ELSEVIER

Comput. Methods Appl. Mech. Engrg. 130 (1996) 163–178

Computer methods in applied mechanics and engineering

On a new time integration method for solving time dependent partial differential equations

Wanxie Zhong[a,1], Zhu Jianing[b,*], Xiang-Xiang Zhong[c]

[a]*Research Institute of Engineering Mechanics, Dalian University of Technology, Dalian, PR China*

[b]*Department of Mathematics and Statistics & NSF Engineering Research Center, Mississippi State University, Mississippi State, MS 39762, USA*

[c]*Department of Applied Mathematics & Statistics, State University of New York, Stony Brook, NY 11794, USA*

Received 31 October 1994; revised 2 March 1995

Abstract

A numerical time integration algorithm that combines the high accuracy of the precise time integration method and the computational efficiency of the finite difference schemes is discussed in this paper. The algorithm is explicit and unconditionally stable. It can also be modified easily to achieve different levels of accuracy by using more integration points. Numerical examples of both linear and non-linear PDE problems are discussed in the paper.

1. Introduction

Time dependent partial differential equations have been used widely in science and engineering for solving many practical problems. While both the finite difference and finite element methods are popular today for the spatial discretization [1–3], the finite difference method remains the most frequently used method for the temporal discretization of time dependent PDEs, particularly in the numerical solution of parabolic and hyperbolic equations. Although the discretization of the temporal and spatial derivatives by finite difference method appears straightforward, the stability properties of various finite difference schemes can be very different. For example, given the equation

$$u_t = -au_x \tag{1}$$

and

$$\begin{aligned} t_n &= t_0 + n\,\Delta t\,, \qquad n = 0, 1, \ldots, N\,, \\ x_j &= x_0 + j\,\Delta x\,, \qquad j = 0, 1, \ldots, J\,, \end{aligned} \tag{2}$$

the FTCS (Forward Time Central Space) scheme gives

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = -a\,\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} \tag{3}$$

* Corresponding author.

[1] Supported by the National Science Foundation of China.

SSDI 0045-7825(95)00876-4

which is well known to be unconditionally unstable. If u_j^n in (3) is replaced by $(u_{j+1}^n + u_{j-1}^n)/2$, then the famous Lax scheme

$$\frac{u_j^{n+1} - (u_{j+1}^n + u_{j-1}^n)/2}{\Delta t} = -a\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x}$$

is conditionally stable when the condition

$$\frac{|a|\,\Delta t}{\Delta x} \leqslant 1 \tag{4}$$

is satisfied. This example demonstrates that the discretizations of spatial and temporal derivatives using finite difference schemes could be quite complicated.

The advantage of finite difference schemes is that at a given time step, each equation at a spatial grid point usually involves only the unknowns from a few neighboring grid points, thus generating sparse matrices which can be handled efficiently in the computation.

On the other hand, the precise time integration method [4, 5] represents the other extreme. It is based on the semi-analytical method which discretizes the spatial dimensions of a given time dependent PDE, such as

$$u_t = \nabla \cdot (a\nabla u) \tag{5}$$

to generate

$$\dot{\boldsymbol{u}} = \boldsymbol{H}\boldsymbol{u} + \boldsymbol{f}(t)\,, \tag{6}$$

where $\boldsymbol{u}$ is the vector containing the values of u defined on a discrete spatial grid. By utilizing the matrix exponential function, the precise time integration method can calculate numerical solution of (6) with an accuracy of machine precision, i.e. no truncation error in the temporal dimension. However, the algorithm involves dense matrix operations which are computationally expensive for large scale engineering problems.

The subdomain precise time integration algorithm to be discussed in this paper combines the computational efficiency of the finite difference schemes and the high accuracy of the precise time integration method. We will first discuss briefly the precise time integration method in the next section, and then introduce the subdomain precise time integration method in Section 3. Stability analysis for the single-point subdomain precise time integration method is given in Section 4, followed by the discussion of the convection and diffusion equation in Section 5. Extensions to non-linear heat equation and higher dimensional problems are given in Sections 6 and 7, respectively. Numerical examples are presented in Section 8, followed by the conclusions in Section 9.

2. Precise time integration of linear systems

To solve Eq. (6) we know from the theory of ordinary differential equations that it is important to get the solution of the corresponding homogeneous equation

$$\dot{\boldsymbol{v}} = \boldsymbol{H}\boldsymbol{v}\,. \tag{7}$$

Assume that the coefficient a in Eqs. (1) or (5) does not depend on time t, then $\boldsymbol{H}$ is a matrix whose elements are independent of t. Therefore, the general solution of Eq. (7) can be written as

$$\boldsymbol{v} = \exp(\boldsymbol{H}\cdot t)\cdot \boldsymbol{v}^{(0)} \tag{8}$$

where $\boldsymbol{v}^{(0)}$ is the initial state. Let the length of a time step be Δt and $\boldsymbol{T} = \exp(\boldsymbol{H}\cdot\Delta t)$, then at the time integration points $t_0 = 0$, $t_1 = \Delta t, \ldots, t_n = n\cdot\Delta t$, we have

$$\boldsymbol{v}(\Delta t) = \boldsymbol{v}^{(1)} = \boldsymbol{T}\cdot\boldsymbol{v}^{(0)}\,, \tag{9}$$

$$\vdots$$

$$\boldsymbol{v}(n\,\Delta t) = \boldsymbol{v}^{(n)} = \boldsymbol{T}\cdot\boldsymbol{v}^{(n-1)}\,. \tag{10}$$

where $\boldsymbol{v}^{(n)}$ is the solution vector at the time $t_n = n\,\Delta t$. It is clear that the key for the accurate computation of solution $\boldsymbol{v}^{(n)}$ is to determine a precise matrix $\boldsymbol{T}$. Once $\boldsymbol{T}$ is calculated, the solutions can be obtained by repeated matrix–vector multiplications. An efficient method for computing this exponential matrix was given in [6]. The method is based on the relation

$$\exp(\boldsymbol{H}\cdot\Delta t) = \left[\exp\left(\boldsymbol{H}\cdot\frac{\Delta t}{m}\right)\right]^m = [\exp(\boldsymbol{H}\cdot\tau)]^m \tag{11}$$

where m can be selected as an integer power of 2, i.e. $m = 2^M$. If m is large enough, then $\tau = \Delta t/m$ will be too small to cause any significant truncation error. We can therefore use the truncated Taylor expansion to approximate $\exp(\boldsymbol{H}\cdot\tau)$ as

$$\exp(\boldsymbol{H}\cdot\tau) \approx \boldsymbol{I} + \boldsymbol{H}\tau + (\boldsymbol{H}\tau)^2/2! + (\boldsymbol{H}\tau)^3/3! + (\boldsymbol{H}\tau)^4/4! = \boldsymbol{I} + \boldsymbol{T}_a\,. \tag{12}$$

With a small τ, the expansion should give a very good approximation to $\exp(\boldsymbol{H}\cdot\tau)$. Since the elements in matrix $\boldsymbol{T}_a$ is very small as compared to the unity in the identity matrix $\boldsymbol{I}$, they should be stored during the computations separately for better numerical accuracy, rather than added to the identify matrix $\boldsymbol{I}$. Otherwise, the accuracy will be lost due to the round-off errors.

With Eq. (12), matrix $\boldsymbol{T}$ can be computed as

$$\boldsymbol{T} = (\boldsymbol{I} + \boldsymbol{T}_a)^{2^M} = (\boldsymbol{I} + \boldsymbol{T}_a)^{2^{M-1}} \times (\boldsymbol{I} + \boldsymbol{T}_a)^{2^{M-1}} = \cdots \tag{13}$$

using only M matrix multiplications. Note that for any matrices $\boldsymbol{T}_b$ and $\boldsymbol{T}_c$, we have

$$(\boldsymbol{I} + \boldsymbol{T}_b)(\boldsymbol{I} + \boldsymbol{T}_c) = \boldsymbol{I} + \boldsymbol{T}_b + \boldsymbol{T}_c + \boldsymbol{T}_b \times \boldsymbol{T}_c\,. \tag{14}$$

Therefore, the matrix computation expressed by Eq. (13) can be accomplished by the following algorithm

```
Compute T_a from Eq. (12)
do i = 1, M
        T_a = 2T_a + T_a × T_a                    (15)
end do
        T = I + T_a                               (16)
```

Eqs. (12), (15) and (16) give the algorithm for computing the exponential matrix $\boldsymbol{T}$.

For the inhomogeneous system (6), assume that the inhomogeneous term $\boldsymbol{f}(t)$ is linear within a time step (t_n, t_{n+1}), then we have

$$\dot{\boldsymbol{u}} = \boldsymbol{H}\boldsymbol{u} + \boldsymbol{r}_0 + \boldsymbol{r}_1(t - t_n)\,, \tag{17}$$

where $\boldsymbol{r}_0$ and $\boldsymbol{r}_1$ are known vectors. This equation can be solved using the superposition principle. Let $\boldsymbol{\Phi}(t - t_n)$ be the fundamental solution of the corresponding homogeneous equation (7), i.e.

$$\dot{\boldsymbol{\Phi}} = \boldsymbol{H}\boldsymbol{\Phi}\,, \qquad \boldsymbol{\Phi}(0) = \boldsymbol{I}\,,$$

then the solution of Eq. (17) can be written as

$$\boldsymbol{u} = \boldsymbol{\Phi}(t - t_n)\cdot[\boldsymbol{u}^{(n)} + \boldsymbol{H}^{-1}(\boldsymbol{r}_0 + \boldsymbol{H}^{-1}\boldsymbol{r}_1)] - \boldsymbol{H}^{-1}[\boldsymbol{r}_0 + \boldsymbol{H}^{-1}\boldsymbol{r}_1 + \boldsymbol{r}_1\cdot(t - t_n)]\,. \tag{18}$$

In the numerical solution process, we do not need the analytic form of $\boldsymbol{\Phi}$, only the matrix $\boldsymbol{\Phi}(t_{n+1} - t_n) = \boldsymbol{\Phi}(\tau) = \boldsymbol{T}$ is needed to calculate $\boldsymbol{u}^{(n+1)}$ from Eq. (18). Therefore, the precise time integration formula for the inhomogeneous equation (6) can be given as

$$\boldsymbol{u}^{(n+1)} = \boldsymbol{T}[\boldsymbol{u}^{(n)} + \boldsymbol{H}^{-1}(\boldsymbol{r}_0 + \boldsymbol{H}^{-1}\boldsymbol{r}_1)] - \boldsymbol{H}^{-1}[\boldsymbol{r}_0 + \boldsymbol{H}^{-1}\boldsymbol{r}_1 + \boldsymbol{r}_1\cdot\tau]\,. \tag{19}$$

Since the main step in the precise time integration is the computation of the exponential matrix $\boldsymbol{T}$, the major error in the time integration is from the truncated expansion (12). The truncation error is

proportional to the term $\|\boldsymbol{H}\cdot\Delta t\|^5/5!$, and the relative error is proportional to $\|\boldsymbol{H}\cdot\Delta t\|^4/5!$. Assume that we have all the eigensolutions of $\boldsymbol{H}$, then

$$\boldsymbol{H}\boldsymbol{Y}=\boldsymbol{Y}\lceil\mu\rfloor,$$

or

$$\boldsymbol{H}=\boldsymbol{Y}\lfloor\mu\rfloor\boldsymbol{Y}^{-1}, \tag{20}$$

where the columns of $\boldsymbol{Y}$ are the eigenvectors of $\boldsymbol{H}$ and the elements in the diagonal matrix $\lceil\mu\rfloor$ are the corresponding eigenvalues. From Eq. (12), we have

$$\exp(\boldsymbol{H}\cdot\tau)=\boldsymbol{Y}\exp(\lceil\mu\rfloor\tau)\boldsymbol{Y}^{-1}=\boldsymbol{Y}\lceil\exp(\mu\tau)\rfloor\boldsymbol{Y}^{-1},$$

so that the approximation given by Eq. (12) becomes

$$\exp(\mu_i\cdot\tau)\approx 1+\mu_i\tau+(\mu_i\tau)^2/2+(\mu_i\tau)^3/3!+(\mu_i\tau)^4/4!. \tag{21}$$

The relative truncation error is proportional to

$$(\mu_i\cdot\tau)^5/5!, \tag{22}$$

where the errors corresponding to different eigenvalues have been separated. To keep the truncation error within the machine accuracy (16 decimal digits in double precision), we must have

$$(\mu_i\cdot\tau)^4\leqslant 10^{-16},$$

or

$$\mathrm{abs}(\mu_i)\cdot\frac{\Delta t}{2^M}\leqslant 10^{-4}.$$

When M is taken as 20, we have

$$\mathrm{abs}(\mu_i)\cdot\Delta t\leqslant 100. \tag{23}$$

Even with $\max_i(\mathrm{abs}(u_i))=100$, a time step size of $\Delta t=1$ will still provide time integration results accurate to machine precision.

3. Subdomain precise time integration method

For simplicity of discussions for the subdomain precise time integration, we use the one-dimensional heat equation to explain the method. Given the equation

$$u_t=u_{xx},\qquad 0\leqslant x\leqslant 2,\quad t\geqslant 0 \tag{24}$$

with

$$u(0,t)=0,\qquad u(2,t)=0, \tag{25}$$

$$\begin{aligned} &u(x,0)=10x,\qquad 0\leqslant x\leqslant 1,\\ &u(x,0)=10(2-x),\qquad 1\leqslant x\leqslant 2, \end{aligned} \tag{26}$$

if the spatial derivative is discretized on a grid of x_i, $i=0,1,\ldots,N$, using the central finite difference scheme, we will have a set of ODEs

$$\dot{u}_i=\frac{1}{\Delta x^2}(u_{i-1}-2u_i+u_i),\quad i=1,\ldots,N-1. \tag{27}$$

Eq. (27) can be written as a matrix equation

$$\dot{\boldsymbol{u}}=\boldsymbol{A}\boldsymbol{u} \tag{28}$$

where $u=\{u_1,u_2,\ldots,u_{N-1}\}^{\mathrm{T}}$ and A is an $(N-1)\times(N-1)$ tri-diagonal matrix with -2 in the main

Table 1
Transient solution of Eq. (24) at different time levels obtained using the whole domain precise time integration with $\Delta t = 0.1$

t	$x = 0.2$	$x = 0.4$	$x = 0.6$	$x = 0.8$	$x = 1.0$
0.1	1.87919	3.63355	5.10390	6.09979	6.45430
0.2	1.52439	2.90645	4.01211	4.72768	4.97546
0.3	1.19802	2.27955	3.13885	3.69121	3.88167
0.4	0.93726	1.78286	2.45404	2.88504	3.03357
0.5	0.73277	1.39383	1.91845	2.25529	2.37136

diagonal and 1 in the sub- and super-diagonals. With $N = 20$, Eq. (28) can be solved easily by the precise time integration method discussed in Section 2 since it is a small problem. The results are given in Table 1.

The basic idea of the subdomain precise time integration is to apply the integration process to only a few rows of the system equations in (28) at a time, instead of the complete system. For example, the 3-point subdomain integration will treat three equations of (28) simultaneously. For an interior point i, we will have

$$\begin{Bmatrix} \dot{u}_{i-1} \\ \dot{u}_i \\ \dot{u}_{i+1} \end{Bmatrix} = \frac{1}{\Delta x^2} \begin{bmatrix} 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & 1 \end{bmatrix} \begin{Bmatrix} u_{i-2} \\ u_{i-1} \\ u_i \\ u_{i+1} \\ u_{i+2} \end{Bmatrix}, \quad i = 2, \ldots, N-2 \tag{29}$$

or

$$\begin{Bmatrix} \dot{u}_{i-1} \\ \dot{u}_i \\ \dot{u}_{i+1} \end{Bmatrix} = \frac{1}{\Delta x^2} \begin{bmatrix} -2 & 1 & \\ 1 & -2 & 1 \\ & 1 & -2 \end{bmatrix} \begin{Bmatrix} u_{i-1} \\ u_i \\ u_{i+1} \end{Bmatrix} + \frac{1}{\Delta x^2} \begin{Bmatrix} u_{i-2} \\ 0 \\ u_{i+2} \end{Bmatrix}, \quad i = 2, \ldots, N-2. \tag{30}$$

This equation system can be considered as a system of three ODEs with an inhomogeneous term. Assume we have finished calculation at time step n, then $u_i^{(n)}$, $i = 1, \ldots, N-1$, are all known. To calculate $u_i^{(n+1)}$, $i = 1, \ldots, N-1$, we have to integrate Eq. (30). When the subdomain precise time integration is applied to Eq. (30), we have to approximate the inhomogeneous term $\{u_{i-2}, 0, u_{i+2}\}^{\mathrm{T}}$. An explicit approximation would be to use the values $u_{i-2}^{(n)}$ and $u_{i+2}^{(n)}$ from the previous time step. In the extreme case, when there is only one subdomain which covers all interior points in the whole domain, the subdomain precise time integration method becomes the whole domain precise time integration method discussed in Section 2.

Having approximated the inhomogeneous term in Eq. (30), we can write it as

$$\dot{v}_i = H v_i + f_i^{(n)}, \tag{31}$$

with

$$v_i = \{u_{i-1}, u_i, u_{i+1}\}^{\mathrm{T}},$$

$$f_i^{(n)} = \frac{1}{\Delta x^2} \{u_{i-2}^{(n)}, 0, u_{i+2}^{(n)}\}^{\mathrm{T}}, \quad i = 2, 3, \ldots, N-2.$$

Let $T = \exp(H \cdot \Delta t)$, which can be calculated using the method discussed in Section 2, we will have, based on (19),

$$v_i^{(n+1)} = T v_i^{(n)} + (T - I) H^{-1} f_i^{(n)}. \tag{32}$$

Note that since in this case H is independent of the indices i and n, the matrix T needs to be calculated only once.

In Eq. (32) three values $u_{i-1}^{(n+1)}$, $u_i^{(n+1)}$, and $u_{i+1}^{(n+1)}$ are calculated simultaneously. It is expected that the value $u_i^{(n+1)}$ should be more accurate than the other two values because there is no approximation for the middle equation in the inhomogeneous term of Eq. (30). We see from Fig. 1 that $u_i^{(n+1)}$,

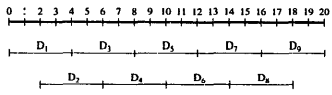

Fig. 1. The original domain D is decomposed into 9 subdomains D_1–D_9. $u_i^{(n+1)}$ with i even is calculated as the middle point in each subdomain. $u_i^{(n+1)}$, $i = 3, 5, \ldots, 17$ are calculated in two overlapping subdomains.

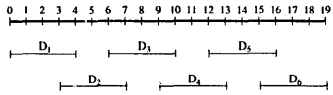

Fig. 2. The original domain D is decomposed into 6 subdomains D_1–D_6. Each $u_i^{(n+1)}$ is calculated in only one subdomain.

$i = i = 2, \ldots, N-2$, can all be calculated as a middle point value in the 3-point subdomain integration method. The values $u_1^{(n+1)}$ and $u_{N-1}^{(n+1)}$ will have to be calculated simultaneously with $u_2^{(n+1)}$ and $u_{N-2}^{(n+1)}$, respectively. In this way, with $N+1$ grid points, there are $N-3$ subdomains and the values of $u_{i-1}^{(n+1)}$ and $u_{i+1}^{(n+1)}$ in the ith subdomain are not used except in the first and last subdomains. To save computation, the overlapping subdomains can be reduced. As shown in Fig. 1, we can use only 9 subdomains, instead of the maximum of 17. The values $u_i^{(n+1)}$ where i is even are calculated as the middle point value of each subdomain, while $u_i^{(n+1)}$'s with $i = 3, 5, \ldots, 17$ are the average values from two subdomains. For the subdomains shown in Fig. 2, there is no overlapping and each $u_i^{(n+1)}$ is calculated only in one subdomain.

To improve the accuracy in the time dimension, multi-step time integration can be easily incorporated into subdomain precise time integration method. For example, if a leap-frog type algorithm is used in the subdomain precise time integration method, we will have

$$v_i^{(n+1)} = T(2\Delta t)v_i^{(n-1)} + (T - I)H^{-1}f_i^n \tag{33}$$

with $T(2\Delta t) = \exp(2H\,\Delta t)$ which is very similar to the results given by Eq. (32). Note that for the first step when $n = 0$, the one step forward integration algorithm given by Eq. (32) must be used to calculate $u_i^{(1)}$, $i = 1, \ldots, N-1$, in order to use the leap-frog type method at later time steps.

The concept of implicit finite difference schemes can also be easily applied to the subdomain precise time integration method. For example, similar to the Crank–Nicolson finite difference scheme, we can approximate the inhomogeneous term f_i in Eq. (31) by

$$f_i = f_i^{(n)} + (t - t_n)(f_i^{(n+1)} - f_i^{(n)})/\Delta t = r_0 + r_1(t - t_n) \tag{34}$$

with $r_0 = f_i^n$, $r_1 = (f_i^{n+1} - f_i^n)/\Delta t$. As discussed in Section 2, Eq. (30) can then be integrated as

$$v_i^{(n+1)} = T[v_i^{(n)} + H^{-1}(r_0 + H^{-1}r_1)] - H^{-1}(r_0 + H^{-1}r_1 + r_1\,\Delta t)\,. \tag{35}$$

Since r_1 in the right-hand side of Eq. (35) depends on $u_{i-2}^{(n+1)}$ and $u_{i+2}^{(n+1)}$, Eq. (35) is an implicit equation with unknowns in the right-hand side.

The main purpose of using implicit finite difference schemes is to overcome the stability constraint associated with the explicit finite difference schemes. As will be discussed in the next section, explicit subdomain precise time integration methods have much better stability property than that of the explicit finite difference schemes. Therefore, the use of implicit subdomain precise time integration is not as pressing as in the case of finite difference schemes.

4. Stability analysis

It should be noted that the number of interior points in a subdomain can be adjusted easily depending on the requirement of accuracy in the time integration. If a subdomain includes all the points in the whole spatial domain, then it becomes the whole domain precise time integration method with a truncation error in the time dimension equal to machine zero. On the other hand, we can use only one

interior point in each subdomain which greatly simplifies the computation and analyses of the algorithms. For the ith subdomain, with only one interior point, we have

$$\dot{u}_i = \frac{1}{\Delta x^2}(u_{i-1} - 2u_i + u_{i+1}) = \frac{2}{\Delta x^2}u_i + \frac{1}{\Delta x^2}(u_{i-1} + u_{i+1}), \quad i = 1, \ldots, N-1. \tag{36}$$

Assume $u_i^{(n)}$, $i = 0, 1, \ldots, N$, are all known, the one-point explicit subdomain precise time integration of (36) leads to

$$u_i(t) = C\,\mathrm{e}^{-2\Delta t/\Delta x^2} + \frac{u_{i-1}^{(n)} + u_{i+1}^{(n)}}{2}, \quad t_n \leqslant t \leqslant t_{n+1}. \tag{37}$$

Using the conditions $u_i(t_n) = u_i^{(n)}$, $u_i(t_{n+1}) = u_i(t_n + \Delta t) = u_i^{(n+1)}$, we have

$$u_i^{(n+1)} = \left(u_i^{(n)} - \frac{u_{i-1}^{(n)} + u_{i+1}^{(n)}}{2}\right)\mathrm{e}^{-2\Delta t/\Delta x^2} + \frac{(u_{i-1}^{(n)} + u_{i+1}^{(n)})}{2}. \tag{38}$$

Eq. (38) is very similar to the FTCS (forward time central space) finite difference scheme. When Δt is small, we can use a first-order approximation for $\mathrm{e}^{-2\Delta t/\Delta x^2}$ to get

$$\mathrm{e}^{-2\Delta t/\Delta x^2} \approx 1 - \frac{2\Delta t}{\Delta x^2}. \tag{39}$$

Substituting (39) back into (38), we get

$$u_i^{(n+1)} = u_i^{(n)} + \frac{\Delta t}{\Delta x^2}(u_{i-1}^{(n)} - 2u_i^{(n)} + u_{i+1}^{(n)}), \tag{40}$$

which is exactly the FTCS finite difference scheme. It is well known that for the FTCS scheme given in (40), the stability condition is

$$\frac{\Delta t}{\Delta x^2} \leqslant \frac{1}{2}, \tag{41}$$

which could be very restrictive for a fine spatial grid with small Δx or three-dimensional problems.

On the other hand, for the scheme given by (38), we can use the Von Neumann stability analysis to get the stability criterion as

$$|\lambda + (1 - \lambda)\cos\theta| \leqslant 1 \tag{42}$$

where $\lambda = \mathrm{e}^{-2\Delta t/\Delta x^2}$ and $0 < \lambda \leqslant 1$. It is not difficult to verify that the inequality in (42) is always satisfied for any λ and θ. Therefore, the explicit one-point subdomain precise time integration given by (38) is unconditionally stable.

For the leap-frog finite difference scheme

$$u_i^{(n+1)} = u_i^{(n-1)} + \frac{2\Delta t}{\Delta x^2}(u_{i-1}^{(n)} - 2u_i^{(n)} + u_{i+1}^{(n)}), \tag{43}$$

it is also well known that the stability requirement is

$$\left|\frac{-b \pm \sqrt{b^2 + 4}}{2}\right| \leqslant 1 \tag{44}$$

where $b = 8\Delta t/\Delta x^2 \sin^2\theta/2$. If $\sin^2\theta/2 \neq 0$, it is obvious that the inequality in (44) cannot be satisfied. Therefore, the method is unconditionally unstable.

With the one-point precise time integration method, a similar three level formula can be obtained from Eq. (36) by integrating from t_{n-1} to t_{n+1} and setting $1/\Delta x^2(u_{i-1} + u_{i+1})$ to be $1/\Delta x^2(u_{i-1}^{(n)} + u_{i+1}^{(n)})$:

$$u_i^{(n+1)} = \left[u_i^{(n-1)} - \frac{u_{i-1}^{(n)} + u_{i+1}^{(n)}}{2}\right]\mathrm{e}^{-4\Delta t/\Delta x^2} + \frac{u_{i-1}^{(n)} + u_{i+1}^{(n)}}{2}. \tag{45}$$

Again, using a first-order approximation for $e^{-4\Delta t/\Delta x^2}$ will lead to the leaf-frog finite difference scheme of (43). The Von Neumann stability analysis of Eq. (45) shows that the stability criterion

$$\left|\frac{1}{2}[(1-\lambda^2)\cos\theta \pm \sqrt{(1-\lambda^2)^2\cos^2\theta + 4\lambda^2}]\right| \leqslant 1\,, \tag{46}$$

where $\lambda = e^{-4\Delta t/\Delta x^2}$, is satisfied unconditionally. Therefore, the one-point subdomain precise time integration algorithm given by (45) is also unconditionally stable.

5. Convection–diffusion equation

Another equation that is widely used in practice is the convection–diffusion equation

$$u_t = Du_{xx} - Cu_x\,. \tag{47}$$

We only consider the case when the diffusion and convection coefficients D and C are constants. Since Eq. (47) has two obvious solutions $u =$ constant and $u = x - Ct$, which is a wave propagating at a velocity C, a good spatial discretization scheme must admit two solutions. Thus, we have

$$(u_i)_t = c_1 u_{i-1} - c_2 u_i + c_3 u_{i+1}\,, \quad i = 1, \ldots, N-1 \tag{48}$$

with

$$\begin{aligned} c_1 &= \left(\frac{C}{2} + c_d\right)\Big/\Delta x_i\,,\\ c_3 &= \left(-\frac{C}{2} + c_d\right)\Big/\Delta x_{i+1}\,,\\ c_2 &= c_1 + c_3\,,\\ \Delta x_i &= x_i - x_{i-1}\,, \qquad \Delta x_{i+1} = x_{i+1} - x_i\,. \end{aligned} \tag{40}$$

The third constant c_d can be determined by requiring that scheme (48) contain the asymptotic steady state solution of (47) which is $u = e^{(Cx)/D}$, thus obtaining

$$\begin{aligned} &p_i = C\,\Delta x_i/D\,, \qquad p_{i+1} = C\,\Delta x_{i+1}/D\,,\\ &f_i = [1 - \exp(-p_i)]/\Delta x_i\,,\\ &g_{i+1} = [\exp(p_{i+1}) - 1]/\Delta x_{i+1}\,,\\ &c_d = \frac{C}{2}(g_{i+1} + f_i)/(g_{i+1} - f_i)\,. \end{aligned} \tag{50}$$

This spatial discretization is upwinding since $c_1 > c_3 > 0$ when $C > 0$. As discussed in Section 3, the subdomain precise time integration can be applied to (48) easily. The single-point leap-frog type algorithm will lead to

$$u_i^{(n+1)} = \left[u_i^{(n-1)} - \frac{c_3 u_{i+1}^{(n)} + c_1 u_{i-1}^{(n)}}{c_2}\right] e^{-2c_2\Delta t} + \frac{c_3 u_{i+1}^{(n)} + c_1 u_{i-1}^{(n)}}{c_2}\,. \tag{51}$$

For stability analysis of algorithm (51), the Von Neumann method leads to the relation

$$\xi = \lambda\xi^{-1} + (1-\lambda)\beta \tag{52}$$

where

$$\begin{aligned} \lambda &= e^{-2c_2\Delta t}\\ \beta &= \frac{c_3}{c_2}e^{i\theta_1} + \frac{c_1}{c_2}e^{i\theta_2}\,. \end{aligned}$$

Since $c_1 > 0$, $c_3 > 0$, and $c_2 = c_1 + c_3$, we have $|\beta| \leq 1$. The two roots η_1 and η_2 of Eq. (52) have the relations

$$\begin{aligned} &\eta_1 \cdot \eta_2 = -\lambda \\ &\eta_1 + \eta_2 = (1-\lambda)\beta \end{aligned} \tag{53}$$

with $0 < \lambda \leq 1$. Let $\eta_1 = r_1 e^{i\theta}$ and $\eta_2 = -r_2 e^{i\theta}$, where $r_1 > 0$, $r_2 > 0$, and

$$\begin{aligned} &r_1 \times r_2 = \lambda , \\ &(r_1 - r_2)\cos\theta + i(r_1 + r_2)\sin\theta = (1-\lambda)\beta . \end{aligned} \tag{54}$$

Squaring the absolute value of both sides of the second equation in (54), we have

$$r_1^2 + r_2^2 - 2r_1 r_2 \cos 2\theta = (1-\lambda)^2 |\beta|^2 .$$

Using the first equation in (54), we can prove that

$$(r_1 + r_2)^2 = (1-\lambda)^2|\beta|^2 + 2\lambda(1 + \cos 2\theta) \leq (1-\lambda)^2 + 4\lambda = (1+\lambda)^2 . \tag{55}$$

After using the first equation in (54) one more time, we finally get

$$(r_i - \lambda)(r_i - 1) \leq 0 , \quad i = 1, 2$$

which proves $\lambda \leq r_i \leq 1$, and hence the unconditional stability of algorithm (51).

6. Non-linear equations

In this section, we study the non-linear heat equation

$$C\frac{\partial T}{\partial t} = \nabla \cdot (D \nabla T) + Q \tag{56}$$

where T is temperature, t is time, Q is heat source, D is the heat capacitance coefficient, and C is the heat conductivity coefficient. Both C and D are functions of T, which makes the equation non-linear.

By introducing the new variables

$$\phi = \int_{T_0}^{T} D(s)\, ds , \tag{57}$$

so that

$$D(T) = \frac{\partial \phi}{\partial T} ,$$

we can reduce Eq. (56) to

$$CD^{-1}\frac{\partial \phi}{\partial t} = \nabla^2 \phi + Q , \tag{58}$$

where Q and CD^{-1} are now functions of ϕ. Most of the existing numerical algorithms for solving Eq. (58) start with spatial discretization to reduce Eq. (58) to a system of ODEs defined at all grid points, then use implicit finite difference schemes, such as Crank–Nicolson [3] method, for time integration.

We now extend the single-point precise time integration method to this non-linear heat equation. The algorithm is explicit and has second order of accuracy. For simplicity of the discussion, we will concentrate on a one-dimensional case.

Similar to the procedure described in Section 3, we discretize Eq. (58) on a uniform spatial grid of $N+1$ points to obtain

$$\dot{\phi}_i = E_i(\phi_{i-1} - 2\phi_i + \phi_{i+1}) + q_i , \quad i = 1, 2, \ldots, N-1 , \tag{59}$$

where

$$E_i = D(\phi_i)C^{-1}(\phi_i)/(\Delta x)^2\,,$$
$$q_i = \bar{E}_i \cdot Q(\phi_i)\,. \tag{60}$$

For time integration, we discretize the time domain by

$$t_n = n \cdot \Delta t\,, \quad n = 0, 1, \ldots\,. \tag{61}$$

When Δt is small, E_i can be approximated within $(t_n, t_n + \Delta t)$ by

$$E_i \simeq k_0^n + k_1^n(\phi_i - \phi_i^n)$$
$$q_i \simeq p_0^n + p_1^n(\phi_i - \phi_i^n) \tag{62}$$

where

$$k_0^n = E(\phi_i^n)\,, \qquad k_1^n = \left.\frac{\mathrm{d}E_i}{\mathrm{d}\phi_i}\right|_{\phi_i^n}\,,$$
$$p_0^n = q_i(\phi_i^n)\,, \qquad p_1^n = \left.\frac{\mathrm{d}q_i}{\mathrm{d}\phi_i}\right|_{\phi_i^n}\,. \tag{63}$$

The single-point precise time integration calls from substituting ϕ_{i+1} and ϕ_{i-1} in Eq. (59) by ϕ_{i+1}^n and ϕ_{i-1}^n, respectively. Eq. (59) can then be written as

$$\dot{\phi}_i = (k_0^n - k_1^n\phi_i^n + k_1^n\phi_i)(\phi_{i+1}^n + \phi_{i-1}^n - 2\phi_i) + p_0^n + p_1^n(\phi_i - \phi_i^n)\,. \tag{64}$$

Note that in Eq. (64), ϕ_i is the only unknown function to be integrated from t_{n-1} to t_{n+1}. For the first time step, the integration will be from t_0 to t_1. Since the right-hand side of Eq. (64) is a quadratic polynomial of ϕ_i, it can be integrated analytically. For simplicity, assume the source term q is zero (meaning p_0^n and p_1^n are zero in (62)), then Eq. (64) is simplified to

$$\dot{\phi}_i = (a + b\phi_i)(\alpha + \beta\phi_i)\,, \tag{65}$$

with

$$a = k_0^n - k_1^n\phi^n\,, \qquad b = k_1^n\,, \qquad \alpha = \phi_{i-1}^n + \phi_{i+1}^n\,, \qquad \beta = -2\,.$$

Integrating Eq. (65), we obtain

$$\frac{\alpha + \beta\phi_i}{a + b\phi_i} = c_n\,\mathrm{e}^{kt}\,, \tag{66}$$

with $k = a\beta - \alpha b$ and c_n the integration constant. Using the condition $\phi_i = \phi_i^{n-1}$ when $t = t_{n-1}$, we obtain

$$c_n = \frac{\alpha + \beta\phi_i^{n-1}}{a + b\phi_i^{n-1}}\,\mathrm{e}^{-kt_{n-1}}\,, \tag{67}$$

which leads to

$$\frac{\alpha + \beta\phi_i^{n+1}}{a + b\phi_i^{n+1}} = \frac{\alpha + \beta\phi_i^{n-1}}{a + b\phi_i^{n-1}}\,\mathrm{e}^{2k\,\Delta t}\,. \tag{68}$$

From Eq. (68), we can solve ϕ_i^{n+1} explicitly.

For the first time step, the integration goes from t_0 to t_1, which gives

$$\frac{\alpha + \beta\phi_i^1}{a + b\phi_i^1} = \frac{\alpha + \beta\phi_i^0}{a + b\phi_i^0}\,\mathrm{e}^{k\,\Delta t} \tag{69}$$

for ϕ_i^1 at the first time step.

7. Extension to higher-dimensional problems

The single-point precise time integration algorithm as given by (38) or (45) can easily be extended to the higher-dimensional problems. For example, for the equation

$$u_t = u_{xx} + u_{yy}$$

with the Dirichlet boundary condition, the standard central finite difference scheme results in

$$\dot{u}_{i,j} = \frac{1}{\Delta h^2}(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}) + \frac{1}{\Delta h^2}(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}), \quad i, j = 1, \ldots, N-1, \tag{70}$$

where we have assumed that a two-dimensional rectangular domain is discretized using a uniform $N \times N$ grid with the grid spacing h, and $u_{i,j}$ represents the function value at the point (i, j). Note that Eq. (70) can be written as

$$\dot{u}_i = -\frac{4}{\Delta h^2}u_{i,j} + \frac{1}{\Delta h^2}(u_{i,j-1} + u_{i-1,j} + u_{i+1,j} + u_{i,j+1}), \quad i, j = 1, \ldots, N-1. \tag{71}$$

Assuming that the solution $u^n_{i,j}$ at all grid points are known from the previous time level t_n, then the single-point precise time integration of Eq. (71) leads to

$$u_i(t) = C\,e^{-4t/\Delta h^2} + \frac{u^{(n)}_{i,j-1} + u^{(n)}_{i-1,j} + u^{(n)}_{i+1,j} + u^{(n)}_{i,j+1}}{4}, \quad t_n \leqslant t \leqslant t_{n+1}. \tag{72}$$

Using the conditions $u_{i,j}(t_n) = u^{(n)}_{i,j}$ and $u_{i,j}(t_{n+1}) = u^{(n+1)}_{i,j}$, we have

$$u^{(n+1)}_{i,j} = \left(u^{(n)}_{i,j} - \frac{u^{(n)}_{i,j-1} + u^{(n)}_{i-1,j} + u^{(n)}_{i+1,j} + u^{(n)}_{i,j+1}}{4}\right) e^{-4\Delta t/\Delta h^2} + \frac{u^{(n)}_{i,j-1} + u^{(n)}_{i-1,j} + u^{(n)}_{i+1,j} + u^{(n)}_{i,j+1}}{4}. \tag{73}$$

This is the single-point precise time integration scheme for the two-dimensional heat equation. If we expand $e^{-4\Delta t/\Delta h^2}$ and keep only two terms of the expansion in Eq. (73), i.e.

$$e^{-4\Delta t/\Delta h^2} \approx 1 - \frac{4\Delta t}{\Delta h^2}, \tag{74}$$

then we will have

$$u^{(n+1)}_{i,j} = u^{(n)}_{i,j} + \frac{\Delta t}{\Delta h^2}(u^{(n)}_{i,j-1} + u^{(n)}_{i-1,j} - 2u^{(n)}_{i,j} + u^{(n)}_{i+1,j} + u^{(n)}_{i,j+1}) \tag{75}$$

which is exactly the FTCS finite difference scheme for a two-dimensional heat equation. It is well known that the algorithm given by Eq. (75) is subject to the stability restriction of

$$\frac{\Delta t}{\Delta x^2} \leqslant \frac{1}{4}.$$

For the stability analysis of the algorithm given by (73), we can use the Von Neumann method to get a stability criterion of

$$\left|\lambda + \frac{1}{2}[(1-\lambda)\cos\theta_1 + (1-\lambda)\cos\theta_2]\right| \leqslant 1 \tag{76}$$

where $\lambda = e^{-4\Delta t/\Delta h^2} < 1$. It is obvious that this criterion is satisfied unconditionally.

The extension of the multipoint precise time integration method to higher-dimensional problems can be carried out in a similar straightforward manner. The algebraic manipulation and analysis, however, will be lengthy and more involved than that for the single point algorithms.

8. Numerical examples

The first numerical example given here is the solution of Eqs. (24)–(26). The spatial domain is discretized using a uniform grid of twenty-one grid points, with $\Delta x = 0.1$. The solution is symmetric with respect to $x = 1.0$. We used first-order forward time subdomain precise time integration method with 1, 3 and 7 interior points, respectively. The numerical values are given in Table 2 .

The values with 19 interior points actually correspond to the whole domain precise time integration result. Note that $\Delta t = 0.005$ is the boundary of numerical instability for the FTCS finite difference scheme. Since the subdomain precise time integration algorithm is unconditionally stable, the numerical values in Table 2 with different Δt are still bounded. It is clear that the results obtained using 7 interior points are reasonably close to the values obtained using 19 interior points which are actually the exact solution of the discretized ODEs to the machine precision.

Tables 3 and 4 give the transient solutions of Eq. (24) at different time levels using 3 and 1 interior points, respectively. Comparing Tables 3 and 4 with Table 1, we see clearly that the results obtained using three interior points agree well with that obtained using 19 points. In many practical engineering computations, numerical results accurate to the second or third digit are considered to be good enough.

Fig. 3 shows the transient solutions of Eq. (24) obtained using the whole domain precise time integration and single-point subdomain precise time integration. The algorithm used is the leap-frog algorithm given by (45). The results obtained using three interior points are so close to that of the whole domain precise time integration that the two curves are not distinguishable in the figure.

The second example is the numerical solution of Eq. (47) with $C = 18$, $D = 1$, and the following boundary and initial conditions

Table 2
Solution of Eq. (24) at $t = 0.50$ obtained using different time step Δt and different number of interior points nip in the subdomain integration

Δt	nip	$x = 0.2$	$x = 0.4$	$x = 0.6$	$x = 0.8$	$x = 1.0$
0.1	19	0.73277	1.39383	1.91845	2.25529	2.37136
0.0025	1	0.80425	1.52979	2.10561	2.47533	2.60273
0.005	1	0.98282	1.86954	2.57337	3.02535	3.18111
0.005	3	0.73701	1.40244	1.93056	2.26965	2.38650
0.010	3	0.76993	1.46537	2.01733	2.37174	2.49386
0.005	7	0.73263	1.39349	1.91787	2.25531	2.37089
0.010	7	0.73164	1.39125	1.91711	2.25271	2.37040

Table 3
Transient solution of Eq. (24) obtained using three interior points with $\Delta t = 0.005$

t	$x = 0.2$	$x = 0.4$	$x = 0.6$	$x = 0.8$	$x = 1.0$
0.1	1.87963	3.63634	5.11061	6.11056	6.46691
0.2	1.52737	2.91348	4.02282	4.74104	4.98980
0.3	1.20202	2.28809	3.15110	3.70589	3.89721
0.4	0.94155	1.79172	2.46658	2.89995	3.04930
0.5	0.73701	1.40244	1.93056	2.26965	2.38650

Table 4
Transient solution of Eq. (24) obtained using one interior point with $\Delta t = 0.0025$

t	$x = 0.2$	$x = 0.4$	$x = 0.6$	$x = 0.8$	$x = 1.0$
0.1	1.90465	3.69137	5.20025	6.22974	6.59783
0.2	1.58058	3.01568	4.16648	4.91297	5.17182
0.3	1.26697	2.41113	3.32069	3.90566	4.10743
0.4	1.00990	1.92110	2.64443	3.10897	3.26907
0.5	0.80425	1.52979	2.10561	2.47533	2.60273

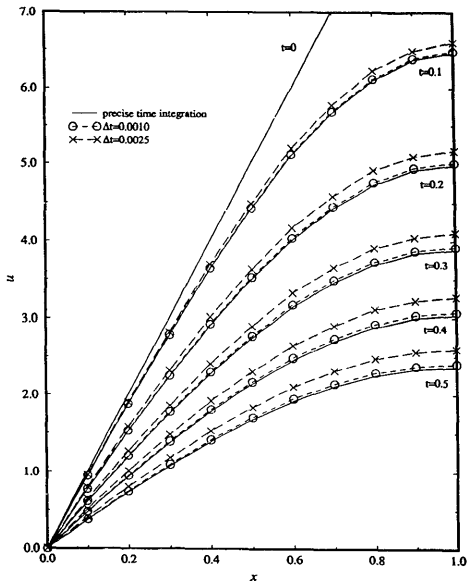

Fig. 3. Transient solutions of Eq. (24).

$u(0, t) = 0\,, \quad t > 0\,,$

$u(2, t) = 20\,, \quad t > 0\,,$

$u(x, 0) = 10x\,, \quad 0 < x < 2\,.$

The spatial domain $[0, 2]$ is divided into twenty small intervals with a non-uniform grid point distribution. The coordinates x_i, $i = 0, \ldots, 20$, of the grid points are

$$\{x_0, \ldots, x_{20}\} = \{0.00, 0.05, 0.10, 0.15, 0.20, 0.30, 0.40, 0.50, 0.60, 0.80, 1.00, 1.20, 1.40, 1.50, 1.60, 1.70, 1.80, 1.85, 1.90, 1.95, 2.00\}\,.$$

The transient solution of Eq. (47) are plotted in Fig. 4, where the results of integrating the semi-discretized equations in (48) by the whole domain precise time integration are represented by the solid curves and those obtained using the single-point subdomain leap-frog precise time integration with $\Delta t = 0.001$ are marked by $\times$. It is clear that the single-point integration gives a very accurate result as compared with the whole domain precise time integration.

The last example is for the non-linear heat equation (56). It involves a special kind of plastic material with a non-linear heat capacitance coefficient

$$\begin{aligned} D^*(T) &= d_0 + d_1(T - T_0) \\ C^*(T) &= c_0 \end{aligned} \tag{77}$$

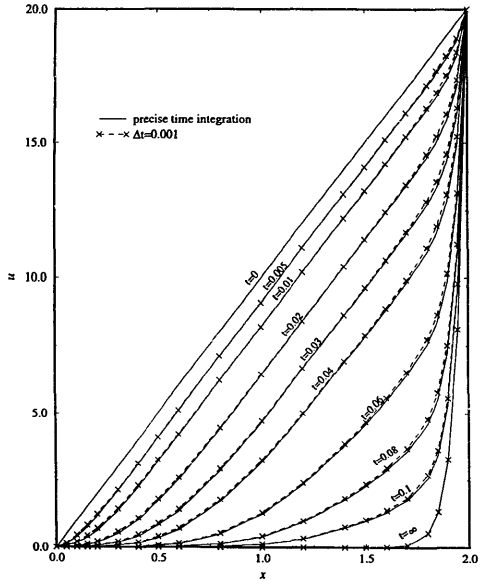

Fig. 4. Transient solutions of Eq. (47).

where

$$T_0 = 290°\mathrm{K}, \quad d_0 = 0.72,$$
$$d_1 = 0.03 \quad \text{and} \quad c_0 = 1.6 \times 10^6.$$

To utilize Eq. (58), we need to express D^* and C^* as functions of ϕ. Using relation (57), we have

$$\phi(T) = d_0(T - T_0) + d_1 \frac{(T - T_0)^2}{2}, \tag{78}$$

which gives

$$T - T_0 = \frac{\sqrt{d_0^2 + 2d_1\phi} - d_0}{d_1}. \tag{79}$$

Thus, we have

$$D(\phi) = \sqrt{d_0^2 + 2d_1\phi}. \tag{80}$$

The coefficients in (62) for E_i then become

$$k_0^n = \frac{\sqrt{d_0^2 + 2d_1\phi_i^n}}{c_0 \cdot (\Delta x)^2}, \qquad k_1^n = \frac{d_1}{c_0 \cdot (\Delta x)^2 \cdot \sqrt{d_0^2 + 2d_1\phi_i^n}}. \tag{81}$$

The spatial domain is $0 \leq x \leq 2.0$, $\Delta x = 0.1$ and $\tau = t/c_0$. We use the leaf-frog type algorithm to study two cases with different initial and boundary conditions:

Case 1:

B.C. $T(0, t) = 0$,

$T(2, t) = 20°C$,

I.C. $T(x, 0) = \begin{cases} 20x & 0 < x \leq 1, \\ 20 & 1 \leq x < 2. \end{cases}$

Case 2:

B.C. $T(0, t) = 60°C$,

$T(2, t) = 20°C$,

I.C. $T(x, 0) = \begin{cases} 60 & 0 < x \leq 1, \\ 100 - 40x & 1 \leq x < 2. \end{cases}$

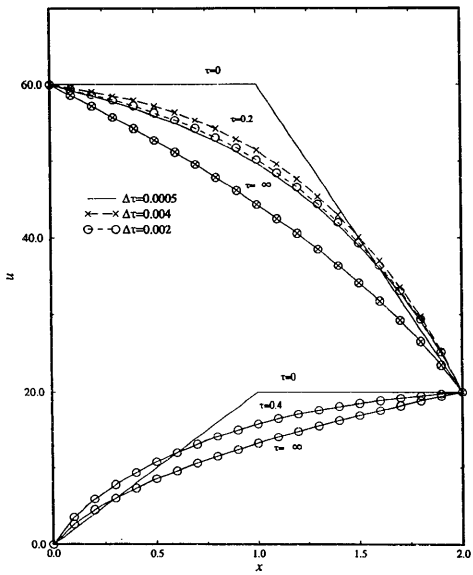

Fig. 5. Transient solutions of Case 1 and Case 2 for Eq. (56).

The computational results are given in Fig. 5, where the solid line represents the results obtained using $\Delta\tau = 0.0005$, which can be considered as the accurate solution of Eq. (59) up to the machine precision, since there is no more improvement as $\Delta\tau$ is further refined. The results marked by $\odot$ and $*$ represent the results obtained using $\Delta\tau = 0.002$ and $\Delta\tau = 0.004$, respectively. We can see that the results converge to the accurate solution uniformly without oscillation as $\Delta\tau$ is refined. The numerical results shown in Fig. 5 are much better than the traditional finite difference FTCS algorithm. We omit the plotting here to save the space.

9. Discussion and concluding remarks

A class of subdomain precise time integration algorithms are discussed in this paper. The algorithms are explicit in nature and unconditionally stable for the single-point subdomain integration. The accuracy can be improved by using more interior points in each subdomain, or using multi-step integration in the temporal dimension, such as the leap-frog algorithm. Note that the leaf-frog algorithm is unconditionally unstable in the traditional finite difference method for the heat equation, but becomes unconditionally stable with the subdomain precise time integration method.

For simplicity, most of the discussions in this paper are limited to one-dimensional cases. However, the method can be extended to the higher-dimensional cases to generate explicit and stable algorithms. It is for the higher-dimensional cases that this method has the potential to significantly improve the efficiency of numerical solutions of PDEs, since in higher-dimensional problems the traditional explicit finite difference schemes suffer much severer stability constraints while the implicit finite difference schemes become very costly for large scale problems.

Extensions of the subdomain precise time integration method to higher-dimensional and non-linear problems and analysis of stability of multi-point subdomain time integration algorithms are under way. More detailed results will be reported in the near future.

References

[1] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery, Numerical Recipes, 2nd edition (Cambridge University Press, New York, 1992).
[2] K.J. Bathe and E.L. Wilson, Numerical Methods in Finite Element Analysis (Prentice-Hall, Englewood Cliffs, New Jersey, 1976).
[3] O.C. Zienkiewicz and R.L. Taylor, The Finite Element Method, 4th edition (McGraw-Hill, New York, 1991).
[4] W.X. Zhong, J.P. Zhu and X.X. Zhong, A precise time integration algorithm for non-linear systems, in: Proc. 3rd World Congress on Computational Mechanics (Chiba, Japan, 1994) 12–17.
[5] W.X. Zhong and F.W. Williams, A precise time step integration method, Proc. Inst. Mech. Engrs. 208 (1994) 427–436.
[6] W.X. Zhong and Z.S. Yang, An algorithm for computing the main eigenpairs in time continuous LQ control, Appl. Math. Mech. 12 (1991) 45–50.