, Inc..

[3] Fatmah M. Alabdali, Huda Omar Bakodah; (2014); A New Modification of the Method of Lines for First Order Hyperbolic PDEs; Applied Mathematics, 2014, 5, 1457-1462 Published Online June 2014 in Sci Res. http://www.scirp.org/journal/am.

[4] Jeremy Kepner;(2009); "Parallel MATLAB for Multicore and Multinode Computers"; by the Society for Industrial and Applied Mathematics (SIAM); ISBN 978-0-898716-73-3.

[5] Karline Soetaert & Filip Meysman; (2010); Solving partial differential equations, using R package Reac Tran;
http://www.nioo.knaw.nl/ppages/ksoetaert.

[6] Khaddaj, S. A. and Liddell, H. M., (1990), "The Solution of Partial Differetial Equations by the Numerical Method of Lines Combined with a Parallel ODE Solver", Numerical Methods in Engineering: Theory and Applications, Vol. I, Edited by Pande, G. N. and Middleton, J., University College of Swansea Elsevier Applied Science Publisher Ltd..

[7] Leon Lapidus &G eorge F. pinder; (1999); NUMERICAL SOLUTION OF ARTIAL DIFFERENTIAL EQUATIONS IN SCIENCE AND ENGINEERING; by John Wiley & Sons, Inc. All rights reserved ISBN 0-471-35944-0 .

[8]Louise Olsen-Kettle; Numerical solution of partial differential equations. http://researchers.uq.edu.au/researcher/768.

[9]M. AKRAM ; (2005); A parallel algorithm for the inhomogeneous heat equations; University College of Information Technology, Punjab University, Old Campus, Lahore-54000, Pakistan; J. Indian Inst. Sci., Sep.–Oct. 2005, 85, 253–264.

[10] M. Javidi; (2006);" THE MOL SOLUTION FOR THE ONE-DIMENSIONAL HEAT EQUATION SUBJECT TO NONLOCAL CONDITIONS"; Kermanshah 67149, Iran; International Mathematical Forum, 1, 2006, no. 12, 597-602.

[11] M.A. Rehman M. S. A. Taj and M. M. Butt ;(2010); FIFTH-ORDER NUMERICAL METHODS FOR HEAT EQUATION SUBJECT TO A BOUNDARY INTEGRALSPECIFICATION; Acta Math. Univ. Comenianae Vol. LXXIX, 1(2010), pp. 89-104.

[12] Malik Shahadat Ali Taj;(1995); Higher Order Parallel Splitting Methods For Parabolic Partial Differential Equations"; thesis submitted for the degree of Doctor of Philosophy Uxbridge, Middlesex, England. UB8 3PH.

[13] Norma Alias; (2010); "Some Parallel Numerical Methods in Solving Partial Differential Equations";V2-396 2010 2nd International Conference on Computer Engineering and Technology.

[14]Nur Izzati Che Jawias& Fudiah Ismail& Mohamed Sulieman and Azmi Jaffar; "Fourth Order Four-stage Diagonally Implicit Runge-Kutta Method For Linear Ordinary Differential Equations"; Malaysian Journal of Mathematical Sciences 4(1):95-`05(2010).

[15] Qais Younis Mahmmod; (2013); Improving Parallel Numerical Solutions Of Partial Differential equations; thesis Submitted to The Council of the College of Education,University of Mosul.

[16] Randall J. Leveque;(2007); Finite Difference Methods for Ordinary and Partial Differential Equations Steady-State and Time-Dependent Problems; by the Society for Industrial and Applied Mathematics.

[17] Samir Hamdi & William E. Schiesser y and Graham W. Gri_thsz; (2009); Method of Lines; Ecole Polytechnique, France; Lehigh University, USA; City University,UK; Scholarpedia, 2(7):2859.

[18] Sang-Bae kim;(1993);" parallel numerical methods for partial differential equations"; thesis of Purdue University in partial requirements for the degree of doctor of philosophy.

[19] Timothy Sauer ; (2012); NUMERICAL Analysis second edition; By Pearson Education, Inc. ISBN-13: 978-0-321-78367-7.
[20] VICTOR J. Law; (2013); NUMERICAL METHODS for CHEMICAL ENGINEERS Using Excel , VBA, and MATLAB; by Taylor & Francis Group, LLC Boca Raton, FL 33487-2742.
[21] William E. Schiesser & Graham W. Griffiths ; (2009) ; A Compendium of Partial Differential Equation Models: Method of Lines Analysis with Matlab; ISBN-13 978-0-511-50853-0.

تحسين الحلول المتوازية لطريقة الخطوط في حل معادلة الحرارة ذات البعد الواحد باستخدام النقاط الخمسة للفروقات المنتهية

قيس يونس محمود يونس[1] ، اكرم سالم محمد[2] ، زياد محمد عبدالله[2]

[1] ممثلية وزارة التربية في اربيل ، وزارة التربية ، اربيل ، العراق

[2]قسم الرياضيات ، كلية علوم الحاسوب والرياضيات ، جامعة تكريت ، تكريت ، العراق

الملخص

الهدف من الدراسة هو الحلول العددية لمعادلة الحرارة ذات البعد الواحد باستخدام طريقة الخطوط (MOL). في بداية طريقة الخطوط (MOL) نقوم بتحويل المشتقة الثانية لما يكافئها باستخدام النقاط الخمسة المركزية للفروقات المنتهية وبالتالي تتحول هذه المعادلة التفاضلية الجزئية الى معادلة تفاضلية اعتيادية (ODEs). ثم نحل منظومة المعادلات التفاضلية الاعتيادية باستخدام احدى الطرق المعرفة لحل (ODEs) مثل استخدام طريقة رونج–كوتا ذات الرتبة الرابعة وطريقة رونج–كوتا فيلبريج. وبسبب تحويل المشتقة الثانية الى ما يكافئها باستخدام النقاط الخمسة للفروقات المنتهية (FDM). والتي ادت الى زيادة حجم منظومة المعادلات الخطية (ODE) وبالتالي الى زيادة وقت الحل، قمنا بتطوير كفاءة طريقة (MOLs) ولحل هذه المشكلة استخدامنا المعالجات المتوازية لتعجيل حل منظومة (ODE). واستخدمنا حاسبات من نوع MIMD في التقنية المتوزاية المطورة الملائمة لتنفيذها.