Journal of Algorithms and Computation

journal homepage: http://jac.ut.ac.ir

A New Numerical Solution for System of Linear Equations

Iman Shojaei*[1] and Hossein Rahami†[2]

[1]Engineering Optimization Research Group, College of Engineering, University of Tehran
[2]School of Engineering Science, College of Engineering, University of Tehran, Tehran, Iran

ABSTRACT

In this paper we have developed a numerical method for solving system of linear equations through taking advantages of properties of repetitive tridiagonal matrices. A system of linear equations is usually obtained in the final step of many science and engineering problems such as problems involving partial differential equations. In the proposed algorithm, the problem is first solved for repetitive tridiagonal matrices (i.e., system of linear equations) and a closed-from relationship is obtained.

Keyword: linear system of equations, partial differential equations, numerical solution, efficient analysis, repetitive tridiagonal matrices, iterative methods..

ARTICLE INFO

Article history:
Research Paper
Received 11, January 2021
Received in revised form 14, April 2021
Accepted 10 May 2021
Available online 01, June 2021

AMS subject Classification: 65F10

1 Abstract continued

This relationship is then used for solving a general matrix through converting the matrix into a repetitive tridiagonal matrix and a remaining matrix that is moved to the right-hand side of the equation. Therefore, the problem is converted into a repetitive tridiagonal matrix problem where we have a vector of unknowns on the right-hand side (in addition

*shojaei.iman@gmail.com
†Corresponding author: H. Rahami. Email: hrahami@ut.ac.ir

to the left-hand side) of the equation. The problem is solved iteratively by first using an initial guess to define the vector on the right-hand side of the equation and then solving the problem using the closed-from relationship for repetitive tridiagonal matrices. The new obtained solution is then substituted in the right-hand side of the equation and the tridiagonal problem is solved again. This process is carried out iteratively until convergence is achieved. Computational complexity of the method is investigated and efficiency of the method is shown through several examples. As indicated in the examples, one of the advantages of the proposed method is its high rate of convergence in problems where the given matrix includes large off-diagonal entries. In such problems, methods like Jacobi, Gauss-Seidel, and Successive Over-Relaxation will either have a low rate of convergence or be unable to converge.

2 Introduction

The final step of many engineering problems is solving a system of linear equations. In general, such systems can be solved using direct or iterative methods. Direct methods, such as Gaussian elimination or LU decomposition, are not usually used for problems with large and/or sparse matrices as these methods are computationally expensive and require storage of data and high speed computations. Therefore, several iterative methods [1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 15, 18, 20, 21, 22, 24, 28] have been developed that can, in general, be classified as two groups of stationary and non-stationary methods.
Since the goal is to solve the equation $\mathbf{Ax} = \mathbf{b}$, one can rewrite the equation as

$$\mathbf{x} = (\mathbf{I} - \mathbf{A})\mathbf{x} + \mathbf{b}$$

to form the iterative formula $\mathbf{x^{(k)}} = (\mathbf{I} - \mathbf{A})\mathbf{x^{(k-1)}} + \mathbf{b}$. In general, this equation can be written in the form $\mathbf{x^{(k)}} = \mathbf{Bx^{(k-1)}} + \mathbf{c}$ where matrix B and vector c can be updated in each iteration. In stationary methods, matrix B and vector c does not change with iterations and remain unchanged. Specifically, Jacobi and Gaus-Seidel methods are well-known examples of stationary methods that have been widely used in literature to solve systems of linear equations. Jacobi method is a straightforward method with simple interpretation and implementation. The drawback of this method, however, is its relatively low rate of convergence. Gauss-Seidel method is similar to the Jacobi method, but uses updated values in each iteration of the algorithm leading to a faster convergence [4, 17, 26, 29]. To further improve the rate of convergence, Successive Over-Relaxation (SOR) algorithm has been suggested that is obtained via adding a parameter ω to the Gauss-Seidel method.
The general form of SOR algorithm is

$$\begin{aligned} x_i^{(k)} =& x_i^{(k-1)} + \omega \frac{R_i^{(k-1)}}{a_{i,i}} \quad i = 1:n \\ R_i^{(k-1)} =& b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{(k)} - \sum_{j=i}^{n} a_{i,j} x_j^{(k-1)} \end{aligned}$$

which is simplified to the Gauss-Seidel method for $\omega = 1$. For the case of $0 < \omega < 1$ (so-called under relaxation), the method can prevent divergence of the solution and damps potential oscillations during iterations. For the case of $1 < \omega < 2$ (so-called over relaxation), the method increases the rate of convergence. Finally, when $\omega > 2$, the solution will diverge. Likewise, Ehrlic (1981) introduced a method, called Ad-Hoc SOR, wherein they used a new approach for updating variables in order to improve the rate of convergence [12].

In non-stationary methods computations involve information that change in each iteration of the solution [13, 14, 23]. Specifically, constant values of the algorithm in each iteration of the solution is updated with the goal of reducing the norm of error. Conjugate Gradient is a well-known example of such methods. Performance and comparison between stationary versus non-stationary methods for solving singular problems can be found in [16].

Solving a system of linear equations using the methods above is challenging when the coefficient matrix is Stieltjes. Stieltjes matrix is a positive definite matrix wherein the off-diagonal entries are negative and diagonal entries are positive. To solve such matrices AGMG and CG-AMG multigrid algorithms have been suggested. Todini and Pilati (1987) used a global Gradient algorithm for hydraulic analysis of pipe networks [25]. Also, Webster (1998) used an efficient multigrid method to analyze fluid flow in pipe networks [27].

In this paper we have proposed a numerical method for solving system of linear equations through taking advantages of properties of repetitive tridiagonal matrices. The method is computationally efficient, $O(n^2)$, and its rate of convergence is high in problems involving matrices of large off-diagonal entries where methods like Jacobi, Gauss-Seidel, and SOR will either have a low rate of convergence or be unable to converge. There are several methods in literature for solving a system of linear equations. Specifically, we have evaluated the performance of our algorithm against three established algorithms in literature: Jacobi, Gauss-Seidel, and SOR methods. Several practical examples including partial differential equation (PDE) problems from mathematical finance have been solved using the proposed method.

3 Jacobi method for solving system of linear equations $Au = c$

Consider matrix $\mathbf{A}$ below

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \dots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \tag{1}$$

Matrix $\mathbf{A}$ is decomposed into a diagonal matrix $\mathbf{D}$ and the remainder $\mathbf{R}$

$$\mathbf{A} = \begin{bmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \dots & \\ & & & a_{nn} \end{bmatrix} + \begin{bmatrix} 0 & a_{12} & \dots & a_{1n} \\ a_{21} & 0 & \dots & a_{2n} \\ \vdots & \vdots & \dots & \vdots \\ a_{n1} & a_{n2} & \dots & 0 \end{bmatrix} = \mathbf{D} + \mathbf{R}. \tag{2}$$

We will have

$$(\mathbf{D} + \mathbf{R})\mathbf{u} = \mathbf{c} \tag{3}$$

and

$$\mathbf{D}\mathbf{u} = \mathbf{c} - \mathbf{R}\mathbf{u} \tag{4}$$

Setting an initial approximation as $\mathbf{u}^0 = \mathbf{D}^{-1}c$ and obtaining a solution

$$\mathbf{u}^1 = \mathbf{D}^{-1}(\mathbf{c} - \mathbf{R}\mathbf{u}^0) \tag{5}$$

Repeating the procedure leads to an iterative algorithm from which the solution is obtained

$$\mathbf{u}^{n+1} = \mathbf{D}^{-1}(\mathbf{c} - \mathbf{R}\mathbf{u}^n) \quad \text{and} \quad n = 0, 1, 2, \dots \tag{6}$$

The condition for convergence of Jacobi method (and any other iterative method) is when the spectral radius of the iteration matrix ($\mathbf{D}^{-1}\mathbf{R}$) is less than 1. One sufficient condition for convergence of the method is that matrix $\mathbf{A}$ is diagonally dominant. Here we presented the algorithm for Jacobi method as the simplest iterative method for system of linear equations. Readers are referred to the relevant literature for detail and formulation of other iterative methods such as Gauss-Seidel and SOR.

4 Closed-Form Solution of Mx = f when M is a tridiagonal matrix

Eigenvalues and eigenvectors of a tridiagonal matrix, of dimension $N-1$, of the form

$$\mathbf{M} = \begin{bmatrix} b & c & & \dots & 0 \\ a & b & c & & \vdots \\ & a & \ddots & c & \\ \dots & & a & b & c \\ 0 & \dots & & a & b \end{bmatrix} \tag{7}$$

are calculated [19, 30] using

$$\begin{aligned} \lambda_n &= b + 2\sqrt{ac}\cos\frac{n\pi}{N} \quad \text{and} \quad n = 1, 2, \dots, N-1 \\ v_j^n &= \left(\frac{a}{c}\right)^{j-1}\sin\frac{nj\pi}{N} \quad \text{and} \quad n = 1, 2, \dots, N-1 \\ v^n &= [v_1^n, v_2^n, \dots, v_{N-1}^n]^t. \end{aligned} \tag{8}$$

And if $a = c$, we will have

$$\begin{aligned} \lambda_n =& b + 2a\cos\frac{n\pi}{N} \quad \text{and} \quad n = 1, 2, \ldots, N-1 \\ v_j^n =& \sin\frac{nj\pi}{N} \quad \text{and} \quad n = 1, 2, \ldots, N-1 \\ v^n =& [v_1^n, v_2^n, \ldots, v_{N-1}^n]^t. \end{aligned} \tag{9}$$

Since the matrix is symmetric (of dimension $N-1$), its eigenvectors $(\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^{N-1})$ provide an orthogonal basis for $N-1$ space. Therefore, we can expand $\mathbf{x}$ and $\mathbf{f}$ in terms of $(\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^{N-1})$ basis:

$$\mathbf{x} = \sum_{i=1}^{N-1} x_i \mathbf{v}^i \quad \text{and} \quad \mathbf{f} = \sum_{i=1}^{N-1} b_i \mathbf{v}^i \tag{10}$$

where f_is are known (i.e., can readily be computed)

$$f_i = (\mathbf{v}^i)^t \mathbf{f} \tag{11}$$

and x_is are our unknowns that should be computed through substitution in $\mathbf{M}\mathbf{x} = \mathbf{f}$

$$\mathbf{M} \sum_{i=1}^{N-1} x_i \mathbf{v}^i = \sum_{i=1}^{N-1} f_i \mathbf{v}^i \tag{12}$$

where we can write

$$\mathbf{M} \sum_{i=1}^{N-1} x_i \mathbf{v}^i = \sum_{i=1}^{N-1} x_i \mathbf{M}\mathbf{v}^i = \sum_{i=1}^{N-1} x_i \lambda_i \mathbf{v}^i \tag{13}$$

We can now re-express Eq. (12) as

$$\sum_{i=1}^{N-1} \lambda_i x_i \mathbf{v}^i = \sum_{i=1}^{N-1} f_i \mathbf{v}^i \tag{14}$$

Because $\mathbf{v}^i$s are linearly independent (they form a basis), we will have

$$\lambda_i x_i = f_i \rightarrow x_i = \frac{f_i}{\lambda_i} \quad \text{and} \quad n = 1, 2, \ldots, N-1. \tag{15}$$

Therefore,

$$x = \sum_{i=1}^{N-1} \frac{f_i}{\lambda_i} \mathbf{v}^i = \sum_{i=1}^{N-1} \mathbf{v}^i \frac{f_i}{\lambda_i} = \sum_{i=1}^{N-1} \frac{\mathbf{v}^i (\mathbf{v}^i)^t}{\lambda_i} \mathbf{f} \tag{16}$$

Finally,

$$\mathbf{x} = \sum_{i=1}^{N-1} \frac{[\sin\frac{i\pi}{N}, \sin\frac{2i\pi}{N}, \ldots, \sin\frac{(N-1)i\pi}{N}]^t [\sin\frac{i\pi}{N}, \sin\frac{2i\pi}{N}, \ldots, \sin\frac{(N-1)i\pi}{N}]}{b + 2a\cos\frac{i\pi}{N}} \mathbf{f} \tag{17}$$

5 A numerical method for the solution of $Au = c$ when A is an arbitrary matrix

Here, we want to get advantages of the closed form solution for tridiagonal matrices to develop an efficient numerical method for a linear system of equations $\mathbf{Au} = \mathbf{c}$ when $\mathbf{A}$ is arbitrary. Consider matrix $\mathbf{A}$ below

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \dots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \tag{18}$$

Lets decompose matrix $\mathbf{A}$ into a tridiagonal matrix, $\mathbf{A}_1$, and the remaining terms, $\mathbf{A}_2$. b and a entries in tridiagonal matrix $\mathbf{A}_1$ are chosen to be, respectively, the average of diagonal entries (i.e., $a_{11}, \dots, a_{nn}$) and the average of upper and lower neighboring-diagonals entries (i.e., $a_{12}, \dots, a_{n-1,n} \dots, a_{21}, \dots, a_{n,n-1}$).

$$\mathbf{A} = \begin{bmatrix} b & a & & \\ a & b & \dots & \\ & \dots & \dots & a \\ & & a & b \end{bmatrix} + \begin{bmatrix} a_{11}-b & a_{12}-a & \dots & a_{1n} \\ a_{21}-a & a_{22}-b & \dots & a_{2n} \\ \vdots & \vdots & \dots & \vdots \\ a_{n1}-b & a_{n2} & \dots & a_{nn}-b \end{bmatrix} = \mathbf{A}_1 + \mathbf{A}_2. \tag{19}$$

We will have

$$(\mathbf{A}_1 + \mathbf{A}_2)\mathbf{u} = \mathbf{c} \tag{20}$$

and

$$\mathbf{A}_1\mathbf{u} = \mathbf{c} - \mathbf{A}_2\mathbf{u}. \tag{21}$$

Replacing $\mathbf{A}_2$ by $\mathbf{A} - \mathbf{A}_1$

$$\mathbf{A}_1\mathbf{u} = \mathbf{c} - (\mathbf{A} - \mathbf{A}_1)\mathbf{u} \tag{22}$$

Lets initialize $\mathbf{u}$ using the following approximation

$$\mathbf{A}_1\mathbf{u}^0 = \mathbf{c} \tag{23}$$

Since $\mathbf{A}_1$ is tridiagonal, according to previous section we will have

$$\mathbf{u}^0 = \sum_{i=1}^{N-1} \frac{[\sin\frac{i\pi}{N}, \sin\frac{2i\pi}{N}, \dots, \sin\frac{(N-1)i\pi}{N}]^t [\sin\frac{i\pi}{N}, \sin\frac{2i\pi}{N}, \dots, \sin\frac{(N-1)i\pi}{N}]}{b + 2a\cos\frac{i\pi}{N}} c \tag{24}$$

Now, we can improve the results through:

$$\mathbf{A}_1\mathbf{u}^1 = \mathbf{c} - (\mathbf{A} - \mathbf{A}_1)\mathbf{u}^0 \tag{25}$$

And in general we can use the following iterative equation to solve the problem

$$\mathbf{A}_1\mathbf{u}^(n+1) = \mathbf{c} - (\mathbf{A} - \mathbf{A}_1)\mathbf{u}^n \tag{26}$$

Since $\mathbf{A}_1$ is tridiagonal, we can write the equation as follows

$$\mathbf{u}^{n+1} = \sum_{i=1}^{N-1} \frac{[\sin\frac{i\pi}{N}, \sin\frac{2i\pi}{N}, \ldots, \sin\frac{(N-1)i\pi}{N}]^t [\sin\frac{i\pi}{N}, \sin\frac{2i\pi}{N}, \ldots, \sin\frac{(N-1)i\pi}{N}]}{b + 2a\cos\frac{i\pi}{N}} [c - (\mathbf{A} - \mathbf{A}_1)\mathbf{u}^n] \quad (27)$$

As such, an iterative formula for solving a general linear system of equations is obtained through properties of tridiagonal matrices. The condition for convergence of the method is when the spectral radius of the iteration matrix, $\mathbf{A}_1^{-1}(\mathbf{A} - \mathbf{A}_1)$, is less than 1.
We did not formally study the potential sufficient conditions for convergence of the proposed method. However, through several examples we have shown that in many cases that Jacobi, Gauss-Seidel, and SOR methods do not converge the proposed method converges.

6 Computational complexity of the method

For one iteration of the method, the dominant computational complexity is $O(n^2)$:

- $(\mathbf{A} - \mathbf{A}_1)\mathbf{u}^n = \mathbf{MV}$: Multiplication of a matrix ($\mathbf{M}$) and a vector ($\mathbf{V}$), of the complexity $O(n^2)$.

For sigma calculations we will have

- $[\sin\frac{i\pi}{N}, \sin\frac{2i\pi}{N}, \ldots, \sin\frac{(N-1)i\pi}{N}][c - (\mathbf{A} - \mathbf{A}_1)\mathbf{u}^n] = \mathbf{P}^t\mathbf{V}$. Multiplication of a row vector ($\mathbf{P}^t$) and a column vector ($\mathbf{V}$), of the complexity $O(n)$. The outcome here would be a scaler (S).

- $[\sin\frac{i\pi}{N}, \sin\frac{2i\pi}{N}, \ldots, \sin\frac{(N-1)i\pi}{N}]^t S = \mathbf{P}S$: Multiplication of a column vector ($\mathbf{P}$) and a scaler (S), of the complexity $O(n)$.

- n times calculation of the two steps above (because of sigma) leads to a computational complexity of $O(n^2)$.

Therefore, the dominant computational complexity of the method for one iteration is $O(n^2)$. If m iterations are required to achieve a desired accuracy, the computational complexity of the method would be $O(mn^2)$. However, since the required number of iterations to achieve convergence is much smaller than the dimension of matrix A (i.e., $m << n$), computational complexity of the method would be $O(n^2)$.

7 Examples

Example 1. Consider the following linear system of equations taken from:

$$\mathbf{A} = \begin{bmatrix} 10 & -1 & 2 & 0 \\ -1 & 11 & -1 & 3 \\ 2 & -1 & 10 & -1 \\ 0 & 3 & -1 & 8 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 6 \\ 25 \\ -11 \\ 15 \end{bmatrix}$$

The system of equations was solved using matrix inversion (i.e., the exact solution), the proposed method, and three iterative methods (Table 1). Matrix $\mathbf{A}_1$ in the proposed method and matrix $\mathbf{D}$ in Jacobi method are as follows:

$$\mathbf{A}_1 = \begin{bmatrix} 9.75 & -1 & 0 & 0 \\ -1 & 9.75 & -1 & 0 \\ 0 & -1 & 9.75 & -1 \\ 0 & 0 & -1 & 9.75 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 11 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 8 \end{bmatrix}$$

Table 1: Comparison between the exact solution, the proposed method, and three iterative methods for solving a matrix equation of 4 unknowns.

$\mathbf{x} = \mathbf{A^{-1}b}$	5 iterations	5 iterations	5 iterations	5 iterations
Exact	Proposed	Jacobi	GaussSeidel	SOR ($\omega = 1.2$)
1.0000	1.0003	0.9981	1.0001	1.0021
2.0000	2.0008	2.0023	2.0000	1.9993
-1.0000	-0.9995	-1.0019	-1.0000	-0.9999
1.0000	1.0008	1.0035	0.9999	1.0009
$\|Exact - Iterative\|$	0.0012	0.0050	0.0001	0.0023

It is observed that the convergence rate of the proposed method is higher than that of the Jacobi and SOR methods but a little bit lower than that of the Gauss-Seidel method.

Example 2. BlackScholes equation
BlackScholes equation is a partial differential equation in mathematical finance that describes the price of the option over time:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 s^2 \frac{\partial^2 V}{\partial s^2} + rs\frac{\partial V}{\partial s} - rV = 0,$$

or in a more general form we can write:

$$\frac{\partial V}{\partial t} + a(s,t)\frac{\partial^2 V}{\partial s^2} + b(s,t)\frac{\partial V}{\partial s} + c(s,t)V = 0.$$

Using Crank-Nicholson finite difference method we will have:

$$\frac{V_{n,j+1} - V_{n,j}}{\Delta t} + \frac{a_{n,j+1}}{2}\left(\frac{V_{n+1,j+1} - 2V_{n,j+1} + V_{n-1,j+1}}{\Delta s^2}\right) + \frac{a_{n,j}}{2}\left(\frac{V_{n+1,j} - 2V_{n,j} + V_{n-1,j}}{\Delta s^2}\right)$$
$$+ \frac{b_{n,j+1}}{2}\left(\frac{V_{n+1,j+1} - V_{n-1,j+1}}{2\Delta s}\right) + \frac{b_{n,j}}{2}\left(\frac{V_{n+1,j} - V_{n-1,j}}{2\Delta s}\right) + \frac{1}{2}c_{n,j+1}V_{n,j+1} + \frac{1}{2}c_{n,j}V_{n,j} = 0.$$

We can rewrite the equation as

$$A_{n,j+1}V_{n-1,j+1} + (1+B_{n,j+1})V_{n,j+1} + C_{n,j+1}V_{n+1,j+1} = -A_{n,j}V_{n-1,j} + (1+B_(n,j))V_{n,j} - C_{n,j}V_{n+1,j},$$

where

$$\begin{aligned} A_{n,j} =& \frac{1}{2}v_1 a_{n,j} - \frac{1}{4}v_2 b_{n,j} \\ B_{n,j} =& -v_1 a_{n,j} + \frac{1}{2}\Delta t c_{n,j}, \\ C_{n,j} =& \frac{1}{2}v_1 a_{n,j} + \frac{1}{4}v_2 b_{n,j}, \\ v_1 =& \frac{\Delta t}{\Delta s^2} \quad \text{and} \quad v_2 = \frac{\Delta t}{\Delta s}. \end{aligned}$$

These equations hold for $n = 1, 2, , N-1$. Boundary conditions provide two additional equations. In a matrix form and for boundary conditions of $V(0,t) = 0$ and $V(s_{\max}, t) = s_{\max} - Ee^{-r(T-t)}$, we will have

$$\begin{aligned} &V_{0,j} = 0, \\ &V_{N,j} = N\Delta s - Ee^{-r(j)\Delta t}, \\ &V_{0,j+1} = 0, \\ &V_{N,j+1} = N\Delta s - Ee^{-r(j+1)t}. \end{aligned}$$

$$\mathbf{M}^{\mathbf{L}}_{j+1}\mathbf{V}_{j+1} + \mathbf{r}^{\mathbf{L}}_{j+1} = \mathbf{M}^{\mathbf{R}}_{j}\mathbf{V}_{j} + \mathbf{r}^{\mathbf{R}}_{j}$$

$$\mathbf{M}^{\mathbf{L}}_{j+1}\mathbf{V}_{j+1} + \mathbf{r}^{\mathbf{L}}_{j+1} = \begin{bmatrix} 1+B_{1,j+1} & C_{1,j+1} & & & \\ A_{2,j+1} & 1+B_{2,j+1} & C_{2,j+1} & & \\ & A_{3,j+1} & \cdots & \cdots & \\ & & \cdots & 1+B_{N-2,j+1} & C_{N-2,j+1} \\ & & & A_{N-1,j+1} & 1+B_{N-1,j+1} \end{bmatrix} \begin{bmatrix} V_{1,j+1} \\ V_{2,j+1} \\ \vdots \\ V_{N-2,j+1} \\ V_{N-1,j+1} \end{bmatrix} + \begin{bmatrix} A_{1,j+1}V_{0,j+1} \\ 0 \\ \vdots \\ 0 \\ C_{N-1,j+1}V_{N,j+1} \end{bmatrix}$$

$$\mathbf{M}^{\mathbf{R}}_{j}\mathbf{V}_{j} + \mathbf{r}^{\mathbf{R}}_{j} = \begin{bmatrix} 1-B_{1,j} & -C_{1,j} & & & \\ -A_{2,j} & 1-B_{2,j} & -C_{2,j} & & \\ & -A_{3,j} & \cdots & \cdots & \\ & & \cdots & 1-B_{N-2,j} & -C_{N-2,j} \\ & & & -A_{N-1,j} & 1-B_{N-1,j} \end{bmatrix} \begin{bmatrix} V_{1,j} \\ V_{2,j} \\ \vdots \\ V_{N-2,j} \\ V_{N-1,j} \end{bmatrix} + \begin{bmatrix} A_{1,j+1}\mathbf{V}_{0,j+1} \\ 0 \\ \vdots \\ 0 \\ C_{N-1,j+1}\mathbf{V}_{N,j+1} \end{bmatrix}$$

$$\mathbf{M}^{\mathbf{L}}_{j+1}\mathbf{V}_{j+1} = \mathbf{M}^{\mathbf{R}}_{j}\mathbf{V}_{j} + \mathbf{r}^{\mathbf{R}}_{j} - \mathbf{r}^{\mathbf{L}}_{j+1}$$

The matrix and vectors on the right hand side of the equation are known so we can write the equation as

$$\mathbf{M}^{\mathbf{L}}_{j+1}\mathbf{V}_{j+1} = \mathbf{b}$$

Now, using the known matrix $\mathbf{M}^{\mathbf{L}}_{j+1}$ and vector b, we can find vector $\mathbf{V}_{j+1}$. The obtained vector is then used on the right hand side of the equation in the next iteration to find vector $\mathbf{V}$ in subsequent step.

Consider the following parameters for the problem:

$\sigma = 0.25$ Volatility of the stock

$r = 0.2$ Interest rate

$S_{max} = 20$ Maximum stock price

$S_{min} = 0$ Minimum stock price

$T = 1$ Maturation of contract
$E = 10$ Exercise price of the underlying
$M = 1600$ Number of time points
$N = 160$ Number of stock price points
$\Delta t = \frac{T}{M} = 0.000625$ Time step
$\Delta s = \frac{S_{max} - S_{min}}{N} = 0.125$ Price step
Starting with $j = 0$, we will have

$$\mathbf{M}_1^{\mathbf{L}}\mathbf{V}_1 = \left(\begin{bmatrix} 1.0000 & 0.0000 & & & & & \\ 0.0000 & 1.0000 & 0.0000 & & & & \\ & 0.0000 & \dots & \dots & & & \\ & & \dots & 0.5186 & 0.2358 & & \\ & & & 0.2487 & 0.5125 & 0.2389 & \\ & & & & 0.2519 & 0.5063 \end{bmatrix}_{159\times159} \mathbf{V}_1 \right) = \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ \dots \\ 9.75 \\ 7.39 \end{bmatrix}_{159\times1}$$

The linear system of equations was solved using the three iterative methods, the proposed method, and matrix inversion (i.e., the exact solution). Matrix $\mathbf{A}_1$ in the proposed method and matrix D in Jacobi method are as follows:

$$\mathbf{A}_1 = \begin{bmatrix} 0.8339 & 0.0828 & & & & & \\ 0.0828 & 0.8339 & 0.0828 & & & & \\ & 0.0828 & \dots & \dots & & & \\ & & \dots & 0.8339 & 0.0828 & & \\ & & & & 0.8339 & 0.0828 & \\ & & & & 0.0828 & 0.8339 \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} 1.0000 & & & & & & \\ & 1.0000 & & & & & \\ & & \dots & & & & \\ & & & 0.5186 & & & \\ & & & & 0.5124 & & \\ & & & & & 0.5063 \end{bmatrix}$$

Vector $\mathbf{V}_1$ of dimension 159×1 was obtained as (Table 2):
It can be observed that the difference between exact and proposed solutions (the norm of difference between the two vectors) is smaller than 0.1 after 8 iterations (Table 2, Fig. 1). The proposed method outperformed Jacobi and Gauss-Seidel methods but had close performance to that of SOR method. To achieve almost same accuracy using Jacobi and Gauss-Seidel methods, we needed 34 and 10 iterations, respectively (Table 2, Fig. 1). In this problem $\omega = 1.2$ was the optimal value of ω in SOR algorithm. It is notable that although SOR algorithm had the same performance as the proposed method, adjusting ω in this algorithm to achieve the best performance requires trial and error leading to several additional iterations.
Now with $\mathbf{V}_1$ in hand, we can continue with the solution to calculate $\mathbf{V}_2$ in the next time step. The procedure is reaped until all unknowns are found ($j = 0, 1, , M = 1600$). This means that to solve the problem using a Jacobi and Gauss-Seidel methods, we will need

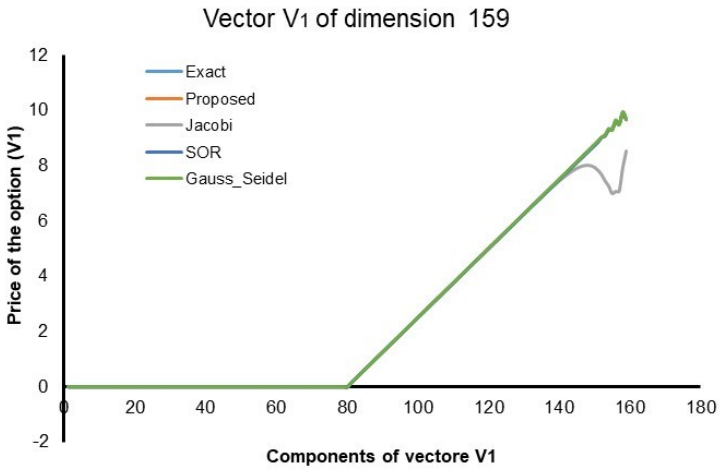

Figure 1: Performance of the proposed and SOR methods after 8 iterations were very close to the exact solution (norm of error ~ 0.1). Gauss-Seidel method reached the same accuracy after 10 iterations whereas Jacobi method needed $\sim$ 34 iterations to become comparable with the exact solution. The indicated results in the figure are for 8 iterations in all methods.

Table 2: Comparison between the exact solution, the proposed method, and three iterative methods for solving a matrix equation of 159 unknowns. After 8 iterations the proposed method outperformed Jacobi and Gauss-Seidel methods but had close performance to that of SOR method. Jacobi and Gauss-Seidel methods reached the desired accuracy after 34 and 10 iterations, respectively. In this problem $\omega = 1.2$ was the optimal value of ω in SOR algorithm. Although SOR algorithm had the same performance as the proposed method, adjusting ω in this algorithm required trial and error leading to several additional iterations.

$\mathbf{V_1} = (\mathbf{M_1^L})^{-1}\mathbf{b}$	8 iterations	8 iterations	8 iterations	8 iterations	34 iterations	10 iterations
Exact	Proposed	Jacobi	GaussSeidel	SOR (=1.2)	Jacobi	GaussSeidel
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
⋮	⋮	⋮	⋮	⋮	⋮	⋮
0.3762	0.3762	0.3762	0.3762	0.3762	0.3762	0.3762
0.5012	0.5012	0.5012	0.5012	0.5012	0.5012	0.5012
0.6262	0.6262	0.6262	0.6262	0.6262	0.6262	0.6262
⋮	⋮	⋮	⋮	⋮	⋮	⋮
9.5474	9.4974	7.0627	9.4623	9.5196	9.5040	9.5051
9.8749	9.9234	7.9059	9.9373	9.8926	9.8395	9.9056
9.6928	9.6620	8.5132	9.6618	9.6851	9.6717	9.6776
$\|Exact - Iterative\|$	0.0914	5.8903	0.1794	0.1082	0.0938	0.0986

$\sim 341600 = 54,400$ and $\sim 101600 = 16,000$ iterations, whereas to solve the problem with

the same accuracy using the proposed method we will need $\sim 81600 = 12,800$ iterations.

Example 3. Another advantage of the proposed method is its power in solving problems with large off-diagonal entries. In such cases Jacobi, Gauss-Seidel, and SOR methods were either slow or unable to converge.
Consider a coefficient matrix $\mathbf{M}$ with relatively large off-diagonal entries above and below the main diagonal and consider same matrix $\mathbf{b}$ as in previous example:

$$\mathbf{M} = \begin{bmatrix} 2.0500 & 0.5000 & & & & & \\ 0.5001 & 2.0500 & 0.5000 & & & & \\ & 0.5002 & \ddots & \ddots & & & \\ & & \ddots & 1.5686 & 0.7358 & & \\ & & & 0.7487 & 1.5625 & 0.7389 & \\ & & & & 0.7519 & 1.5563 & \end{bmatrix}_{159\times 159} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 9.75 \\ 7.39 \end{bmatrix}_{159\times 1}$$

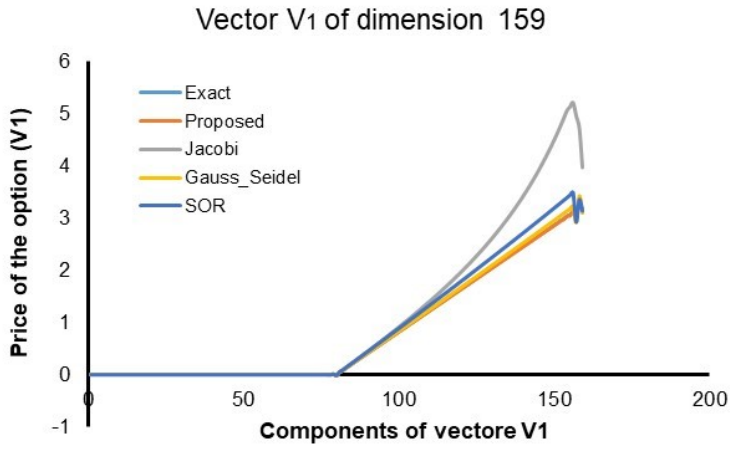

Figure 2: : In a problem with relatively large off-diagonal entries the norm of error for the proposed solution was ~ 0.02 after 3 iterations (the exact and proposed solutions are not distinguishable visually in the figure). Jacobi, Gauss-Seidel, and SOR methods needed 55, 14, and 11 iterations respectively to achieve the same accuracy. The indicated results in the figure are for 3 iterations in all methods.

It can be observed that the convergence rate of Jacobi, Gauss-Seidel, and SOR methods is lower than the proposed method (Table 3, Fig. 2). Now, lets further increase the value

Table 3: Comparison between the exact solution, the proposed method, and three iterative methods for a problem with relatively large off-diagonal entries. The proposed method reached the error of ~ 0.02 after 3 iterations whereas Jacobi, Gauss-Seidel, and SOR methods needed 55, 14, and 11 iterations respectively to achieve the same accuracy. In this problem $\omega = 1.2$ was the optimal value of ω in SOR algorithm.

$\mathbf{V_1 = M^{-1}b}$	3 iterations	3 iterations	3 iterations	3 iterations	55 iterations	14 iterations	11 iterations
Exact	Proposed	Jacobi	GaussSeidel	SOR ($\omega = 1.2$)	Jacobi	GaussSeidel	SOR ($\omega = 1.2$)
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0.7379	0.7379	0.8159	0.7505	0.8013	0.7379	0.7379	0.7380
0.7789	0.7789	0.8634	0.7922	0.8457	0.7789	0.7789	0.7790
0.8199	0.8199	0.9112	0.8340	0.8902	0.8199	0.8199	0.8200
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
3.1328	3.1215	4.9600	2.9120	2.9274	3.1402	3.1329	3.1279
3.2304	3.2404	4.7282	3.4229	3.3536	3.2359	3.2304	3.2335
3.1907	3.1847	3.9699	3.0977	3.1399	3.1936	3.1907	3.1893
$\|Exact - Iterative\|$	0.0217	7.4424	0.5266	1.6869	0.0238	0.0217	0.0225

of off-diagonal entries:

$$\mathbf{M} = \begin{bmatrix} 2.0500 & 0.5000 & & & & & \\ 0.5001 & 2.0500 & 0.5000 & & & & \\ & 0.5002 & \dots & \dots & & & \\ & & \dots & 1.5686 & 0.7358 & & \\ & & & 0.7487 & 1.5625 & 0.7389 & \\ & & & & 0.7519 & 1.5563 & \end{bmatrix}_{159\times 159}$$

It can be observed that while in the proposed method the error after 15 iterations is less than 0.06, the other three methods cannot converge. In the SOR method, all other values of $0 < \omega < 2$ also lead to divergence of the solution.

8 Discussion and Conclusions

There are several methods to solve a system of linear equations. In general, these methods can be classified as direct and iterative methods. Given the computational demand of direct methods, iterative methods are more often used in practice. One of simplest iterative methods used to solve a system of linear equations is Jacobi method. This method, however, has a slow rate of convergence and may also diverge in many problems with non-dominant diagonals of coefficient matrices. Therefore, several algorithms (Gauss-Seidel and SOR amongst others) have been developed to improve the efficiency of Jacobi method. These methods, however, have their own advantages and disadvantages so that we still need to develop new efficient methods for many challenging problems.
In this paper a new iterative method for system of linear equations was presented. We first developed a closed-form solution for repetitive tridiagonal matrices. Then, for a given

Table 4: Comparison between the exact solution, the proposed method, and three iterative methods for a problem with dominant off-diagonal entries. The error for the proposed solution is ~ 0.05 after 15 iterations. The iterative methods could not converge. In the SOR method, all other values of $0 < \omega < 2$ also lead to divergence of the solution.

$\mathbf{V} = \mathbf{M^{-1}b}$	15 iterations	15 iterations	15 iterations	15 iterations
Exact	Proposed	Jacobi	GaussSeidel	SOR ($\omega = 1.2$)
0.9948	1.0002	0	0	0
-0.5098	-0.5117	0	0	0
-0.7335	-0.7383	0	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0.8424	0.8380	0	0	0
-0.7934	-0.7886	0	0	0
-0.4439	-0.4413	0	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
1.6268	1.6300	$1.2608e+12$	$2.2481e+54$	$3.8421e+62$
1.7270	1.7315	$9.0905e+11$	$-7.2086e+54$	$-1.4778e+63$
0.0330	0.0266	$4.8016e+11$	$1.9694e+55$	$4.8308e+63$
$\|Exact - Iterative\|$	0.0569	Diverged	Diverged	Diverged

arbitrary matrix, it was decomposed into a tridiagonal matrix and the remainder. The remainder was added to the right-hand side of the equation such that the vector of unknowns was generated on both sides of the equation. As such, a typical iterative formulation was obtained where the solution was achieved through iterations over the closed-form solution of tridiagonal matrices. One of the advantages of the proposed method is its high rate of convergence without additional computational cost (the computational complexity of the method is $O(n^2)$). Furthermore, as opposed to many iterative algorithms developed to improve the convergence rate of Jacobi method, the proposed method does not need any additional parameters to be adjusted. For instance, in SOR algorithm there exists a parameter adjustment of which requires trial and error which adds additional cost to the problem. As indicated in several examples, the proposed algorithm is efficient in tackling problems involving matrices of large off-diagonal entries whereas methods like Jacobi, Gauss-Seidel, and SOR either have a slow rate of convergence or are unable to converge in such problems. This feature of the algorithm is because of taking directly the off-diagonal entries into account when formulating the tridiagonal matrices.

References

[1] Abolpour Mofrad, A., Sadeghi, M.R., Panario, D., Solving sparse linear systems of equations over finite fields using bit-flipping algorithm, Linear Algebra and its Applications, 439(7) (2013), 1815-1824.

[2] Ahamed, A.K.Ch., and Magouls, F., Efficient implementation of Jacobi iterative method for large sparse linear systems on graphic processing units, The Journal of Supercomputing, 73(8) (2017) 3411-3432.

[3] Alyahya, H., Mehmood, R., and Katib, I., Parallel Iterative Solution of Large Sparse Linear Equation Systems on the Intel MIC Architecture, Smart Infrastructure and Applications, (2019) 377-407.

[4] Barrett, R., Berry, M., Chan, T.F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and Van der Vorst, H., Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd ed. Philadelphia, PA: SIAM, (1994).

[5] Bhrawy, A.H., An efficient Jacobi pseudospectral approximation for nonlinear complex generalized Zakharov system, Appl. Math. Comput., 247 (2014) 30-46.

[6] Bhrawy, A.H., and Zaky, M.A., A method based on the Jacobi tau approximation for solving multi-term time-space fractional partial differential equations, J. Comput. Phys., 281 (2015) 876-895.

[7] Bohacek, J., Kharicha, A., Ludwig, A., Wu, M., Holzmann, T., and Karimi-Sibakib, E., A GPU solver for symmetric positive-definite matrices vs. traditional codes, Computers & Mathematics with Applications, 78(9) (2019) 2933-2943.

[8] Dias da Cunha, R., and Hopkins, T., The Parallel Iterative Methods (PIM) package for the solution of systems of linear equations on parallel computers, Applied Numerical Mathematics, 19(1-2) (1995) 33-50.

[9] Doha, E.H., Bhrawy, A.H., and Ezz-Eldien, S.S., A new Jacobi operational matrix: An application for solving fractional differential equations, Applied Mathematical Modelling, 36 (2012) 4931-4943.

[10] Doha, E.H., Bhrawy, A.H., and Ezz-Eldien, S.S., Efficient Chebyshev spectral methods for solving multi-term fractional orders differential equations, Applied Mathematical Modelling, 35(12) (2011) 5662-5672.

[11] Doha, E.H., Bhrawy, A.H., and Hafez, R.M., A Jacobi-Jacobi dual-Petrov-Galerkin method for third- and fifth-order differential equations, Math. Comput. Modell., 53 (2011) 1820-1832.

[12] Ehrlich, L.W., An Ad-Hoc SOR method, J. Comput. Phys., 44(1) (1981) 31-45.

[13] Golub, G., and O'Leary, D., Some history of the conjugate gradient and Lanczos methods, SIAM Rev., 31 (1989) 50-102.

[14] Golub, G., and Van Loan, C., Matrix Computations, second edition, The Johns Hopkins University Press, Baltimore, (1989).

[15] Guan, J., Kalhoro, Z.A., and Chandio, A.A., Tridiagonal iterative method for linear systems, University of Sindh Journal of Information and Communication Technology, (USJICT), 1(1) (2017).

[16] Hadjidimos, A., Optimum stationary and nonstationary iterative methods for the solution of singular linear systems, Numerische Mathematik, 51(5) (1987) 517530.

[17] Hageman, L., and Young, D., Applied Iterative Methods, New York: Academic Press, (1981).

[18] Hezari, D., Edalatpour, V., and Khojasteh Salkuyeh D., Preconditioned GSOR iterative method for a class of complex symmetric system of linear equations, Numerical Linear Algebra with Applications, 22(4) (2015) 761-776.

[19] Kaveh, A., Rahami, H., and Shojaei, I., Swift Analysis of Civil Engineering Structures Using Graph Theory Methods, Springer Nature, (2020)

[20] Kebede, T., Second Degree Refinement Jacobi Iteration Method for Solving System of Linear Equation, International Journal of Computing Science and applied mathematics, 3(1) (2017) 5-10.

[21] Pan, V.Y., and Qian, G., Solving linear systems of equations with randomization, augmentation and aggregation, Linear Algebra and its Applications, 437(12) (2012) 2851-2876.

[22] Pratapa, Ph.P., Suryanarayana, Ph., and Pask, J.E., Anderson acceleration of the Jacobi iterative method: An efficient alternative to Krylov methods for large, sparse linear systems, Journal of Computational Physics, 306(1) (2016) 43-54.

[23] Santos, N.R., and Evans, D.J., Non-stationary iterative solution of positive definite linear systems, International Journal of Computer Mathematics, (2007) 289-304.

[24] Tian, Zh., Tian, M., Zhang, Y., and Wen, P., An iteration method for solving the linear system Ax=b, Computers & Mathematics with Applications, 75(8) (2018) 2710-2722.

[25] Todini, E., and Pilati, S., A Gradient Algorithm for the Analysis of Pipe Networks, in International Conference on Computer Applications for Water Supply and Distribution. Leicester, UK., (1987).

[26] Varga, R., Matrix Iterative Analysis. Englewood Cliffs, NJ: Prentice-Hall, (1962).

[27] Webster, R., Efficient Algebraic Multigrid Solvers with Elementary Restriction and Prolongation, International Journal for Numerical Methods in Fluids, 28 (1998) 317-336.

[28] Xiao-yong, Z., and Junlin, L., Convergence analysis of Jacobi pseudo-spectral method for the Volterra delay integro-differential equations, Appl. Math. info. Sci., 9 (2015) 135-145.

[29] Young, D., Iterative Solutions of Large Linear Systems, New York: Academic Press, (1971).

[30] Yueh, W.Ch., Eigenvalues of Several Tridiagonal Matrices, Applied Mathematics E-Notes, 5 (2005) 66-74.