Solving Tridiagonal Symmetric Systems of Equations Using Circuit Theory Approach

Mirko Todorovski, Jordančo Angelov and Jovica Vuletić

***Abstract*—This paper introduces a novel solution method for solving symmetric tridiagonal systems. The main idea behind it is to construct a specific electric circuit with the same node-voltage equations as the original system. This circuit has a specific "ladder" structure that is efficiently solved using a methodology known as admittance summation method.**

The proposed method avoids possible zero divisions by exploiting a specific circuit structure. This specific property is an equivalent to a pivoting strategy used in other methods.

Performance tests show that the proposed method is comparable to Thomas algorithm, Gaussian elimination adapted for tridiagonal systems and Matlab backslash operator.

The procedure executes $O(N)$ times meaning that computation time is linearly proportional to system size. The whole method is coded very concisely.

***Index Terms*—Electric circuits, admittance summation, tridiagonal systems, linear algebra.**

I. Introduction

A Non-singular tridiagonal linear system of equations $\mathbf{A} \cdot \mathbf{u} = \mathbf{r}$ is often solved using matrix factorization. One of the most efficient approaches is to a use diagonal pivoting method with LBL^T decomposition of $\mathbf{A}$, where $\mathbf{L}$ is unit lower triangular and $\mathbf{B}$ is a block diagonal matrix with 1×1 and 2×2 blocks. Diagonal pivoting methods have been developed for symmetric tridiagonal matrices that do not require row or column interchanges [1].

It is usually simpler to solve these systems using tailor-made algorithms that exploit the special structure of $\mathbf{A}$ [2]. Very often, in these simple algorithms there is no pivoting and for this reason they can fail even when the underlying matrix is non-singular. In practice, this does not frequently occur [3], but it is desirable to have an algorithm that can handle such cases.

This paper presents an algorithm that solves tridiagonal symmetric system of equations, representing it with an electric circuit whose node-voltage equations are the same as the original system. The equivalent circuit is solved using a specially tailored method for this type of circuits which has a ladder-like structure comprised of series of vertical and horizontal elements [4]. The solution method does not require pivoting. Additionally, it only accounts for possible zero divisions that are avoided by exploiting the circuits specific structure.

Manuscript received July 18, 2020; revised March 6, 2021.

M. Todorovski is a Full Professor at the Institute of Power Transmission Systems, Faculty of Electrical Engineering and Information Technologies, University Ss. Cyril and Methodius, Skopje, Macedonia (corresponding author, e-mail: mirko@feit.ukim.edu.mk).

J. Angelov is an Assistant Professor at the Institute of Power Transmission Systems, Faculty of Electrical Engineering and Information Technologies, University Ss. Cyril and Methodius, Skopje, Macedonia (e-mail: jordanco@feit.ukim.edu.mk).

J. Vuletić is an Assistant Professor at the Institute of Power Transmission Systems, Faculty of Electrical Engineering and Information Technologies, University Ss. Cyril and Methodius, Skopje, Macedonia (e-mail: jovicav@feit.ukim.edu.mk).

II. Tridiagonal System of Equations Circuit Representation

For a circuit comprised of resistors and independent current sources with n nodes, a node-voltage equations can be written in terms of conductances as follows [5]:

$$\mathbf{G} \cdot \mathbf{u} = \mathbf{i}, \tag{1}$$

where $\mathbf{G}$ is a $n \times n$ symmetric matrix called the conductance matrix, while $\mathbf{u}$ and $\mathbf{i}$ are vectors of length n.

The unknown voltage at node k is u_k and i_k is independent current source known current directly connected to node k. Currents entering the node are treated as positive while those exiting as negative. For elements of $\mathbf{G}$, the following rules apply (2):

$$G_{kj} = \begin{cases} k = j: & \text{sum of conductances connected to node } k, \\ k \neq j: & \text{negative sum of conductances directly connecting nodes } k \text{ and } j, \\ 0 & \text{if there is no direct connection between nodes } k \text{ and } j. \end{cases} \tag{2}$$

Let's consider a simple electric circuit given in Figure 1. This circuit has a special structure called "ladder", because it resembles a ladder with series of vertical and horizontal elements.

All vertical elements are connected between a circuit node and ground. Vertical resistors are labeled with $R_1, R_2, \ldots, R_n$, while independent current sources are labeled with $i_1, i_2, \ldots, i_n$. In this case, an element index corresponds to a node number.

All horizontal elements are connected between two consecutive circuit nodes. These elements are resistors labeled $Z_1, Z_2, \ldots, Z_{n-1}$, with indices corresponding to the node number on their left-hand side.

According to (2), it can be seen that matrix $\mathbf{G}$ is tridiagonal since each node has at most two connections with other nodes. For example, the first row of $\mathbf{G}$ is comprised of the following elements:

$$G_{11} = \frac{1}{R_1} + \frac{1}{Z_1}, \quad G_{12} = -\frac{1}{Z_1}, \quad G_{13} = 0, \quad G_{14} = 0.$$

If, for the sake of simplicity all resistors assume a resistance value of $1\,\Omega$ and independent current sources are $i_1 = 0$ A, $i_2 = 2$ A, $i_3 = 3$ A and $i_4 = 5$ A respectively, then the circuit from Figure 1 is completely described with the following system of equations which is clearly tridiagonal.

$$\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 \\ 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 3 \\ 5 \end{bmatrix}, \tag{3}$$

The set of equations subject to solution possesses the

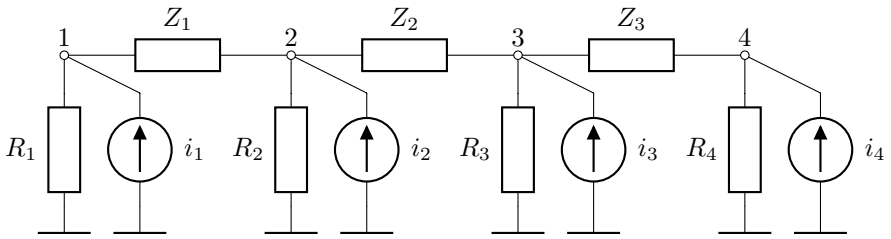

Fig. 1. Simple Electric Circuit

general form given with (4).

$$\begin{bmatrix} a_1 & b_1 & & & & & \\ b_1 & a_2 & b_2 & & & & \\ & b_2 & a_3 & b_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & b_{n-2} & a_{n-1} & b_{n-1} \\ & & & & & b_{n-1} & a_n \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ \vdots \\ r_{n-1} \\ r_n \end{bmatrix}. \tag{4}$$

It goes without saying that not the full $n \times n$ matrix is stored but only its nonzero elements as two row vectors $\mathbf{a}$ and $\mathbf{b}$ for the main diagonal and diagonals below and above the main diagonal.

At this point the following question arises: given the system of equations (4), could an electric circuit be constructed whose node-voltage equations are the same? The answer is an obvious one and the procedure to form such a circuit comes from the rule of formation for conductance matrix $\mathbf{G}$ (2) and by introducing a conductance $Y_k = 1/R_k$, $k = 1, 2, \ldots, n$, with the following parameters:

$$Z_k = -\frac{1}{b_k}, \quad k = 1, 2, \ldots, n, \tag{5a}$$
$$Y_1 = a_1 + b_1, \tag{5b}$$
$$Y_k = a_k + b_{k-1} + b_k, \quad k = 2, 3, \ldots, n-1, \tag{5c}$$
$$Y_n = a_n + b_{n-1}, \tag{5d}$$
$$i_k = r_k, \quad k = 1, 2, \ldots, n. \tag{5e}$$

Equation (5a) is self explanatory since off-diagonal elements in $\mathbf{G}$ are negatives of branch conductance, which in turn is a reciprocal of branch resistance, hence $b_k = -1/Z_k$ and the derivation of equation (5a).

The diagonal term for node k is a sum of conductances of three elements: $1/R_k$, $1/Z_k$ and $1/Z_{k-1}$. Bear in mind that $Y_k = 1/R_k$ is introduced and considering (5a), equation (5c) is derived. The other two equations (5b) and (5d) are practically the same, with the exception that at nodes 1 and n there are only two connected elements instead of three.

III. Admittance Summation Method

Circuits with structure as in Figure 1 are very often encountered in power distribution networks and there are suitable solving methods that do not require solving any set of simultaneous equations. The most suitable method for solving such circuits is admittance summation method [4].

The method basically employs the rules for serial/parallel element connection that enables circuit reduction to a single resistance. Circuit nodes are eliminated one by one during the process, until a single node remains. This is equivalent to a variable elimination until a single variable remains, which is a trivial problem to solve. At the end, all other variables are obtained with a procedure equivalent to back-substitution used in LU decomposition methods [6].

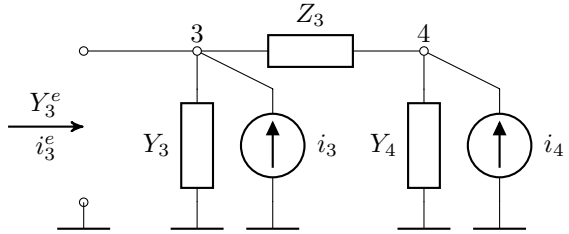

Fig. 2. Definition of Y^e and i^e for node 3

A. Backward sweep

For each node k, besides the known conductance Y_k, a driving point conductance Y_k^e is introduced as part of the circuit fed by node k, including Y_k. Additionally, an equivalent current generator i_k^e is introduced for the part of the circuit fed by node k. The current of this generator consists of all generator currents from nodes fed by node k. The following is presented in Figure 2 for node 3 of the circuit from Figure 1. From the definition of Y_k^e and i_k^e, it is obvious that for the last node n, $Y_n^e = Y_n$ and $i_n^e = i_n$. Therefore the circuit reduction starts from the last node n.

At the beginning, the following equations are set:

$$Y_k^e = Y_k, \quad k = 1, 2, \ldots, n, \tag{6a}$$
$$i_k^e = i_k, \quad k = 1, 2, \ldots, n, \tag{6b}$$

and proceed with backward-sweep node elimination going from node n towards node 1.

Let's assume that we are at node k which is connected through a branch with resistance Z_{k-1} to node $k-1$. In

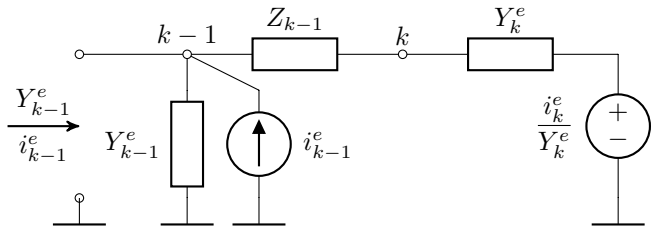

Fig. 3. Backward sweep: processing node k — step 1

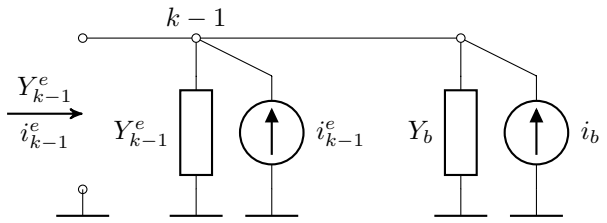

Fig. 4. Backward sweep: processing node k — step 2

addition, the parallel combination of conductance Y_k^e and current generator i_k^e is converted into a serial combination of the same conductance and voltage generator whose voltage, following the principle of source transformation [7], is i_k^e/Y_k^e (Figure 3).

The two resistors with resistance Z_{k-1} and conductance Y_k^e are connected in series and they can be replaced with a single resistor with conductance Y_b calculated as follows:

$$Y_b = \frac{1}{Z_{k-1} + 1/Y_k^e} = \frac{Y_k^e}{1 + Z_{k-1}Y_k^e} = D_k Y_k^e,$$

and accounting for (5a), the following applies:

$$D_k = \frac{1}{1 + Z_{k-1}Y_k^e} = \frac{b_{k-1}}{b_{k-1} - Y_k^e}. \tag{7}$$

Afterwards, the voltage generator connected in series with Y_b can be transformed into a current generator whose current is:

$$i_b = Y_b \frac{i_k^e}{Y_k^e} = D_k i_k^e,$$

as presented in Figure 4.

Finally, the parallel connection of Y_{k-1}^e and Y_b, as well as i_{k-1}^e and i_b can be replaced with a single conductance and current generator as follows:

$$Y_{k-1}^{e\prime} = Y_{k-1}^e + D_k Y_k^e, \quad k = n, n-1, \ldots, 2, \tag{8a}$$

$$i_{k-1}^{e\prime} = i_{k-1}^e + D_k i_k^e, \quad k = n, n-1, \ldots, 2, \tag{8b}$$

providing the desired driving point conductance $Y_{k-1}^{e\prime}$ for the part of the circuit fed by node $k-1$ and the corresponding driving point equivalent current generator $i_{k-1}^{e\prime}$.

When calculating D_k with (7), due consideration should be given for cases where $1 + Z_{k-1}Y_k^e = 0$. This is possible when $Z_{k-1} = -1/Y_k^e$, meaning that serial connection of Z_{k-1} and $-1/Y_k^e$ has zero resistance and the circuit from Figure 3 takes on the form as that from Figure 5. Condition $Z_{k-1} = -1/Y_k^e$ can also be written as $b_{k-1} - Y_k^e = 0$.

Figure 5 suggests that there is an ideal voltage generator connected to node $k-1$, so that its voltage is $u_{k-1} = i_k^e/Y_k^e$. This implies that elements Y_{k-1}^e and i_{k-1}^e can be ignored since they do not change anything, because the voltage at node $k-1$ is fixed via the ideal voltage generator. Next, the serial connection of Z_{k-2} and i_k^e/Y_k^e can be converted into a parallel connection of Z_{k-2} and an ideal current generator with current equal to $(i_k^e/Y_k^e)/Z_{k-2}$, both connected at node $k-2$. As a conclusion, conductance $1/Z_{k-2} = -b_{k-2}$ and a current generator $-b_{k-2}i_k^e/Y_k^e$ are added to node $k-2$ as follows:

$$Y_{k-2}^{e\prime} = Y_{k-2}^e - b_{k-2}, \tag{8c}$$

$$i_{k-2}^{e\prime} = i_{k-2}^e - b_{k-2}\frac{i_k^e}{Y_k^e}, \tag{8d}$$

afterwards proceeding with node $k-3$.

From Fig. 5 it can be seen that the voltage at node $k-1$ is known as well:

$$u_{k-1} = i_k^e/Y_k^e. \tag{9}$$

However, in this case it's not possible to calculate the voltage at node k using the circuits from Fig. 3 and 5. Therefore, row $k-1$ from matrix $\mathbf{A}$ is used, which leads to

$$u_k = \frac{r_{k-1} - b_{k-2}u_{k-2} - a_{k-1}u_{k-1}}{b_{k-1}}. \tag{10}$$

B. Forward sweep

After performing a backward sweep, the circuit is reduced to a single conductance Y_1^e and current generator i_1^e, hence the calculation of voltage at node 1 is as follows:

$$u_1 = \frac{i_1^e}{Y_1^e}. \tag{11a}$$

From Fig. 6, the voltage at node k can be calculated as:

$$u_k = u_{k-1} - Z_{k-1}i_{k-1} = u_{k-1} - Z_{k-1}(Y_k^e u_k - i_k^e),$$

from where:

$$u_k = \frac{u_{k-1} + Z_{k-1}i_k^e}{1 + Z_{k-1}Y_k^e}.$$

Finally, when substituting (7) and (5a) the voltage at node k is

$$u_k = D_k\left(u_{k-1} - \frac{i_k^e}{b_{k-1}}\right). \tag{11b}$$

Due consideration should be given when applying (11b) for cases where $b_k = 0$. Since Z_k connects nodes k and $k+1$, having $b_k = 0$ implies that these two nodes are connected via infinite resistance $Z_k = \infty$, i.e. they are disconnected. As a consequence, node $k+1$ is a starting node for another circuit going right from it. For this circuit, driving point conductance Y_{k+1}^e and equivalent current generator i_{k+1}^e is already calculated, so its voltage is:

$$u_{k+1} = \frac{i_{k+1}^e}{Y_{k+1}^e} \quad \text{if} \quad b_k = 0. \tag{12}$$

When coding the above procedure which is done in Matlab for the purposes of this paper, there's no need of separate vectors for $\mathbf{Y^e}$ and $\mathbf{Y}$, nor for $\mathbf{i^e}$ and $\mathbf{i}$. Single vectors $\mathbf{Y}$ and $\mathbf{i}$ can be used and overwrite their elements as needed.

IV. Solution Method

The admittance summation is performed in-place with overwriting elements in both vectors $\mathbf{Y}$ and $\mathbf{r}$ going from the last node n to node 1. Please note that a vector $\mathbf{i}$ is not introduced. Instead, operations are performed in the original

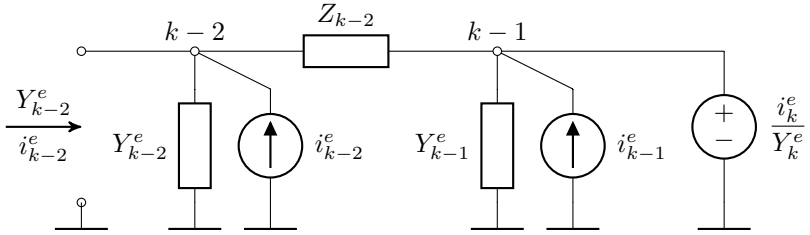

Fig. 5. Backward sweep: processing node — special case

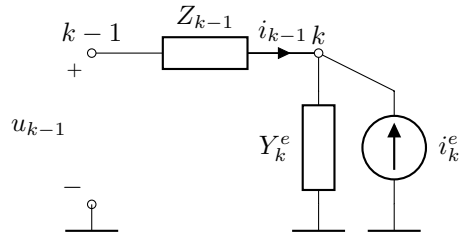

Fig. 6. Calculation of u_k

vector $\mathbf{r}$ since they are the same. The procedure developed in the previous section can be summarized into three simple steps: Initialization, Backward sweep and Forward sweep.

Initial values for elements in $\mathbf{Y}$ are obtained from (5b) — (5d). It can be seen that they are defined as sums of nonzero elements in each row of the tridiagonal matrix. Since its nonzero elements are stored in two row vectors $\mathbf{a}$ and $\mathbf{b}$, the following vector equation can be written:

$$\mathbf{Y} = \mathbf{a} + [0 \;\; \mathbf{b}] + [\mathbf{b} \;\; 0], \tag{13a}$$

where $[0 \;\; \mathbf{b}]$ is a concatenation of 0 and row vector $\mathbf{b}$, while $[\mathbf{b} \;\; 0]$ is another concatenation where 0 is added after $\mathbf{b}$. In the initial phase, the original right-hand side of vector $\mathbf{r}$ is also saved:

$$\mathbf{r}^o = \mathbf{r}. \tag{13b}$$

The next two steps are given as algorithms in Algorithm 1 and Algorithm 2. The approach is used for more concise presentation of the admittance summation procedure presented in Section III-A and the voltage calculation procedure presented in Section III-B. Both algorithms account for possible special cases and avoid zero divisions that in a way is a replacement for pivot strategy which is not needed in the proposed method. Although in Algorithm 2 there are three "if" statements, in practice, only the first one will be checked since the other two may appear in special circumstances when equations (10) or (12) have to be applied as explained in the previous sections.

The following section covers time performance of a "raw" version of the method where zero division check is omitted. Practically this version only executes lines 5–7 from Algorithm 2 and line 4 from Algorithm 1, which is a bit faster since all "if" statements are avoided. The "raw" version of the method is given in Algorithm 3.

V. Tridiagonal Systems — Gaussian Elimination

Very often a system of equations has the properties that allow for safe usage of Gaussian elimination without pivoting. The class of matrices for which this is true are called

Algorithm 1 Backward sweep

1: **Initialize:**

$u_i \leftarrow 0, \quad D_i \leftarrow 0, \quad s_i \leftarrow$ False

$i = 1, 2, \ldots, n$

2: $k = n$

3: **while** $k \geq 2$ **do**

4: **if** $b_{k-1} - Y_k \neq 0$ **then**

5: $D_k \leftarrow \dfrac{b_{k-1}}{b_{k-1} - Y_k}$

6: $Y_{k-1} \leftarrow Y_{k-1} + D_k Y_k$

7: $r_{k-1} \leftarrow r_{k-1} + D_k r_k$

8: **else**

9: $Y_{k-2} \leftarrow Y_{k-2} - b_{k-2}$

10: $r_{k-2} \leftarrow r_{k-2} - b_{k-2}\dfrac{r_k}{Y_k}$

11: $u_{k-1} \leftarrow r_k / Y_k$

12: $s_k \leftarrow$ True

13: $k \leftarrow k - 1$

14: **end if**

15: $k \leftarrow k - 1$

16: **end while**

diagonally dominant matrices with properties expressed by the inequality:

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|. \tag{14}$$

These matrices arise naturally in applications involving the discretization by finite differences of partial differential equations, in the study of splines and other areas [8]. If the coefficient matrix has this property, then in the steps of Gaussian elimination rows can be used consecutively as pivot rows since the pivot elements are not zero (14). After one step is completed, we would like to know that next row 2 can be used as the next pivot row, which is governed by the next theorem.

Theorem 1. *Gaussian elimination without pivoting preserves the diagonal dominance of a matrix.*

Proof: It is enough to consider the first step in Gaussian elimination, in which zeros are created in column 1, since subsequent steps are exactly the same as the first, except for being applied to matrices of smaller size.

Algorithm 2 Forward sweep

$u_1 \leftarrow \dfrac{r_1}{Y_1}$
for $k = 2$ **to** n **do**
  if $D_k \neq 0$ **then**
    $u_k = D_k \left(u_{k-1} - \dfrac{r_k}{b_{k-1}} \right)$
    go to 15
  end if
  if $b_{k-1} = 0$ **then**
    $u_k = \dfrac{r_k}{Y_k}$
    go to 15
  end if
  if $s_k =$ True **then**
    $u_k = \dfrac{r^o_{k-1} - b_{k-2}u_{k-2} - a_{k-1}u_{k-1}}{b_{k-1}}$
    go to 15
  end if
end for

Algorithm 3 The Admittance Summation — "raw" version

for $k = n$ **to** 2 **do**
  $D_k \leftarrow \dfrac{b_{k-1}}{b_{k-1} - Y_k}$
  $Y_{k-1} \leftarrow Y_{k-1} + D_k Y_k$
  $r_{k-1} \leftarrow r_{k-1} + D_k r_k$
end for
$u_1 \leftarrow \dfrac{r_1}{Y_1}$
for $k = 2$ **to** n **do**
  $u_k = D_k \left(u_{k-1} - \dfrac{r_k}{b_{k-1}} \right)$
end for

Let $\mathbf{A}$ be an $n \times n$ matrix that is diagonally dominant. Accounting for zeros created in column 1 and having in mind that row 1 is unchanged, we have to prove that

$$\left| a_{ii}^{(2)} \right| > \sum_{\substack{j=2 \\ j \neq i}}^{n} \left| a_{ij}^{(2)} \right|, \quad i = 2, 3, \ldots, n,$$

where superscript (2) denotes step 2.

Taking into account the operations included in the Gaussian elimination the above can be written as:

$$|a_{ii} - (a_{i1}/a_{11})a_{1i}| > \sum_{\substack{j=2 \\ j \neq i}}^{n} |a_{ij} - (a_{i1}/a_{11})a_{1j}|,$$

for $i = 2, 3, \ldots, n$.

It is sufficient to prove the stronger inequality:

$$|a_{ii}| - |(a_{i1}/a_{11})a_{1i}| > \sum_{\substack{j=2 \\ j \neq i}}^{n} |a_{ij}| + |(a_{i1}/a_{11})a_{1j}|,$$

which may be rewritten as:

$$|a_{ii}| - \sum_{\substack{j=2 \\ j \neq i}}^{n} |a_{ij}| > \sum_{j=2}^{n} |(a_{i1}/a_{11})a_{1j}|.$$

From the diagonal dominance, for row i the following applies:

$$|a_{ii}| - \sum_{\substack{j=2 \\ j \neq i}}^{n} |a_{ij}| > |a_{i1}|,$$

hence, it's suffice to prove that:

$$|a_{i1}| \geq \sum_{j=2}^{n} |(a_{i1}/a_{11})a_{1j}|.$$

This is true because of diagonal dominance in row 1, that is:

$$|a_{11}| > \sum_{j=2}^{n} |a_{1j}| \quad \Rightarrow \quad 1 > \sum_{j=2}^{n} |a_{1j}/a_{11}|.$$

■

In its simplest form, Gaussian elimination is applied assuming that the coefficient matrix is such that pivoting is not necessary when solving the system. This is true if the matrix is symmetric and positive definite [2].

This paper uses simple Gaussian elimination where the right-hand side vector $\mathbf{r}$ is processed simultaneously with the left-hand side matrix $\mathbf{A}$. In Step 1, a multiple of row 1 is subtracted from row 2 to obtain 0 in the position where b_1 was placed originally. Please note that a_2 and r_2 are altered but b_2 is not. The multiple used is b_1/a_1, therefore Step 1 consists of these replacements:

$$a_2 \leftarrow a_2 - \frac{b_1}{a_1} b_1,$$
$$r_2 \leftarrow r_2 - \frac{b_1}{a_1} r_1.$$

All remaining steps producing zeros in the other rows during the forward elimination phase are exactly the same as the first one, only with different indices.

In the backward substitution phase, the first step is:

$$u_n \leftarrow \frac{r_n}{a_n},$$

then the next step is as follows:

$$u_{n-1} \leftarrow \frac{r_{n-1} - b_{n-1} r_n}{a_{n-1}},$$

and the whole procedure is presented in Algorithm 4.

Gaussian elimination with full pivoting, treats both rows and columns in an order different from the natural order. In each step, the pivot element a_{ij} is chosen so that $|a_{ij}|$ is the largest in the entire matrix. This determines that row i will be the pivot row and column j will be the pivot column. Zeros are created in column j by subtracting multiples of row i from the other rows. During the process of Gaussian elimination with pivoting, two permutation vectors are required to keep track of possible row/column permutations.

VI. Results

The proposed method is compared to two other methods specially developed for tridiagonal systems, as well as with

Algorithm 4 Tridiagonal Gaussian Elimination

1: **for** $i = 2$ **to** n **do**
2: $\quad m \leftarrow \dfrac{b_{i-1}}{a_{i-1}}$
3: $\quad a_i \leftarrow a_i - m \cdot b_{i-1}$
4: $\quad r_i \leftarrow r_i - m \cdot r_{i-1}$
5: **end for**
6: $u_n \leftarrow \dfrac{r_n}{a_n}$
7: **for** $i = n - 1$ **to** 1 **step** -1 **do**
8: $\quad u_i \leftarrow \dfrac{r_i - b_i r_{i+1}}{a_i}$
9: **end for**

the Matlab backslash operator. The first method is a simple Gaussian elimination tailored for tridiagonal systems [2], while the second one is the Thomas algorithm [3].

Both methods assume that the tridiagonal matrix is such that pivoting is not necessary for solving the system. This is the case for matrices that are diagonally dominant, which is expressed by the following inequalities:

$$\begin{aligned} &|a_1| > |b_1|, \\ &|a_k| > |b_k| + |b_{k-1}|, \quad k = 2, 3, \ldots, n-1, \\ &|a_n| > |b_{n-1}|, \end{aligned}$$

and it is known that Gaussian elimination without pivoting preserves the diagonal dominance of a matrix [2].

On the other hand, Matlab backslash operator solves a system of linear equations ($\mathbf{A}\backslash\mathbf{r}$) with sparse matrices using LU decomposition with pivoting strategy, so it can solve any system with non-singular matrix regardless of its diagonal dominance.

For example, one system with diagonally dominant matrix is (3) and all methods easily calculate the solution $u_1 = 1$, $u_2 = 2$, $u_3 = 3$ and $u_4 = 4$.

Since there is no pivoting in the methods from [2] and [3], they both fail due to zero pivot when the matrix is not diagonally dominant even when it is non-singular. An example of such system of equations is given with (15):

$$\begin{bmatrix} 2 & 2 & 0 & 0 \\ 2 & 2 & 3 & 0 \\ 0 & 3 & 4 & 5 \\ 0 & 0 & 5 & 1 \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 4 \\ 7 \\ 16 \\ 8 \end{bmatrix}. \tag{15}$$

This system has a solution $u_1 = 3$, $u_2 = -1$, $u_3 = 1$ and $u_4 = 3$ and the solution is obtained only with the Matlab backslash operator and with the proposed method. Therefore, the main feature of the proposed method is the capability to solve systems with matrices that are not diagonally dominant even though it does not use pivoting. At the same time, it is as simple as the other two methods.

Two additional systems of equations were constructed to check whether the proposed method solves cases with special conditions. They are the following:

$$\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 \\ 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 3 \\ 5 \end{bmatrix}, \tag{16}$$

and

$$\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 3 & 0 & 0 \\ 0 & 0 & 3 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 3 \\ 5 \end{bmatrix}. \tag{17}$$

In (16), the special case is encountered at node 4 where $Z_3 = 1\,\Omega$ and $Y_4 = -1$ S which gives a serial combination of 2 elements with total resistance equal to 0. Such a situation is detected in line 4 of Algorithm 2 and in line 11 voltage at node 3 is calculated as i_4^e/Y_4^e and flag variable s_4 set to "True", which signals to Algorithm 2 in line 11 that equation (10) has to be used to calculate u_4.

The special case in (17) is at node 3 where $Z_2 = \infty$, meaning that nodes 2 and 3 are not connected. This is detected in line 7 of Algorithm 2 and then voltage at node 3 is calculated using (12). In the next step, voltage calculation continues normally using line 4 since node 3 is a root node for another "regular ladder" circuit.

Before exploring the proposed method performance, let's illustrate the steps in solving the system (16). In the following paragraphs, symbols T and F are used as True and False flags respectively.

Initial step: $\mathbf{a} = [2\ 3\ 3\ 0]$, $\mathbf{b} = [-1\ -1\ -1]$, $\mathbf{r} = [0\ 2\ 3\ 5]$, $\mathbf{Y} = [1\ 1\ 1\ 1]$, $\mathbf{D} = [0\ 0\ 0\ 0]$, $\mathbf{s} = [F\ F\ F\ F]$ and $\mathbf{u} = [0\ 0\ 0\ 0]$.

Backward sweep:
$k = 4$, $b_3 - Y_4 = 0$
$Y_2 = Y_2 - b_2 = 1 - (-1) = 2$
$r_2 = r_2 - b_2 \cdot r_4/Y_4 = 2 - (-1) \cdot 5/(-1) = -3$
$u_3 = r_4/Y_4 = 5/(-1) = -5$
$s_4 = T$
$k = k - 2 = 2$

$k = 2$, $b_1 - Y_2 \neq 0$
$D_2 = b_1/(b_1 - Y_2) = -1/(-1 - 2) = 0.333333$
$Y_1 = Y_1 + D_2 \cdot Y_2 = 1 + 0.333333 \cdot 2 = 1.66667$
$r_1 = r_1 + D_2 \cdot r_2 = 0 + 0.333333 \cdot (-3) = -1$

Forward sweep:
$u_1 = r_1/Y_1 = -1/1.66667 = -0.6$

$k = 2$, $D_2 \neq 0$
$u_2 = D_2 \cdot (u_1 - r_2/b_1) = 0.333333 \cdot [-0.6 - (-3)/(-1)] = -1.2$

$k = 3$, u_3 already known
$k = 4$, $s_4 = T$
$u_4 = (r_3^o - b_2 \cdot u_2 - a_3 \cdot u_3)/b_3 = [3 - (-1) \cdot (-1.2) - 3 \cdot (-5)]/(-1) = -16.8$

Solution: $\mathbf{u} = [-0.6\ -1.2\ -5\ -16.8]$

In order to investigate the proposed method performance, circuit from Figure 1 is multiplied to several million nodes with same resistors of $1\,\Omega$, so that we have tried to solve tridiagonal system with several million unknowns. Computation time comparison for all 4 methods is given in Figure 7. Computation time of the "raw" version of the proposed method is also given in the same figure. It's observable that for systems with size from 1 to 10 million unknowns, the proposed method solves the systems with practically the same time as methods from [2] and [3], while the Matlab backslash operator performs about 1.5 times faster. Full version of the proposed method is about $10 - 15\%$ slower

TABLE I
MATRIX TYPES USED IN THE NUMERICAL EXPERIMENTS

Matrix type	Description
1	Randomly generated matrix from a uniform distribution on $[-1, 1]$.
2	`gallery('dorr', n, 1e-4)` — Ill-conditioned, diagonally dominant matrix.
3	Randomly generated matrix from a uniform distribution on $[-1, 1]$. The 50^{th} sub-diagonal element is then multiplied by 1×10^{-50}.
4	Main diagonal elements randomly generated from a uniform distribution on $[-1, 1]$. Off-diagonal elements each chosen with 50% probability as either zero or generated randomly from a uniform distribution on $[-1, 1]$.
5	Toeplitz matrix with diagonal elements equal to 1×10^{8}. Off-diagonal elements are randomly generated from a uniform on $[-1, 1]$.
6	Toeplitz matrix with diagonal elements equal to 1×10^{-8}. Off-diagonal elements are randomly generated from a uniform on $[-1, 1]$.
7	`inv(gallery('kms', n, 0.5))` Inverse of a Kac-Murdock-Szegő Toeplitz matrix ($A_{ij} = 0.5^{\lvert i-j \rvert}$).
8	`A = gallery('lesp', n)` A matrix whose eigenvalues are real and smoothly distributed in the interval $[-2n - 3.5, -4.5]$.
9	`A = gallery('randsvd', n, 1e15, 1, 1, 1)` Randomly generated matrix with condition number 1×10^{15} and one large singular value.
10	`A = gallery('randsvd', n, 1e15, 2, 1, 1)` Randomly generated matrix with condition number 1×10^{15} and one small singular value.
11	`A = gallery('randsvd', n, 1e15, 3, 1, 1)` Randomly generated matrix with condition number 1×10^{15} and geometrically distributed singular values.
12	`A = gallery('randsvd', n, 1e15, 4, 1, 1)` Randomly generated matrix with condition number 1×10^{15} and arithmetically distributed singular values.

than the "raw" version due to numerous checks within if statements.

Computation time is measured in 64-bit Matlab ver. R2016b on Intel® Core™ i5–8500 CPU @ 3.00GHz under Linux operating system. A `timeit` function is used which calls the specified method multiple times and returns the median of measurements considering the first-time run costs.

VII. NUMERICAL EXPERIMENTS

Numerical tests were performed using 12 types of non-singular tridiagonal linear systems containing a wide range of difficulty. Several ill-conditioned matrices were chosen as part of a test set in order to compare algorithm robustness. Very roughly it can be expected that for ill-conditioned matrices with large condition number, the rate at which the solution will change with respect to a change in right-hand side vector is large. Thus, the solution is very sensitive even to a small change in the right-hand side vector. The test matrices were taken from the literature [1]. Table I contains a description of each tridiagonal matrix type from the test set.

For cases where test matrices from Matlab matrix gallery were non-symmetric, a symmetrization procedure is applied as in [9]. In a non-symmetric tridiagonal matrix, subdiagonal and superdiagonal vectors are not the same. Denoting them with $\mathbf{c}$ and $\mathbf{d}$ respectively and using a symmetrization procedure, a single vector $\mathbf{b}$ is generated that replaces both of them using the following equation:

$$b_i = \sqrt{c_i d_i}, \quad i = 1, 2, \ldots, n-1,$$

This ensures that original and symmetrized matrix both have the same eigenvalues and approximately the same condition number. The latter is important for the algorithm robustness test.

A system of equations with $n = 100$ was generated for each matrix type where the right-hand side vector consists of ones. The same system of the form $\mathbf{A} \cdot \mathbf{u} = \mathbf{r}$ was solved by each algorithm. Table II shows the relative errors associated for each method calculated as

$$E = \frac{\|\mathbf{A} \cdot \hat{\mathbf{u}} - \mathbf{r}\|_2}{\|\mathbf{r}\|_2},$$

where $\hat{\mathbf{u}}$ is the solution obtained by each solver. Relative errors are given in columns marked 1-4 for the following algorithms: Matlab solver, Thomas algorithm, Gaussian elimination and Proposed method. The last column gives the condition number of matrix $\mathbf{A}$. Figure 8 presents a visual representation of relative errors for all methods.

TABLE II
RELATIVE ERRORS FOR SOLVING TRIDIAGONAL SYSTEMS

Matrix type	Method 1	2	3	4	Condition number
1	6.35×10^{-16}	1.16×10^{-15}	5.85×10^{-15}	1.01×10^{-14}	5.23×10^{2}
2	8.95×10^{-17}	7.45×10^{-17}	8.23×10^{-17}	1.07×10^{-16}	7.84×10^{17}
3	2.72×10^{-16}	7.69×10^{-16}	7.60×10^{-16}	1.30×10^{-14}	2.59×10^{2}
4	1.39×10^{-16}	2.36×10^{-16}	1.50×10^{-16}	1.67×10^{-15}	1.57×10^{2}
5	9.99×10^{-17}	1.17×10^{-16}	9.99×10^{-17}	1.37×10^{-16}	1.00
6	8.17×10^{-15}	1.60×10^{-9}	9.30×10^{-10}	1.24×10^{-9}	5.02×10^{5}
7	1.65×10^{-16}	1.66×10^{-16}	1.65×10^{-16}	3.47×10^{-16}	8.98
8	1.46×10^{-16}	1.46×10^{-16}	1.46×10^{-16}	1.50×10^{-16}	4.47×10^{1}
9	1.41×10^{-4}	8.19×10^{-5}	6.57×10^{-5}	6.32×10^{-4}	1.03×10^{15}
10	2.48×10^{-5}	1.95×10^{-4}	1.56×10^{-3}	1.56×10^{-3}	9.46×10^{14}
11	2.19×10^{-5}	3.21×10^{-5}	4.16×10^{-5}	3.44×10^{-5}	7.77×10^{14}
12	1.68×10^{-3}	2.14×10^{-3}	2.56×10^{-3}	3.00×10^{-3}	9.03×10^{14}

1. Matlab solver, 2. Thomas algorithm, 3. Gaussian elimination, 4. Proposed method

From Table II it can be seen that all algorithms are comparable on a wide range of linear systems. When the matrix $\mathbf{A}$ is well-conditioned (types: 1, 3, 4, 5, 6, 7 and 8), all algorithms present solutions that are obtained with relative errors close to machine precision and they are comparable to each other. On the contrary, significant differences occur when the matrix $\mathbf{A}$ is ill-conditioned (types: 2, 9, 10, 11 and 12), when relative errors are 10 or more magnitudes higher. But even in these cases the proposed algorithm performs no worse than the others.

For example, on types 9-12 all methods perform particularly poorly, including the Matlab backslash command, with an error near 10^{-3}. Actually, due to the ill-conditioning of matrix $\mathbf{A}$, all methods deteriorate in performance nearly failing to solve the tridiagonal linear system. However, their relative errors are within an order of magnitude from each other.

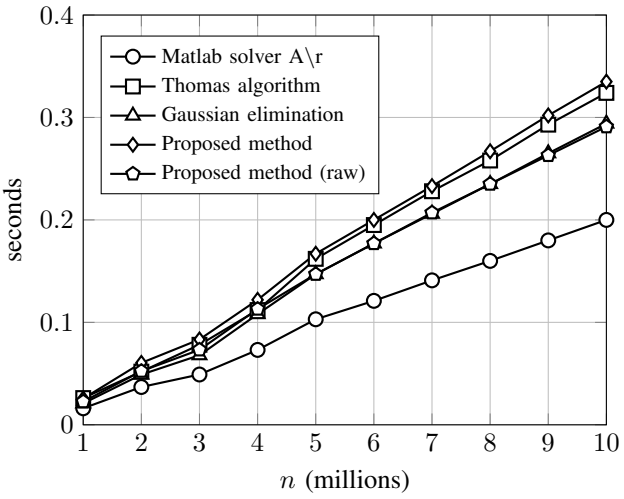

Fig. 7. Time Comparison – All Methods for Different System Size n

VIII. CONCLUSION

In this paper a novel solution method for symmetric tridiagonal systems is presented. The method converts the system of equations into an equivalent circuit with node-voltage equations being the same as in the original system. Afterwards, the circuit is efficiently solved using a procedure tailored for such circuits. The method avoids possible zero divisions exploiting the specific structure of the circuit, which is an equivalent to a pivoting strategy used in other methods.

Tests have shown that the performance of the proposed method is comparable to the Thomas algorithm and Gaussian elimination adapted for tridiagonal systems, as well as the Matlab backslash operator both in speed and robustness.

Procedures of forward and backward sweeps both take only $O(N)$ operations meaning that computation time is linearly proportional to system size. The whole method can be coded very concisely and the possible ill conditions are overcomed without pivoting (equation reordering). Instead of pivoting, a special technique is used, emerged from the solution procedure tailored for circuits with "ladder" structure that are used as an equivalent representation of tridiagonal systems.

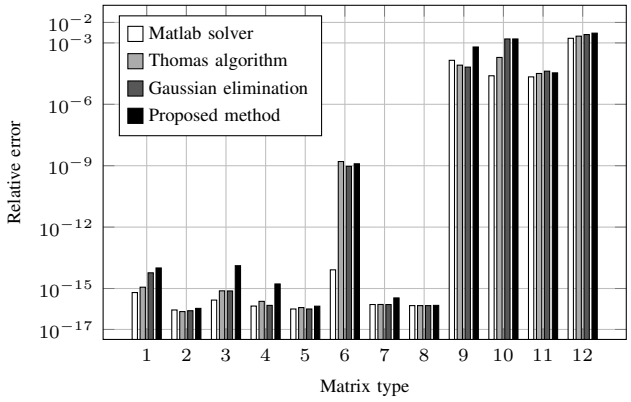

Fig. 8. Relative errors for solving tridiagonal systems

IX. The Code

The code for the proposed method developed in this paper is written in Matlab. It is given as an open-source in a GitHub repository along with all input data in [10]. https://github.com/todorovski-m/tridiagonal

References

[1] J. B. Erway, R. F. Marcia, and J. A. Tyson, "Generalized Diagonal Pivoting Methods for Tridiagonal Systems without Interchanges," *IAENG International Journal of Applied Mathematics*, vol. 40, no. 4, pp269-275, 2010.

[2] D. Kincaid and W. Cheney, "Numerical Analysis – Mathematics of Scientific Computing," ch. 4.3, *American Mathematical Society*, pp179-180, 2002.

[3] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "Numerical Recipes – The Art of Scientific Computing," 3rd ed., ch. 2.4, *Cambridge University Press*, pp56-57, 2007.

[4] D. Rajičić and R. Taleski, "Two novel methods for radial and weakly meshed network analysis," *Electric Power Systems Research*, vol. 48, no. 2, pp79-87, 1998.

[5] C. K. Alexander and M. N. O. Sadiku, "Fundamentals of Electric Circuits," 5th ed., ch. 3.6 Nodal and Mesh Analyses by Inspection, *McGraw-Hill*, pp100-101, 2012.

[6] G. Strang, "Introduction to Linear Algebra," 5th ed., ch. 2.6 Elimination = Factorization: A = LU, *Wellesley-Cambridge Press*, pp97-108, 2016.

[7] C. K. Alexander and M. N. O. Sadiku, "Fundamentals of Electric Circuits," 5th ed., ch. 4.4 Source Transformation, *McGraw-Hill*, pp135-136, 2012.

[8] W. F. Ames, "Numerical Methods for Partial Differential Equations," 3rd ed., ch. 1-8 Finite Elements, *Elsevier*, pp27-33, 1992.

[9] Q. Al-Hassan, "An inverse eigenvalue problem for general tridiagonal matrices," *International Journal of Contemporary Mathematical Sciences*, vol. 4, no. 13-16, pp625-634, 2009.

[10] M. Todorovski, "Matlab code for solving tridiagonal symmetric systems of equations," *GitHub Repository*, 2020. https://github.com/todorovski-m/tridiagonal