On Numerical Solution of the Parabolic Equation with Neumann Boundary Conditions

M. Akram

University College of Information Technology
University of the Punjab, Old Campus
Lahore-54000, Pakistan
m.akram@pucit.edu.pk

Abstract
In this paper an $O(h^2 + l^2)$ $L_0$-stable parallel algorithm is developed for the numerical solution of the parabolic partial differential equation $u_t + u_{xx} = s(x, t), \ 0 < x < X, \ 0 < t \leq T$, subject to $u(x, 0) = g(x), \ 0 \leq x \leq X, \ u_x(0, t) = f_1(t)$ and $u_x(X, t) = f_2(t)$.

Keywords: Parabolic equation; Rational approximation; Parallel algorithm.
Mathematics Subject Classification: 65M06, 65M20, 68W10.

1 Introduction

Parabolic Differential Equations (PDE’s) are commonly used in the fields of Engineering and Science for simulating physical processes. In a variety of cases, approximations are used to convert PDE’s to ordinary differential equations (ODE’s) or even to algebraic equations. However, because of the ever increasing requirement for more accurate modelling of physical process, engineers and scientists are more and more required to solve the actual PDE’s that govern the physical problem under investigated. The solutions of PDE’s describe possible physical reactions that have to be fixed through boundary conditions. These equations involve two or more independent variables that determine the behavior of the dependent variable as described by a differential equation, usually of second or higher order.

Consider the second-order parabolic partial differential equation

$$u_t + u_{xx} = s(x, t), \ 0 < x < X, \ t > 0,$$  \hspace{1cm} (1)

subject to Neumann time-dependent boundary conditions on boundary $\partial R$ of the open-region $R$ defined by lines $x = 0$ and $x = X$ given by

$$u_x(0, t) = f_1(t), \ 0 < t \leq T,$$  \hspace{1cm} (2)
\[ u_x(X, t) = f_2(t), \; \; 0 < t \leq T, \]  
(3)

with initial condition

\[ u(x, 0) = g(x), \; \; 0 \leq x \leq X, \]  
(4)

where \( f_1(t) \), \( f_2(t) \), \( g(x) \) and \( s(x, t) \) are known, while the function \( u(x, t) \) is to be determined. The existence and uniqueness properties of this problem are presented in literature. Several sequential numerical methods (implicit as well as explicit) have been proposed in the literature for the solution of this problem \([5, 6, 7]\). The aim of this paper is to present \( O(h^2 + l^2) \) \( L_0 \)-stable parallel algorithm for the numerical solution of parabolic equation subject to Neumann boundary conditions.

Parallel Processing refers to the concept of speeding-up the execution of a program by dividing the program into multiple fragments that can execute simultaneously, each on its own processor. A program being executed across \( n \) processors might execute \( n \) times faster than it would using a single processor.

The paper is organized in the following way: the numerical scheme is described in Section 2 and the parallel algorithm is presented in Section 3. The numerical results produced by this scheme are given in Section 4 and conclusion is given in Section 5.

2 Derivation of the Scheme

The interval \( 0 \leq x \leq X \) is divided into \( N + 1 \) subintervals each of width \( h \), so that \( (N + 1)h = X \) and the time variable \( t \) is discretized in the steps of length \( l \). Thus at each time level \( t = t_n = nl (n = 0, 1, 2, \cdots) \), the open region \( R = \{ 0 < x < X \} \times [t > 0] \) and its boundary \( \partial R \) consisting of lines \( x = 0 \) and \( x = X \) and the axis \( t = 0 \) have been superimposed by rectangular mesh with \( N \) points within \( R \) and open point along each side of \( \partial R \). The solution \( u(x, t) \) of (1) is sought at each point \((kh, nl) \) in \( R \times [t > 0] \), where \( k = 1, 2, \cdots, N \) and \( n = 0, 1, 2, \cdots \). The solution of an approximating numerical method will be denoted by \( U(x, t) \). The space derivatives in (1) and boundary conditions will be replaced by the following second-order central-difference approximations:

\[
\frac{\partial^2 u(x, t)}{\partial x^2} \approx \frac{1}{h^2} \{ u(x-h, t) - 2u(x, t) + u(x+h, t) \},
\]  
(5)

\[
\frac{\partial u(x, t)}{\partial x} \approx \frac{1}{2h} \{ u(x+h, t) - u(x-h, t) \}.
\]  
(6)
Thus the boundary conditions \( u(0,t) \) and \( u(X,t) \) can be determined using (2) and (3) with (6) respectively. Applying (1) to all the interior mesh points within \( R \) at time level \( t = nl \) with the space derivative replaced by (5) leads to a system of \( N \) first-order ODE’s of the form

\[
\frac{dU(t)}{dt} = AU(t) + v(t), \quad t > 0
\]

with the initial condition

\[
U(0) = g
\]

in which the matrix \( A \) is of order \( N \) and given by

\[
A = \frac{1}{h^2} \begin{bmatrix}
-2 & 2 & 0 & 0 & \cdots \\
1 & -2 & 1 & 0 & \cdots \\
0 & 1 & -2 & 1 & \cdots \\
& 0 & 1 & -2 & 1 \\
& & & 0 & -2
\end{bmatrix}
\]

In equation (7) the vector \( v(t) \) arises from the use of boundary conditions \( u(0,t) \) and \( u(X,t) \) in (5). Solving (7) subject to (8) gives

\[
U(t) = \exp(lA)g + \int_0^t \exp((t + l - s)A)v(s)ds;
\]

which satisfies [1] the recurrence relation

\[
U(t + l) = \exp(lA)U(t) + \int_t^{t+l} \exp((t + l - s)A)v(s)ds;
\]

in which \( l \) is a constant time step in the discretization of the time variable \( t \geq 0 \) at the points \( t_n = nl(n = 0, 1, 2, \cdots, N) \). To approximate the matrix exponential function in (11), we consider the rational approximate to \( \exp(lA) \) of the form

\[
\exp(lA) = \frac{I + (1 - a)lA}{I - alA + (a - \frac{1}{2})(lA)^2}.
\]

The denominator of \( \exp(lA) \) has distinct real zeroes by choosing \( a \approx 0.54 \) and \( L_0 \)-stability is introduced in [4]. The integral term appearing in (11) is approximated by a quadrature formula of the form

\[
\int_t^{t+l} \exp((t + l - s)A)v(s)ds \simeq \sum_j W_jv(s_j)
\]
where all $s_i (i = 1, 2)$ are different and weights $W_1 = W_1(lA)$ and $W_2 = W_2(lA)$ are matrices. It can easily be shown that when $v(s) = [1, 1, 1, \ldots, 1]^T$,

$$W_1 + W_2 = M_1$$  \hspace{1cm} (13)

where $M_1 = A^{-1}(\exp(lA) - I)$.

when $v(s) = [s, s, s, \ldots, s]^T$,

$$s_1W_1 + s_2W_2 = M_2$$  \hspace{1cm} (14)

where $M_2 = A^{-1}\{t\exp(lA) - A^{-1}A^{-1}(\exp(lA) - I)\}$, $I$ is the identity matrix of order $N$. Taking $s_1 = t$, $s_2 = t + l$ and then solving (13) and (14) simultaneously and replacing by $\exp(lA)$ gives

$$U(t + l) = \exp(lA)U(t) + \frac{l}{2}[S(lA)v(t) + T(lA)v(t + l)]; t = 0, l, 2l, \cdots \hspace{1cm} (15)$$

in which

$$S(lA) = [I - alA + (a - \frac{1}{2})(lA)^2]^{-1}, \hspace{1cm} (16)$$

$$T(lA) = [I - alA + (a - \frac{1}{2})(lA)^2]^{-1}[I - 2(a - \frac{1}{2})lA]. \hspace{1cm} (17)$$

3 The parallel algorithm

In order to implement (15) in parallel, the functions $\exp(lA)$, $S(lA)$ and $T(lA)$ are decomposed into their partial-fraction forms [4] given by

$$\exp(lA) = p_1(I - r_1lA)^{-1} + p_2(I - r_2lA)^{-1},$$

$$S(lA) = p_3(I - r_1lA)^{-1} + p_4(I - r_2lA)^{-1},$$

$$T(lA) = p_5(I - r_1lA)^{-1} + p_6(I - r_2lA)^{-1},$$

with

$$p_1 = \frac{1 - a + r_1}{r_1 - r_2}, \quad p_2 = \frac{1 - a + r_2}{r_2 - r_1},$$

$$p_3 = \frac{r_1}{r_1 - r_2}, \quad p_4 = \frac{r_2}{r_2 - r_1},$$

$$p_5 = \frac{r_1}{r_1 - r_2}, \quad p_6 = \frac{r_2}{r_2 - r_1}.$$
and

\[ p_5 = \frac{1 - 2a + r_1}{r_1 - r_2}, \quad p_6 = \frac{1 - 2a + r_2}{r_2 - r_1}, \]

where

\[ r_1 = \frac{2a - 1}{a + \sqrt{a^2 - 4a + 2}}, \quad r_2 = \frac{2a - 1}{a - \sqrt{(a^2 - 4a + 2)}}. \]

The solution vector \( U(t + l) \) in (15) may be obtained in parallel using two processors running concurrently as follows:

**Processor 1**

1. Input \( l, r_1, U(0), A \)
2. Compute \( p_1, p_3, p_5, (I - r_1lA) \)
3. Decompose \( (I - r_1lA) = L_1U_1 \)
4. Evaluate \( v(t), v(t + l) \)
5. Use \( z_1(t) = \frac{l}{2}(p_3v(t) + p_5v(t + l)) \)
6. Solve \( L_1U_1y_1(t) = p_1U(t) + z_1(t) \)

**Processor 2**

1. Input \( l, r_2, U(0), A \)
2. Compute \( p_2, p_4, p_6, (I - r_2lA) \)
3. Decompose \( I - r_2lA = L_2U_2 \)
4. Evaluate \( v(t), v(t + l) \)
5. Use \( z_2(t) = \frac{l}{2}(p_4v(t) + p_6v(t + l)) \)
6. Solve \( L_2U_2y_2(t) = p_2U(t) + z_2(t) \)

\( U(t + l) = y_1(t) + y_2(t) \).

GOTO step (4) for next time step.

Implementing the algorithm, Processor 1 generates decomposition of \( I - r_1lA \) and Processor 2 generates decomposition of \( I - r_2lA \) simultaneously. Parallel computing is the simultaneous execution of the same task (split up and specially adapted) on multiple processors in order to obtain faster results. The idea is based on the fact that the process of solving a problem usually can be divided into smaller tasks, which may be carried out simultaneously with some coordination.
4 Numerical Examples

In order to test the behavior of L0-stable scheme, three problems from the literature are considered. The parallel algorithm is tested on a sequential computer for the solutions of parabolic equations. Comparisons are made in terms of accuracy with Crank Nicholson formula developed in [7].

**Example 1.** Consider the parabolic partial differential equation with

\[
\begin{align*}
  s(x, t) &= -2e^{x-t} \\
  g(x) &= e^x \\
  f_1(t) &= e^{-t} \\
  f_2(t) &= e^{1-t}
\end{align*}
\]

<table>
<thead>
<tr>
<th></th>
<th>0.125</th>
<th>0.1</th>
<th>0.05</th>
<th>0.025</th>
</tr>
</thead>
<tbody>
<tr>
<td>(l=0.125)</td>
<td>0.23D-3</td>
<td>0.11D-3</td>
<td>0.86D-4</td>
<td>0.31D-4</td>
</tr>
<tr>
<td>(l=0.1)</td>
<td>0.13D-3</td>
<td>0.76D-4</td>
<td>0.33D-4</td>
<td>0.21D-4</td>
</tr>
<tr>
<td>(l=0.05)</td>
<td>0.71D-4</td>
<td>0.23D-4</td>
<td>0.82D-5</td>
<td>0.54D-5</td>
</tr>
<tr>
<td>(l=0.025)</td>
<td>0.64D-4</td>
<td>0.21D-4</td>
<td>0.77D-5</td>
<td>0.11D-5</td>
</tr>
</tbody>
</table>

The analytical solution is \(u(x,t) = e^{x-t}\). Using algorithm , this problem is solved for \(h = 0.125, 0.1, 0.05, 0.025\) with \(l = 0.125, 0.1, 0.05, 0.025\). The absolute relative errors \(|(u - U)/u|\) at time \(t = 1\) are shown in Table 1.

**Example 2.** Consider the parabolic partial differential equation with

\[
\begin{align*}
  s(x, t) &= xt^2 \\
  g(x) &= \sin(x) \\
  f_1(t) &= 1 \\
  f_2(t) &= \sin(t)
\end{align*}
\]

Using the algorithm , this problem is solved for \(h = 0.125, 0.1, 0.05, 0.025\)
Table 2: Maximum relative errors for Example 2 at $t = 1$.

<table>
<thead>
<tr>
<th>$h$ $\rightarrow$</th>
<th>0.125</th>
<th>0.1</th>
<th>0.05</th>
<th>0.025</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l=0.125$</td>
<td>0.13D-2</td>
<td>0.10D-2</td>
<td>0.81D-3</td>
<td>0.56D-3</td>
</tr>
<tr>
<td>$l=0.1$</td>
<td>0.12D-2</td>
<td>0.34D-3</td>
<td>0.21D-3</td>
<td>0.12D-3</td>
</tr>
<tr>
<td>$l=0.05$</td>
<td>0.74D-3</td>
<td>0.53D-3</td>
<td>0.17D-3</td>
<td>0.83D-4</td>
</tr>
<tr>
<td>$l=0.025$</td>
<td>0.32D-3</td>
<td>0.31D-3</td>
<td>0.56D-4</td>
<td>0.14D-4</td>
</tr>
</tbody>
</table>

with each $l = 0.125, 0.1, 0.05, 0.025$. The absolute relative errors $|(u - U)/u|$ at time $t = 1.0$ are shown in Table 2.

**Example 3.** Consider the parabolic partial differential equation with

\[
\begin{align*}
  s(x, t) &= 0 \\
  g(x) &= sin(\pi x) \\
  f_1(t) &= \pi e^{-\pi^2 t} \\
  f_2(t) &= -\pi e^{\pi^2 t}
\end{align*}
\]

The problem has analytical solution $u(x, t) = e^{-\pi^2 t} sin(\pi x)$. The results of $u(0.5, 1.0)$ with different number of time steps are shown in Table 3. The results obtained using the new scheme developed in this paper are slightly more accurate than those from the scheme of [7]. Note that the new scheme will require less CPU time. It is clear that as far as efficiency is concerned, the scheme introduced in this paper is a better candidate for the model problem.

## Conclusion

In this paper the parallel algorithm has been applied to the parabolic equation subject to Neumann boundary conditions. The algorithm may be implemented on a parallel architecture using two processors. The second-order spatial derivative was discretized to result in an approximating system of ODEs. The exact solution of this system of first order ODEs satisfies a recurrence relation involving the matrix exponential function. This function is approximated by a rational function possessing real and distinct poles which consequently readily admits a partial fraction expansion thereby allowing the distribution of the work in solving the corresponding linear algebraic systems on concurrent
Table 3: Maximum relative errors for Example 3 at $t = 1$.

<table>
<thead>
<tr>
<th>Time Steps</th>
<th>[7]</th>
<th>Parallel Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.001127</td>
<td>0.0005185</td>
</tr>
<tr>
<td>20</td>
<td>0.00828</td>
<td>0.0004231</td>
</tr>
<tr>
<td>30</td>
<td>0.002514</td>
<td>0.0003367</td>
</tr>
<tr>
<td>40</td>
<td>0.003616</td>
<td>0.0002864</td>
</tr>
<tr>
<td>50</td>
<td>0.004299</td>
<td>0.0005361</td>
</tr>
<tr>
<td>60</td>
<td>0.004718</td>
<td>0.0003572</td>
</tr>
<tr>
<td>70</td>
<td>0.004974</td>
<td>0.0003281</td>
</tr>
<tr>
<td>80</td>
<td>0.005130</td>
<td>0.0003242</td>
</tr>
<tr>
<td>90</td>
<td>0.005225</td>
<td>0.0002351</td>
</tr>
<tr>
<td>100</td>
<td>0.005284</td>
<td>0.0001213</td>
</tr>
<tr>
<td>110</td>
<td>0.005319</td>
<td>0.0001334</td>
</tr>
<tr>
<td>120</td>
<td>0.005341</td>
<td>0.0001432</td>
</tr>
<tr>
<td>130</td>
<td>0.005354</td>
<td>0.0001131</td>
</tr>
<tr>
<td>140</td>
<td>0.005362</td>
<td>0.0001331</td>
</tr>
<tr>
<td>150</td>
<td>0.005367</td>
<td>0.0001234</td>
</tr>
</tbody>
</table>

processors. The method needs only real arithmetic in its implementation. The use of only real arithmetic especially in the three-dimensional models can yield large saving in CPU time used.

References


Received: April 14, 2006