The Numerical Solution of Heat Equation by the Fourth-Order Iterative Alternating Decomposition Explicit Method with MPI

Simon Uzezi Ewedafe, Rior Hirowati Shariffudin

Abstract*: The numerical solution of the heat equation in one space dimension is obtained using the Fourth-Order Iterative Alternating Decomposition Explicit Method (4-IADE) on a parallel platform with Message Passing Interface (MPI). Here, a higher fourth-order Crank-Nicolson type scheme is used in the approximation which gives rise to a Penta diagonal matrix in the solution of the system at each time level. The method employs a splitting strategy which is applied alternately at each half time step. The method is shown to be computationally stable and appropriate parameters chosen to accelerate convergence. The accuracy of the method is comparable to that of existing well known methods. Results obtained by this method for several different problems were compared with the exact solution and agreed closely with those obtained by other finite-difference methods with correlation between speedup and efficiency.

***Index Terms*: efficiency, heat equation, MPI, speedup, 4-IADE.**

I. INTRODUCTION

Many numerical methods have been suggested for the solution of the heat equation. Two types of finite-difference equations which have been studied previously are explicit difference equations and implicit difference equations [1], [2]. Sequential numerical methods for solving time dependable problems have been explored extensively [3]-[5]. Attempts have also been made towards parallel solutions on distributed memory MIMD machines. In analyzing time-dependable convection and diffusion problems [6], [7] using numerical methods, the authors found that the machine time required was large and the cost prohibitive. This was true regardless of the method used. When explicit methods were used, the time step was restricted to value very much smaller than the maximum allowable for the solution of the heat equation because of the presence of the convective terms in the original partial differential equations. On the other hand, when implicit methods were tried, the problem of solving large scale linear systems was encountered [3], [8].

Parallel computing environment has been very effective for high performance computing and with richer computational resource it can scale application performance with high bandwidth network. Therefore, an implicit finite difference method is required which is simpler and requires no special skill in programming whilst the computational procedure used must place no upper limit on the size of the time increment so long as it is finite. In this paper, we examine the application and numerical performance of the fourth-order IADE method to solving the time dependable problem and to test its accuracy calculations compare with those from the exact solution for a model problem with MPI using various mesh sizes. The results obtained by the present method also compared satisfactorily with those obtained by other numerical methods with convergence criteria satisfied.

Revised Manuscript Received on September 22, 2019.

First Author name, His Department Name, University/ College/ Organization Name, City Name, Country Name.
Second Author name, His Department Name, University/ College/ Organization Name, City Name, Country Name.
Third Author name, His Department Name, University/ College/ Organization Name, City Name, Country Name.

II. BACKGROUND

The problem which we considered here is the numerical solution of the unsteady diffusion equation of the form:

$$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2}, \quad 0 \le x \le 1, \quad t > 0 \qquad (1)$$

Subject to given initial and Dirichlet boundary conditions over a rectangular domain with a uniformly spaced network whose mesh points are $x_i = i\Delta x,\ t_j = j\Delta t$ for $i = 0,1,\ldots,m,m+1$ and $j = 0,1,\ldots,n,n+1$ with $\Delta x = 1/(m+1)$ and $\Delta t = T/(n+1)$. The mesh ratio is taken as $\lambda = \Delta t/(\Delta x)^2$. A weighted approximation to the differential equation (1.1) at the point $((x_i, t_{j+1/2})$ is given by

$$\begin{aligned} &-\lambda\theta u_{i-1,j+1} + (1+2\lambda\theta)u_{i,j+1} - \lambda\theta u_{i+1,j+1} \\ &= \lambda(1-\theta)u_{i-1,j} + \left[1-2\lambda(1-\theta)\right]u_{i,j} \\ &+\lambda(1-\theta)u_{i+1,j} \end{aligned}$$

$$i = 1,2,\ldots m \qquad (2)$$

the above approximation corresponds to the fully implicit, the Crank-Nicolson and the classical explicit methods when θ takes the values 1, ½, and 0 respectively. According to [7], who developed the Iterative Alternating Decomposition Explicit (IADE) method to solve (1). The second-order IADE scheme entailed the decomposition of a tridiagonal

matrix which arises from the difference method used to approximate the parabolic equation. By employing the fractional scheme of Yanenko and the Mitchell-Fairweather (MF) variant, this method proves to be highly accurate, fast, convergent and stable. It possess a (2,4) accuracy with respect to *x* and *t*. The algorithm is more accurate than the recently developed generalized Alternating Group Explicit (AGE) scheme of order (2,2). Here, the higher fourth-order Crank-Nicolson type scheme is introduced in the approximation, giving rise to a pentadiagonal matrix in the solution of the system at each time level. The fourth-order IADE scheme is expected to produce a better solution in terms of accuracy and rate of convergence.

III. FORMULATION OF THE FOURTH-ORDER IADE METHOD

A fourth-order Crank-Nicolson type scheme for the numerical solution of (1) is given as follows:

$$\frac{1}{\Delta t}(u_{i,j+1}-u_{i,j}) = \frac{1}{2(\Delta x)^2}\left(\delta_x^2-\frac{1}{12}\delta_x^4\right)(u_{i,j+1}+u_{i,j}) \quad (3)$$

where δ is the usual central difference operator. By defining constants such as

$$a=\frac{\lambda}{24}, b=-\frac{2\lambda}{3}, c=\frac{4+5\lambda}{4}, \quad d=-\frac{2\lambda}{3}, e=\frac{\lambda}{24}, \hat{c}=\frac{4-5\lambda}{4} \quad (4)$$

Eq. (3) becomes:

$$au_{i-2,j+1}+bu_{i-1,j+1}+cu_{i,j+1}+du_{i+1,j+1}+eu_{i+2,j+1} = -au_{i-2,j}-bu_{i-1,j}+\hat{c}u_{i,j}-du_{i+1,j}-eu_{i+2,j}$$

for $i=2,3,\ldots,m-1$.

The above approximation can be displayed in a matrix form as

$$\begin{bmatrix} c & d & e & & & & \\ b & c & d & e & & 0 & \\ a & b & c & d & e & & \\ & \ddots & \ddots & \ddots & \ddots & & \\ & & a & b & c & d & e \\ & 0 & & a & b & c & d \\ & & & & a & b & c \end{bmatrix}_{(m-2)x(m-2)} \begin{bmatrix} u_2 \\ u_3 \\ \vdots \\ \vdots \\ . \\ u_{m-2} \\ u_{m-1} \end{bmatrix}_{j+1} = \begin{bmatrix} f_2 \\ f_3 \\ \vdots \\ \vdots \\ . \\ f_{m-2} \\ f_{m-1} \end{bmatrix} \quad (5)$$

where

$$u=(u_{2,j+1},u_{3,j+1},\ldots,u_{m-1,j+1})^T, \quad f=(f_2,f_3,\ldots f_{m-1})^T$$

$$f_2=-b(u_{1,j}+u_{1,j+1})+\hat{c}u_{2,j}-du_{3,j}-eu_{4,j}$$

$$f_3=-a(u_{1,j}+u_{1,j+1})-bu_{2,j}+\hat{c}u_{3,j}-du_{4,j}-eu_{5,j}$$

$$f_i=-au_{i-2,j}-bu_{i-1,j}+\hat{c}u_{i,j}-du_{i+1,j}-eu_{i+2,j}, \quad \text{for } i=4,5,\ldots,m-3$$

$$f_{m-2}=-au_{m-4,j}-bu_{m-3,j}+\hat{c}u_{m-2,j}-du_{m-1,j}-e(u_{m,j}+u_{m,j+1})$$

$$f_{m-1}=-au_{m-3,j}-bu_{m-2,j}+\hat{c}u_{m-1,j}-d(u_{m,j}+u_{m,j+1}) \quad (6)$$

The column vector **f** of order m-2 consists of boundary values and known u values at time level j. We seek to find the values of u at time level (j+1). The Mitchell-Fairweather variant of the IADE scheme of Sahimi et al for a fixed acceleration parameter r > 0 is given by:

$$(rI+G_1)u^{(p+1/2)}=(rI-gG_2)u^{(p)}+f$$
$$(rI+G_2)u^{(p+1)}=(rI-gG_1)u^{(p+1/2)}+gf \quad (7)$$

where

$$g=(6+r)/6.$$

The coefficient matrix A in (3.5) is decomposed into

$$A=G_1+G_2-\frac{1}{6}G_1G_2 \quad (8)$$

where

$$G_1=\begin{bmatrix} 1 & & & & \\ l_1 & 1 & & & 0 \\ \hat{m}_1 & l_2 & \ddots & & \\ & \hat{m}_2 & \ddots & \ddots & \\ & & \ddots & l_{m-4} & 1 \\ & 0 & & \hat{m}_{m-4} & l_{m-3} \quad 1 \end{bmatrix}_{(m-2)x(m-2)}$$

and

$$G_2 = \begin{bmatrix} \hat{e}_1 & \hat{u}_1 & \hat{v}_1 & & & \\ & \hat{e}_2 & \hat{u}_2 & \hat{v}_2 & 0 & \\ & & \ddots & \ddots & & \\ & 0 & & \hat{e}_{m-4} & \hat{u}_{m-4} & \hat{v}_{m-4} \\ & & & & \hat{e}_{m-3} & \hat{u}_{m-3} \\ & & & & & \hat{e}_{m-3} \end{bmatrix}_{(m-2)x(m-2)} \quad (9)$$

the constituent matrices G_1 and G_2 must be in the form of lower and upper tridiagonal matrices respectively, in order to retain the pentadiagonal structure of A. Equation (8) leads to:

$$\hat{e}_1 = \frac{6}{5}(c-1)$$

$$\hat{u}_1 = \frac{6}{5}d$$

$$\hat{v}_1 = \frac{6}{5}e, \qquad i = 1,2,\ldots,m-4$$

$$l_1 = \frac{6b}{6-\hat{e}_1}$$

$$\hat{e}_2 = \frac{1}{5}\left(6(c-1) + l_1\hat{u}_1\right)$$

$$for\ i = 2,3,\ldots,m-3:$$

$$\hat{u}_1 = \frac{1}{5}\left(6d + l_{i-1}\hat{v}_{i-1}\right), \qquad \hat{m}_{i-1} = \frac{6a}{6-\hat{e}_i},$$

$$l_i = \frac{1}{6-\hat{e}_i}\left(6b + \hat{m}_{i-1}\hat{u}_{i-1}\right),$$

$$\hat{e}_{i+1} = \frac{1}{5}\left(6(c-1) + l_i\hat{u}_i + \hat{m}_{i-1}\hat{v}_{m-1}\right) \quad (10)$$

Since G_1 and G_2 are three banded matrices, then $(G_1 + rI)$ and $(G_2 + rI)$ can be inverted easily. From (7) we have

$$u^{(p+1/2)} = (rI + G_1)^{-1}(rI - gG_2)u^{(p)} + (rI + G_1)^{-1} f$$

$$u^{(p+1)} = (rI + G_2)^{-1}(rI - gG_1)u^{(p+1/2)} + g(rI + G_2)^{-1} f \quad (11)$$

Giving us the following computational formulae at each of the half-iterates.

(i) at the (p + ½) iterate:

$$u_2^{(p+1/2)} = \frac{1}{R}\left(E_1u_2^{(p)} + W_1u_3^{(p)} + V_1u_4^{(p)} + f_2\right)$$

$$u_3^{(p+1/2)} = \frac{1}{R}\left(E_2u_3^{(p)} + W_2u_4^{(p)} + V_2u_3^{(p)} - l_1u_2^{(p+1/2)} + f_3\right)$$

$$u_i^{(p+1/2)} = \frac{1}{R}\begin{pmatrix} E_{i-1}u_i^{(p)} + W_{i-1}u_{i+1}^{(p)} + V_{i-1}u_{i+2}^{(p)} - \hat{m}_{i-3}u_{i-2}^{(p+1/2)} \\ -l_{i-2}u_{i-1}^{(p+1/2)} + f_i \end{pmatrix},$$

$$i = 4,5,\ldots,m-3$$

$$u_{m-2}^{(p+1/2)} = \frac{1}{R}\begin{pmatrix} E_{m-3}u_{m-2}^{(p)} + W_{m-3}u_{m-1}^{(p)} - \hat{m}_{m-5}u_{m-4}^{(p+1/2)} - \\ l_{m-4}u_{m-3}^{(p+1/2)} + f_{m-2} \end{pmatrix}$$

$$u_{m-1}^{(p+1/2)} = \frac{1}{R}\left(E_{m-2}u_{m-1}^{(p)} - \hat{m}_{m-4}u_{m-3}^{(p+1/2)} - l_{m-3}u_{m-2}^{(p+1/2)} + f_{m-1}\right) \quad (12)$$

with

$$\begin{aligned} R &= 1 + r \\ E_i &= r - g\hat{e}_i, & i &= 1,2,\ldots,m-2 \\ W_i &= -g\hat{u}_i, & i &= 1,2,\ldots,m-3 \\ V_i &= -g\hat{v}_i, & i &= 1,2,\ldots,m-4 \end{aligned} \quad (13)$$

(ii) at the (p+1) iterate:

$$u_{m-1}^{(p+1)} = \frac{1}{Z_{m-2}}\left(S_{m-4}u_{m-3}^{(p+1/2)} + Q_{m-3}u_{m-2}^{(p+1/2)} + Pu_{m-1}^{(p+1/2)} + gf_{m-1}\right)$$

$$u_{m-2}^{(p+1)} = \frac{1}{Z_{m-3}}\begin{pmatrix} S_{m-5}u_{m-4}^{(p+1/2)} + Q_{m-4}u_{m-3}^{(p+1/2)} + Pu_{m-2}^{(p+1/2)} \\ -\hat{u}_{m-3}u_{m-1}^{(p+1)} + gf_{m-2} \end{pmatrix}$$

$$u_i^{(p+1)} = \frac{1}{Z_{i-1}}\begin{pmatrix} S_{i-3}u_{i-2}^{(p+1/2)} + Q_{i-2}u_{i-1}^{(p+1/2)} + Pu_i^{(p+1/2)} \\ -\hat{u}_{i-1}u_{i+1}^{(p+1)} - \hat{v}_{i-1}u_{i+2}^{(p+1)} + gf_i \end{pmatrix},$$

$$i = 4,5,\ldots,m-3$$

$$u_3^{(p+1)} = \frac{1}{Z_2}\begin{pmatrix} Q_1u_2^{(p+1/2)} + Pu_3^{(p+1/2)} - \hat{u}_2u_4^{(p+1)} \\ -\hat{v}_2u_5^{(p+1)} + gf_3 \end{pmatrix}$$

$$u_2^{(p+1)} = \frac{1}{Z_1}\left(Pu_2^{(p+1/2)} - \hat{u}_1u_3^{(p+1)} - \hat{v}_1u_4^{(p+1)} + gf_2\right) \quad (14)$$

with

$$P = r - g$$
$$Z_i = r + \hat{e}_i, \quad i = 1,2,\ldots,m-2$$
$$Q_i = -gl_i, \quad i = 1,2,\ldots,m-3 \qquad (15)$$
$$S_i = -g\hat{m}_i, \quad i = 1,2,\ldots,m-4$$

The IADE algorithm is completed explicitly by using the required equations at levels (p + ½) and (p + 1) in alternate sweeps along all the points in the interval (0, 1) until convergence is reached.

IV. MPI COMMUNICATION SERVICE DESIGN WITH PARALLEL STRATEGIES

MPI like most other network-oriented middleware services communicates data from one worker to another across a network. However, MPI's higher level of abstraction provides an easy-to-use interface more appropriate for distributing parallel computing applications. MPI serves as an important foundation for a large group of applications. Conversely, MPI provides a wide variety of communication operations including both blocking and non-blocking sends and receives and collective operations such as broadcast and global reductions. We concentrate on basic message operations: blocking send, blocking receives, non-blocking send, and non-blocking receive. Note that MPI provides a rather comprehensive set of messaging operations. MPI primitive communication operation is the blocking send to blocking receive. A blocking send (*MPI_Send)* does not return until both the message data and envelope have been safely stored. When the blocking sends returns, the sender is free to access and overwrite the send buffer. Note that these semantics allow the blocking send to compute even if no matching receive has been executed by the receiver. A blocking receives (*MPI_Recv)* returns when a message that matches its specification has been copied to the buffer. However, an alternative to blocking communication operations MPI provides non-blocking communication to allow an application to overlap communication and computation. This overlap improves application performance. In non-blocking communication, initialization and completion of communication operations are distinct. A non-blocking send has both a send start call (*MPI_Isend)* initializes the send operation and it return before the message is copied from the send buffer. The send complete call (*MPI_Wait)* completes the non-blocking send by verifying that the data has been copied from the send buffer. It is this separation of send start and send complete that provides the application with the opportunity to perform computations.

The performance metric most commonly used is the speedup and efficiency which gives a measure of the improvement of performance experienced by an application when executed on a parallel system. Speedup is the ratio of the serial time to the parallel version run on *N* workers. Efficiency is the ability to judge how effective the parallel algorithm is expressed as the ratio of the speedup to *N* workers. In traditional parallel systems it is widely define as:

$$S(N) = T(s)/T(N), \qquad E(N) = S(N)/N \qquad (16)$$

where *S(n)* is the speedup factor for the parallel computation, *T(s)* is the CPU time for the best serial algorithm, *T(n)* is the CPU time for the parallel algorithm using *N* workers, *E(n)* is the total efficiency for the parallel algorithm.

V. NUMERICAL EXPERIMENTS AND COMPARATIVE RESULTS

A number of experiments were conducted to demonstrate the application of the fourth-order algorithm on the problems and where appropriate the solutions were compared with that of the other alternating methods either given by [5] or obtained by the author. The acceleration parameter r was chosen so as to provide the most rapid convergence. Unless otherwise stated the convergence criterion was taken as eps = 10^{-4}.

A. Experiment

The problem considered was taken from [Saulev 91964)],

$$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2} \qquad 0 \le x \le 1 \qquad (17)$$

subject to the to the initial condition

$$U(x,0) = 4x(1-x) \qquad 0 \le x \le 1 \qquad (18)$$

and the boundary conditions

$$U(0,t) = U(1,t) = 0 \qquad t \ge 0 \qquad (19)$$

the exact solution was given by

$$U(x,t) = \frac{32}{\pi^2} \sum_{k=1,(2)}^{\infty} \frac{1}{k^3} e^{-\pi^2 k^2 t} \sin(k\pi x) \qquad (20)$$

We observed in Table I – III below the results for the various schemes.

Table- I: absolute error of numerical solutions of Experiment 1, $\lambda = 0.5,\ t = 0.25,\ \Delta t = 0.005,\ \Delta x = 0.1,\ eps = 10^{-4}$

X *method*	*0.1*	*0.2*	*0.3*	*0.4*	*0.5*	*0.6*	*0.7*	*0.8*	*0.9*	*Av. Abs. Err*	*RMS** *It.*
AGE-PR											
IMP 3	$2.1\text{x}10^{-3}$	$4.2\text{x}10^{-3}$	$5.6\text{x}10^{-3}$	$6.6\text{x}10^{-3}$	$6.9\text{x}10^{-3}$	$6.6\text{x}10^{-3}$	$5.7\text{x}10^{-3}$	$4\text{x}10^{-3}$	$2.2\text{x}10^{-3}$	$4.9\text{x}10^{-3}$	$2.7\text{x}10^{-5}$
CN 2	$5.3\text{x}10^{-4}$	$1.0\text{x}10^{-3}$	$1.4\text{x}10^{-3}$	$1.6\text{x}10^{-3}$	$1.7\text{x}10^{-3}$	$1.6\text{x}10^{-3}$	$1.4\text{x}10^{-3}$	$1\text{x}10^{-3}$	$5.5\text{x}10^{-4}$	$1.2\text{x}10^{-3}$	$1.7\text{x}10^{-6}$
AGE-DR											

IMP	$2.6x10^{-3}$	$5x10^{-3}$	$6.7x10^{-3}$	$8x10^{-3}$	$8.3x10^{-3}$	$7.9x10^{-3}$	$6.8x10^{-3}$	$4.9x10^{-3}$	$2.7x10^{-3}$	$5.9x10^{-3}$	$3.88x10^{-5}$
6											
CN	$9.7x10^{-4}$	$1.9x10^{-3}$	$2.5x10^{-3}$	$3x10^{-3}$	$3.1x10^{-3}$	$3x10^{-3}$	$2.6x10^{-3}$	$1.8x10^{-3}$	$9.9x10^{-4}$	$2.2x10^{-3}$	$5.5x10^{-6}$
6											
IADE-MF (2^{nd} – order)											
IMP	$2.2x10^{-3}$	$4.3x10^{-3}$	$5.8x10^{-3}$	$6.8x10^{-3}$	$7.1x10^{-3}$	$6.7x10^{-3}$	$5.7x10^{-3}$	$4.1x10^{-3}$	$2.1x10^{-3}$	$5x10^{-3}$	$2.8x10^{-5}$
3											
CN	$5.1x10^{-4}$	$9.9x10^{-4}$	$1.4x10^{-3}$	$1.6x10^{-3}$	$1.7x10^{-3}$	$1.6x10^{-3}$	$1.4x10^{-3}$	$1.x10^{-3}$	$5.6x10^{-4}$	$1.2x10^{-3}$	$1.6x10^{-6}$
3											
IADE-MF (4^{th} – order)	-	$2.6x10^{-7}$	$3x10^{-7}$	$5.9x10^{-6}$	$2.1x10^{-6}$	$1.1x10^{-7}$	$7.5x10^{-7}$	$5.8.x10^{-7}$	-	$1.1x10^{-6}$	$4.5x10^{-12}$
4											
EXACT SOLN	0.027	0.051	0.071	0.083	0.088	0.083	0.071	0.051	0.027	-	-	-

Table- II: The parallel speed-up S_{par} and the efficiency E_{par} for a mesh of 200x200, for MPI.

Schemes	*N*	T_w	T_m	T_{sd}	T_{sc}	*MPI*		
						T	S_{par}	E_{par}
	1	*2314*	*53*	*23*	*1169.6*	*1083.63*	*1.000*	*1.000*
	2	*1968*	*53*	*19*	*632*	*566.46*	*1.913*	*0.957*
	4	*1341*	*51*	*19*	*407.61*	*365.10*	*2.968*	*0.742*
	8	*1089*	*51*	*19*	*305.97*	*285.99*	*3.789*	*0.474*
IADE-4th Order	*16*	*862*	*51*	*19*	*164.62*	*182.92*	*5.924*	*0.370*
	20	*718*	*51*	*19*	*133.36*	*166.41*	*6.512*	*0.326*
	24	*701*	*51*	*19*	*107.54*	*146.73*	*7.385*	*0.308*
	30	*629*	*51*	*19*	*83.44*	*127.08*	*8.527*	*0.284*
	38	*611*	*51*	*19*	*68.2*	*114.51*	*9.463*	*0.249*
	48	*528*	*51*	*19*	*47.31*	*97.47*	*11.118*	*0.232*

Table- III: The parallel speed-up S_{par} and the efficiency E_{par} for a mesh of 300x300, with MPI.

Schemes	*N*	T_w	T_m	T_{sd}	T_{sc}	*MPI*		
						T	S_{par}	E_{par}
	1	*2968*	*116*	*53*	*1454*	*1328*	*1.000*	*1.000*
	2	*2575*	*114*	*51*	*700.6*	*689.87*	*1.925*	*0.963*
	4	*2388*	*114*	*51*	*377.63*	*439.59*	*3.021*	*0.755*
	8	*2011*	*114*	*51*	*308.04*	*340.95*	*3.895*	*0.487*
IADE-4th Order	*16*	*1932*	*114*	*51*	*131.06*	*217.6*	*6.103*	*0.381*
	20	*1634*	*114*	*51*	*84.19*	*195.35*	*6.798*	*0.340*
	24	*1538*	*114*	*51*	*57.79*	*175.48*	*7.568*	*0.315*
	30	*1184*	*114*	*51*	*24.56*	*154.19*	*8.613*	*0.287*
	38	*1099*	*114*	*51*	*10.71*	*136.89*	*9.701*	*0.255*
	48	*987*	*114*	*31*	*0.11*	*123.43*	*10.759*	*0.224*

Comparative results from problem 1 using the 4th-order IADE and the other alternating implicit schemes for $\lambda = 0.5,\ t = 0.25,\ eps = 10^{-4}$ are given in Table I.

The numerical solutions to the heat equation for the same problem obtained by means of the various different schemes at the same time levels are presented. The Tables also show that the 4-IADE scheme achieves a low iteration count by using a convergence requirement of $\varepsilon = 10^{-4}$. This indicates that the algorithm is a fast numerical solver. Since the IADE scheme employs the Mitchell-Fairweather variant which is second-order accurate in space and fourth-order accurate in time whereas the AGE-PR variant is second-order accurate in both space and time. As for the DR variant, its truncation error is $O\left[(\Delta x)^2 + (\Delta t)\right]$, hence its accuracy is the least among under all the methods under consideration. The crank Nicolson type approximation gives rise to a pentadiagonal system of equations with fourth-order accuracy, thus producing a more accurate solution as compared to the second-order IADE. However, a drawback in using the former is that it incurs more work than the latter in the derivation of its formulae. In conclusion, the fourth-order IADE scheme has merits as an alternative method with respect to stability, accuracy and rate of convergence. Its computational properties are well-suited for implementation on parallel computers. The total CPU time is composed of three parts: the CPU time for the master task, the average worker CPU time for data communication and the average worker CPU time for computation, $T = T_M + T_{SD} + T_{SC}$. Data communication at the end of every iteration is necessary in this strategy. The convergence rate behavior the ratio of the iteration number for the best sequential CPU time on one processor and the iteration number for the parallel CPU time on N processor describe the increase in the number of iterations required by the parallel method to achieve a specified accuracy as compared to the serial method as shown in Table II and III. This increase is caused mainly by the deterioration in the rate of convergence with increasing number of processors and sub-domains. Because the best serial algorithm is not known generally, we take the existing parallel program running on one processor to replace it. When the processor number becomes large, *E*par decreases with *n* due to the effect of both the convergence rate and the parallel efficiency. With the MPI version we were able to show the implementation for the efficiency of different mesh sizes with different block numbers as observed.

REFERENCES

1. D. J. Evans, and M. S. Sahimi, "The Alternating Group Explicit Method (AGE) to solve Parabolic and Hyperbolic Partial Differential Equations," *Ann. Rev. Num. Fluid Mech. And Heat Trans. Ed. By Chang-Lin Tien, Hemisphere Pub. Corp.*, vol. 2, 1988, pp. 288-388.
2. G. D. Smith, *Numerical Solution of Partial Differential Equations – Finite Difference Methods*, Oxford University Press, 2nd Edition, 1978.
3. D. J. Evans, and M. S. Sahimi, "The Numerical Solution of Burger's Equation by the AGE Method," *Intern. J. Computer Math.*, vol. 29, 1989, pp. 39-64.
4. *R. Mitchell, and D. F. Griffiths, The Finite Difference Method in Partial Differential Equations, John Wiley and Sons, Chichester, 1980.*
5. V. K. Saulev, *Integration of Equations of Parabolic Type by the Method of Nets*, Pergamon Press, Oxford 1964.
6. L. W. Johnson, and R. D. Riess, *Numerical Analysis*, 2nd Edition, Addison-Wesley Publication Company, 1982, Reading.
7. M. S. Sahimi, A. Ahmad, and A. A. Bakar, "The Iterative Alternating Decomposition Explicit (IADE) Method to Solve the Heat Conduction Equation," *Intern. J. Computer Math.*, 1993, pp219 – 229.
8. N. N. Yanenko, *The Method of Fractional Steps*, Springer-Verlag, New York, 1971.

AUTHORS PROFILE

Ewedafe Simon Uzezi received the BSc. and MSc. degrees in Industrial-Mathematics and Mathematics from Delta Stae University and The University of Lagos in 1998 and 2003 respectively. He further obtained his PhD. in Numerical Parallel Computing 2010 from the University of Malaya, Kuala Lumpur Malaysia. In 2011 he joined UTAR in Malaysia as an Assistant Professor and later lectured in Oman and Nigeria as Senior Lecturer in Computing. Currently he is a Senior Lecturer in the Department of Computing UWI , Kingston, Jamaica.

Rior Hirowati Shariffudin., Professor in the Institute of Mathematical Sciences, Universiti Malaya, Malaysia