International Journal of Computer Mathematics
Vol. 86, No. 3, March 2009, 431–440

Taylor & Francis
Taylor & Francis Group

Parallel iterative methods for parabolic equations

Wenrui Hao* and Shaohong Zhu

School of Mathematics Science and LPMC, Nankai University, Tianjin, Peoples' Republic of China

(*Received 10 April 2007; revised version received 26 June 2007; second revision received 24 July 2007; accepted 25 July 2007*)

For the problems of the parabolic equations in one- and two-dimensional space, the parallel iterative methods are presented to solve the fully implicit difference schemes. The methods presented are based on the idea of domain decomposition in which we divide the linear system of equations into some non-overlapping sub-systems, which are easy to solve in different processors at the same time. The iterative value is proved to be convergent to the difference solution resulted from the implicit difference schemes. Numerical experiments for both one- and two-dimensional problems show that the methods are convergent and may reach the linear speed-up.

Keywords: parabolic equation; parallel iterative method; domain decomposition; difference scheme

2000 AMS Subject Classification: 65M06; 65F10; 65N22

1. Introduction

While numerically solving the problem of the parabolic equation, the explicit difference scheme can be used on parallel computers, but it is disadvantageous in that it usually has severe restrictions about the time step length due to the conditional stability. The fully implicit difference (ID) scheme does not have such severe constrain because of the unconditional stability, but a global linear system of equations needs to be solved at each time level, which implies that it can not be directly implemented on parallel computers. Facing the fact, there are two choices: one is to construct new difference schemes that are capable of parallel implementation and have weaker restriction than the explicit difference scheme, thus appearing the alternating schemes [2,4,7] and the domain decomposition ones [1,3,6,8,9] and the other is to consider how to solve the linear system of equations deduced from the fully ID scheme in parallel method. Under the first choice, the domain decomposition schemes can reach larger speed-up and have simpler construction, but they are still conditionally stable even though they have weaker restriction than the explicit difference scheme. Therefore, in this paper, we focus attention on the fully ID schemes. We present the parallel iterative (PI) methods in order that the fully ID schemes can be implemented on parallel computers.

*Corresponding author. Email: haowr@mail.nankai.edu.cn

ISSN 0020-7160 print/ISSN 1029-0265 online

DOI: 10.1080/00207160701598446
http://www.informaworld.com

The PI methods presented here are based on the idea of domain decomposition. To get the solution of the ID scheme, we divide the global linear system of equations into some non-overlapping sub-systems, which can be solved in different processors at the same time, and the solution of sub-system is regarded as the iterative value. The iterative value is proved to be convergent to the difference solution. Numerical experiments show that the methods are efficient.

The rest of this paper is organized as follows. In Section 2, the PI method is constructed for the parabolic problem in one-dimensional space, and the convergence of the method is proved. In Section 3, we extend the method to the problem of two-dimensional space. In Section 4, numerical results are provided to show the performance of the methods. Finally, in Section 5, the conclusion is given.

2. PI method for one-dimensional problem

We divide the domain $[0, l] \times [0, T]$ into grids with the space step h and the time step τ, to solve the following initial-boundary value problem by the finite difference method

$$u_t = au_{xx}, \qquad a > 0, \ \ 0 < x < l, \ \ 0 < t \le T, \tag{1}$$

$$u(0, t) = u(l, t) = 0, \qquad 0 < t \le T, \tag{2}$$

$$u(x, 0) = u_0(x), \qquad 0 \le x \le l, \tag{3}$$

where $u_0(x)$ is a given function. Denote $x_j = jh (j = 0, 1, \ldots, J, Jh = l)$, $t^n = n\tau (n = 1, 2, \ldots, N, N\tau = T)$ and $r = \tau/h^2$. Let U_j^n express the approximate value of $u(x_j, t^n)$, $(U_j^{n+1} - U_j^n)/\tau$ and $(U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1})/h^2$ approximate $u_t(x_j, t^{n+1})$ and $u_{xx}(x_j, t^{n+1})$, respectively. Then, the fully ID scheme for Equation (1) is as follows [5].

$$(1 + 2ar)U_j^{n+1} - ar(U_{j-1}^{n+1} + U_{j+1}^{n+1}) = U_j^n, \quad j = 1, 2, \ldots, \ J - 1, \ \ 0 \le n \le N - 1,$$

From [5], we can know that the scheme is unconditionally stable. Denote $U^{n+1} = (U_1^{n+1}, U_2^{n+1}, \ldots, U_{J-1}^{n+1})^{\mathrm{T}}$ and

$$C_k = \begin{bmatrix} 1 + 2ar & -ar & & & & \\ -ar & 1 + 2ar & -ar & & & \\ & \ddots & \ddots & \ddots & & \\ & & -ar & 1 + 2ar & -ar & \\ & & & -ar & 1 + 2ar & \end{bmatrix}_{k \times k}, \quad \forall k.$$

Then, the difference solution converts into the solution of the following linear system of equation

$$C_{J-1}U^{n+1} = U^n. \tag{4}$$

This system can be solved by some classical sequential methods, such as the Thomas Algorithm, SOR iterative method and so on.

Now we present the PI method for the system (4). Assume that K_i are a series of integers satisfying $0 = K_0 < K_1 < K_2 < \cdots < K_m < K_{m+1} = J - 1$. Denote

$$A = \begin{bmatrix} C_{K_1} & & & & 0 \\ & C_{K_2 - K_1} & & & \\ & & \ddots & & \\ & & & C_{K_m - K_{m-1}} & \\ 0 & & & & C_{J-1-K_m} \end{bmatrix},$$

$$B = \begin{bmatrix} O_{K_1-1} & & & & & & & & & & 0 \\ & 0 & ar & & & & & & & & \\ & ar & 0 & & & & & & & & \\ & & & O_{K_2-K_1-2} & & & & & & & \\ & & & & 0 & ar & & & & & \\ & & & & ar & 0 & & & & & \\ & & & & & & \ddots & & & & \\ & & & & & & & \ddots & & & \\ & & & & & & & & O_{K_m-K_{m-1}-2} & & \\ & & & & & & & & & 0 \quad ar & \\ & & & & & & & & & ar \quad 0 & \\ 0 & & & & & & & & & & O_{J-K_m-2} \end{bmatrix},$$

where for any k, O_k is the $k \times k$ zero matrix. Then, we can rewrite C_{J-1} as $C_{J-1} = A - B$. Thus, let $U^{n+1(s)} = (U_1^{n+1(s)}, U_2^{n+1(s)}, \ldots, U_{J-1}^{n+1(s)})^{\mathrm{T}}$ be the sth iterative value, we get the following iterative method

$$AU^{n+1(s+1)} = U^n + BU^{n+1(s)}, \quad s \geq 0, \tag{5}$$

or

$$C_{K_1}\left(U_1^{n+1(s+1)}, \ldots, U_{K_1}^{n+1(s+1)}\right)^{\mathrm{T}} = \left(U_1^n, \ldots, U_{K_1-1}^n, U_{K_i}^n + arU_{K_1}^{n+1(s)}\right)^{\mathrm{T}}, \tag{6}$$

$$C_{K_{i+1}-K_i}\left(U_{K_i+1}^{n+1(s+1)}, \ldots, U_{K_{i+1}}^{n+1(s+1)}\right)^{\mathrm{T}} = \left(U_{K_i+1}^n + arU_{K_i}^{n+1(s)}, U_{K_i+2}^n, \ldots, U_{K_{i+1}}^n + arU_{K_{i+1}+1}^{n+1(s)}\right)^{\mathrm{T}} (i = 1, \ldots, m-1). \tag{7}$$

$$C_{K_{m+1}-K_m}\left(U_{K_m+1}^{n+1(s+1)}, \ldots, U_{K_{m+1}}^{n+1(s+1)}\right)^{\mathrm{T}} = \left(U_{K_m+1}^n + arU_{K_m}^{n+1(s)}, U_{K_m+2}^n, \ldots, U_{K_{m+1}}^n\right)^{\mathrm{T}}. \tag{8}$$

The initial value of iteration may be taken as $U^{n+1(0)} = U^n$. The global system (4) is divided into $m+1$ sub-systems (6), (7) and (8). They can be solved by $m+1$ processors at the same time.

In the following, we prove the convergence of the iterative method. From Equation (5), the iterative matrix is $A^{-1}B$. So, the iterative method is convergent if $\|A^{-1}B\|_\infty < 1$.

Denote the ith column of the matrix B by B_i, then B can be rewritten as $B = (B_1, B_2, \ldots, B_{J-1})$. Thus,

$$\|A^{-1}B\|_\infty \leq \|A^{-1}\sum_j B_j\|_\infty, \tag{9}$$

Denote the $\sum_j B_j = B1$, then $B1 = (B1_{K_1}^T, B1_{K_2-K_1}^T, \ldots, B1_{J-1-K_m}^T)^{\mathrm{T}}$, where $B1_{K_1} = (0, \ldots, ar)^{\mathrm{T}}$, $B1_{K_{i+1}-K_i} = (ar, 0, \ldots, 0, ar)^{\mathrm{T}} (i = 1, 2, \ldots, m-1)$, $B1_{J-1-K_m} = (ar, \ldots, 0)^{\mathrm{T}}$.

Further, let $x = (x_{k_1}, x_{k_2-k_1}, \ldots, x_{J-1-k_m}) = A^{-1}B1$, then $x_{K_{i+1}-K_i}$ the equation $C_{K_{i+1}-K_i}$ $x_{K_{i+1}-K_i} = B1_{K_{i+1}-K_i}, i = 1, 2, \ldots, m-1$. Next, we solve the equation to get $\|A^{-1}B\|_\infty$. To simplify, denote $x_{K_{i+1}-K_i}$ as y, then the following relation is valid.

$$\begin{gathered} (1+2ar)y_1 - ary_2 = ar \\ -ary_1 + (1+2ar)y_2 - ary_3 = 0 \\ \vdots \\ -ary_{K_{i+1}-K_i-1} + (1+2ar)y_{K_{i+1}-K_i} = ar. \end{gathered}$$

Define $y_0 = 1$ and $y_{K_{i+1}-K_i+1} = 1$, the above relation can be rewritten as

$$-ary_{i-1} + (1 + 2ar)y_i - ary_{i+1} = 0, (i = 1, 2, \ldots, K_{i+1} - K_i).$$

The solution of the difference equation is

$$y_i = c_1 q_1^i + c_2 q_2^i,$$

where $q_1 = (1 + 2ar + \sqrt{1 + 4ar})/2ar$, $q_2 = (1 + 2ar - \sqrt{1 + 4ar})/2ar$ and the parameter c_1, c_2 are determined by

$$\begin{cases} c_1 + c_2 = 1 \\ c_1 q_1^{K_{i+1}-K_i+1} + c_2 q_2^{K_{i+1}-K_i+1} = 1. \end{cases}$$

Thus,

$$y_i = \frac{q_2^s - 1}{q_2^s - q_1^s} q_1^i + \frac{1 - q_1^s}{q_2^s - q_1^s} q_2^i, \quad s = K_{i+1} - K_i + 1, \ \ s \geq 2$$

and

$$c_1 < c_2, \quad q_1 q_2 = 1.$$

By discussing the monotonicity of sequence y_i, we know that the sequence $\{y_i\}_{i=1}^{s-1}$ reach the maximum value when $i = 1$ or $i = s - 1$. It means that $y_i \leq \max(y_{K_{i+1}-K_i}, y_1) = (1/q_1)(q_1^s + q_1^2/q_1^s + 1) \leq (2q_1/1 + q_1^2) = (2ar/1 + 2ar) < 1$, $(1 \leq i \leq K_{i+1} - K_i)$. Moreover, when the $m \to \infty$, $y_i \to (1/q_1)$, Similarly, we can know that x_{K_1} and x_{K_m} have the same attribute.

Thus, $\|x\|_\infty \leq 2ar/(1 + 2ar) < 1$, it means that we get

$$\|A^{-1}B\|_\infty \leq \frac{2ar}{1 + 2ar} < 1.$$

This implies that the iterative method (5) is convergent.

3. The iterative method for two-dimensional problem

In this section, we extend the PI method to the following two-dimensional problem

$$u_t = u_{xx} + u_{yy}, \quad (x, y) \in \Omega = [0, l] \times [0, l], \ \ 0 < t \leq T, \tag{10}$$

$$u(x, y, t) = 0, \quad (x, y) \in \partial\Omega, \ \ 0 < t \leq T, \tag{11}$$

$$u(x, y, 0) = u_0(x, y), \quad (x, y) \in \Omega. \tag{12}$$

Denote $h = l/J$, $\tau = T/N$ for any given positive integers J and N, and $r = (\tau/h^2)$. Let $U_{i,j}^n$ be the approximate solution at the grid point $(x_i, y_j, t^n) = (ih, jh, n\tau)$. Then, the ID scheme for

problems (10)–(12) is as follows [5].

$$(1+4r)U_{i,j}^{n+1} - r(U_{i,j+1}^{n+1} + U_{i,j-1}^{n+1} + U_{i+1,j}^{n+1} + U_{i-1,j}^{n+1}) = U_{i,j}^{n}, \quad 1 \le i, \quad j \le J-1, \tag{13}$$

$$U_{i,0}^{n} = U_{i,J}^{n} = U_{0,j}^{n} = U_{J,j}^{n} = 0, \quad 0 \le i, \quad j \le J, \tag{14}$$

$$U_{i,j}^{0} = u_0(x_i, y_j), \quad 0 \le i, \quad j \le J. \tag{15}$$

Denote $U^n = (U_{1,1}^n, \ldots, U_{1,J-1}^n, U_{2,1}^n, \ldots, U_{2,J-1}^n, \ldots, U_{J-1,1}^n, \ldots, U_{J-1,J-1}^n)^{\mathrm{T}}$ and

$$G_k = \begin{bmatrix} H & -rI & & & \\ -rI & H & -rI & & \\ & \ddots & \ddots & \ddots & \\ & & -rI & H & -rI \\ & & & -rI & H \end{bmatrix}, \quad \forall k,$$

in which I is the unit matrix,

$$H = \begin{bmatrix} 1+4r & -r & & & \\ -r & 1+4r & -r & & \\ & \ddots & \ddots & \ddots & \\ & & -r & 1+4r & -r \\ & & & -r & 1+4r \end{bmatrix}_{(J-1)\times(J-1)},$$

and the subscript of G expresses the number of H in the matrix. Then, the difference schemes (13) and (14) can be expressed as

$$G_{J-1}U^{n+1} = U^n. \tag{16}$$

For some K_i, which are a series of integers satisfying $1 = K_0 < K_1 < K_2 < \cdots < K_m < K_{m+1} = J-1$, we can rewrite G_{J-1} as $G_{J-1} = E - D$, where

$$E = \begin{bmatrix} G_{K_1} & & & & 0 \\ & G_{K_2-K_1} & & & \\ & & \ddots & & \\ & & & G_{K_m-K_{m-1}} & \\ 0 & & & & G_{J-1-K_m} \end{bmatrix},$$

$$D = \begin{bmatrix} O_{K_1-1} & & & & & & & & & 0 \\ & O & rI & & & & & & & \\ & rI & O & & & & & & & \\ & & & O_{K_2-K_1-2} & & & & & & \\ & & & & O & rI & & & & \\ & & & & rI & O & & & & \\ & & & & & & \ddots & & & \\ & & & & & & & \ddots & & \\ & & & & & & & & O_{K_m-K_{m-1}-2} & \\ & & & & & & & & & O \quad rI \\ & & & & & & & & & rI \quad O \\ 0 & & & & & & & & & & O_{J-K_m-2} \end{bmatrix},$$

So, Equation (16) becomes as follows

$$(E - D)U^{n+1} = U^n. \tag{17}$$

Denote

$$U^{n+1(m)} = \left(U_{1,1}^{n+1(m)}, \ldots, U_{1,J-1}^{n+1(m)}, U_{2,1}^{n+1(m)}, \ldots, U_{2,J-1}^{n+1(m)}, \ldots, U_{J-1,1}^{n+1(m)}, \ldots, U_{J-1,J-1}^{n+1(m)}\right)^{\mathrm{T}}.$$

We get the following iterative method from Equation (17)

$$EU^{n+1(m+1)} = U^n + DU^{n+1(m)}, \quad m \geq 0. \tag{18}$$

E is a block diagonal matrix, so we can solve all the blocks at the same time.

In the following, we prove the convergence of the iterative method by computing $\|E^{-1}D\|_\infty$. Similar to §2, we consider the following equation.

$$Ex = D1$$

and $D1{=}(D1_{K_1}^T, D1_{K_2-K_1}^T, \ldots, D1_{J-1-K_m})^{\mathrm{T}}$ $D1_{K_1}{=}(0, \ldots, e)^{\mathrm{T}}$, $D1_{K_{i+1}-K_i}{=}(e, 0, \ldots, 0, e)^{\mathrm{T}}$ $(i = 1, 2, \ldots, m-1)$, $D1_{J-1-K_m} = (e, \ldots, 0)^{\mathrm{T}}$, where $e = (1, \ldots, 1)$.

$$|x_{i_0}| = \max_i |x_i| = \max_i |E^{-1}D1|$$

Therefore, from i_0th equation of $Ex = D1$

$$|E_{i_0 i_0}||x_{i_0}| = \left| D1_{i_0} - \sum_{j \neq i_0} E_{i_0 j} x_j \right| \leq \left| D1_{i_0} \right| + |x_{i_0}| \sum_{j \neq i_0} |E_{i_0 j}|$$

$$|x_{i_0}| \leq \frac{|D1_{i_0}|}{(|E_{i_0 i_0}| - \sum_{j \neq i_0} |E_{i_0 j}|)} \leq \max_i \frac{|D1_i|}{(|E_{ii}| - \sum_{j \neq i} |E_{ij}|)} = \frac{r}{1+r}$$

Thus, $\|E^{-1}D\|_\infty \leq (r/(1+r) < 1)$. This implies that the iterative scheme (18) is convergent.

4. Numerical experiment

In this section, we provide some results of numerical experiments to show the performance of the iterative methods.

Example 4.1 (*one-dimensional problem*) We consider the one-dimensional problem defined in Equations (1)–(3) with $u_0(x) = \sin(\pi x)$ and $a = 1$. The exact solution is $u(x, t) = \mathrm{e}^{-\pi^2 t} \sin(\pi x)$. We have made the numerical experiments in the Nankai Star cluster system by taking $h = 0.001$, $T = 1$ and the iterative accuracy as 10^{-6}. The speed-up of the PI method to the fully ID method is obtained. Tables 1 and 2 provide the numerical results for $r = 1$ and $r = 5$, respectively, where CPUs expresses the number of the processors. From Tables 1 and 2, we can observe that it takes much less, nearly 30 times, by using iterative methods than the classical methods. Meanwhile, the iterative methods do not sacrifice the accuracy of the solution. Moreover, as the equation scale increases, the efficiency increases. This point alone has a great significance in applied field.

Example 4.2 (*two-dimensional problem*) Now we consider the two-dimensional problem defined in Equations (10)–(12) with $u_0(x, y) = \sin(\pi x)\sin(\pi y)$. The exact solution is

Table 1. Numerical results with $r = 1$ for example 4.1.

Method	CPUs	Running time (s)	Speed-up	Efficiency %	$\max_{j,n} \lvert u_j^n - U_j^n \rvert$
ID	1	134.242651	–	–	0.000091
	2	93.875979	1.43	71.5	0.000081
	4	36.478981	3.68	92.0	0.000068
PI	8	18.414630	7.29	91.1	0.000037
	16	9.144595	14.68	91.8	0.000019
	32	4.479234	29.97	93.7	0.000010

Table 2. Numerical results with $r = 5$, for example 4.1.

Method	CPUs	Running time (s)	Speed-up	Efficiency %	$\max_{j,n} \lvert u_j^n - U_j^n \rvert$
ID	1	42.312038	–	–	0.000066
	2	29.588837	1.43	71.5	0.000096
	4	11.555299	3.66	91.5	0.000068
PI	8	5.861692	7.22	90.3	0.000037
	16	2.854293	14.82	92.6	0.000019
	32	1.402454	30.17	94.3	0.000009

$u(x,t) = \mathrm{e}^{-2\pi^2 t} \sin(\pi x) \sin(\pi y)$. We make the numerical experiments by taking $h = 0.001$, $r = 5$ and $T = 1$. To get the implicit difference solution and the iterative solutions of every sub-systems, we employ two classical methods. One is the block chasing method, the other is the SOR iterative method with the relaxation factor $\omega = 0.8$. The numerical results are shown in Tables 3 and 4. From the Tables 3 and 4, it is proved that the iterative methods have much more advantage in saving time cost than the classical methods. Whereas, the accuracy of the solution does not decrease.

Table 3. Numerical results by the block chasing method for example 4.2.

Method	CPUs	Running time (s)	Speed-up	Efficiency %	$\max_{i,j,n} \lvert u_{ij}^n - U_{ij}^n \rvert$
ID	1	3752.526468	–	–	0.000023
	2	2886.558822	1.30	65.0	0.000056
	4	1204.276787	3.12	77.9	0.000078
PI	8	614.765149	6.10	76.3	0.000025
	16	271.136305	13.84	86.7	0.000069
	32	130.007153	28.86	90.2	0.000033

Table 4. Numerical results by the SOR method for example 4.2.

Method	CPUs	Running time (s)	Speed-up	Efficiency %	$\max_{i,j,n} \lvert u_{ij}^n - U_{ij}^n \rvert$
ID	1	4428.572859	–	–	0.000189
	2	3256.303573	1.36	68.0	0.000466
	4	1392.632971	3.18	79.5	0.000012
PI	8	710.846372	6.23	77.9	0.000156
	16	313.638302	14.12	88.3	0.000009
	32	151.455981	29.24	91.4	0.000010

It is seen from the two numerical examples that the PI methods are convergent and may reach the linear speed-up. With the increase of CPUs, the efficiency is greatly raised, especially for the two-dimensional problem.

Next, for Example 4.1, we compare the PI method presented in Section 2 with the following known parallel finite difference methods.

1. The AGE method due to Evans and Abdullah [4], which is stable for any real number r.
2. The ASEI method presented in [2], which is also stable for any r. In the experiments, we divided the space domain into three segments.
3. The domain decomposition algorithm (DDA1) presented in [3], in which we took $H = 2h$. According to [3], the stability condition of the DDA1 must be $r \leq 2$.
4. The domain decomposition algorithm (DDA2) presented in [6] . The stability condition given in [6] is $r \leq 4$.
5. The high-order domain decomposition algorithm (HDDA) presented in [9] . According to [9], the stability condition of the HDDA is $r \leq 0.90$.

Taking $h = 1/40$, we calculated the absolute errors $|e_j^n| = |u_j^n - U_j^n|$ at the 400th time step. Tables 5–7 give the absolute errors at some points for $r = 0.5$, $r = 1.0$ and $r = 5.0$, respectively. It can be seen from Tables 5–7 that the high-order domain decomposition algorithm (HDDA) presented in [9] has indeed higher accuracy than others, but for large r the numerical solution blows up due to the serious restrict of stability condition. Except the HDDA, the PI method has the best accuracy for any r, then is the ASEI method.

Furthermore, we observe the running time between the PI method and ASEI method. Taking $h = 0.001$, $r = 4.0$, computing 400 time steps, the results are showed in Table 8, where

Table 5. The errors of numerical solutions at the 400th time step for $J = 40$, $r = 0.5$.

	x								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$\|e_j^n\| \times 10^5$									
PI	2.49	5.15	8.08	11.31	14.76	10.20	7.28	5.53	2.44
AGE	7.71	18.1	26.6	32.3	34.7	33.4	28.7	21.0	11.1
ASEI	5.66	10.7	14.7	15.5	16.3	15.5	14.7	10.7	5.66
DDA1	21.4	40.6	55.6	64.7	67.2	64.7	55.6	40.6	21.4
DDA2	20.2	38.1	51.8	57.9	60.9	58.9	51.1	37.6	19.9
HDDA	0.470	0.938	1.40	1.84	2.24	1.84	1.40	0.936	0.466
Exact solutions $\times 10^1$	0.90	1.71	2.36	2.77	2.91	2.77	2.36	1.71	0.90

Table 6. The errors of numerical solutions at the 400th time step for $J = 40$, $r = 1.0$.

	x								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$\|e_j^n\| \times 10^5$									
PI	2.39	4.69	6.80	8.60	9.99	8.84	6.06	4.56	2.30
AGE	31.3	63.6	89.5	106.0	113.0	108.0	91.9	66.9	35.2
ASEI	2.82	5.34	7.27	6.41	6.80	6.41	7.27	5.34	2.82
DDA1	20.6	39.0	53.4	62.2	64.6	62.2	53.4	39.0	20.6
DDA2	19.7	37.4	51.1	59.4	61.4	58.9	50.7	37.1	19.6
HDDA	–	–	–	–	–	–	–	–	–
Exact solutions $\times 10^2$	2.62	4.98	6.86	8.07	8.48	8.07	6.86	4.98	2.62

Table 7. The errors of numerical solutions at the 400th time step for $J = 40, r = 5.0$.

	x								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$\mid e_j^n \mid \times 10^8$									
PI	1.36	2.58	3.55	4.17	4.39	4.17	3.55	2.58	1.36
AGE	125.0	239.0	329.0	387.0	407.0	387.0	329.0	239.0	125.0
ASEI	4.38	8.37	1.16	1.63	1.71	1.63	1.61	8.37	4.38
DDA1	23.4	44.4	61.0	71.5	74.9	71.5	61.0	44.4	23.4
DDA2	–	–	–	–	–	–	–	–	–
HDDA	–	–	–	–	–	–	–	–	–
Exact solutions $\times 10^6$	1.36	2.58	3.55	4.17	4.39	4.17	3.55	2.58	1.36

Table 8. Comparison of running time between PI method and ASEI method.

	PI		ASEI	
CPUs	Running time	$\max_j \mid u_j^n - U_j^n \mid \times 10^7$	Running time	$\max_j \mid u_j^n - U_j^n \mid \times 10^7$
2	14.858067	2.35	14.257261	5.47
4	5.778137	3.25	5.779254	7.23
8	3.077124	1.54	3.272149	5.12
16	1.462457	6.95	1.793564	6.95
32	0.701225	0.48	0.813546	1.32

the iterative accuracy of the PI method is taken as 10^{-8}. It can be seen from Table 8 that the running time of PI method is less than that of ASEI method, under the same number of processors.

5. Conclusion

In conclusion, the PI methods based on the fully ID schemes are presented to solve one- and two-dimensional heat equations. The methods are proved to be convergent to the difference solutions. Numerical results show that the PI methods may reach the linear speed-up. With the increase of CPUs, the efficiency of the methods is greatly raised, especially for the two-dimensional problem. Compared with some parallel finite difference schemes, the accuracy of the PI method is best, except for the high-order domain decomposition algorithm (HDDA) presented in [9]. Meanwhile, the PI method is competitive with ASEI method in accuracy as well as in running time.

References

[1] U.M. Ascher, S.J. Ruuth, and B.T.R. Wetton, *Implicit-explicit methods for time-dependent partial differential equations*, SIAM J. Numer. Anal. 32 (1995), pp. 797–823.
[2] Z. Bo-lin, *Alternating segment explicit-implicit method for diffusion equation*, J. Numer. Method Comp. Appl. 12 (1991), pp. 245–253.
[3] C.N. Dawson, Q. Du, and T.F. Dupont, *A finite difference domain decomposition algorithm for numerical solution of the Heat equation*, Math. Comp. 57 (1991), pp. 63–71.
[4] D.J. Evans and A.R.B. Abdullah, *Group explicit method for parabolic equations*, Int. J. Comp. Math. 14 (1983), pp. 73–105.
[5] K.W. Morton and D.F. Mayers, *Numerical Solution of Partial Differential Equations 2nd Edition*, Cambridge University Press.
[6] G. Yuan, S. Zhu, and L. Shen, *Domain decomposition algorithm based on the group explicit formula for the heat equation*, Intern. J. Comp. Math. 82 (2005), pp. 1295–1306.

[7] B. Zhang and W. Li, *On alternating Crank–Nicolson scheme*, Parallel Comput. 20 (1994), pp. 897–902.
[8] S. Zhu, G. Yuan, and W. Sun, *Convergence and stability of explicit/implicit schemes for parabolic equations with discontinuous coefficients*, Int. J. Numer. Anal. Model. 1 (2004), pp. 131–145.
[9] S. Zhu, Z. Yu, and J. Zhao, *A high-order parallel finite difference algorithm*, Appl. Math. Comput. 183 (2006), pp. 365–372.