On Solving a General Bordered Tridiagonal Linear System

A. A. Karawia*
Computer science unit, Deanship of educational services, Qassim University,
P.O.Box 6595, Buraiddah 51452, Saudi Arabia.
Email: kraoieh@qu.edu.sa

Q. M. Rizvi
Computer science unit, Deanship of educational services, Qassim University,
P.O.Box 6595, Buraiddah 51452, Saudi Arabia.
Email: sirqaim@gmail.com

ABSTRACT

In this paper, the authors present reliable symbolic algorithms for solving a general bordered tridiagonal linear system. The first algorithm is based on the LU decomposition of the coefficient matrix, and its computational cost is $O(n)$. The second is based on the Sherman-Morrison-Woodbury formula. Algorithms are implementable to the Computer Algebra System (CAS) such as MAPLE, MATLAB and MATHEMATICA. Three examples are given to illustrate these algorithms.

Keywords - Bordered tridiagonal matrices; LU factorization; Sherman-Morrison-Woodbury Formula; Computer algebra systems (CAS).

1. INTRODUCTION

A general bordered tridiagonal linear system takes the form:

$$
\begin{bmatrix}
    a_1 & b_1 & 0 & 0 & \cdots & 0 & y_1 \\
    c_2 & a_2 & b_2 & 0 & \cdots & 0 & y_2 \\
    0 & c_3 & a_3 & b_3 & \cdots & 0 & y_3 \\
    \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
    0 & \cdots & 0 & c_{n-2} & a_{n-2} & b_{n-2} & y_{n-2} \\
    0 & \cdots & 0 & c_n & a_n & b_n & y_n \\
\end{bmatrix}
\begin{bmatrix}
    x_1 \\
    x_2 \\
    x_3 \\
    \vdots \\
    x_{n-2} \\
    x_{n-1} \\
\end{bmatrix}
= \begin{bmatrix}
    1 \\
    0 \\
    0 \\
    \vdots \\
    0 \\
\end{bmatrix}
$$

where $n > 3 \ (1.1)$

Many problems in mathematics and applied science require the solution of a general bordered tridiagonal linear system. For example, the solution of certain partial differential equations, spline approximation, computation of electric power system, etc. [1-4]. This area of research is still very active and has been considered by many authors. Recently in [5], the author presents an approach to find a solution to the linear system (1.1) in heterogeneous environments. In this approach, a bordered tridiagonal linear system is first converted into three or more tridiagonal linear systems that are independent from each other, then the solution of the bordered tridiagonal linear system is obtained by solving the tridiagonal linear systems via parallel computing under heterogeneous environments. The motivation of the current paper is to establish efficient algorithms for solving a general bordered tridiagonal linear system of the form (1.1).

2. A NEW SYMBOLIC ALGORITHM

In this section, we shall focus on the construction of a new symbolic algorithm for solving a general bordered tridiagonal linear system. To do this, we begin by considering the LU decomposition [6] of the coefficient matrix in (1.1):

$$
\begin{bmatrix}
    a_1 & b_1 & 0 & 0 & \cdots & 0 & y_1 \\
    c_2 & a_2 & b_2 & 0 & \cdots & 0 & y_2 \\
    0 & c_3 & a_3 & b_3 & \cdots & 0 & y_3 \\
    \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
    0 & \cdots & 0 & c_{n-2} & a_{n-2} & b_{n-2} & y_{n-2} \\
    0 & \cdots & 0 & c_n & a_n & b_n & y_n \\
\end{bmatrix}
= \begin{bmatrix}
    d_1 & h_1 & 0 & 0 & \cdots & 0 \\
    0 & d_2 & h_2 & 0 & \cdots & 0 \\
    0 & 0 & d_3 & h_3 & \cdots & 0 \\
    \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
    0 & \cdots & 0 & d_{n-2} & h_{n-2} & \beta_{n-2} \\
    0 & \cdots & 0 & d_{n-1} & h_{n-1} & \beta_{n-1} \\
\end{bmatrix}
$$

where $d_i, \alpha_i$ and $\beta_i$ in (2.1) satisfy:

$$
d_i = \begin{bmatrix}
    a_i & \frac{b_i}{d_{i-1}} & c_i \\
    b_i & \frac{d_{i-1}}{d_{i-1}} & e_i \\
    \vdots & \ddots & \ddots \\
    a_{i-n+2} & \frac{b_{i-n+2}}{d_{i-n+1}} & e_{i-n+2} \\
    a_{i-n+1} & \frac{b_{i-n+1}}{d_{i-n}} & e_{i-n+1} \\
    \alpha_i & \sum_{j=1}^{i} \alpha_j \beta_j & \beta_i \\
\end{bmatrix}
$$

This paper is organized as follows: In Section 2, a new symbolic algorithm is constructed. The Sherman-Morrison-Woodbury algorithm is presented in Section 3. Three illustrative examples are given in Section 4. Conclusions of the work are presented in Section 5.

*This article is authorized for use only by Recent Science Journal Authors, Subscribers and Partnering Institutions*

*Home Address: Mathematics Department, Faculty of Science, Mansoura University, Mansoura 35516, Egypt. E-mail: abibka@mans.edu.eg*
di = 0; i

and

We also have the determinant of the coefficient matrix in (1.1):

\[ \text{Determinant} = \prod_{i=1}^{n} d_i. \]

The solution of the system (1.1) is given by

\[ x_i = \begin{cases} 
\frac{\alpha_i}{d_i} & \text{if } i = 1, \\
\frac{\beta_i}{d_i} & \text{if } i = 2, 3, \ldots, n-2, \\
\frac{1}{d_i} (z_i - \beta_i x_i) & \text{if } i = n-1, \\
\frac{1}{d_i} (z_i - \beta_i x_i) & \text{if } i = n-2, n-3, \ldots, 2, \\
\frac{1}{d_i} (z_i - \beta_i x_i) & \text{if } i = 2, 3, \ldots, n-1, \\
\frac{1}{d_i} (z_i - \beta_i x_i) & \text{if } i = n. 
\end{cases} \]

where

\[ y_i = \begin{cases} 
y_i & \text{if } i = 1, \\
y_i & \text{if } i = 2, 3, \ldots, n-1, \\
y_i & \text{if } i = n, 
\end{cases} \]

Step 1: Set \( d_1 = a_1 \), \( \alpha = q_1 / d_1 \), \( \beta = p_1 / d_1 \). If \( d_1 = 0 \) then \( d_1 = t \) end if.

Step 2: For \( i = 2, 3, \ldots, n-1 \), Compute

\[ d_i = a_i - b_i / d_{i-1}, \]

If \( d_i = 0 \) then \( d_i = t \) end if.

Step 3: For \( i = 2, 3, \ldots, n-2 \), Compute

\[ a_i = (q_i - \alpha_i \cdot b_{i-1}) / d_i, \]

\[ \beta_i = p_i - \beta_i \cdot c_i / d_i. \]

Step 4: Set \( d_n = (c_n - \alpha_n \cdot b_{n-1}) / d_{n-1} \), \( a_n = c_n / d_n \), \( \beta_n = p_n / c_i / d_n \).

Step 5: Compute \( \text{Determinant} = \prod_{i=1}^{n} d_i \).

Step 6: Set \( z_n = y_n \).

Step 7: For \( i = 2, 3, \ldots, n-1 \), Compute

\[ z_i = y_i - \sum_{j=1}^{i-1} \alpha_j z_j / d_i. \]

Step 8: Compute

\[ x_i = z_i / d_i \] and \( x_n = (z_n - \beta_n x_n) / d_n. \)

Step 9: For \( i = n-2, n-3, \ldots, 1 \), Compute

\[ x_i = (z_i - b_i x_{i+1}) / d_i. \]

Step 10: Substitute \( t = 0 \) in all expressions of the solution vector \( x = [x_1, x_2, \ldots, x_n] \).

The symbolic Algorithm 2.1 will be referred to as SBTLS algorithm. The computational cost of SBTLS algorithm is \( 19n - 34 \) operations. In [7], the PERTLS algorithm is special case of our algorithm when \( p_i = q_i = 0, i = 2, 3, n - 2 \).

3. THE SHERMAN MORRISON WOOD-BURY ALGORITHM

In this section, a new symbolic algorithm is going to be formulated for solving a general bordered tridiagonal linear system of the form (1.1) based on the Sherman-Morrison-Woodbury formula and any symbolic tridiagonal linear solver.

A general bordered tridiagonal linear system of the form (1.1) can be written in the form:

\[ \begin{bmatrix} M_1 & V \\ U^T & M_2 \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} y'' \\ y'' \end{bmatrix} \] (3.1)

where

\[ M_1 = \begin{bmatrix} a_1 & b_1 & 0 & \cdots & 0 \\ c_1 & a_2 & b_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & a_{n-1} \end{bmatrix}, \]

\[ M_2 = \begin{bmatrix} a_n \\ \vdots \\ a_{n-1} \end{bmatrix}, \]

\[ M_2 = \begin{bmatrix} b_1 & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{n-1} & 0 \\ 0 & \cdots & 0 & 0 & a_n \end{bmatrix}, \]

\[ U = \begin{bmatrix} g_1 & q_1 & \cdots & q_{n-1} & c_n \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_1 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 \end{bmatrix}, \]

\[ x' = [x_1, x_2, \ldots, x_{n-1}, 1], \]

\[ y'' = [y_1, y_2, \ldots, y_{n-1}], \]

\[ y'' = [y_1]. \]

Thus (3.1) is equivalent to:

\[ M_1 x' + V x'' = y' \]

\[ U^T x' + M_2 x'' = y'' \] (3.2)
Assume that $a_0 \neq 0$. After the elimination of $x''$ from (3.2), we get the linear system:

$$Mx' = y'$$ (3.3)

where

$$M = M_1 - VM_2^1U_1^T,$$

and $y' = y'' - VM_2^1y'$.

If we apply the Sherman-Morrison-Woodbury formula [8] to $M$, we will obtain:

$$M^{-1} = M_1^{-1} + M_1^{-1}V (M_2 - U^TM_1^{-1}V)^{-1}U^TM_1^{-1},$$

and

$$x' = M^{-1}y' = r + M_1^{-1}V (M_2 - U^TM_1^{-1}V)^{-1}U^Tr.$$ where $r$ is the solution of $M_f = y'$.

It is clear that the solution $x''$ can be found from the above formula by successive calculation of the expressions

$$r = M_1^{-1}y', q = M_1^{-1}V , (M_2 - U^TM_1^{-1}V)^{-1},$$

and

$$(M_2 - U^TM_1^{-1}V)^{-1}U^Tr.$$ The main part of the above calculations is finding the first two expressions, which is equivalent to solving two $(n - 1)$-by-$(n - 1)$ tridiagonal linear systems with the same coefficient matrix $M_1$ and different right-hand sides. After finding the value of $x''$, we can get $x'$ from the second equation of (3.2) by the formula

$$x'' = M_1^{-1}(y'' - U^Tx')$$

At this point, it is convenient to formulate our second result. It is a symbolic algorithm for computing the solution of a general bordered tridiagonal linear system of the form (1.1) and can be considered as a natural generalization of the the Sherman-Morrison algorithm in [7].

3.1 Algorithm

To compute the solution of a general bordered tridiagonal linear system (1.1), we may proceed as follows:

**Step 1:** Find $M_1, M_2, U_1^T, V, x''; x'$ and $y' = y'' - VM_2^1y'$.

**Step 2:** Solve $M_f = y'$, and $M_g = V$.

**Step 3:** Compute

$$x = r + q(M_2 - U^Tq)^{-1}U^Tr,$$

and $x' = M_1^{-1}(y'' - U^Tx').$

**Step 4:** Compute the solution

$$x = \begin{bmatrix} x'' \\ x' \end{bmatrix} \Bigg|_{j=0}.$$

Two systems $M_f = y'$ and $M_g = V$ in algorithm 3.1 can be solved in parallel by any symbolic tridiagonal algorithm. The symbolic Algorithm 3.1 will be referred to as SMWBTLS algorithm.

4. ILLUSTRATIVE EXAMPLES

In this section, three examples are given to demonstrate the viability of the proposed algorithms.

4.1 Example

Let

$$\begin{bmatrix} 32 & 3 & 0 & 0 & 0 & 0 & 0 & 9 \\ 27 & 26 & 52 & 0 & 0 & 62 & x_2 & 24 \\ 0 & 55 & 63 & 59 & 0 & 35 & x_3 & 43 \\ 0 & 0 & 99 & 12 & 24 & 0 & 71 & x_4 = 97 \\ 0 & 0 & 0 & 74 & 61 & 51 & x_5 & 51 \\ 0 & 0 & 0 & 0 & 1 & 68 & 42 & x_6 & 52 \\ 29 & 65 & 9 & 45 & 72 & 59 & 33 & x_7 = 56 \end{bmatrix}.$$ 

i) By applying the BTLE algorithm in [5], it yields $x_{BTLE} = [3.8709, -2.2703, 3.1714, 1.8793, -1.0861, 2.6376, -3.0065]^T$.

ii) By applying the SBTLE algorithm, it yields $x_{SBTLE} = [3.8638, -2.2838, 3.1464, 1.9121, -1.0871, 2.6192, -2.9767]^T$.

iii) By applying MATLAB command $A\backslash b$, it yields $x_{MATLAB} = [3.8638, -2.2838, 3.1464, 1.9121, -1.0871, 2.6192, -2.9767]^T$.

iv) By applying SMWBTLS algorithm, it yields $x_{SMWB} = [3.8638, -2.2838, 3.1464, 1.9121, -1.0871, 2.6192, -2.9767]^T$.

4.2 Example

Let

$$\begin{bmatrix} 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \\ 13 & 2 & 12 & 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 9 & 1 & 5 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 3 & 15 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2 & 3 & 10 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 7 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -5 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 & 1 & 1 & 1 & 12 \\ 3 & 2 & 1 & 7 & 5 & -2 & 4 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} = \begin{bmatrix} 7 \\ 30 \\ 17 \\ 20 \\ 30 \\ 12 \\ 6 \\ 16 \\ 11 \end{bmatrix}.$$ 

i) By applying the BTLE algorithm in [5], it breaks down

ii) By applying the SBTLE algorithm, it yields $x_{SBTLE} = [12067595/(8154227 t+12067595), (8516457 t+12067595)/(8154227 t+12067595), (6642982 t+12067595)/(8154227 t+12067595), (17567204 t+12067595)/(8154227 t+12067595), (9396731 t+12067595)/(8154227 t+12067595), (41265687 t+12067595)/(8154227 t+12067595), (4932382 t+12067595)/(8154227 t+12067595), (14315844 t+12067595)/(8154227 t+12067595), (5598816 t+12067595)/(8154227 t+12067595)]^T.$ 

4.3 Example

We consider the following general bordered linear system in order to demonstrate the efficiency of SBTLE algorithm and SMWBTLS algorithm. Let’s have a look at the following bordered linear system:
It can be verified that the exact solution is $x = [1, 1, ..., 1]^T$. Results are given in the next table in which $\varepsilon = \|x - \bar{x}\|$.

Table 1. Errors and CPU times in seconds for SBTLE Algorithm, SMWBTLE Algorithm, and Gauss Elimination Algorithm when $n=500, 1000, 5000,$ and $10000$.

<table>
<thead>
<tr>
<th>n</th>
<th>SBTLE Algorithm</th>
<th>SMWBTLE Algorithm</th>
<th>Gauss Elimination Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>$3.41 \times 10^{-8}$</td>
<td>$1.46 \times 10^{-7}$</td>
<td>$4.40 \times 10^{-6}$</td>
</tr>
<tr>
<td>1000</td>
<td>$6.91 \times 10^{-8}$</td>
<td>$2.96 \times 10^{-7}$</td>
<td>$8.90 \times 10^{-6}$</td>
</tr>
<tr>
<td>5000</td>
<td>$3.49 \times 10^{-7}$</td>
<td>$1.50 \times 10^{-6}$</td>
<td>$4.49 \times 10^{-7}$</td>
</tr>
<tr>
<td>10000</td>
<td>$6.99 \times 10^{-7}$</td>
<td>$3.00 \times 10^{-6}$</td>
<td>$8.99 \times 10^{-7}$</td>
</tr>
</tbody>
</table>

Based on the result in Table 1, we can conclude that SBTLE Algorithm is more accurate and faster than the other algorithms. Also, SMWBTLE Algorithm is better than Gauss Elimination Algorithm.

5. CONCLUSIONS

In this paper, new symbolic computational algorithms have been developed for computing the solution of a general bordered tridiagonal linear system. The algorithms are reliable, computationally efficient and remove the cases where the numeric algorithms fail.

REFERENCES


