Available at
http://pvamu.edu/aam
Appl. Appl. Math.
ISSN: 1932-9466

Applications and Applied Mathematics:
An International Journal
(AAM)

Vol. 10, Issue 2 (December 2015), pp. 995-1006

Algorithm for Solving Tri-diagonal Finite Volume Discretized Linear Systems

J. S. V. R. Krishna Prasad
Department of Mathematics,
M. J. College, Jalgaon -425 001, Maharashtra, India
krishnaprasadjsvr@yahoo.com

Parag V. Patil
Department of Applied Science,
SSBT's College of Engineering and Technology,
Bambhori, Jalgaon -425 001, Maharashtra, India
patilparagv7@gmail.com

Received: October 9, 2014; Accepted: August 11, 2015

Abstract

In this paper we present efficient computational algorithms for solving finite volume discretized tri-diagonal linear systems. The implementation of the algorithm for steady state finite volume structured grids linear system using MS Excel is presented. An example is given in order to illustrate the algorithms.

Keywords: Computational algorithm; Discretized linear system; Finite volume method; MS Excel; Structured grids; Steady state; Tri-diagonal Matrix

MSC 2010 No.: 65F30, 65N22

1. Introduction

Tri-diagonal matrices play a central role in the solution of linear systems of equations in the different disciplines of science and engineering. We study algorithm for most commonly occurring problem in scientific computing, the solution of linear systems having a backward tri-diagonal coefficient matrices. This kind of linear system occurs in many field of numerical computation; see EI-Mikkawy (2004, 2005) and Karawia (2007).

Linear solution methods can broadly be classified into two categories: direct, and iterative. Examples of direct methods are Gauss elimination, LU decomposition, Cramer's rule, and matrix inversion. For more details, see Allen and Isaacson (1997). On the other hand, iterative methods can easily be formulated to take advantage of the matrix sparsity. Since these methods successively improve the solution by the application of a fixed number of operations, we can stop the process when the solution at any given iteration has been obtained to a sufficient level of accuracy.

The remainder of the paper is organized as follows. Section 2 contains the theoretical justification of the iterative algorithm for a special tri-diagonal matrix, and Section 3 contains the formulation of one and two dimensional finite volume discretized equations of linear systems, while Section 4 contains computational algorithms for tri-diagonal finite volume discretized linear system. An illustrative example and the implementation of algorithm using MS excel are presented in Section 5. Finally, Section 6 concludes the paper.

2. Tri-Diagonal Matrix Algorithm

The finite volume discretized system of linear equations is of the form

$$\mathbf{Ax} = \mathbf{b}$$

Here, **A** is an $N \times N$ matrix and **x** is a vector of the unknowns. The efficient solution of such systems is an important component of computational fluid dynamics (CFD) analysis. One important characteristic of our linear systems is that they contain large number of zeroes in the matrix **A** and the discrete equation at a cell has non-zero coefficients for only the neighbouring cells. The system of equations resulting from a one-dimensional grid, for example, has non-zero entries only on the diagonal and two adjacent "lines" on either side. For a mesh of 5 cells, the matrix has the form

$$A = \begin{bmatrix} x & x & 0 & 0 & 0 \\ x & x & x & 0 & 0 \\ 0 & x & x & x & 0 \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \end{bmatrix}.$$

Here, *x* denotes the non-zero entries. The linear systems involving such matrices are known as tri-diagonal matrices. The solution of such matrices can be attributed to Thomas (1949) who developed a technique for rapidly solving tri-diagonal systems which is known as the Thomas algorithm or the tri-diagonal matrix algorithm (TDMA). The tri-diagonal linear system plays a very important role in solving finite volume discretized equations, see Versteeg and Malalasekera (1995). The TDMA is actually a direct method for one dimensional situation, but it can be applied iteratively in a line-by-line fashion, to solve multidimensional problems and is frequently used in CFD problems.

The tri-diagonal matrix algorithm (TDMA) is a simplified form of Gaussian elimination that can be used to solve tri-diagonal system of equations. A tri-diagonal system for *n* unknowns may be written as

$$\begin{aligned} u_1 &= d_1 \\ -a_2u_1 + b_2u_2 - c_2u_3 &= d_2 \\ -a_3u_2 + b_3u_3 - c_3u_4 &= d_3 \\ \cdot &\quad \cdot \\ \cdot &\quad \cdot \\ \cdot &\quad \cdot \\ -a_{n-1}u_{n-2} + b_{n-1}u_{n-1} - c_{n-1}u_n &= d_{n-1} \\ -a_nu_{n-1} + b_nu_n - c_nu_{n+1} &= d_n \\ u_{n+1} &= d_{n+1}. \end{aligned} \tag{1}$$

In the above set of equations u_1 and u_{n+1} are known boundary values. The general form of any single equation is

$$-a_i u_{i-1} + b_i u_i - c_i u_{i+1} = d_i. \tag{2}$$

The equations of the system Equation (1) can be rewritten as

$$u_2 = \frac{c_2}{b_2} u_3 + \frac{a_2}{b_2} u_1 + \frac{d_2}{b_2} \tag{3}$$

$$u_3 = \frac{c_3}{b_3} u_4 + \frac{a}{b_3} u_2 + \frac{d_3}{b_3} \tag{4}$$

. .
. .
. .

$$u_n = \frac{c_n}{b_n} u_{n+1} + \frac{a_n}{b_n} u_{n-1} + \frac{d_n}{b_n}. \tag{5}$$

The TDMA is based on the Gaussian elimination procedure and consists of two parts - a forward elimination phase and a backward substitution phase. The forward elimination process starts by removing u_2 from Equation (4) by substitution from Equation (3) to get

$$u_3 = \left(\frac{c_3}{b_3 - a_3 \frac{c_2}{b_2}} \right) u_4 + \left(\frac{a_3 \left(\frac{a_2}{b_2} u_1 + \frac{d_2}{b_2} \right) + d_3}{b_3 - a_3 \frac{c_2}{b_2}} \right). \tag{6}$$

If we let

$$A_2 = \frac{c_2}{b_2}$$

$$B_2 = \frac{a_2}{b_2} u_1 + \frac{d_2}{b_2},$$

then Equation (6) can be written as

$$u_3 = \left(\frac{c_3}{b_3 - a_3 A_2} \right) u_4 + \left(\frac{a_3 B_2 + d_3}{b_3 - a_3 A_2} \right). \tag{7}$$

If we let

$$A_3 = \frac{c_3}{b_3 - a_3 A_2}$$

and

$$B_3 = \frac{a_3 B_2 + d_3}{b_3 - a_3 A_2},$$

then Equation (7) can be rewritten as

$$u_3 = A_3 u_4 + B_3. \tag{8}$$

From the system (1), we obtain

$$A_i = \frac{c_1}{b_1}, \qquad \text{if } i = 1,$$
$$= \frac{c_i}{b_i - a_i A_{i-1}}, \qquad \text{if } i = 2, 3, \ldots, (n-1),$$

$$B_i = \frac{d_1}{b_1}, \qquad \text{if } i = 1,$$
$$= \frac{a_i B_{i-1} + d_i}{b_i - a_i A_{i-1}}, \qquad \text{if } i = 2, 3, \ldots, n.$$

Equation (8) can be used to eliminate u_3 and the procedure can be repeated up to the last equation of the system. This constitutes the forward elimination process. For the back substitution, we use the general form of Equation (8).

$$u_i = A_i u_{i+1} + B_i, \qquad \text{if } i = n-1, n-2, \ldots, 1.$$

Note that the conditions of diagonal dominance for the system (1) that are sufficient for stability of the tri-diagonal elimination can actually be relaxed. In fact, one can only require that the coefficients of system (1) satisfy the inequalities:

$$|b_1| \geq |c_1|,$$
$$|b_i| \geq |a_i| + |c_i|, \qquad \text{if } i = 2, 3, \ldots, (n-1),$$
$$|b_n| \geq |c_n|.$$

3. Problem Formulation

Problem I

The TDMA can be applied to solve a system of equations for one dimensional structured grid problem. For more details see Prasad and Patil (2014). Consider the grid in Figure 1 and for west-east (w-e) line the finite volume discretized equation is re-arranged in the form

$$-a_W u_W + a_P u_P - a_E u_E = S_u. \tag{9}$$

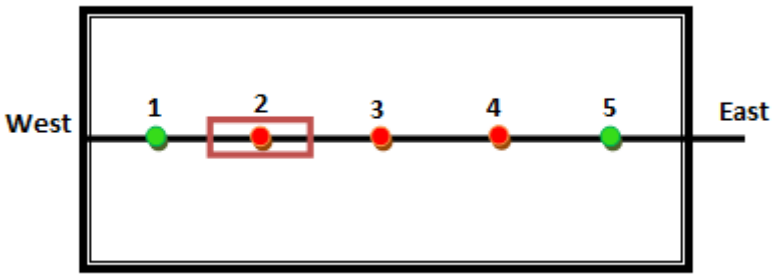

Figure1. Finite volume one dimensional grids

Problem II

The TDMA can be applied iteratively to solve a system of equations for two dimensional structured grid problems. See Prasad and Patil (2014). Consider the grid in Figure 2 and a general two dimensional finite volume discretized equation of the form

$$a_P u_P = a_W u_W + a_E u_E + a_S u_S + a_N u_N + S_u.$$

To solve the system, TDMA is applied along north-south (n-s) lines. The finite volume discretized equation is rearranged in the form

$$-a_S u_S + a_P u_P - a_N u_N = a_W u_W + a_E u_E + S_u \,. \tag{10}$$

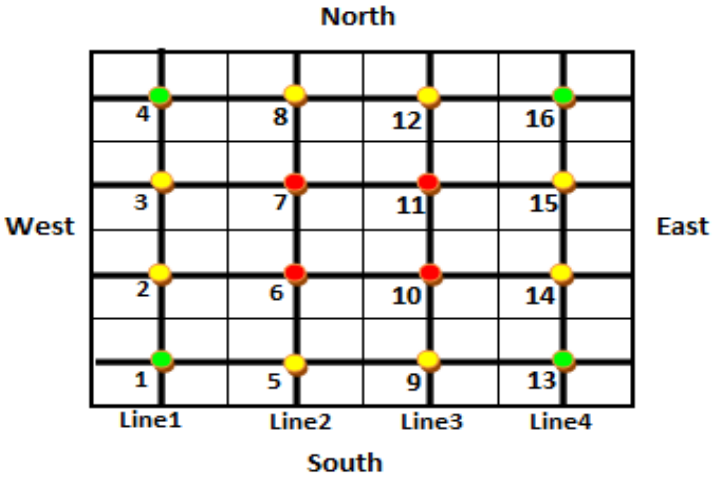

Figure 2. Finite volume two dimensional grids

4. Computational Algorithms

Algorithm 4.1 To solve the one dimensional general backward tri-diagonal linear systems Equation (9), we may proceed as follows:

Step 1: Set the given vectors

$$a_w = a,$$
$$a_p = b,$$
$$a_E = c,$$
$$S_u = d.$$

Step 2: For $i = 1$, set

$$A_0 = 0,$$
$$B_0 = 0,$$

and compute

$$A_1 = \frac{c_1}{b_1 - a_1 A_0}$$

and

$$B_1 = \frac{a_1 B_0 + d_1}{b_1 - a_1 A_0}.$$

Step 3: For $i = 2, \ldots, n$ and compute

$$A_i = \frac{c_i}{b_i - a_i A_{i-1}}$$

and

$$B_i = \frac{a_i B_{i-1} + d_i}{b_i - a_i A_{i-1}}.$$

Step 4: For $i = n, \ldots, 1$. Set $u_{n+1} = 0$, and compute

$$u_i = A_i u_{i+1} + B_i.$$

Algorithm 4.2 To solve the two dimensional general backward tri-diagonal linear systems Equation (10) along the north-south line-by-line fashion, we may proceed as follows:

Iteration 1

Step 1: Set the given vectors

$$a_N = a,$$
$$a_p = b,$$
$$a_S = c,$$
$$u_w = \text{ zero vector},$$
$$u_E = \text{ zero vector},$$
$$a_w u_w + a_E u_E + S_u = d.$$

Step 2: For $i = 1$, set

$$A_0 = 0,$$
$$B_0 = 0,$$

and compute

$$A_1 = \frac{c_1}{b_1 - a_1 A_0}$$

and

$$B_1 = \frac{a_1 B_0 + d_1}{b_1 - a_1 A_0}.$$

Step 3: For $i = 2, \ldots, n$ and compute

$$A_i = \frac{c_i}{b_i - a_i A_{i-1}}$$

and

$$B_i = \frac{a_i B_{i-1} + d_i}{b_i - a_i A_{i-1}}.$$

Step 4: For $i = n, \ldots, 1$. Set $u_{n+1} = 0$, and compute

$$u_i = A_i u_{i+1} + B_i.$$

(The End of the First Line)

Step 5: Set the given vectors

$$a_N = a,$$
$$a_p = b,$$
$$a_S = c,$$
$$u_E = \text{zero vector},$$
$$u_W = u \text{ (Result of First Line)},$$
$$a_w u_w + a_E u_E + S_u = d.$$

Step 6: Repeated Steps 2, 3, and 4, then we get the result of second line.

(The End of the Second Line)

Step 7: Set the given vectors

$$a_N = a,$$
$$a_p = b,$$
$$a_S = c,$$
$$u_E = \text{zero vector},$$
$$u_W = u \text{ (Result of Second Line)},$$
$$a_w u_w + a_E u_E + S_u = d.$$

Step 8: Repeated Steps 2, 3, and 4, then we get the result of third line.
(The End of the Third Line)

Step 9: Set the given vectors

$$a_N = a,$$
$$a_p = b,$$
$$a_S = c,$$
$$u_E = \text{zero vector},$$
$$u_W = u \text{ (Result of Third Line)},$$
$$a_w u_w + a_E u_E + S_u = d.$$

Step 10: Repeated Steps 2, 3, and 4, then we get the result of fourth line.
(The End of the Fourth Line)

Iteration 2

Step 1: Set the given vectors

$$a_N = a,$$
$$a_p = b,$$
$$a_S = c,$$
$$u_w = \text{zero vector},$$
$$u_E = u \text{ (Result of Second Line-Iteration 1)},$$
$$a_w u_w + a_E u_E + S_u = d.$$

Step 2: For $i = 1$, set

$$A_0 = 0,$$
$$B_0 = 0,$$

and compute

$$A_1 = \frac{c_1}{b_1 - a_1 A_0}$$

and

$$B_1 = \frac{a_1 B_0 + d_1}{b_1 - a_1 A_0}.$$

Step 3: For $i = 2, \ldots, n$ and compute

$$A_i = \frac{c_i}{b_i - a_i A_{i-1}}$$

and

$$B_i = \frac{a_i B_{i-1} + d_i}{b_i - a_i A_{i-1}}.$$

Step 4: For $i = n, \ldots, 1$. Set $u_{n+1} = 0$, and compute

$$u_i = A_i u_{i+1} + B_i.$$

(The End of the First Line)

Step 5: Set the given vectors

$a_N = a,$
$a_p = b,$
$a_S = c,$
$u_E = u$ (Result of Third Line-Iteration 1),
$u_W = u$ (Result of First Line),
$a_w u_w + a_E u_E + S_u = d.$

Step 6: Repeated Steps 2, 3, and 4, then we get the result of second line.

(The End of the Second Line)

Step 7: Set the given vectors

$a_N = a,$
$a_p = b,$
$a_S = c,$
$u_E = u$ (Result of Fourth Line-Iteration 1),
$u_W = u$ (Result of Second Line),
$a_w u_w + a_E u_E + S_u = d.$

Step 8: Repeated Steps 2, 3, and 4, then we get the result of third line.

(The End of the Third Line)

Step 9: Set the given vectors

$a_N = a,$
$a_p = b,$
$a_S = c,$
$u_E =$ Zero vector,
$u_W = u$ (Result of Third Line),
$a_w u_w + a_E u_E + S_u = d.$

Step 10: Repeated Steps 2, 3, and 4, then we get the result of fourth line.

(The End of the Fourth Line)

The entire iteration 2 procedure is repeated until a converged solution is obtained.

5. An Illustrative Examples

In this section we are going to give illustrative examples.

Example 5.1
To solve the one dimensional general backward tri-diagonal linear systems Equation (9) of size 5 given below, by using the algorithm 4.1 and MS Excel.

$$\begin{bmatrix} 20 & 5 & 0 & 0 & 0 \\ 5 & 15 & 5 & 0 & 0 \\ 0 & 5 & 15 & 5 & 0 \\ 0 & 0 & 5 & 15 & 5 \\ 0 & 0 & 0 & 5 & 10 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix} = \begin{bmatrix} 1100 \\ 100 \\ 100 \\ 100 \\ 100 \end{bmatrix}.$$

Table 1. The Algorithm 4.1 for one dimensional grid

Node	a	b	c	d	A_i	B_i	u_i
					0	0	
1	0	20	5	1100	0.25	55	64.2276
2	5	15	5	100	0.3636	27.2727	36.9106
3	5	15	5	100	0.3793	17.9310	26.5041
4	5	15	5	100	0.3816	14.4737	22.6016
5	5	10	0	100	0	21.3008	21.3008
							0

Example 5.2.

To solve the two dimensional general backward tri-diagonal linear systems Equation (10) of size 4 along the north-south line-by-line fashion given below, by using the algorithm 4.2 and MS Excel.

For First Line:

$$\begin{bmatrix} 1000 & 250 & 0 & 0 \\ 250 & 1250 & 250 & 0 \\ 0 & 250 & 1250 & 250 \\ 0 & 0 & 250 & 1000 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 624.87 \\ 875 \\ 1125 \\ 1375.12 \end{bmatrix}$$

For Second Line:

$$\begin{bmatrix} 750 & 250 & 0 & 0 \\ 250 & 1000 & 250 & 0 \\ 0 & 250 & 1000 & 250 \\ 0 & 0 & 250 & 750 \end{bmatrix} \begin{bmatrix} u_5 \\ u_6 \\ u_7 \\ u_8 \end{bmatrix} = \begin{bmatrix} -0.125 \\ 0 \\ 0 \\ 0.125 \end{bmatrix}$$

For Third Line:

$$\begin{bmatrix} 750 & 250 & 0 & 0 \\ 250 & 1000 & 250 & 0 \\ 0 & 250 & 1000 & 250 \\ 0 & 0 & 250 & 750 \end{bmatrix} \begin{bmatrix} u_9 \\ u_{10} \\ u_{11} \\ u_{12} \end{bmatrix} = \begin{bmatrix} -0.125 \\ 0 \\ 0 \\ 0.125 \end{bmatrix}$$

For Fourth Line:

$$\begin{bmatrix} 1000 & 250 & 0 & 0 \\ 250 & 1250 & 250 & 0 \\ 0 & 250 & 1250 & 250 \\ 0 & 0 & 250 & 1000 \end{bmatrix} \begin{bmatrix} u_{13} \\ u_{14} \\ u_{15} \\ u_{16} \end{bmatrix} = \begin{bmatrix} 1124.87 \\ 1375 \\ 1625 \\ 1875.12 \end{bmatrix}$$

Table 2. The Algorithm 4.2 for two dimensional grid after first iteration using MS excel

Lines	Node	a	b	c	S_u	a_w	a_E	T_w	T_E	d	A_i	B_i	u_i
											0	0	
1	1	0	1000	250	624.87	0	250	0	0	624.87	0.25	0.6249	**0.9202**
	2	250	1250	250	875	0	250	0	0	875	0.2105	0.8684	**1.1811**
	3	250	1250	250	1125	0	250	0	0	1125	0.2088	1.1209	**1.4855**
	4	250	1000	0	1375.12	0	250	0	0	1375.12	0	1.7465	**1.7465**
2	5	0	750	250	-0.125	250	250	**Result of First Line Iteration 1**	0	229.91	0.3333	0.3066	**0.5081**
	6	250	1000	250	0	250	250		0	295.28	0.2727	0.4057	**0.6045**
	7	250	1000	250	0	250	250		0	371.38	0.2683	0.5074	**0.7288**
	8	250	750	0	0.125	250	250		0	436.75	0	0.8253	**0.8253**
3	9	0	750	250	-0.125	250	250	**Result of Second Line Iteration 1**	0	126.89	0.3333	0.1692	**0.2721**
	10	250	1000	250	0	250	250		0	151.13	0.2727	0.211	**0.3086**
	11	250	1000	250	0	250	250		0	182.21	0.2683	0.2522	**0.358**
	12	250	750	0	0.125	250	250		0	206.44	0	0.3946	**0.3946**
4	13	0	1000	250	1124.87	250	0	**Result of Third Line Iteration 1**	0	1192.89	0.25	1.1929	**1.6809**
	14	250	1250	250	1375	250	0		0	1452.16	0.2105	1.474	**1.952**
	15	250	1250	250	1625	250	0		0	1714.5	0.2088	1.7397	**2.2703**
	16	250	1000	0	1875.13	250	0		0	1973.77	0	2.5413	**2.5413**

Table 3. The Algorithm 4.2 for two dimensional grid after second iteration using MS excel

Lines	Node	a	b	c	S_u	a_w	a_E	T_w	T_E	d	A_i	B_i	u_i
											0	0	
1	1	0	1000	250	624.87	0	250	0	**Result of Second Line Iteration 1**	751.88	0.25	0.7519	**1.0983**
	2	250	1250	250	875	0	250	0		1026.13	0.2105	1.0224	**1.3857**
	3	250	1250	250	1125	0	250	0		1307.21	0.2088	1.3052	**1.7255**
	4	250	1000	0	1375.12	0	250	0		1581.44	0	2.0128	**2.0128**
2	5	0	750	250	-0.125	250	250	**Result of First Line Iteration 2**	**Result of Third Line Iteration 1**	342.47	0.3333	0.4566	**0.7452**
	6	250	1000	250	0	250	250			423.57	0.2727	0.5866	**0.8657**
	7	250	1000	250	0	250	250			520.87	0.2683	0.7164	**1.0232**
	8	250	750	0	0.125	250	250			601.98	0	1.1437	**1.1437**
3	9	0	750	250	-0.125	250	250	**Result of Second Line Iteration 2**	**Result of Fourth Line Iteration 1**	606.39	0.3333	0.8085	**1.2858**
	10	250	1000	250	0	250	250			704.41	0.2727	0.9889	**1.4318**
	11	250	1000	250	0	250	250			823.37	0.2683	1.1489	**1.6238**
	12	250	750	0	0.125	250	250			921.39	0	1.7698	**1.7698**
4	13	0	1000	250	1124.87	250	0	**Result of Third Line Iteration 2**	0	1446.32	0.25	1.4463	**2.0288**
	14	250	1250	250	1375	250	0		0	1732.95	0.2105	1.7638	**2.3298**
	15	250	1250	250	1625	250	0		0	2030.94	0.2088	2.0644	**2.6887**
	16	250	1000	0	1875.13	250	0		0	2317.57	0	2.9897	**2.9897**

The entire iteration 2 procedure is repeated and its converged solution is obtained after 7th iterations as shown in Table 4.

Table 4 The Algorithm 4.2 for two dimensional grid after seventh iteration using MS excel

Node/ Iterations	1	2	3	4	5	6	7
1	0.9202	1.0983	1.1796	1.3735	1.4886	1.5517	1.5857
2	1.1811	1.3857	1.4736	1.6797	1.7992	1.8637	1.8982
3	1.4855	1.7255	1.8227	2.0456	2.1713	2.2378	2.2730
4	1.7465	2.0128	2.1167	2.3518	2.4818	2.5498	2.5854
5	0.5081	0.7452	1.3147	1.6559	1.8436	1.9452	1.9997
6	0.6045	0.8657	1.4792	1.8360	2.0289	2.1322	2.1872
7	0.7288	1.0232	1.6967	2.0753	2.2756	2.3812	2.4371
8	0.8253	1.1437	1.8612	2.2554	2.4608	2.5682	2.6246
9	0.2721	1.2858	1.7586	2.0139	2.1522	2.2267	2.2666
10	0.3086	1.4318	1.9330	2.1973	2.3386	2.4140	2.4542
11	0.3580	1.6238	2.1642	2.4412	2.5867	2.6636	2.7042
12	0.3946	1.7698	2.3386	2.6246	2.7731	2.8509	2.8919
13	1.6809	2.0288	2.1890	2.2749	2.3213	2.3462	2.3596
14	1.9520	2.3298	2.4980	2.5864	2.6336	2.6588	2.6722
15	2.2703	2.6887	2.8678	2.9598	3.0082	3.0337	3.0473
16	2.5413	2.9897	3.1767	3.2712	3.3204	3.3463	3.3599

6. Conclusion

The technique described here is very effective and easy to implement as compared to costly software such as MAPLE, MATHMATICA and MATLAB for solving a backward tri-diagonal finite volume structured grid linear systems which appear in many applications.

Acknowledgments

The authors would like to express their gratitude to the referees and the Editor-in-Chief Professor Aliakbar Montazer Haghighi for helpful suggestions which improved the earlier draft of this paper.

References

Allen M. B. and Isaacson E. L. (1997). *Numerical Analysis for Applied Science*, Wiley-Interscience, John Wiley and Sons.

Chawla M. and Khazal R. R. (2002). A Parallel Elimination Method for Periodic Tri-diagonal Systems, International Journal of Computer Mathematics, Vol 79, No. 4.

Cheney W. and Kincaid D. (1985). *Numerical Mathematics and Computing*, Wadsworth, Inc.

EI-Mikkawy M. E. A. (2004). A Fast Algorithm for evaluating nth Order Tri-diagonal Determinants, Journal of Computational and Applied Mathematics, Vol. 166, No. 2.

EI-Mikkawy M. E. A. (2005). A new Computational Algorithm for Solving Periodic Tri-diagonal Linear Systems, Applied Mathematics and Computation, Vol. 161 No. 2.

Karawia A. A. (2007). Two Algorithms for Solving a General Backward Tri-diagonal Linear System, Applied Mathematics and Computation, Vol. 194, No. 2.

Karawia A. A. (2006). A Computational Algorithm for Solving Periodic Penta-diagonal Linear System, Applied Mathematics and Computation, Vol. 174.

Prasad Krishna J. S. V. R. and Patil P. V. (2014). Finite Volume Numerical Grid Technique for Solving One and Two Dimensional Heat Flow Problems, Research Journal of Mathematical and Statistical Sciences (RJMSS), Vol. 2, No. 8.

Prasad Krishna J. S. V. R. and Patil P. V. (2014). Finite Volume Numerical Grid Technique for Multidimensional Problems, International Journal of Science and Research (IJSR), Vol. 3, No. 7.

Versteeg H. K. and Malalasekera W. (1995). *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*, Longman Scientific and Technical, England.