A Temporally-Piecewise Adaptive Algorithm to Solve Transient Convection-Diffusion Heat Transfer Problems

Xiao Zhao¹, Haitian Yang¹,² and Qiang Gao¹

Abstract: A piecewised adaptive algorithm in the time domain is presented to solve the transient convection-diffusion heat transfer problem. By expanding all variables at a time interval, an initial and boundary value problem is decoupled into a series of recursive boundary value problems which can be solved by FEM or other well developed numerical schemes to deal with boundary value problems. A steady computing accuracy can be adaptively maintained via the power increase of the expansion, particularly when the step size varies in the whole computing process. Additionally for the nonlinear cases, there is no requirement of iteration and additional assumption for the proposed algorithm. Three numerical examples are provided to verify the presented algorithm, and satisfactory results have been achieved.

Keywords: convection-diffusion; heat transfer; adaptive algorithm; time domain; FEM.

1 Introduction

There have been a number of well developed numerical methods to solve various steady/transient convection-diffusion problems. These methods are generally classified into two pools. One consists of those to deal with the boundary value problem, such as the characteristic finite volume method [Rui, H. X. (2008)], and time-space finite element method, etc. In this pool, the Galerkin FE method and its alternatives constitute an important family that includes SUPGM (Streamline Upwind Petrov-Galerkin method) [Brooks, A. N.; Hughes, T. J. R. (1982), Franca, L. P.; Valentin, F. (2000)], GLSM (Galerkin/Least-Squares Method) [Hughes T. J. R.; Franca, L. P.; Hulbert, G. M. (1989)], USFEM (Unusual Stabilized Finite Element Method) [Franca, L. P.; Valentin, F. (2000)], the upstream-weighted higher

¹ State Key Lab of Structural analysis for industrial Equipment, Dept. of Engineering Mechanics, Dalian University of Technology, Dalian, 116024, Dalian, P.R. China
² Corresponding author. E-mail address: Haitian@dlut.edu.cn, Tel.: +86-41184708394; fax: +86-41184708393


Enlightened by the \( p \)-refinement skill of FEM [Zienkiewicz, O. C.; Taylor, R. L. (2000)], a temporally piecewised adaptive algorithm is developed to solve transient convection-diffusion heat transfer problems in this paper, by which an initial and boundary value problem is decoupled into a series of recurrent boundary value problems which can be solved by FEM or other well-developed numerical techniques to deal with boundary value problems, and a steady computing accuracy
can adaptively be maintained via the power increase of the expansion. Additionally for the nonlinear cases there is no requirement of additional assumption and iteration for the proposed algorithm.

3 numerical examples are presented to verify the proposed approach, and good accordance can be observed in comparison with the results given by the analytical, Runge-Kutta and Crank-Nicolson methods.

2 Recursive governing equations of transient convection-diffusion heat transfer problems

The governing equation of transient convection-diffusion heat transfer problems can be written as [Platten, J. K.; Legros, J. C. (1984)]

\[ c \left[ T', + u_i T_i \right] = \left[ k_i T_i \right]', + Q \quad x_i \subset \Omega \quad (1) \]

where \( T \) denotes temperature, \( t \) refers to time, \( Q \) is a source or reaction term, \( c \) and \( k_i \) are heat capacity and thermal conductivity, respectively, \( u_i \) represents the vector of fluid velocity, \( x_i \) is the vector of the coordinates, and \( \Omega \) represents the space domain of the problem, and subscript \( i \) refers to a summation index (\( i=1,2 \) for the 2D problem, \( i=1,2,3 \) for the 3D problem).

The initial condition is describe by

\[ T|_{t=0} = T_0 \quad (2) \]

where \( T_0 \) is a prescribed function.

The boundary condition is given by

\[ T = T_B \quad x_i \in \Gamma_1 \quad (3) \]

\[ n_i k_i T_i = q \quad x_i \in \Gamma_2 \quad (4) \]

where \( T_B \) and \( q \) are prescribed functions, \( \Gamma = \Gamma_1 + \Gamma_2 \) represents the whole boundary of \( \Omega \), and \( n_i \) refers to the outward unit normal along \( \Gamma_2 \).

We divide time domain into a number of intervals, initial points and sizes of intervals are defined by \( t_0, t_1, t_2, \ldots, t_{\lambda}, \ldots \) and \( t_{s_1}, t_{s_2}, \ldots, t_{s_2}, \ldots \), respectively.

At \( \lambda-th (\lambda \geq 1) \) interval, all variables are expanded in the form

\[ T = \sum_{m=0}^{\lambda} T^m s^m \quad (5) \]

\[ c = \sum_{m=0}^{\lambda} c^m s^m \quad (6) \]
\[ k_i = \sum_{m=0}^{\infty} k_i^m s^m \]  
\[ Q = \sum_{m=0}^{\infty} Q_m s^m \]  
\[ u_i = \sum_{m=0}^{\infty} u_i^m s^m \]  
\[ T_B = \sum_{m=0}^{\infty} T_B^m s^m \]  
\[ x_i \in \Gamma_1 \]  
\[ q = \sum_{m=0}^{\infty} q_m s^m \]  
\[ x_i \in \Gamma_2 \]  
\[ s = \frac{t - t_{\lambda-1}}{t_{s_k}} \]  

where \( T^m \) denotes the expansion coefficient of \( T, c^m, k_i^m, Q^m, u_i^m, q^m \) and \( T_B^m \) represent expansion coefficients of \( c, k_i, Q, u_i, q \) and \( T_B \), respectively.

Utilizing

\[ \frac{\partial}{\partial t} = \frac{\partial}{\partial s} \frac{\partial s}{\partial t} = \frac{1}{t_{s_k}} \frac{\partial}{\partial s} \]  

we have

\[ T_n = \sum_{m=0}^{n+1} \frac{(m+1)}{t_{s_k}} T^{m+1} s^m \]  

Substituting Eqns. (5)–(11) and (14) for Eqns. (1), (3) and (4) and equating the powers of \( s \) at two sides of the equations then yields

\[ \frac{(n+1)}{t_{s_k}} c^{0} T^{n+1} + \sum_{m=0}^{n} c^{n-m+1} T^{m} + \sum_{m=0}^{n} \sum_{p=0}^{n-m} c^{n-m-p} u_i^p (T^m)_{,i} \]

\[ = \sum_{m=0}^{n} \left[ k_i^{n-m} (T^m)_{,i} \right]_{,i} + Q^n \]  

\[ T^n = T^n_B \text{ on } \Gamma_1 \]  

\[ \sum_{m=0}^{n} n_i k_i^{n-m} T_{,i}^m = q^n \text{ on } \Gamma_2 \]
3 Implementation of FEM

In the above section, an initial and boundary value problem is decoupled into a series of recursive boundary value problems which can be solved using FEM or other well-developed numerical techniques to deal with boundary value problems. A utilization of a weighted residual method on Eqns. (15) \sim (17) leads to [Zienkiewicz, O. C.; Taylor, R. L. (2000)]

\[
\int_{\Omega} W \left( \frac{n+1}{t_{\lambda}} c_0 T^{n+1} \right) d\Omega + \sum_{m=0}^{n} \int_{\Omega} W \left( m \frac{n-m+1}{t_{\lambda}} T^{m} \right) d\Omega \\
+ \sum_{m=0}^{n} \sum_{p=0}^{n-m} \int_{\Omega} W c^{n-m-p} u_i^p (T^m)_{i} d\Omega \\
- \sum_{m=0}^{n} \int_{\Omega} W \left[ k_i^{n-m} (T^m)_{i} \right]_{j} d\Omega = \int_{\Omega} W Q^n d\Omega - \int_{\Gamma_2} W \left( \sum_{m=0}^{n} n_i k_i^{n-m} T^{m} - q^n \right) d\Gamma_2
\]

(20)

where \( W \) denotes a weighting function. The application of integration by parts gives

\[
\int_{\Omega} W \left( \frac{n+1}{t_{\lambda}} c_0 T^{n+1} \right) d\Omega + \sum_{m=0}^{n} \int_{\Omega} W \left( m \frac{n-m+1}{t_{\lambda}} T^{m} \right) d\Omega \\
+ \sum_{m=0}^{n} \sum_{p=0}^{n-m} \int_{\Omega} W c^{n-m-p} u_i^p (T^m)_{i} d\Omega \\
- \sum_{m=0}^{n} \int_{\Omega} W_i \left[ k_i^{n-m} (T^m)_{i} \right]_{j} d\Omega = \int_{\Omega} W Q^n d\Omega + \int_{\Gamma_2} W q^n d\Gamma_2
\]

(21)

The choice of the weighting function \( W \) leads to various approximation methods, in the Galerkin FE method, \( T_j \) and \( W \) are approximated by

\[
T_j = [N] \{T\}_e^j \quad (22)
\]

\[
W = [N] \{W\}_e \quad (23)
\]

where \([N]\) represents a matrix of shape functions, \( \{T\}_e^j \) and \( \{W\}_e \) represent the local nodal vectors of \( T_j \) and \( W \), respectively. Substituting Eqns. (20) \sim (21) into Eqn. (19) and considering the arbitrariness of
\{W\}_e \text{ then yields }

\[(n+1) C^0 \{T\}^{n+1} + \sum_{m=0}^{n} \frac{m}{t_{s}} C^{n-m+1} \{T\}^{m} + \sum_{m=0}^{n} \sum_{p=0}^{n-m} M^{n-m} \{T\}^{m} + \sum_{m=0}^{n} K^{n-m} \{T\}^{m} = \{Q\}^{n} + \{q\}^{n} \quad (24)\]

where \(\{T\}^{m}\) represents the \(m\)-th coefficient vector of \(\{T\} = \sum_{m=0}^{n} \{T\}^{m} s^{m}\).  \(\{T\}\) is the global nodal vector of \(T\),

\[C^0 = \sum_{e} \int_{\Omega_e} [N]^T c^0 [N] d\Omega_e \quad (25)\]

\[C^{n-m+1} = \sum_{e} \int_{\Omega_e} [N]^T c^{n-m+1} [N] d\Omega_e \quad (26)\]

\[M^{n-m} = \sum_{e} \int_{\Omega_e} c^{n-m-p} [N]^T u^p_i [N]_i d\Omega_e \quad (27)\]

\[K^{n-m} = \sum_{e} \int_{\Omega_e} [N]_i^T k^{n-m}_i [N]_i d\Omega_e \quad (28)\]

\[\{Q\}^{n} = \sum_{e} \int_{\Omega_e} [N]^T Q^n d\Omega_e \quad (29)\]

\[\{q\}^{n} = \sum_{e} \int_{\Gamma_{2e}} [N]^T q^n d\Gamma_{2e} \quad (30)\]

where \([N]_i\) denotes a matrix of derivatives of the shape functions.

We also derived a recursive equation similar to Eqn. (22) by combining Eqns. (15) \sim (17) with the multilevel discontinuous Galerkin methods [Gopalakrishnan, J.; Kanschat, G. (2003)]. Owing to the limited capacity for an article, this recursive equation and its derivation are not presented here.

### 4 Adaptive process

At the first time interval, \(\{T\}^0\) is prescribed by the initial condition, at the \(\lambda\)-th time interval (\(\lambda > 1\)), \(\{T\}^0\) is provided by

\[
\{T\}^0 = \{T\}^m |_{m=0} \text{ (at } \lambda \text{-th time interval })
\]

\[
= \sum_{m=0}^{n} \{T\}^m s^m |_{s=1} = \sum_{m=0}^{n} \{T\}^m \text{ (at } (\lambda - 1)\text{-th time interval })
\quad (31)
\]
when \( \{T\}^0 \) is given, \( \{T\}^m \ (m \geq 1) \) can be calculated using Eqn. (22), and \( \{T\} \) can be attained by

\[
\{T\} = \sum_{m=0}^{\infty} \{T\}^m s^m
\]

(32)

An adaptive computing process at a time interval can be realized with the increase of \( m \) until the following criteria is tenable.

\[
\frac{\| \{T\}^m s^m \|_2}{\| \sum_{j=0}^{m-1} \{T\}^j s^j \|_2} \leq \beta
\]

(33)

where \( \beta \) is a prescribed error tolerance, and \( \| \cdot \|_2 \) represents a L\(^2\)-norm.

If Eqn. (31) holds on continuously for three times at \( m = M_{\text{max}} \), the computation at the current time interval will stop and step into the next one. At different time intervals, \( M_{\text{max}} \) usually is different. Regarding to Eqn. (30), the local truncation order of the proposed algorithm is \( M_{\text{max}} + 1 \) at a time interval.

The above process is similar to the \( p\)-refinement process in which the adaptive computing is fulfilled by increasing terms of polynomials [Harper, C. (1976)].

5 Numerical verification

This section provides 3 numerical examples with time dependent boundary conditions. The thermal conductivity in the Example 1 and the source term in the Example 3 are temperature dependent. The eight node quadrilateral element is employed in FE analysis.

For the simplicity, all the computing parameters are assumed dimensionless.

A L\(^2\)-norm error [Sudirham, J. J.; van der Vegt, J. J. W.; van Damme, R. M. J. (2006)] and a summation of relative errors are employed to evaluate the computing accuracy, i.e.

\[
\text{err}(t) = \| \{T_n(t)\} - \{T_a(t)\} \|_2
\]

(34)

\[
\text{REerr}(t) = \sqrt{\sum_{r=1}^{n} \left( \frac{T_n(r,t) - T_a(r,t)}{T_a(r,t)} \right)^2}
\]

(35)

where \( \{T_n(t)\} \) and \( \{T_a(t)\} \) stand for the vectors of numerical and analytical solutions at all FE nodes, and \( r \) refers to \( r\)-th component of \( \{T_n(t)\} \) and \( \{T_a(t)\} \).

All the computing tasks are completed on a PC (Pentium D, CPU: 2.80 GHz, Memory: 1 Gbyte), and all the computing programs are coded in Matlab.
Example 1 considers a nonlinear problem in a $1 \times 0.1$ rectangular domain where $k_1$ is temperature dependent [Shih, T. M. (1983)]. The Galerkin FEM is employed to deal with boundary value problems, and a $10 \times 1$ uniform FE grid is used, as schematically illustrated in Fig. 1.
The boundary condition is specified by

\[ T_B(0, x_2, t) = \frac{2^2}{24 - 6t}, \quad T_B(1, x_2, t) = \frac{3^2}{24 - 6t}. \]

The initial condition is

\[ T_0 = \frac{(x_1 + 2)^2}{24}. \]

Other computing parameters are given by

\[ Q = 0, \quad c = 1.0, \quad u_1 = 0.0, \quad u_2 = 0.0 \]

\[ k_1 = T, \quad k_2 = 0, \quad t \in [0, 3.8], \quad \beta = 1e - 8. \]

The exact solution of this problem is given by [Shih, T. M. (1983)]

\[ T = \frac{(x_1 + 2)^2}{24 - 6t}. \]
Regarding to Eqn. (10), \( T_B \) is expended in the form

\[
T_B(0,x_2,t) = \frac{2^2}{24 - 6t} = \frac{2}{3} \cdot \frac{1}{4 - (t_{\lambda-1} + s \cdot t_{x_2})} \\
= \frac{2}{3 \cdot (4 - t_{\lambda-1})} \cdot \frac{1}{1 - \frac{t_{x_2} \cdot s}{4 - t_{\lambda-1}}} = \frac{2}{3 \cdot (4 - t_{\lambda-1})} \cdot \left[ \sum_{n=0}^{\lambda} \left( \frac{t_{x_2} \cdot s}{4 - t_{\lambda-1}} \right)^n \right] (\lambda \geq 1)
\]

and

\[
T_B(1,x_2,t) = \frac{(2+1)^2}{24 - 6t} = \frac{3}{2} \cdot \frac{1}{4 - (t_{\lambda-1} + s \cdot t_{x_2})} \\
= \frac{3}{2 \cdot (4 - t_{\lambda-1})} \cdot \frac{1}{1 - \frac{t_{x_2} \cdot s}{4 - t_{\lambda-1}}} = \frac{3}{2 \cdot (4 - t_{\lambda-1})} \cdot \left[ \sum_{n=0}^{\lambda} \left( \frac{t_{x_2} \cdot s}{4 - t_{\lambda-1}} \right)^n \right] (\lambda \geq 1)
\]

The solution obtained by the proposed algorithm is exhibited in Tab. 1 and Figs. 4-8, and compared with the solutions given by Heun’s method [Burden, R. L.; Faires, J. D. (1988)], the Modified Euler method [Burden, R. L.; Faires, J. D. (1988)] and the 4-order Runge-Kutta method [Burden, R. L.; Faires, J. D. (1988)]. The comparisons indicate that the presented algorithm is a bit computationally expensive than three others, but it is able to keep a steady computing accuracy with the increase of step size, as shown in Fig. 7. When \( t_{x_2} = 2 \times 10^{-3} (\lambda = 1, 2, 3, ..., 500) \), all the other three methods are unable to obtain reasonable solutions, as shown in Fig. 8. Figs. 9 and 10 mean that a relatively smaller \( \beta \) is required to maintain a steady computing accuracy when step size becomes relatively larger.

Example 2 considers a convection-diffusion problem in a domain \( \Omega = (0, 1)^2 \) [Sudirham J. J.; van der Vegt, J. J. W.; van Damme, R. M. J. (2006)]. The Galerkin FEM is employed to deal with boundary value problems, and a 20\times20 uniform FE grid is used, as schematically illustrated in Fig. 2.

The initial condition is

\[
T_0 = \sin(\pi x_1) \cdot \sin(\pi x_2),
\]

The boundary condition is specified by

\[
T_B(0,x_2,t) = \sin(-\pi u_{1t}) \cdot \sin(\pi (x_2 - u_{2t})) \cdot \exp(-2D\pi^2 t),
\]

\[
T_B(1,x_2,t) = \sin(\pi (1 - u_{1t})) \cdot \sin(\pi (x_2 - u_{2t})) \cdot \exp(-2D\pi^2 t),
\]

\[
T_B(x_1,0,t) = \sin(\pi (x_1 - u_{1t})) \cdot \sin(-\pi u_{2t}) \cdot \exp(-2D\pi^2 t),
\]
The computing parameters are given by

\[ Q = 0, \quad c = 1.0, \quad u_1 = 1, \quad u_2 = 1, \quad k_1 = 1, \quad k_2 = 1, \quad \beta = 1 \times 10^{-8}. \]

The analytical solution is given by [Sudirham, J. J.; van der Vegt, J. J. W.; van Damme, R. M. J. (2006)].

\[ T_B(x_1,1,t) = \sin(\pi(x_1 - u_1 t)) \cdot \sin(\pi(1 - u_2 t)) \cdot \exp(-2D\pi^2 t), \]

Since \( T_B \) seems difficult to expand directly in the term of \( s \) it is approximated using a polynomial \( P_{ij}(x_1,x_2,t,\kappa) \) at each FE node \( (x_1,j,x_2,j) \) along the boundary.

\[ P_{ij}(x_1,x_2,t,\kappa) = a_0(x_1,x_2) + a_1(x_1,x_2) t + a_2(x_1,x_2) t^2 + \cdots + a_\kappa(x_1,x_2) t^\kappa \]
Figure 8: A companion of $T(x_1,x_2,t)|_{x_1=0.5,x_2=0}$ with $t_{s_\lambda} = 2 \times 10^{-3}$ ($\lambda = 1,2,3,...,500$)

Figure 9: The effects of $\beta$ on $err(t)$ with $t_{s_\lambda} = 2 \times 10^{-3}$ ($\lambda = 1,2,3,...,500$)

Figure 10: The effects of $\beta$ on $err(t)$ with $t_{s_\lambda} = 5 \times 10^{-4}$ ($\lambda = 1,2,3,...,2000$)

$a_0(x_{1,i},x_{2,j})$, $a_1(x_{1,i},x_{2,j})$, $a_2(x_{1,i},x_{2,j})$, ..., $a_\kappa(x_{1,i},x_{2,j})$

are determined by fitting $P_{ij}(x_{1,i},x_{2,j},t,\kappa)$ with $T_B(x_{1,i},x_{2,j},t)$ at $N_f$ points uniformly distributed within the whole time domain. $T_B^m$ can be obtained easily via $P_{ij}(x_{1,i},x_{2,j},t,\kappa)$. The fitting criteria is
Algorithm to Solve Transient Convection-Diffusion Heat Transfer Problems

Table 1: Numerical Comparison of Example 1 (at $t = 1$)

<table>
<thead>
<tr>
<th>Proposed Method</th>
<th>Modified Euler Method</th>
<th>Heun’s Method</th>
<th>4-order R-K Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{s_j}$</td>
<td>$5 \times 10^{-4}$</td>
<td>$5 \times 10^{-4}$</td>
<td>$5 \times 10^{-4}$</td>
</tr>
<tr>
<td>$err$</td>
<td>0.000082574</td>
<td>0.00019082</td>
<td>0.000190825</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>898.22</td>
<td>176.5</td>
<td>182.58</td>
</tr>
<tr>
<td>$t_{s_j}$</td>
<td>$1 \times 10^{-3}$</td>
<td>$1 \times 10^{-3}$</td>
<td>$1 \times 10^{-3}$</td>
</tr>
<tr>
<td>$err$</td>
<td>0.000082574</td>
<td>0.69251</td>
<td>2.052</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>457.36</td>
<td>88.61</td>
<td>88.375</td>
</tr>
<tr>
<td>$t_{s_j}$</td>
<td>$2 \times 10^{-3}$</td>
<td>$2 \times 10^{-3}$</td>
<td>$2 \times 10^{-3}$</td>
</tr>
<tr>
<td>$err$</td>
<td>0.00008261978</td>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>644.31</td>
<td>\</td>
<td>\</td>
</tr>
</tbody>
</table>

$\left( \sum_{l=1}^{N_f} \left( T_B(x_{1i}, x_{2j}, t_l) - P_{ij}(x_{1i}, x_{2j}, t_l, \kappa) \right)^2 \right)^{\frac{1}{2}} < \eta, (\eta \leq \beta)$

In this example $N_f = 2000, \kappa = 5, \eta = 1 \times 10^{-8}$.

A numerical comparison is exhibited in Tab. 2 and Figs. 11-13 where both the Modified Euler method and the 4-order Runge-Kutta method failed when $t_{s_{\lambda}} = 10^{-4} (\lambda = 1, 2, 3, ..., 2000)$ and $t_{s_{\lambda}} = 10^{-3} (\lambda = 1, 2, 3, ..., 200)$. Example 3 considers an autonomous advective Fisher Equation in which $Q = \gamma \cdot T^2(x_{1i}, x_{2j}, t_l) \cdot (1 - T(x_{1i}, x_{2j}, t_l))$ is temperature dependent in the domain $\Omega = (0, 1)^2$ [Caliari, M.; Vianello, M.; Bergamaschi, L. (2007)]. The multilevel discontinuous Galerkin FEM is employed to deal with boundary value problems, and a $25 \times 25$ uniform FE grid is used, as schematically illustrated in Fig. 3.

The other computing parameters are

$\gamma = 0.01, \quad u_1 = u_2 = \alpha = -1,$
\[ k_1 = k_2 = \varepsilon = 0.001, \quad \beta = 1 \times 10^{-8}, \quad c = 1.0. \]

The initial and boundary conditions are prescribed by

\[ T_0 = \frac{1}{1 + \exp(a \cdot (x_1 + x_2) + p)}, \]

\[ T_B(0,x_2,t) = \frac{1}{1 + \exp(a \cdot (x_2 - b \cdot t) + p)}, \]

\[ T_B(1,x_2,t) = \frac{1}{1 + \exp(a \cdot (1 + x_2 - b \cdot t) + p)}. \]
Algorithm to Solve Transient Convection-Diffusion Heat Transfer Problems

Figure 15: A companion of $err(t)$ with $t_{s\lambda} = 1 \times 10^{-2}$ ($\lambda = 1, 2, 3, ..., 50$)

Figure 16: A companion of $err(t)$ with $t_{s\lambda} = 2 \times 10^{-2}$ ($\lambda = 1, 2, 3, ..., 25$)

Figure 17: A companion of $REerr(t)$ of the proposed method with different $t_{s\lambda}$

Figure 18: The companions of $Mmax$ and $err(t)$ with constant and variable $t_{s\lambda}$

$$T_B(x_1, 0, t) = \frac{1}{1 + \exp \left( a \cdot (x_1 - b \cdot t) + p \right)};$$

$$T_B(x_1, 1, t) = \frac{1}{1 + \exp \left( a \cdot (x_1 + 1 - b \cdot t) + p \right)};$$

The exact solution is given by [Caliari, M.; Vianello, M.; Bergamaschi, L. (2007)]

$$T(x_1, x_2, t) = \frac{1}{1 + \exp \left( a \cdot (x_1 + x_2 - b \cdot t) + p \right)},$$

where $a = \sqrt{\frac{\gamma}{4\xi}}, b = 2\alpha + \sqrt{\gamma\epsilon}$ and $p = a(b - 1)$. 
Table 2: Numerical Comparison of Example 2 (at $t = 0.2$)

<table>
<thead>
<tr>
<th>$t_{sj}$</th>
<th>Proposed Method</th>
<th>Modified Euler Method</th>
<th>4-order R-K Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{sj}$</td>
<td>$5 \times 10^{-5}$ ($\lambda = 1, 2, 3, ..., 4000$)</td>
<td>$0.000037905$</td>
<td>$0.0000778$</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>119226</td>
<td>56887</td>
<td>58682</td>
</tr>
<tr>
<td>$t_{sj}$</td>
<td>$1 \times 10^{-4}$ ($\lambda = 1, 2, 3, ..., 2000$)</td>
<td>$0.000037905$</td>
<td>\</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>96301</td>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>$t_{sj}$</td>
<td>$1 \times 10^{-3}$ ($\lambda = 1, 2, 3, ..., 200$)</td>
<td>$0.000037905$</td>
<td>\</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>99505</td>
<td>\</td>
<td>\</td>
</tr>
</tbody>
</table>

$T^m_B$ is determined in the same way as Example 2 with $\eta = 1 \times 10^{-8}$, $\kappa = 4$, $N_f=5000$. A numerical comparison with the Crank-Nicolson method [Caliari, M.; Vianello, M.; Bergamaschi, L. (2007)] is given in Tab. 3 and Figs. 14-16. REerr $(t)$ curves in Fig. 17 exhibit a steady computing accuracy maintained by the proposed method with different size of $t_{sj}$. Numerical results with constant ($t_{sj} = 1 \times 10^{-3}, (\lambda = 1, 2, 3, ..., 500)$) and variable step sizes (as shown in Tab. 4) in the whole computing process are compared in Fig. 18. As we expected, the adaptive process effectively guarantees a steady computing accuracy when the step size varies.

However the Crank-Nicolson method that is a sound baseline benchmark for an advection-diffusion solver [Caliari, M.; Vianello, M.; Bergamaschi, L. (2007)] failed when $t_{sj} = 0.02$ ($\lambda = 1, 2, 3, ..., 25$) and $t_{sj} = 0.05$ ($\lambda = 1, 2, 3, ..., 10$), and when the step size varies.
Table 3: Numerical Comparison of Example 3 (at $t = 0.5$)

<table>
<thead>
<tr>
<th></th>
<th>Proposed Method</th>
<th>Crank-Nicolson Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{ni}$</td>
<td>$0.005 (\lambda = 1, 2, 3, \ldots, 100)$</td>
<td></td>
</tr>
<tr>
<td>$err(t)$</td>
<td>0.37021</td>
<td>0.37004</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>30517</td>
<td>17103</td>
</tr>
<tr>
<td>$t_{ni}$</td>
<td>$0.01 (\lambda = 1, 2, 3, \ldots, 50)$</td>
<td></td>
</tr>
<tr>
<td>$err(t)$</td>
<td>0.37021</td>
<td>0.3714</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>9375.3</td>
<td>7889.2</td>
</tr>
<tr>
<td>$t_{ni}$</td>
<td>$0.02 (\lambda = 1, 2, 3, \ldots, 25)$</td>
<td></td>
</tr>
<tr>
<td>$err(t)$</td>
<td>0.37021</td>
<td>_</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>4746.9</td>
<td>_</td>
</tr>
<tr>
<td>$t_{ni}$</td>
<td>$0.05 (\lambda = 1, 2, 3, \ldots, 10)$</td>
<td></td>
</tr>
<tr>
<td>$err(t)$</td>
<td>0.37021</td>
<td>_</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>11166</td>
<td>_</td>
</tr>
</tbody>
</table>

6 Conclusion

A temporally piecewised adaptive algorithm is presented to solve transient convection-diffusion heat transfer problems. The major merits of the proposed algorithm include

1. An initial and boundary value problem is decoupled into a series of boundary value problems which can be solved by FEM, finite volume method, meshless method, or other well-developed numerical techniques to deal with boundary value problems.

2. For the nonlinear case, there is no any requirement of additional assumption and iteration for the proposed algorithm.
Table 4: The variable process of $t_{s\lambda}$ in Example 3

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>Time $[t_{s\lambda-1},t_{s\lambda}]$</th>
<th>Step $t_{s\lambda}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$[0,0.01]$</td>
<td>$1 \times 10^{-3}$</td>
</tr>
<tr>
<td>2</td>
<td>$[0.01,0.015]$</td>
<td>$5 \times 10^{-3}$</td>
</tr>
<tr>
<td>3</td>
<td>$[0.015,0.023]$</td>
<td>$8 \times 10^{-3}$</td>
</tr>
<tr>
<td>4</td>
<td>$[0.023,0.033]$</td>
<td>$1 \times 10^{-2}$</td>
</tr>
<tr>
<td>5</td>
<td>$[0.033,0.053]$</td>
<td>$2 \times 10^{-2}$</td>
</tr>
<tr>
<td>6</td>
<td>$[0.033,0.093]$</td>
<td>$4 \times 10^{-2}$</td>
</tr>
<tr>
<td>7</td>
<td>$[0.543,0.548]$</td>
<td>$5 \times 10^{-1}$</td>
</tr>
<tr>
<td>8</td>
<td>$[0.548,0.556]$</td>
<td>$8 \times 10^{-3}$</td>
</tr>
<tr>
<td>9</td>
<td>$[0.556,0.566]$</td>
<td>$1 \times 10^{-2}$</td>
</tr>
<tr>
<td>10</td>
<td>$[0.566,0.586]$</td>
<td>$2 \times 10^{-2}$</td>
</tr>
</tbody>
</table>

3. The adaptive process can provide a steady computing accuracy when step size varies.

Acknowledgement: The research leading to this paper is funded by NSF [10421002, 10772035, 10721062, 11072043], and NKBRSF [2005CB321704, 2010CB832703].

References


