Journal of Algorithms and Computation

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

A Numerical Method for Eigensolution of Tridiagonal Matrices

I. Shojaei*[1] and H. 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 an iterative method to solve eigenproblem for non-repetitive tridiagonal matrices. The importance of eigensolution for tridiagonal matrices is that in many algorithms the eigneproblem for an arbitrary matrix is first converted to the eigenproblem for a tridiagonal matrix and then the problem is tackled. Our proposed method was developed through taking advantages of some unique properties of repetitive and non-repetitive tridiagonal matrices. First, we established closed-form solutions for the system of linear equations $\mathbf{Mx} = \mathbf{f}$ for the condition $\mathbf{M}$ is tridiagonal. When $\mathbf{M}$ is a repetitive tridiagonal matrix, the unknown vector $\mathbf{x}$, the vector $\mathbf{f}$, and the coefficient matrix $\mathbf{M}$ are expanded using orthogonal basis of matrix $\mathbf{M}$ and closed-form relationships are obtained. For non-repetitive matrix $\mathbf{M}$, the tridiagonal matrix algorithm is used to efficiently solve the matrix equation. We then

Keyword: iterative method, eigensolution, repetitive tridiagonal matrices, non-repetitive tridiagonal matrices, efficient solution, system of linear equations.

ARTICLE INFO

Article history:
Research paper
Received 15, January 2021
Received in revised form 21, April 2021
Accepted 12 May 2021
Available online 01, June 2021

AMS subject Classification: 65F15.

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

1 Abstract continued

using orthogonal basis of matrix **M** and closed-form relationships are obtained. For non-repetitive matrix **M**, the tridiagonal matrix algorithm is used to efficiently solve the matrix equation. We then implemented these solutions in an iterative relationship for eigenproblem where eigenpairs of non-repetitive tridiagonal matrices were obtained through successive solution of the tridiagonal matrix equation efficiently solved above. Furthermore, closed-form relationships for eigenpairs of repetitive tridiagonal matrices were implemented in the algorithm as start point for eigensolution of non-repetitive tridiagonal matrices so that the required number of iterations was significantly reduced. Computational complexity of the proposed method is $O(n^2)$ that is competitive with the best existing algorithms in literature. As indicated through several numerical examples, the advantages of the proposed algorithm include high rate of convergence, computational efficiency in each iteration, simple implementation, and availability of an objective start point for initialization.

2 Introduction

There are several applications for eigenpairs of a matrix. For instance, eignepairs can be used for dimensionality reduction in order to detect a set of features that are more important in description of a random phenomenon such that a simpler and more efficient model is obtained. One of such techniques is principal component analysis wherein eigenpairs of covariance matrix are used to compact the original matrix of observations. Therefore, without missing significant information, efficiency and efficacy of the model increase. Another application of eigenpairs is in defining modal matrices that are used for diagonalization in many science and engineering problems. Modal matrices may take different patterns according to the form of eigenpairs. For the case eigenvalues are distinguished, modal matrix will be a diagonal matrix, whereas for repeated eigenvalues as well as complex conjugate eigenvalues block diagonal matrices are obtained. Many other large-scale problems of engineering and scientific computing also require eigensolution so that developing efficient methods for eigenproblem have always been of interest among researchers.

Given different applications of eigenpairs in science and engineering, several methods have been developed for eignesolution. These methods includes Power iteration, Inverse iteration, Rayleigh quotient iteration, QR algorithm, and Lanczos algorithm among many others. Specifically, Lanczos method is a general method that can be combined with other algorithms to convert a Hermitian matrix into a similar tridiagonal matrix. The obtained tridiagonal matrix can then be solved using other algorithms. A unified overview of theory, algorithms, and practical software for eigenvalue problems can be found in [2, 5]. Bai et al. [2] developed an informal decision tree for choosing the best state-of-the-art algorithms and software for a given eigenproblem. Ji et al. [11] established implementation of the block power method based on Spark in order to approximate the dominant eigenpairs

of large-scale sparse matrices. Yuan et al. [17] proposed a solution, called truncated power method, to approximately solve the underlying nonconvex optimization problem for sparse eigenvalue problem. Solving generalized eigenvalue problems and their application in large scale computational models and engineering problems can be found in [1, 14]. Lanczos [13] established an iterative method for the solution of eigenvalue problem of linear differential and integral operators. Several other works have also been conducted to study eigenproblem for asymmetric matrices [4], Hermitian matrices [3], block matrices [6, 7], band matrices [8, 9], and non-linear problems [15]. New algorithms for eigensolution using Multilevel Newtons Method and Subspace Methods were developed by He et al. [10] and Watkins [16].

In this paper we have developed an iterative method to solve eigenproblem for tridiagonal matrices. In many algorithms for eigensolution, the desired matrix is first converted to a similar tridiagonal matrix and then eigenproblem for the new matrix is solved. As such, developing efficient eigensolutions for tridiagonal matrices can be regarded as an independent vital problem. Our proposed method was developed through taking advantages of unique properties of repetitive tridiagonal matrices in eigenvalue problem and unique properties of both repetitive and non-repetitive matrices in solution of linear system of equations. In summary, we established closed-form solutions for the system of linear equations $\mathbf{Mx} = \mathbf{f}$ for the condition $\mathbf{M}$ is tridiagonal. We then implemented these solutions in an iterative relationship for eigenproblem where eigenpairs of non-repetitive tridiagonal matrices were obtained through successive solution of the tridiagonal matrix equations efficiently solved above. Furthermore, closed-form relationships for eigenpairs of repetitive tridiagonal matrices were implemented in the algorithm as the start point for eigensolution of non-repetitive tridiagonal matrices such that the required number of iterations significantly decreased.

3 Closed-form solution of a system of linear equations for tridiagonal matrices

Eigenpairs of a repetitive tridiagonal matrix $\mathbf{M}$ with the following form

$$\mathbf{M} = \begin{bmatrix} b & c & & \dots & 0 \\ a & b & c & & \vdots \\ & a & \ddots & \ddots & \\ \vdots & & \ddots & b & c \\ 0 & \dots & & a & b \end{bmatrix}_{N-1} \tag{1}$$

can be obtained using the following equation [12, 18]

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

For the case where matrix $\mathbf{M}$ is symmetric ($i.e., a = c$), eigenpairs in Eq(2) are simplified to $\lambda_n = b + 2a\cos\frac{n\pi}{N}$ and $v_j^n = \sin\frac{nj\pi}{N}$.
To solve the matrix equation $\mathbf{Mx} = \mathbf{f}$, one can expand the unknown vector $\mathbf{x}$ and the given vector $\mathbf{f}$ using the orthogonal eigenvectors $(v^1, v^2, \ldots, v^{N-1})$ of matrix $\mathbf{M}$:

$$\mathbf{x} = \sum_{k=1}^{N-1} x_k \mathbf{v}^k \quad \text{and} \quad \mathbf{f} = \sum_{k=1}^{N-1} f_k \mathbf{v}^k \tag{3}$$

Since $\mathbf{f}$ is given, f_ks can be calculated as $f_k = (v^k)^t\mathbf{f}$. x_ks are unknowns of the problem, now lets substitute Eq (3) in $\mathbf{Mx} = \mathbf{f}$:

$$\begin{aligned}
&\mathbf{M}\sum_{k=1}^{N-1} x_k \mathbf{v}^k = \sum_{k=1}^{N-1} f_k \mathbf{v}^k \\
&\mathbf{M}\sum_{k=1}^{N-1} x_k \mathbf{v}^k = \sum_{k=1}^{N-1} x_k \mathbf{M}\mathbf{v}^k = \sum_{k=1}^{N-1} x_k \lambda_k \mathbf{v}^k \\
&\sum_{k=1}^{N-1} \lambda_k x_k \mathbf{v}^k = \sum_{k=1}^{N-1} f_k \mathbf{v}^k
\end{aligned} \tag{4}$$

From the last term in Eq (4) we can write

$$\lambda_k x_k = f_k \rightarrow x_k = \frac{f_k}{\lambda_k} \quad \text{and} \quad k = 1, 2, \ldots, N-1, \tag{5}$$

Therefore [12],

$$\begin{aligned}
&\mathbf{x} = \sum_{k=1}^{N-1} \frac{f_k}{\lambda_k}\mathbf{v}^k = \sum_{k=1} N-1\mathbf{v}^k \frac{f_k}{\lambda_k} = \sum_{k=1}^{N-1} \frac{\mathbf{v}^k(\mathbf{v}^k)^t}{\lambda_k}\mathbf{f} \\
&\mathbf{x} = \sum_{k=1}^{N-1} \frac{[\sin\frac{k\pi}{N}, \sin\frac{2k\pi}{N}, \ldots \sin\frac{(N-1)k\pi}{N}]^t[\sin\frac{k\pi}{N}, \sin\frac{2k}{N}, \ldots, \sin\frac{(N-1)k\pi}{N}]}{b + 2a\cos\frac{k\pi}{N}}\mathbf{f}
\end{aligned} \tag{6}$$

For the case matrix $\mathbf{M}$ is not repetitive as in Eq (7)

$$\mathbf{M} = \begin{bmatrix} b_1 & c_1 & & \dots & 0 \\ a_2 & b_2 & c_2 & & \vdots \\ & a_3 & \ddots & \ddots & \\ \vdots & & \ddots & b_{n-1} & c_{n-1} \\ 0 & \dots & & a_n & b_n \end{bmatrix}_{N-1} \tag{7}$$

the tridiagonal matrix algorithm is used to solve the system of linear equations in $O(n)$ operations and vector x is obtained as follows:

$$\begin{aligned} c'_k &= \begin{cases} \frac{c_k}{b_k}, & k = 1 \\ \frac{c_k}{b_k - a_k c'_{k-1}}, & k = 2, 3, , n-1 \end{cases} \\ f'_k &= \begin{cases} \frac{f_k}{b_k}, & k = 1 \\ \frac{f_k - a_k f'_{k-1}}{b_k - a_k c'_{k-1}}, & k = 2, 3, , n-1 \end{cases} \end{aligned} \tag{8}$$

And by the back substitution

$$\begin{aligned} x_n &= f'_n \\ x_k &= f'_k - c'_k x_{k+1} \end{aligned}$$

And

$$\mathbf{x} = [x_1, x_2, , x_n]^t.$$

4 An Iterative Eigensolution Algorithm

In previous section we presented the closed-form relationships to solve eigenproblem and system of linear equations for repetitive tridiagonal matrices and to solve system of linear equations for non-repetitive tridiagonal matrices. We can now take advantages of these closed-form solutions to develop an efficient iterative algorithm for eigensolution of non-repetitive tridiagonal matrices. Consider the following eigenvalue problem:

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{v} \tag{9}$$

where matrix A is a non-repetitive tridiagonal matrix of the form in Eq (7). We can write the following approximation

$$(\mathbf{A} - \lambda^0 \mathbf{I})\mathbf{v}^1 = \mathbf{v}^0 \tag{10}$$

and lets initialize λ^0 and $\mathbf{v}^0$ as the eigenpairs of matrix A_1 that is a repetitive tridiagonal matrix (see Eq(1)) corresponding to matrix $A.a, b$, and c entries in matrix A_1 are chosen to

be, respectively, the average of diagonal entries ($i.e., b_1, \ldots, b_n$) and the average of upper and lower neighboring-diagonals entries ($i.e., a_2, \ldots, a_n and c_1, \ldots, c_{n-1}$) of matrix A:

$$\mathbf{A}_1\mathbf{v}^0 = \lambda^0\mathbf{v}^0 \tag{11}$$

Because $\mathbf{A}_1$ is a repetitive tridiagonal matrix (see Eq(1)), its eigenpairs are readily available from Eq(2).
Now, we can write the general form of Eq(10) for the i^{th} eigenpair as follows

$$(\mathbf{A} - \lambda_i^m\mathbf{I})\mathbf{v}_i^{m+1} = \mathbf{v}_i^m \tag{12}$$

where according to Eq(2)

$$\begin{aligned} &\lambda_i^0 = \lambda_i = b + 2a\cos\frac{i\pi}{N} \\ &v_j^i = \sin\frac{ij\pi}{N} \quad \text{and} \quad j = 1, 2, \ldots, N-1 \\ &\mathbf{v}_i^0 = \mathbf{v}^i = [v_1^i, v_2^i, \ldots, v_{N-1}^i]^t \end{aligned} \tag{13}$$

In Eq(12), for $m = 0$, the values for λ_i^0 and $\mathbf{v}_i^0$ are replaced using relationships above. The obtained system of linear equations is efficiently solved using the relationships developed in previous section (see Eq(6) and Eq(8)) and $\mathbf{v}_i^{m+1}$ (that is $\mathbf{v}_i^1$ for the first step, $m = 0$) is found. It should be noted that in Eq(12), when solving the system of linear equations, the term $\mathbf{A} - \lambda_i^m\mathbf{I}$ is considered as our tridiagonal matrix such that $\mathbf{v}_i^{m+1}$ is calculated as follows:

$$\begin{aligned} c'_k &= \begin{cases} \frac{c_k}{b_k - \lambda_i^m} & k = 1 \\ \frac{c_k}{(b_k - \lambda_i^m) - a_k c'_{k-1}} & k = 2, 3, \ldots, n-1 \end{cases} \\ (v_{ik}^m)' &= \begin{cases} \frac{v_{ik}^m}{b_k - \lambda_i^m} & k = 1 \\ \frac{v_{ik}^m - a_k(v_{ik-1}^m)'}{(b_k - \lambda_i^m) - a_k c'_{k-1}} & k = 2, 3, \ldots, n \end{cases} \end{aligned} \tag{14}$$

And by the back substitution

$$\begin{aligned} v_{in}^{m+1} &= (v_{in}^m)' \\ v_{ik}^{m+1} &= (v_{ik}^m)' - c'_k v_{ik+1}^{m+1} \end{aligned}$$

and

$$\mathbf{I}_i^{m+1} = [v_{i1}^{m+1}, v_{i2}^{m+1}, \ldots, v_{in}^{m+1}]^t$$

Now, using Rayleigh quotient we will have

$$\lambda_i^{m+1} = \frac{(\mathbf{v_i^{m+1}})^\mathbf{t}\mathbf{A}\mathbf{v_i^{m+1}}}{(\mathbf{v_i^{m+1}})\mathbf{A}\mathbf{v_i^{m+1}}} \tag{15}$$

λ_i^{m+1} (that is λ_i^1 for the first step, $m = 0$) is obtained. The new $\mathbf{v_i^{m+1}}$ and λ_i^{m+1} ($\mathbf{v_i^1}$ and λ_i^1 for the first step) are then replaced in Eq(12) and the system of linear equations is solved again. This process is repeated until convergence is achieved. As such i^{th} eignepair is obtained and we can similarly define next eigenpairs. It should also be noted that during all iterations for the solution of i^{th} eigenpair, we always remove the components of $\mathbf{v_i^{m+1}}$ and $\mathbf{v_i^m}$ that are parallel to the previously-defined eigenvectors (i.e., $\mathbf{v_1}$ to $\mathbf{v_{i-1}}$) . This will help us to avoid convergence of the solution to previous eigenpairs. After removing the excessive components, $\mathbf{v_i^{m+1}}$ and $\mathbf{v_i^m}$ are also normalized to have a unit length. In the proposed eigensolution algorithm we are taking advantages of 1) the efficient solution of system of linear equations for tridiagonal matrices (Eq(6) or Eq(8)) and 2) an efficient start point (λ_i^0 and $\mathbf{v_i^0}$) taken from repetitive tridiagonal matrices. The algorithm can be summarized as follows:

1. Form the approximation $(\mathbf{A} - \lambda_i^m \mathbf{I})\mathbf{v_i^{m+1}} = \mathbf{v_i^m}$ for i^{th} eigenpair of matrix $\mathbf{A}$
2. Construct matrix $\mathbf{A_1}$ that is a repetitive tridiagonal matrix corresponding to matrix $\mathbf{A}$
3. Calculate i^{th} eigenpair of matrix $\mathbf{A_1}$ using Eq(2) or Eq(13) and call it λ_i^0 and $\mathbf{v_i^0}$. Remove the components of $\mathbf{v_i^0}$ that are parallel to the previous eigenvectors and then normalize it.
4. In the equation $(\mathbf{A} - \lambda_i^m \mathbf{I})\mathbf{v_i^{m+1}} = \mathbf{v_i^m}$, set $m = 0$, and replace λ_i^0 and $\mathbf{v_i^0}$ from step 3.
5. Calculate $\mathbf{v_i^{m+1}}$ (that is $\mathbf{v_i^1}$ for the first step, $m = 0$) using Eq(14). Remove the components of $\mathbf{v_i^{m+1}}$ that are parallel to the previous eigenvectors and then normalize $\mathbf{v_i^{m+1}}$.
6. Update λ_i^{m+1} Using Eq(15),$\lambda_i^{m+1} = \frac{(\mathbf{v_i^{m+1}})^t \mathbf{A} \mathbf{v_i^{m+1}}}{(\mathbf{v_i^{m+1}})^t \mathbf{v_i^{m+1}}}$.
7. Replace $\mathbf{v_i^{m+1}}$ in the right-hand and λ_i^{m+1} in the left-hand of the system of linear equations $(\mathbf{A} - \lambda_i^m \mathbf{I})\mathbf{v_i^{m+1}} = \mathbf{v_i^m}$ and repeat from step 5 until convergence is achieved.

5 Computational complexity of the method

Since the proposed solution is iterative, first the computational complexity for one iteration is calculated:

- For calculation of $\mathbf{v_i^{m+1}}$ from Eq(14)
  - The solution can be obtained in $O(n)$ operations. This should be conducted n times for n eigenparis (the algorithm above was developed for i^{th} eigenpair). Therefore, the computational complexity is $O(n^2)$.

- For calculation of λ_i^n in Eq(15)
  - $\mathbf{A}\mathbf{v_i^{m+1}}$: Multiplication of a tridiagonal matrix ($\mathbf{A}$) and a vector ($\mathbf{V}$) that will be of the computational complexity $3O(n)$. The outcome here would be a vector ($\mathbf{F}$)
  - $(\mathbf{v_i^{m+1}})^\mathbf{t}\mathbf{F}$: Multiplication of a row vector $(\mathbf{v_i^{m+1}})^\mathbf{t}$ and a column vector ($\mathbf{F}$), that would be of the complexity $O(n)$. Here, the outcome (i.e., the numerator) will be a scaler
  - $(\mathbf{v_i^{n+1}})^\mathbf{t}\mathbf{v_i^{n+1}}$: Multiplication of a row vector $(\mathbf{v_i^{n+1}})^\mathbf{t}$ and a column vector $\mathbf{v_i^{n+1}}$, that would be of the complexity $O(n)$. Here, the outcome (i.e., the denominator) will be a scaler
  - The three steps above should be conducted n times for n eigenparis. Therefore, the computational complexity is $5O(n^2)$.

As such, for one iteration of the proposed method the dominant computational complexity is $O(n^2)$. We have a total of $6O(n^2)$ from the two steps above (1 from the first step and 5 from the second step). If m is the number of required iterations to attain a desired accuracy for eigensolution, one needs to complete operations of the complexity $O(6mn^2)$ for the entire method. In the numerical examples section, it was shown that the required number of iterations m is much smaller than dimension of matrix $\mathbf{A}$, that is $6m << n$. Therefore, computational complexity of the proposed method is $O(n^2)$.
There are two factors that contribute to the efficiency of the proposed eigensolution algorithm. First, we have used efficient numerical methods for the solution of system of linear equations for tridiagonal matrices (see step 5 of the eigensolution algorithm as well as Eq(14), and second we have used an efficient start point (λ_i^0 and $\mathbf{v_i^0}$) in the eigensolution algorithm, which was taken from closed-form eigensolution of repetitive tridiagonal matrices. In the following section the method is applied to several numerical examples.

6 Numerical Examples

Example 1. Consider the following symmetric matrix of dimension $n = 50$ for which an eigensolution is sought:

$$\mathbf{A} = \begin{bmatrix} 6.9597 & 12.9286 & & & & & \\ 12.9286 & 15.4335 & 10.3042 & & & & \\ & 10.3042 & \ddots & \ddots & & & \\ & & \ddots & 14.8193 & 8.7557 & & \\ & & & 8.7557 & 13.1183 & 8.4659 & \\ & & & & 8.4659 & 14.4679 & \end{bmatrix}_{50\times 50}$$

The repetitive tridiagonal matrix $\mathbf{A_1}$ in the proposed algorithm and its eigenpairs λ_i^0 and $\mathbf{v_i^0}$ are as follows:

$$\mathbf{A_1} = \begin{bmatrix} 11.2942 & 10.5654 & & & & & \\ 10.5654 & 11.2942 & 10.5654 & & & & \\ & 10.5654 & \ddots & \ddots & & & \\ & & \ddots & 11.2942 & 10.5654 & & \\ & & & 10.5654 & 11.2942 & 10.5654 & \\ & & & & 10.5654 & 11.2942 & \end{bmatrix}$$

$$\boldsymbol{\lambda} = \begin{bmatrix} -9.7966 & & & & & & \\ & -9.6765 & & & & & \\ & & \ddots & & & & \\ & & & 32.0653 & & & \\ & & & & 32.2649 & & \\ & & & & & 32.3850 & \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} -0.0337 & -0.0659 & \dots & -0.1282 & -0.0506 & -0.0175 \\ 0.0436 & 0.0807 & \dots & -0.3743 & -0.1320 & -0.0887 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -0.0956 & -0.2270 & \dots & -0.0573 & 0.0411 & -0.0093 \\ 0.0478 & 0 & \dots & 0.0867 & -0.0240 & -0.0121 \\ -0.0418 & 0.0213 & \dots & 0.0097 & -0.0008 & 0.0188 \end{bmatrix}$$

The problem was solved using the proposed method as well as four established iterative methods: power iteration, inverse iteration, Rayleigh quotient iteration, and QR algorithm. All methods were also compared against the exact solution of the problem (Table 1):

Table 1: Comparison between the exact solution, the proposed method, and four iterative methods for eigensolution of a symmetric matrix of dimension $n = 50$. After 6 iterations the proposed method reached the desired accuracy of the order 10^{-2}. Rayleigh quotient method reached the same accuracy after 9 iterations. The power, inverse, and QR methods reached the maximum allowed iterations (500 iterations) before meeting the desired accuracy of 10^{-2}. The CPU time for the proposed method was 0.0202 s that was the fastest among all methods. Additional analyses indicated that the power and QR methods needed, respectively, 571 and 843 iterations to reach the desired accuracy; requiring 0.7429 s and 0.4259 s to complete. The inverse method did not meet the desired accuracy even after 5000 iterations.

After 6 iterations the proposed method reached the desired accuracy of the order 10^{-2}. Rayleigh quotient method reached the same accuracy after 9 iterations. The power, inverse, and QR methods reached the maximum allowed iterations (500 iterations) before meeting the desired accuracy of $10^{-}2$ (Table 1). The convergence rate of the proposed

Table 1:

$\boldsymbol{\lambda}$	6 iterations	500 iterations	500 iterations	9 iterations	500 iterations
Exact	Proposed	Power	Inverse	Rayleigh quotient	QR
-14.1150	-14.1150	-14.1150	-14.1150	-14.1150	-14.1150
-11.9367	-11.9365	-11.9367	-11.9367	-11.9367	-11.8090
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
9.2255	9.2255	9.2255	9.2255	9.2255	9.2255
12.4291	12.4291	12.4291	12.4291	12.4291	12.0921
14.3926	14.3926	14.3926	14.3645	14.3926	14.3926
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
35.1767	35.1767	35.1767	35.1766	35.1767	35.1773
36.3573	36.3573	36.3573	36.3573	36.3573	36.3573
37.4807	37.4807	37.4807	37.8447	37.4807	37.4802
$\|$Exact-Iterative$\|$	0.0015	0.0254	5.0401	$5.1e-07$	0.4170
Time Complexity $n = 50$	0.0202 s	0.6217 s	0.5843 s	0.0434 s	0.1827 s

method was higher than that of the four iterative methods. Specifically, the CPU time for the proposed method was 0.0202 s that was the fastest among all methods (Table 1). Additional analyses indicated that the power and QR methods needed, respectively, 571 and 843 iterations to reach the desired accuracy; requiring 0.7429 s and 0.4259 s to complete. The inverse method did not meet the desired accuracy even after 5000 iterations.

Example 2. BlackScholes equation
Consider the following matrix equation obtained from numerical formulation of BlackScholes equation [12]:

$$\mathbf{M_1^L V_1} = \left(\begin{bmatrix} 1.0000 & 0.0000 & & & & & \\ 0.0000 & 1.0000 & 0.0000 & & & & \\ & 0.0000 & \ddots & \ddots & & & \\ & & \ddots & 0.5186 & 0.2358 & & \\ & & & 0.2487 & 0.5125 & 0.2389 & \\ & & & & 0.2519 & 0.5063 \end{bmatrix}_{159\times 159} \mathbf{V_1} \right)$$

$$= \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 9.75 \\ 7.39 \end{bmatrix}_{159\times 1}$$

Efficient solution of the system of linear equations using repetitive tridiagonal matrices has already been discussed [12]. Here eigensolution of the coefficient matrix $\mathbf{M_1^L}$ is aimed.

The problem was solved using the proposed method, power iteration, inverse iteration, Rayleigh quotient iteration, and QR method. All methods were also compared against the exact solution of the problem (Table 2).
The tridiagonal matrix $\mathbf{A_1}$ in the proposed algorithm and its eigenpairs λ_i^0 and $\mathbf{v_i^0}$ were as follows:

$$\mathbf{A_1} = \begin{bmatrix} 0.8339 & 0.0828 & & & & & \\ 0.0828 & 0.8339 & 0.0828 & & & & \\ & 0.0828 & \ddots & \ddots & & & \\ & & \ddots & 0.8339 & 0.0828 & & \\ & & & 0.0828 & 0.8339 & 0.0828 & \\ & & & & 0.0828 & 0.8339 & \end{bmatrix}$$

$$\boldsymbol{\lambda} = \begin{bmatrix} 0.06684 & & & & & & \\ & 0.06685 & & & & & \\ & & \ddots & & & & \\ & & & 0.9991 & & & \\ & & & & 0.9993 & & \\ & & & & & 0.994 & \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} 3.2e-05 & -4.0e-05 & \dots & -5.1e-05 & -4.0e-05 & 3.2e-05 \\ -6.7e-05 & 8.4e-05 & \dots & -0.0001 & -8.4e-05 & 6.7e-05 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.0468 & 0.0590 & \dots & -0.0749 & 0.0590 & 0.0468 \\ -0.0352 & -0.0410 & \dots & -0.0521 & 0.0410 & 0.0325 \\ 0.0169 & 0.0213 & \dots & -0.0271 & 0.0213 & 0.0169 \end{bmatrix}$$

Table 2: Comparison between the exact solution, the proposed method, and four iterative methods for eigensolution of a matrix of dimension $n = 159$ obtained from numerical solution of BlackScholes equation. After 5 iterations the proposed method reached the desired accuracy of the order 10^{-2}. The Rayleigh quotient, inverse, and power methods reached the same accuracy after 6, 76, and 301 iterations, respectively. The QR method reached the maximum allowed iterations (500 iterations) before meeting the desired accuracy of 10^{-2}. The CPU time for the proposed algorithm was 0.2522 s that was about 3, 7, and 27 times faster than Rayleigh quotient, inverse, and power methods, respectively. Additional analyses indicated that the QR method needed 1861 iterations, taking 8.3970 s, to reach the desired accuracy.

After 5 iterations the proposed method reached the desired accuracy of the order 10^{-2}. The Rayleigh quotient, inverse, and power methods reached the same accuracy after 6, 76, and 301 iterations, respectively. The QR method reached the maximum allowed iterations (500 iterations) before meeting the desired accuracy of 10^{-2} (Table 2). The CPU time for the proposed algorithm was 0.2522 s that was about 3, 7, and 27 times faster

$\boldsymbol{\lambda}$	5 iterations	301 iterations	76 iterations	6 iterations	500 iterations
Exact	Proposed	Power	Inverse	Rayleigh quotient	QR
0.0774	0.0774	0.0774	0.0774	0.0774	0.0774
0.1325	0.1325	0.1325	0.1325	0.1325	0.1325
⋮	⋮	⋮	⋮	⋮	⋮
0.9679	0.9679	0.9679	0.9679	0.9679	0.9652
0.9693	0.9693	0.9692	0.9693	0.9693	0.9660
0.9707	0.9707	0.9713	0.9707	0.9706	0.9668
⋮	⋮	⋮	⋮	⋮	⋮
0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
0.9999	1.0000	1.0000	1.0000	0.9999	0.9999
0.9999	1.0000	1.0000	1.0000	1.0000	1.0000
‖Exact-Iterative‖	0.0051	0.0099	0.0094	0.0088	0.0715
Time Complexity $n = 159$	0.2522 s	6.8457 s	1.6853 s	0.7782 s	1.6434 s

than Rayleigh quotient, inverse, and power methods, respectively. Additional analyses indicated that the QR method needed 1861 iterations, taking 8.3970 s, to reach the desired accuracy.

Example 3. Consider a coefficient matrix $\mathbf{M}$ of the same dimension with relatively larger entries as compared to Example 1:

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

The tridiagonal matrix $\mathbf{A_1}$ in the proposed algorithm and its eigenpairs λ_i^0 and $\mathbf{v_i^0}$ are as follows:

$$\mathbf{A_1} = \begin{bmatrix} 1.8839 & 0.5795 & & & & & \\ 0.5861 & 1.8839 & 0.5795 & & & & \\ & 0.5861 & \ddots & \ddots & & & \\ & & \ddots & 1.8839 & 0.5795 & & \\ & & & 0.5861 & 1.8839 & 0.5795 & \\ & & & & 0.5861 & 1.8839 & \end{bmatrix}$$

Table 2: Comparison between the exact solution, the proposed method, and four iterative methods for eigensolution of a matrix of dimension $n = 159$. After 5 iterations the proposed method reached the desired accuracy of the order 10^{-2}. The Rayleigh quotient method reached the same accuracy after 11 iterations. The power, inverse, and QR methods reached the maximum allowed iterations (500 iterations). The CPU time for the proposed algorithm was 0.2528 s that was about 5 times faster than Rayleigh quotient method. Additional analyses indicated that the power and QR methods needed, respectively, 543 and 1525 iterations to reach the desired accuracy; requiring 14.8105 s and 7.6588 s to complete. The inverse method did not meet the desired accuracy even after 5000 iterations.

$\boldsymbol{\lambda}$	5 iterations	500 iterations	500 iterations	11 iterations	500 iterations
Exact	Proposed	Power	Inverse	Rayleigh quotient	OR
0.1618	0.1618	0.1618	0.1618	0.1618	0.1618
0.2421	0.2421	0.2421	0.2421	0.2421	0.2421
⋮	⋮	⋮	⋮	⋮	⋮
2.2973	2.2973	2.2973	2.2761	2.2973	2.2979
2.3184	2.3184	2.3184	2.3132	2.3184	2.3210
2.3393	2.3393	2.3393	2.3212	2.3393	2.3414
⋮	⋮	⋮	⋮	⋮	⋮
3.0480	3.0480	3.0479	3.0482	3.0480	3.0336
3.0491	3.0491	3.0489	3.0483	3.0491	3.0396
3.0498	3.0498	3.0496	3.0498	3.0498	3.0455
‖Exact-Iterative‖	0.0098	0.0100	1.0659	0.0099	0.1828
Time Complexity $n = 159$	0.2528 s	11.1972 s	12.6215 s	1.3718 s	1.3839 s

$$\boldsymbol{\lambda} = \begin{bmatrix} 0.7184 & & & & & & \\ & 0.7191 & & & & & \\ & & \ddots & & & & \\ & & & 3.0475 & & & \\ & & & & 3.0486 & & \\ & & & & & 3.0493 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} 1.6e-04 & 0.0102 & \dots & -0.0061 & -0.00051 & 9.3e-04 \\ -0.0018 & -0.0038 & \dots & -0.0122 & -0.0102 & 0.0019 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.1077 & -0.0585 & \dots & -0.0217 & -0.0116 & 0.0102 \\ 0.0510 & 0.0047 & \dots & -0.0146 & 0.0078 & 0.0068 \\ -0.1453 & -0.0741 & \dots & -0.0074 & -0.0039 & 0.0034 \end{bmatrix}$$

After 5 iterations the proposed method reached the desired accuracy of the order 10^{-2}. The Rayleigh quotient method reached the same accuracy after 11 iterations. The power, inverse, and QR methods reached the maximum allowed iterations (500 iterations) (Table

2). The CPU time for the proposed algorithm was 0.2528 s that was about 5 times faster than Rayleigh quotient method (Table 2). Additional analyses indicated that the power and QR methods needed, respectively, 543 and 1525 iterations to reach the desired accuracy; requiring 14.8105 s and 7.6588 s to complete. The inverse method did not meet the desired accuracy even after 5000 iterations.
Now, lets change the coefficient matrix $\mathbf{M}$ by increasing the values of off-diagonal entries:

$$\mathbf{M} = \begin{bmatrix} 2.0500 & 4.0000 & & & & & \\ 4.0001 & 2.0500 & 4.0000 & & & & \\ & 4.0002 & \ddots & \ddots & & & \\ & & \ddots & 1.5686 & 4.2358 & & \\ & & & 4.2487 & 1.5625 & 4.2389 & \\ & & & & 4.2519 & 1.5563 & \end{bmatrix}_{159\times 159}$$

The tridiagonal matrix $\mathbf{A_1}$ in the proposed algorithm and its eigenpairs λ_i^0 and $\mathbf{v_i^0}$ are as follows:

$$\mathbf{A_1} = \begin{bmatrix} 1.8839 & 4.0795 & & & & & \\ 4.0861 & 1.8839 & 4.0795 & & & & \\ & 4.0861 & \ddots & \ddots & & & \\ & & \ddots & 1.8839 & 4.0795 & & \\ & & & 4.0861 & 1.8839 & 4.0795 & \\ & & & & 4.0861 & 1.8839 & \end{bmatrix}$$

$$\boldsymbol{\lambda} = \begin{bmatrix} -6.2801 & & & & & & \\ & -6.2754 & & & & & \\ & & \ddots & & & & \\ & & & 10.0354 & & & \\ & & & & 10.0432 & & \\ & & & & & 10.0480 & \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} 0.0091 & -0.0111 & \dots & 0.0083 & 0.0041 & 0.0017 \\ -0.0213 & -0.0131 & \dots & 0.0166 & 0.0082 & 0.0035 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.0163 & 0.0770 & \dots & 0.0092 & -0.0142 & 0.0072 \\ 0.0244 & -0.0321 & \dots & 0.0062 & -0.0095 & 0.0048 \\ 0.0163 & 0.0289 & \dots & 0.0031 & -0.0048 & 0.0024 \end{bmatrix}$$

The proposed method reached the desired accuracy of the order 10^{-2} after 6 iterations. The Rayleigh quotient method reached the same accuracy after 11 iterations. The power, inverse, and QR methods reached the maximum allowed iterations (500 iterations) without meeting the desired accuracy. The CPU time for the proposed algorithm was 0.2526 s

Table 3: The proposed method reached the desired accuracy of the order 10^{-2} after 6 iterations. The Rayleigh quotient method reached the same accuracy after 11 iterations. The power, inverse, and QR methods reached the maximum allowed iterations (500 iterations) without meeting the desired accuracy. The CPU time for the proposed algorithm was 0.2526 s that was about 6 times faster than Rayleigh quotient method. We performed additional analyses up to 5000 iterations for the power, inverse, and QR methods but they did not meet the desired accuracy of 10^{-2}.

$\boldsymbol{\lambda}$	6 iterations	500 iterations	500 iterations	11 iterations	500 iterations
Exact	Proposed	Power	Inverse	Rayleigh quotient	QR
-6.7521	-6.7521	-6.7521	-6.7521	-6.7521	-6.6686
-6.6124	-6.6124	-6.6124	-6.6124	-6.6124	-6.6102
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
4.7115	4.7115	4.0523	3.9472	4.71156	4.3487
4.8614	4.8614	4.0604	4.1019	4.8614	4.3925
5.0100	5.0100	4.2557	4.3615	5.0100	4.4325
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
10.0359	10.0359	10.0360	10.0312	10.0359	9.9955
10.0438	10.0438	10.0433	10.0330	10.0438	10.0152
10.0485	10.0485	10.0477	10.0338	10.04852	10.0351
$\|$Exact-Iterative$\|$	$1.2e-04$	5.5669	14.3944	0.0087	3.1861
Time Complexity $n = 159$	0.2526 s	13.8993 s	11.3784 s	1.5030 s	1.5129 s

that was about 6 times faster than Rayleigh quotient method. We performed additional analyses up to 5000 iterations for the power, inverse, and QR methods but they did not meet the desired accuracy of 10^{-2}.

Example 4. Consider matrix $\mathbf{M}$ of dimension $n = 500$. The matrix entries were randomly generated where the diagonal and off-diagonal entries were considered to be positive and negative, respectively. Such a pattern is seen in the final coefficient matrix of many engineering problems such as stiffness matrices in structural mechanics problems:

$$\mathbf{M} = \begin{bmatrix} 40.8072 & -24.2129 & & & & & \\ -24.2129 & 84.0156 & -13.1912 & & & & \\ & -13.1912 & \ddots & \ddots & & & \\ & & \ddots & 47.9983 & -19.8795 & & \\ & & & -19.8795 & 61.8956 & -14.7955 & \\ & & & & -14.7955 & 35.4551 & \end{bmatrix}_{500\times 500}$$

The tridiagonal matrix $\mathbf{A_1}$ in the proposed algorithm and its eigenpairs λ_i^0 and $\mathbf{v_i^0}$ are as

Table 4: Comparison between the exact solution, the proposed method, and four iterative methods for eigensolution of a matrix of dimension $n = 500$. The diagonal and off-diagonal entries of the matrix were considered to be positive and negative, respectively. Such a pattern is seen in the final coefficient matrix of many engineering problems such as stiffness matrices in structural mechanics. After 10 iterations the proposed method reached the desired accuracy of the order 10^{-2}. The Rayleigh quotient method reached the same accuracy after 12 iterations. The power, inverse, and QR methods reached the maximum allowed iterations (500 iterations) before meeting the desired accuracy of 10^{-2}). The CPU time for the proposed algorithm was 4.7158 s that was about 10 times faster than Rayleigh quotient method. We performed additional analyses up to 5000 iterations for the power, inverse, and QR methods but they did not meet the desired accuracy of 10^{-2}.

$\boldsymbol{\lambda}$	10 iterations	500 iterations	500 iterations	12 iterations	500 iterations
Exact	Proposed	Power	Inverse	Rayleigh quotient	QR
-17.6568	-17.6568	-17.6568	-14.6747	-17.6568	-17.2662
-14.6747	-14.6747	-13.1904	-14.2335	-14.6747	-10.7323
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
52.1357	52.1357	52.1357	52.9599	52.1357	53.2381
52.9599	52.9599	53.1414	53.4833	52.9599	53.5436
53.4833	53.4833	53.4857	53.6320	53.4833	53.7103
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
120.3324	120.3324	120.3324	117.9816	120.3324	117.0530
120.9336	120.9336	120.9336	119.1215	120.9336	117.9679
122.2814	122.2814	122.2814	119.7364	122.2814	118.8489
$\|$Exact-Iterative$\|$	$1.9e-04$	7.4583	12.9249	$8.4e-06$	28.1120
Time Complexity $n = 500$	4.7158 s	173.2776 s	170.213558 s	46.0952 s	33.8641 s

follows:

$$\mathbf{A_1} = \begin{bmatrix} 53.4284 & -21.7844 & & & & & \\ -21.7844 & 53.4284 & -21.7844 & & & & \\ & -21.7844 & \ddots & \ddots & & & \\ & & \ddots & 53.4284 & -21.7844 & & \\ & & & -21.7844 & 53.4284 & -21.7844 & \\ & & & & -21.7844 & 53.4284 \end{bmatrix}$$

$$\boldsymbol{\lambda} = \begin{bmatrix} 9.8604 & & & & & & \\ & 9.8630 & & & & & \\ & & \ddots & & & & \\ & & & 96.9895 & & & \\ & & & & 96.9938 & & \\ & & & & & 96.9964 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} -0.0005 & 0.0017 & \dots & -0.0019 & 0.0003 & -0.0008 \\ -0.0003 & 0.0003 & \dots & 0.0005 & -0.0019 & 0.0004 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -0.0008 & -0.0008 & \dots & -0.0030 & 0.0004 & -0.0007 \\ -0.0008 & 0.0010 & \dots & -0.0066 & -0.0006 & -0.0021 \\ 0.0003 & 0.001894 & \dots & 0.0026 & 0.0015 & -0.0002 \end{bmatrix}$$

It was observed that after 10 iterations the proposed method reached the desired accuracy of the order 10^{-2}. The Rayleigh quotient method reached the same accuracy after 12 iterations. The power, inverse, and QR methods reached the maximum allowed iterations (500 iterations) before meeting the desired accuracy of 10^{-2} (Table 4). The CPU time for the proposed algorithm was 4.7158 s that was about 10 times faster than Rayleigh quotient method (Table 4). We performed additional analyses up to 5000 iterations for the power, inverse, and QR methods but they did not meet the desired accuracy of 10^{-2}.

7 Discussion and Conclusions

There are several methods in literature for eigensolution of matrices. Given the computational demand of analytical methods, iterative methods are often used in practice; especially for the solution of large-scale matrices. Some of such iterative methods include Power iteration, Inverse iteration, Rayleigh quotient iteration, and QR algorithm among others. Given that it is possible to first convert a desired matrix into a similar tridiagonal matrix using Lanczos algorithm and then solve the problem, the focus of the present study has been efficient eigensolution of tridiagonal matrices. The proposed method was developed through taking advantages of unique properties of repetitive tridiagonal matrices in eigenvalue problem and unique properties of both repetitive and non-repetitive matrices in solution of linear system of equations. Computational complexity of the method was of the order $O(n^2)$ being competitive with the best existing algorithms in literature. The method was compared against Power iteration, Inverse iteration, Rayleigh quotient iteration, and QR algorithm through several numerical examples and outperformed for the number of iterations and the total required time to achieve a desired accuracy. The main advantages of the method include the high rate of convergence, computational efficiency in each iteration, simple implementation, and availability of an objective start point for initialization.

References

[1] Akkaoui, Q., Capiez-Lernout, E., Soize, C. and Ohayon, R., Solving generalized eigenvalue problems for large scale fluid-structure computational models with mid-power computers, Computers & Structures, 205, 45-54, 2018.

[2] Bai, Z., Demmel, J., Dongarra, J., Ruhe, A. and van der Vorst, H., Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide. SIAM, Philadelphia, 2000.

[3] Bai, Z., Miao, C. and Jian, S., On multistep Rayleigh quotient iterations for Hermitian eigenvalue problems, Computers & Mathematics with Applications, 77(9): 2396-2406, 2019.

[4] Beilina,L., Karchevskii, E., Karchevskii, M., Algorithms for the Nonsymmetric Eigenvalue Problem, Numerical Linear Algebra: Theory and Applications , 345-374, 2017.

[5] Bosch, J., Greif, C., Geometric and Computational Spectral Theory, Numerical solution of linear Eigenvalue problems, Electronic ISBN: 978-1-4704-4258-3, 2017.

[6] Delvaux, S., Equilibrium problem for the eigenvalues of banded block Toeplitz matrices. Mathematische Nachrichten. 285(16): 1935-1962, 2012.

[7] Delvaux, S., Frederix, K. andVan Barel, M., An algorithm for computing the eigenvalues of block companion matrices. Numerical Algorithms, 62: 261287, 2012.

[8] Ekstrm, S.E., Garoni, C., A matrix-less and parallel interpolationextrapolation algorithm for computing the eigenvalues of preconditioned banded symmetric Toeplitz matrices. Numerical Algorithms. 80: 819848, 2019.

[9] Ekstrm, S.E., SerraCapizzano, S., Eigenvalues and eigenvectors of banded Toeplitz matrices and the related symbols. Numerical Linear Algebra with Applications. 25(5): e2137, 2018.

[10] He, Y., Li, Y., Xie, H., You, C., Zhang, N., A Multilevel Newtons Method for Eigenvalue Problems. Applications of Mathematics. 63: 281303, 2018.

[11] Ji, H., Weinberg, S.H., Li, M., Wang, J., Li, Y., An Apache Spark Implementation of Block Power Method for Computing Dominant Eigenvalues and Eigenvectors of Large-Scale Matrices, IEEE International Conferences on Big Data and Cloud Computing (BDCloud), 2016.

[12] Kaveh, A., Rahami, H. Shojaei, I., Swift Analysis of Civil Engineering Structures Using Graph Theory Methods. Springer, Cham, Switzerland, 2020.

[13] Lanczos,C., An Iteration Method for the Solution of the Eigenvalue Problem of Linear 8. Differential and Integral Operators. Journal of Research of the National Bureau of Standards, 45(4): 255-282. https://doi.org/10.6028/jres.045.026. 1950.

[14] Moler, C.B. and Stewart, G.W., An algorithm for generalized matrix eigenvalue problems. SIAM J. Numer. Anal., 10:241256, 1973.

[15] Xu, F., Huang, Q., Cascadic adaptive finite element method for nonlinear eigenvalue problem based on complementary approach. Journal of Computational and Applied Mathematics. 372, 2020. https://doi.org/10.1016/j.cam.2020.112720.

[16] Watkins, D.S., The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods. SIAM, Philadelphia, 2007.

[17] Yuan, X.T. and Zhang, T., Truncated Power Method for Sparse Eigenvalue Problems, Journal of Machine Learning Research, 14, 899925, 2013.

[18] Yueh, W.C., Eigenvalues of several tridiagonal matrices. Applied Mathematics E-notes. 5: 6674, 2005.