

Paper Type: Original Article



An Experimental Comparison of Two Preconditioned Iterative Methods to Solve the Elliptic Partial Differential Equations

Seyyed Ahmad Edalatpanah * 

Department of Applied Mathematics, Ayandegan Institute of Higher Education, Tonekabon, Iran; s.a. edalatpanah@gmail.com.

Citation:



Edalatpanah, S. A. (2022). An experimental comparison of two preconditioned iterative methods to solve the elliptic partial differential equations. *Computational algorithms and numerical dimensions*, 1 (1), 1-24.

Received: 22/08/2021

Reviewed: 12/09/2021

Revised: 14/09/2021

Accepted: 25/10/2021

Abstract

System of linear equations plays an important role in science and engineering. One of the applications of this system occurs in the discretization of the partial differential equations. This paper aims to investigate an experimental comparison between two kinds of iterative models for solving the elliptic partial differential equations. Different tools of solution such as stationary and non-stationary iterative methods with preconditioning models have been studied. Two types of discretization schemes (centered and hybrid) have been also considered for the comparison of the solution.

Keywords: System of linear equations, iterative methods, preconditioning technique, partial differential equations, finite differences methods.

1 | Introduction

Solving of system of linear equations (SLE) often occur in a wide variety of area, including numerical differential equations, eigenvalue problems, economics models, design and computer analysis of circuits, power system networks, chemical engineering processes, physical and biological sciences [1]-[4]. Such systems are typically solved by two different types of methods; iterative methods and direct methods. The nature of the problem at hand determines which method is more suitable.

A direct method for solving large sparse linear systems of the form of $Ax=b$ involves explicit factorization of the coefficient matrix A into the product of two certain matrices such as triangular matrices [5], [6]. This is a highly time and memory consuming step; nevertheless, direct methods are important because of their generality and robustness. For linear systems arising in certain applications, they are the only feasible solution methods.



Computational
Algorithms and
Numerical Dimensions.

This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0>).



Corresponding Author: s.a. edalatpanah@gmail.com

<https://doi.org/10.22105/cand.2022.155122>

Furthermore, direct methods provide an effective means for solving multiple systems with the same coefficient matrix and different right-hand side vectors because the factorizations need to be performed only ones. The trouble with sparse direct methods is that, because fill-in, their memory requirement grows faster than the size of the problems [7].

Iterative methods are generally much simpler to describe in details than direct methods, because the lack of fill-in and simple matrix access remove the need for sophisticated data structures and graph theoretic analysis. On the other hand, the numerical behavior of iterative methods is much more complicated than that of direct methods [7]-[10].

To sum up, in sparse iterative methods, the spectral properties of the coefficient matrix determine the effectiveness of the method. Therefore, it is a challenge to make them robust enough to serve in portable libraries and environment used for a wide variety of problem domains. In this paper we apply two preconditioned iterative method for solving PDE problems and also compare these two kinds of strategys.

2 | Iterative Methods for Systems of Linear Equations

Consider system of linear equations as

$$Ax = b, \quad (1)$$

A system of nonlinear equations, given in fixed point form

$$x = T(x), \quad (2)$$

can be solved by choosing an initial approximation $x^{(0)} \in \mathbb{R}^n$ and by successively calculating more accurate approximations using the iteration:

$$x^{(k)} = T(x^{(k-1)}), \quad k = 1, 2, 3, \dots \quad (3)$$

Stationary iterative methods for the *Eq. (1)* have the basic form

$$x^{(k)} = Bx^{(k-1)} + c, \quad k = 1, 2, 3, \dots, \quad (3)$$

Where $B \in \mathbb{R}^{n \times n}$ and $c \in \mathbb{R}^n$ do not depend on k . Initially, however, the linear system *Eq. (1)* must be transformed into a form to which the iterative method is applicable. For instance, the coefficient matrix can be written as:

$$A = D - L - U, \quad (4)$$

where $D = \text{diag}(a_{11}, \dots, a_{nn})$, L is a strictly lower triangular matrix and U is the respective upper triangular part of A . For any splitting, $A = M - N$, where M is nonsingular, the iterative method for solving linear systems of *Eq. (1)* is

$$x^{(k)} = M^{-1}Nx^{(k-1)} + M^{-1}b, \quad k = 1, 2, 3, \dots, \quad (5)$$

This iterative process converges to the unique solution $x = A^{-1}b$ for initial vector value $x^0 \in \mathbb{R}^n$ if and only if the spectral radius $\rho(M^{-1}N) < 1$, where $T = M^{-1}N$ is called the iteration matrix.

Here, we introduce some stationary iterative methods, shortly.



2.1 | Stationary Iterative Methods for SLE

The Jacobi method ([5], [6])

For this method $M=D$ and $N=M-A$. A single iteration step of the Jacobi method corresponds to the local solution of the linear system for a single variable. The resulting method is easy to implement, but often converges very slowly.

The Gauss-Seidel method ([5], [6])

this is like the Jacobi method, except that it uses updated approximations as soon as they are available. In general, it will converge faster than the Jacobi method. For this method we have: $M=D-L$.

The SOR method ([5]-[7])

This can be derived from the Gauss-Seidel method by introducing an extrapolation parameter w . For the optimal choice of w , the convergence speed can be increased substantially. The SOR method is defined by $M = \frac{1}{w}(D - wL)$ And $N = \frac{1}{w}((1 - w)D + wU)$.

So, we have $T_{SOR} = (D - wL)^{-1}[(1 - w)D + wU]$.

The SSOR method ([7])

Each iteration step of the Symmetric SOR (SSOR) method consists of two semi-iterations the first of which is a usual (forward) SOR iteration followed by a backward SOR iteration, namely an SOR where the roles of L and U have been interchanged.

The AOR method ([8])

This method uses two parameters r (called relaxation parameter) and w (called extrapolation parameter). The AOR method is defined by $M = \frac{1}{w}(D - rL)$ and $N = \frac{1}{w}((1 - w)D + (w - r)L) + wU$. So, we have:

$$T_{AOR} = (D - rL)^{-1}[(1 - w)D + (w - r)L + wU].$$

The FIM method ([9])

Each iteration step of the FIM consists of n semi-iterations as follows:

$$\left\{ \begin{array}{l} X^{(i+\frac{1}{n})} = D^{-1}[b + LX^{(i+\frac{1}{n})} + UX^{(i)}], \\ X^{(i+\frac{2}{n})} = D^{-1}[b + LX^{(i+\frac{2}{n})} + UX^{(i+\frac{1}{n})}], \\ \vdots \\ X^{(i+\frac{n-1}{n})} = D^{-1}[b + LX^{(i+\frac{n-1}{n})} + UX^{(i+\frac{n-2}{n})}], \\ X^{(i+1)} = D^{-1}[b + LX^{(i+1)} + U\{(1 - w)X^{(i+\frac{n-1}{n})} + wX^{(i+\frac{n-2}{n})}\}] \quad .w \in \mathbb{R} \end{array} \right.$$

2.2 | Non-Stationary Iterative Methods

Non-stationary iterative methods differ from stationary methods in that the computations involve information that changes at each iteration step. This continuously updated information consists primarily of inner products of residuals or other vectors arising from the method. The best known non-stationary iterative methods for solving linear systems are:

The Conjugate Gradient method (CG) generates a sequence of conjugate vectors which are the residuals of iterates. These vectors are also the gradients of a quadratic functional, the minimization of which is equivalent to the solution of the linear systems. The CG method is extremely effective when the coefficient matrix is symmetric and positive definite [11], [12].

The MINRES method and the SYMMLQ method are alternatives to the CG method which are used if the coefficient matrix is symmetric but possibly indefinite. The SYMMLQ method generates the same solutions as the CG method if the coefficient matrix is symmetric and positive definite [5], [14].

The CGNE method and the CGNR method are specific CG methods for problems with non-symmetric and non-singular coefficient matrices. These methods are based on the fact that the $A^T A$, AA^T and AA^T are always symmetric and positive definite. The CGNE method solves the system $AA^T Y = b$ for any y and then computes the solution $x = A^T Y$. The CGNR method solves $(A^T A)x = \bar{b}$ for x where $\bar{b} = A^T b$. The convergence of these methods may be slow since the spectrum of $A^T A$, AA^T and AA^T will be less favorable than the spectrum of A [5], [13].

The GMRES method computes a sequence of orthogonal vectors (as in the MINRES method), and combines them using a least-squares solve and update. However, unlike the MINRES method, it requires storing the whole sequence, so a large amount of storage is needed. This method is useful for general non-symmetric matrices [12], [14].

The BiCG method generates two sequences of vectors: one based on a system with the original matrix A and one A^T , which are made mutually orthogonal, or bi-orthogonal. The BiCG method is useful when the matrix is non-symmetric and non-singular [12], [13], [15].

The QMR method applies a least-squares solve and update to the BiCG residuals, thereby smoothing out the irregular convergence behavior of the BiCG method. It also largely avoids the breakdowns that can occur in the BiCG method [13], [15].

The CGS method is a variant of the BiCG method that applies the updating operations of both sequences to the same vectors. An advantage is that this method does not need the multiplications by A^T , but on the other hand convergence may be much more irregular than for the BiCG method [13], [14], [16].

The BiCGSTAB method is a variant of the BiCG method, like the CGS method. The difference is that the BiCGSTAB method uses different updates for the sequence corresponding to A^T in order to obtain smoother convergence than the CGS method [13], [14], [17].



3 | Preconditioning Methods for Systems of Linear Equations

A matrix P is called a preconditioner if the matrix P enables a transformation of the linear system into an equivalent system (with the same solution) which has more favorable spectral properties. For *Eq. (1)* preconditioning, transforms the system to

$$PAx = Pb, \quad P \in \mathbb{R}^{n \times n}. \quad (6)$$

Furthermore, it can be transformed to

$$AFy = b, \quad x = Fy, \quad (7)$$

where P and F are linear operators, called left and right preconditioners respectively. And for example, for stationary iterative methods we have:

$$x^{(i+1)} = M_P^{-1} N_P x^{(i)} + M_P^{-1} Pb, \quad i = 0, 1, \dots$$

where $PA = M_P - N_P$ and M_P is nonsingular.

Also

$$y^{(i+1)} = M_F^{-1} N_F y^{(i)} + M_F^{-1} b, \quad i = 0, 1, \dots$$

where $AF = M_F - N_F$ and M_F is nonsingular.

The majority of preconditioners fall in the first category; i.e. *Eq. (6)*.

The purpose of preconditioning is to change the matrix of the system, in order to accelerate the convergence of iterative solvers.

We note that applying a preconditioner involves extra cost and most preconditioners involve an amount of computational work proportional to the number n of variables in their application. They thus multiply the amount of computational work per iteration by a constant factor. On the other hand, the number of iterations is usually only improved by a constant, i.e., the saving in computational works is independent of the matrix size. Therefore, there is a trade-off between the cost of constructing and applying the preconditioner and the gain of increased convergence speed.

To improve the convergence rate of a basic iterative method, various models of preconditioning systems have been proposed. Here, we introduce some models of preconditioning, shortly.

Jacobi Preconditioning ([18],[19])

The Jacobi or point-preconditioner consists of just the diagonal of the matrix A :

$$P := \text{diag}(a_{11}, a_{22}, \dots, a_{nn}).$$

Theoretically, there is no need for any extra storage, but in practice storage is allocated for the reciprocals of the diagonal elements in order to avoid repeated (un-necessary) division operations.

If the index set $J := \{1, \dots, n\}$ is partitioned into disjoint subsets J_i , a block version can be derived:

$$P_{ij} = \begin{cases} a_{ij}, & \text{if } i, j \in J_k \\ 0, & \text{else.} \end{cases}$$

The preconditioner is now a block-diagonal matrix; this is the Jacobi block-preconditioning.

Jacobi preconditioners need very little storage, and they are easy to implement, even on parallel computers. On the other hand, more sophisticated preconditioners usually yield much better improvements in the rate of convergence.

SSOR preconditioning ([5], [20])

If the symmetric matrix A is decomposed as $A = D + L + L^T$, then the SSOR matrix is defined as:

$$P := (D + L)D^{-1}(D + L)^T,$$

or, parameterized by ω ,

$$P_\omega := \frac{1}{2-\omega} \left(\frac{1}{\omega} D + L \right) \left(\frac{1}{\omega} D \right)^{-1} \left(\frac{1}{\omega} D + L \right)^T.$$

Using the optimal value of the relaxation factor ω reduces the number of iterations, but in practice, this optimal value is prohibitively expensive to compute. As P is given in factored form, this method shares many of the properties of preconditioning methods based on incomplete factorizations. Since the factorization of the SSOR preconditioner is given a priori, there is no danger of a breakdown in the construction phase.

Note. Above preconditioners called also preconditioners based on relaxation technique. For example, if the preconditioner P be as $P=D$, then this preconditioner is called Jacobi. Also if $P = \frac{1}{w}(D - wL)$ then we have SOR preconditioner where for $w = I$, we have Gauss–Seidel preconditioner and so on.

Incomplete factorization [13], [20]-[23]

Originally, preconditioners were based on direct solution methods in which part of the computation in skipped. This leads to the notion of incomplete LU (or ILU) factorization.

The construction of an incomplete factorization may break down (due to a division by zero pivot element). The existence of an incomplete factorization is guaranteed for many factorization strategies, if the original matrix A has certain properties [23], [24].

An important consideration for incomplete factorization preconditioners is the cost of the factorization process. These costs bear fruit only if the iterative method without preconditioning converges very slowly, or if the same matrix P can be used for solving several linear systems, as can be done, for instance, in Newton's method for large nonlinear systems with a sparse Jacobian matrix.

ADI preconditioning

In 1994, Starke proposed the use of the Alternating Direction Implicit (ADI) method [25] as a preconditioning technique for the Krylov subspace methods for non-symmetric linear systems. Let $A = A_1 + A_2$, where A_1 and A_2 are nonsingular. The alternative direction implicit method for solving the linear system $Ax = b$ is in the following form:

$$\begin{cases} (A_1 + r_1 I)u_{i+\frac{1}{2}} = b - (A_2 - r_1 I)u_i, \\ (A_2 + r_2 I)u_{i+1} = b - (A_1 - r_2 I)u_{i+\frac{1}{2}}. \end{cases}$$

The ADI preconditioner is as $P := (A_1 + r_1 I)(A_2 + r_2 I)$. Parameters r_1, r_2 are acceleration parameters. Varga [6] and Young [7] presented the optimum value for r_1, r_2 .

preconditioners of (I+S)-type

In the literature, various authors have suggested different models of $(I+S)$ -type preconditioner for the above-mentioned problem. In 1987 Milaszewicz [26] presented the preconditioner $(I+S')$, where the elements of the first column below the diagonal of A eliminate. Gunawardena, Jain and Snyder considered [27] a modification of Jacobi and Gauss-Seidel methods and reported in 1991 that the

convergence rate of Gauss–Seidel method using the following preconditioning matrix is superior to that of the standard Gauss–Seidel method .Where,

$$P=I+S.$$

And,

$$S=(s_{i,j})=\begin{cases} -a_{i,j}, & \text{for } j=i+1, i=1,2,\dots,n-1 \\ 0, & \text{for otherwise.} \end{cases}$$

Kohno et al. [28] proposed an extended modification of Jacobi and Gauss-Seidel methods. Their preconditioner is $(I + s_\alpha)$, where

$$s_\alpha=(s_{ij})_{n \times n}=\begin{cases} -\alpha_i a_{ij}, & j=i+1, 0 \leq \alpha_i \leq 1 \\ 0, & \text{otherwise.} \end{cases}$$

They also presented some numerical investigation for the choice of the optimal parameter. In [29] Usui et al. proposed to adopt

$$P=I+U, (\text{or } I+L),$$

as the preconditioned matrix, where U (L) is strictly upper (strictly lower) triangular of matrix A . They obtained excellent convergence rate compared with that by other iterative methods. Kotakemori et al. in [30] used

$$P=I+S_{\max},$$

where S_{\max} is

$$S_{\max}=(s_{ij}^m)=\begin{cases} -a_{i,V_i}, & \text{for } i=1,2,\dots,n-1, j>i \\ 0, & \text{for otherwise,} \end{cases}$$

and,

$$V_i=\min_{j \in \{j \mid \max_j |a_{ij}|\}} \quad \text{for } i < n.$$

Harano and Niki [31] considered the preconditioner

$$P=I+(1+\gamma)(L+U)$$

where γ is small positive number.

Hadjidimos et al. [32] extended, generalized and compared the previous works. Wang and Song [33] presented a general form of the preconditioners for nonsingular M -matrices. Saberi Najafi and Edalatpanah in [34] established

$$P = I + S_{\min},$$

where for $i = 1 : n-1$ and $j > i$:

$$S_{\min} = \begin{cases} 0, & \text{if } j \in Q_i, \\ -a_{i,j}, & \text{for otherwise,} \end{cases}$$

and,

$$Q_i = \left\{ j \mid j < i \text{ \& } |a_{i,j}| = \min_k |a_{i,k}| \right\} \quad \text{for } i < n-1.$$

Furthermore, some other researchers have considered different models in the literature [35]-[44].

3 | Results

In this section, we apply some iterative methods for solving elliptic PDE with Finite Differences Methods (FDM). Here, we establish our research about this topic.

Consider the following elliptic PDE (advection-diffusion equation):

$$\varepsilon \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + a(x, y) \frac{\partial u}{\partial x} + b(x, y) \frac{\partial u}{\partial y} = f(x, y),$$

With the following bonditions;

Dirichlet on $x=0, y=0$, either Neumann at $y=Y$ and $x=X$ or Dirichlet on $y=Y$ and $x=X$,

where

$$\varepsilon > 0; x \in [0, X] \text{ and } y \in [0, Y] \quad a(x, y) = 1 + x^2; b(x, y) = Xe^{-y},$$

and,

$$f(x, y) = \varepsilon \left(\frac{y}{X^2} e^{-\frac{x}{X}} + e^{-\left(\frac{x}{X} + \frac{y}{Y}\right)} \left(\frac{2}{Y} - \frac{y}{Y^2} - \frac{y}{X^2} \right) \right) + e^{-\frac{x}{X}} \left(Xe^{-y} - \frac{y(1+x^2)}{X} \right) + e^{-\left(\frac{x}{X} + \frac{y}{Y}\right)} \left(\frac{y(1+x^2)}{X} - Xe^{-y} + \frac{X}{Y} y e^{-y} \right).$$

This equation has the following analytical solution:

$$u(x, y) = e^{-x/X} \left(1 - e^{-y/Y} \right) y$$

Based on FDM, we solve the equation after the following differencing:

- I. Central differencing.
- II. Upwind differencing.

To see the convergence behavior of above FDMs by two iterative methods, we select Saberi Najafi and Edalatpanah's method (FIM [5]) as stationary iterative method and a good preconditioned nons-tationary iterative method called the preconditioned (Incomplete LU Decomposition) BiCGSTAB method. And finally, we show:

- Plot the residual reduction.
- Plot the error map.
- Estimate the truncation error.
- Plot the smoothing factor of the iteration.

For the above problem, two types of discretization schemes (centered and hybrid) have been compared for the comparison of the solution.

For both schemes, we choice:

$$\frac{\partial^2 u}{\partial x^2} |_{i,j} = \frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{\Delta x^2} + O(\Delta x^2),$$

$$\frac{\partial^2 u}{\partial y^2} |_{i,j} = \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{\Delta y^2} + O(\Delta y^2).$$

And for hybrid Scheme:

$$\frac{\partial u}{\partial x} |_{i,j} = \frac{U_{i,j} - U_{i-1,j}}{\Delta x} + O(\Delta x),$$

$$\frac{\partial u}{\partial x} |_{i,j} = \frac{U_{i,j} - U_{i,j-1}}{\Delta y} + O(\Delta y).$$

Then by above equation, we get:

$$[\varepsilon \Delta x^2 - b_{ij} \Delta x^2 \Delta y] U_{i,j-1} + [\varepsilon \Delta y^2 - a_{ij} \Delta x \Delta y^2] U_{i-1,j} + [-2\varepsilon \Delta y^2 - 2\varepsilon \Delta x^2 + a_{ij} \Delta x \Delta y^2 + b_{ij} \Delta x^2 \Delta y] U_{i,j} + [\varepsilon \Delta y^2] U_{i+1,j} + [\varepsilon \Delta x^2] U_{i,j+1} = f_{ij} \Delta x^2 \Delta y^2,$$

where,

$$a_{ij} = 1 + x_i^2, b_{ij} = X e^{-y_j}$$

Furthermore, for centered Scheme:

$$\frac{\partial u}{\partial x} |_{i,j} = \frac{U_{i+1,j} - U_{i-1,j}}{2\Delta x} + O(\Delta x)$$

$$\frac{\partial u}{\partial x} |_{i,j} = \frac{U_{i,j+1} - U_{i,j-1}}{2\Delta y} + O(\Delta y)$$

So we obtain,

$$[\varepsilon \Delta x^2 - 0.50b_{ij} \Delta x^2 \Delta y] U_{i,j-1} + [\varepsilon \Delta y^2 - 0.50a_{ij} \Delta x \Delta y^2] U_{i-1,j} + [-2\varepsilon \Delta y^2 - 2\varepsilon \Delta x^2] U_{i,j} + [\varepsilon \Delta y^2 + 0.50a_{ij} \Delta x \Delta y^2] U_{i+1,j} + [\varepsilon \Delta x^2 + 0.50b_{ij} \Delta x^2 \Delta y] U_{i,j+1} = f_{ij} \Delta x^2 \Delta y^2.$$

Now, we present the results:

If we choose $\varepsilon = 4$, $X = Y = 4$, Tolerance=1e-20 and number of mesh size on both direction=20. Then with FIM 5-step with $k=-3.7$, we have:

For dirichlet boundary condition

Figs. 1-2, show the numerical solution of the mentioned equation with two types of discretization schemes.

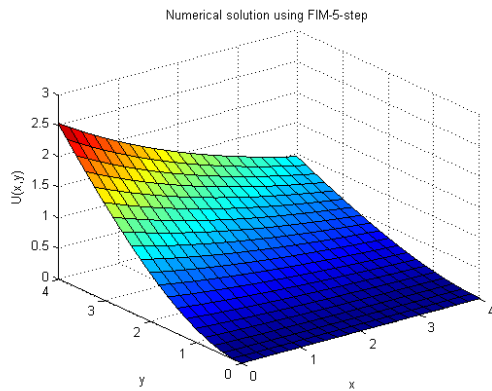


Fig. 1. Solution with upwind (hybrid) scheme.

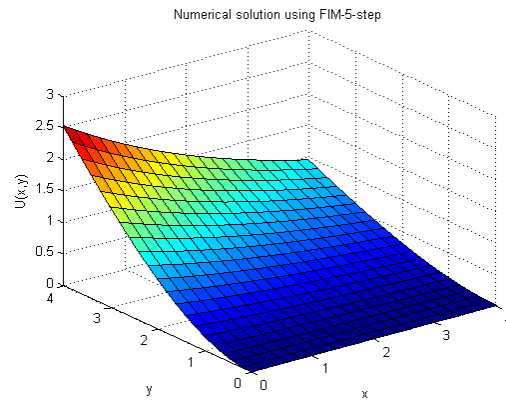


Fig. 2. Solution with centered scheme.

Figs. 3-4, show the error plot for the solution of this problem by FIM and two types of discretization schemes.

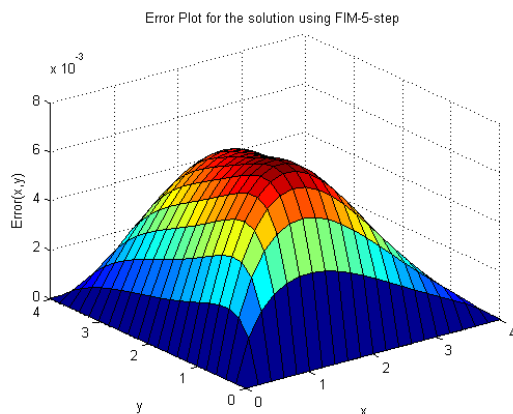


Fig. 3. Error plot with upwind (hybrid) scheme.

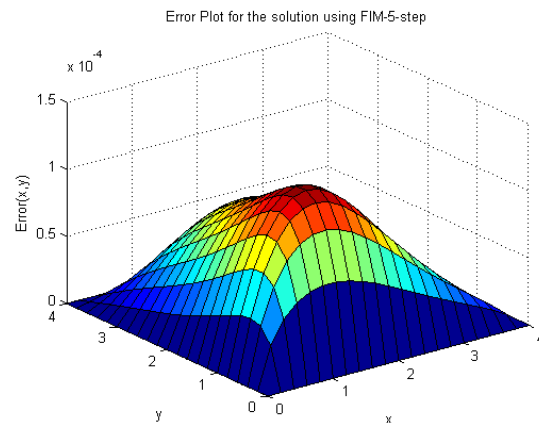


Fig. 4. Error plot with centered scheme.

For neumann boundary condition

Figs. 5-6, show the numerical solution of the mentioned equation with two types of discretization schemes.

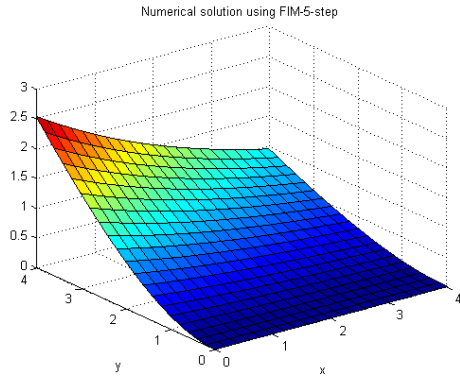


Fig. 5. Solution with upwind (hybrid) schem.

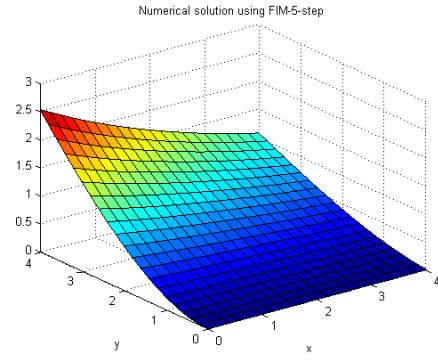


Fig. 6. Solution with centered scheme.

Figs. 7-8, show the error plot for the solution of this problem by FIM and two types of discretization schemes.

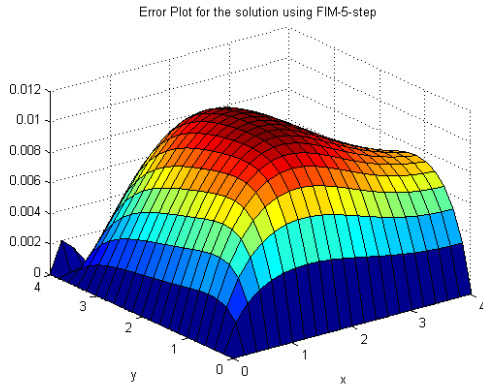


Fig. 7. Solution with centered scheme.

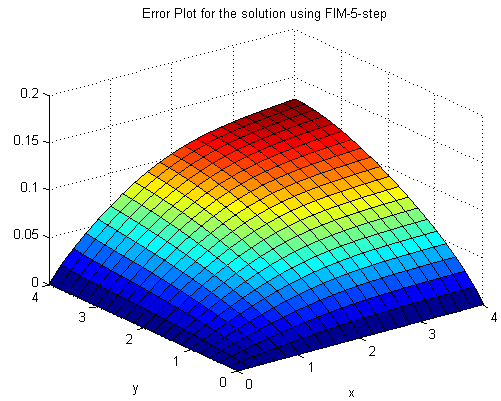


Fig. 8. Solution with centered scheme.

In Table 1, the convergence behaviors of the FIM and a good non-stationary preconditioned method called ILU preconditioned BiCGSTAB method (Pre- BiCGSTAB) are compared by CPU time, number of iterations (Iter) and the maximum error.

Table 1. Comparison of the results between two different iterative methods.

		Dirichlet Boundary Condition		Neumann Boundary Condition	
		Hybrid Scheme	Centered Scheme	Hybrid Scheme	Centered Scheme
Iter	FIM	23	24	128	136
	Pre-BiCGSTAB	35	39	47	55
Maximum Error	FIM	0.0073	1.1829e-04	0.0113	0.1252
	Pre-BiCGSTAB	0.0073	1.1404e-04	0.0110	0.1256
CPU time	FIM	0.1643	0.1688	0.0118	0.3668
	Pre-BiCGSTAB	2.9106	3.2193	3.8977	4.5627

From the result we can see that:

- I. As mentioned in the above table, iteration required for the Neumann case is more as compared to its Dirichlet counterpart.
- II. For the Dirichlet boundary condition, the number of iteration and CPU time for the hybrid scheme is less than Centered scheme. However, the maximum error for the centered scheme is less.
- III. For the Neumann condition, the number of iterations, maximum error and CPU time for the hybrid scheme is less than centered scheme.
- IV. FIM method is superior to the ILU preconditioned BiCGSTAB method.
- V. For low value of epsilon, both methods don't work. The matrix may be become ill-conditioned. To show the convergence behaviors of these methods, here we test the problem with different parameters of ϵ for Dirichlet boundary condition.



Table 2. Results by both methods for epsilon=0.0001.

$\epsilon=0.0001$	Hybrid Scheme	Centered Scheme
FIM	<p>Iter =2, Cpu time = 0.6075 Maximum_Error = 0.0406</p>	Solution oscillates and does not converge after 1000 iteration
	<p>Iter =30, Cpu time =2.6066 Maximum_Error =0.0406</p>	
Pre-BiCGSTAB		



CAND

13

Table 3. Results by the FIM method for epsilon=0.001.

$\epsilon=0.01$	Hybrid Scheme	Centered Scheme
FIM	<div><p>Error Plot for the solution using FIM-5-step</p><p>Iter =5, Cpu time = 0.13817 Maximum_Error = 0.04148</p></div>	Solution oscillates and does not converge after 1000 iteration

Table 4. Results by the Pre-BiCGSTAB method for epsilon=0.001.

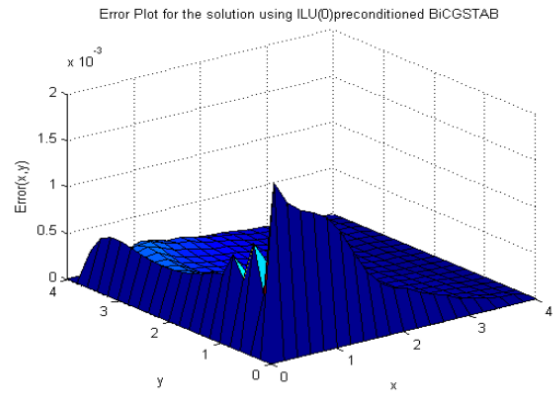
$\epsilon=0.01$	Hybrid Scheme	Centered Scheme
Pre-BiCGSTAB	<div><p>Error Plot for the solution using ILU(0)preconditioned BiCGSTAB</p><p>Iter =29, Cpu time = 2.47187 Maximum_Error = 0.041448</p></div>	Solution oscillates and does not converge after 1000 iteration

Table 5. Results by both methods for epsilon=0.1.

$\epsilon=0.1$	Hybrid Scheme	Centered Scheme
FIM	Solution oscillates and does not converge	Solution oscillates and does not converge

Pre-BiCGSTAB

Solution oscillates and does not converge



Iter =91, Cpu time = 7.636989
Maximum_Error = 0.0018694

Table 6. Results by FIM method for epsilon=1.

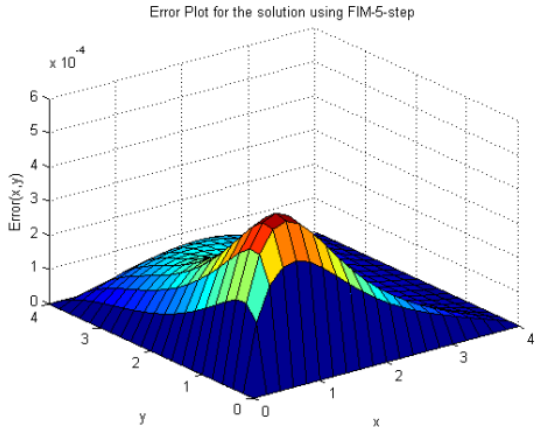
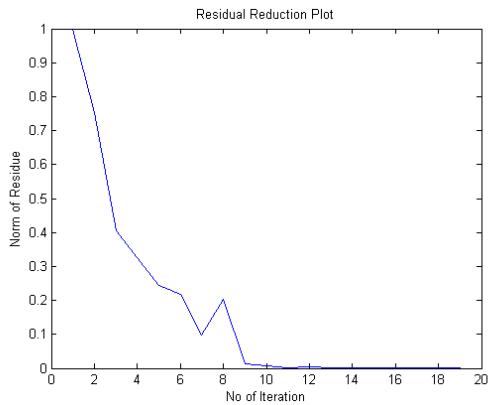
$\epsilon=1$	Hybrid Scheme	Centered Scheme
<p>Error Plot</p> <p>Solution oscillates and does not converge</p>		 <p>Iter =19, Cpu time = 1.653104 Maximum_Error = 4.57593229e-04</p>
<p>Residual Reduction Plot</p> <p>-----</p>		



Table 7. Error plot of the Pre-BiCGSTAB method for epsilon=1.

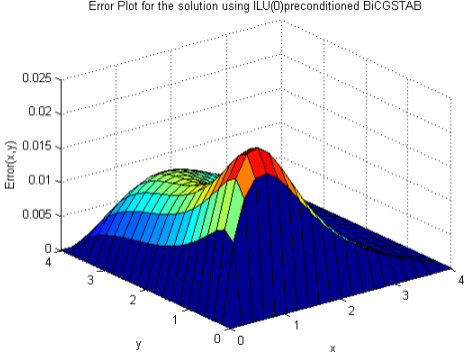
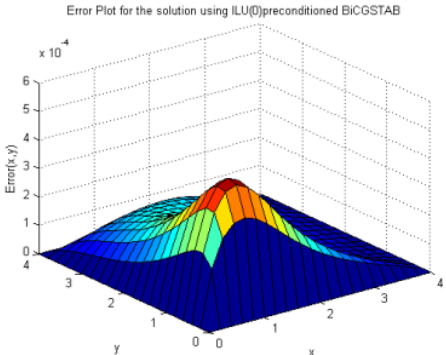
$\epsilon=1$	Hybrid Scheme	Centered Scheme
Error Plot		
	Iter =228, Cpu time= 18.9332 Maximum_Error= 0.0235	Iter =19, Cpu time= 1.6502 Maximum_Error= 4.5759e-04

Table 8. Residual plot of the Pre-BiCGSTAB method for epsilon=1.

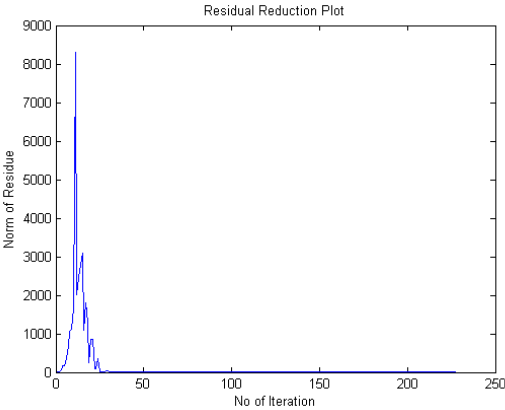
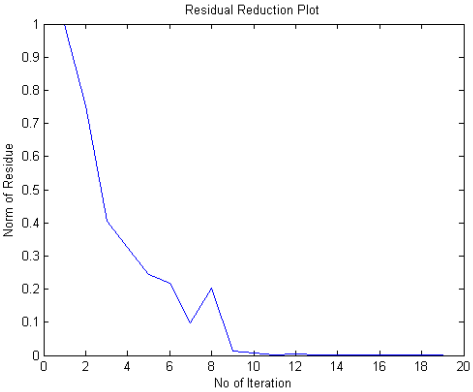
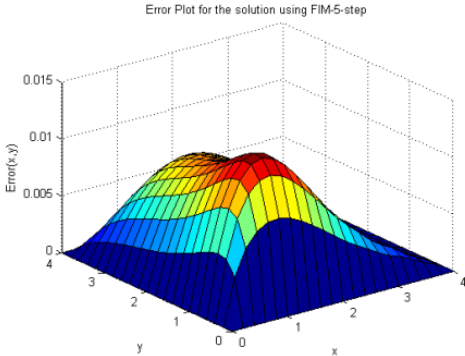
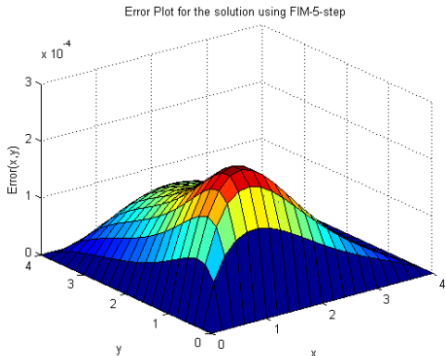
$\epsilon=1$	Hybrid Scheme	Centered Scheme
Residual Reduction Plot		

Table 9. Results by the FIM method for epsilon=2.

$\epsilon=1$	Hybrid Scheme	Centered Scheme
Error Plot		
	Iter =93, Cpu time = 0.301401397 Maximum_Error = 0.0129400359	Iter =18, Cpu time = 0.1635525769 Maximum_Error = 2.398448563e-04
Residual Reduction Plot		

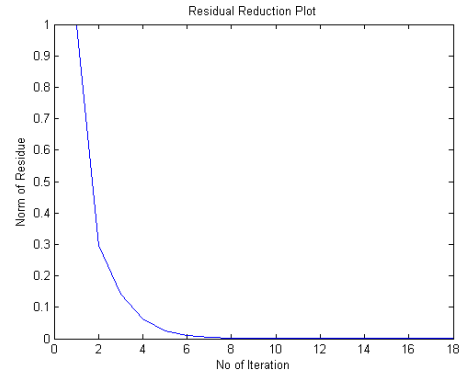
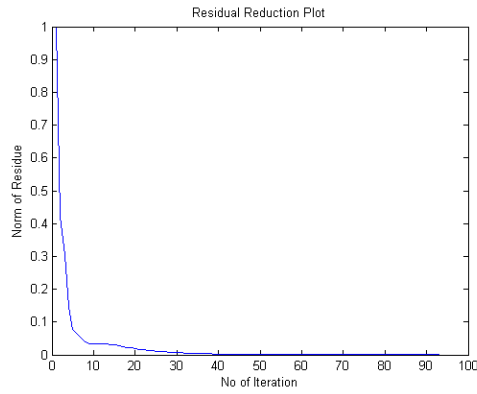
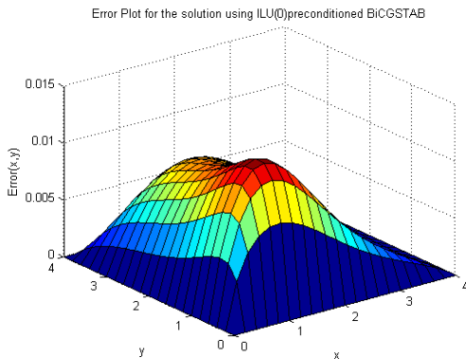
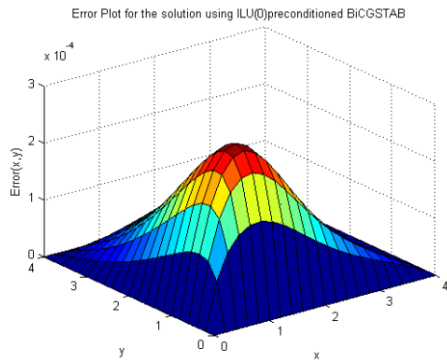
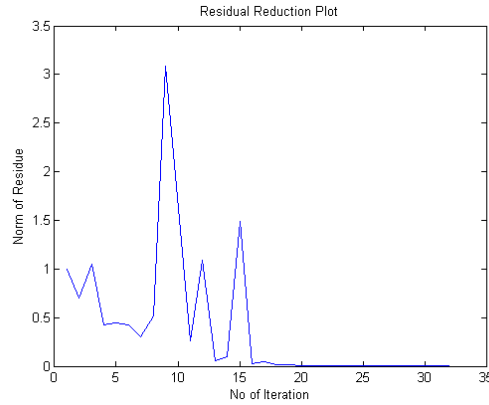
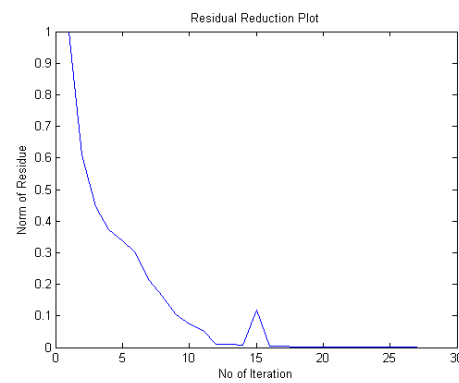


Table 10. Results by the Pre-BiCGSTAB method for epsilon=2.

$\epsilon=2$	Hybrid Scheme	Centered Scheme
Error Plot	 <p>Error Plot for the solution using ILU(0)preconditioned BiCGSTAB. The plot shows a 3D surface of the error function over the domain [0, 4] x [0, 4]. The error is highest in the center and decreases towards the boundaries.</p>	 <p>Error Plot for the solution using ILU(0)preconditioned BiCGSTAB. The plot shows a 3D surface of the error function over the domain [0, 4] x [0, 4]. The error is highest in the center and decreases towards the boundaries.</p>
Residual Reduction Plot	<p>Iter =32, Cpu time = 2.72658957 Maximum_Error = 0.0128128</p>  <p>Residual Reduction Plot for Hybrid Scheme. The y-axis is 'Norm of Residue' (0 to 3.5) and the x-axis is 'No of Iteration' (0 to 35). The plot shows a sharp initial drop in the norm of the residue, followed by a gradual decrease, reaching near zero by iteration 35.</p>	<p>Iter =28, Cpu time = 2.35891506 Maximum_Error = 2.82797714639e-04</p>  <p>Residual Reduction Plot for Centered Scheme. The y-axis is 'Norm of Residue' (0 to 1) and the x-axis is 'No of Iteration' (0 to 30). The plot shows a sharp initial drop in the norm of the residue, followed by a gradual decrease, reaching near zero by iteration 30.</p>

By above results we can conclude that:

- In both methods, for the large value of ϵ , the convergence of the solution is much more ensured. The error is less for large ϵ . The stiffness matrix is well conditioned for large values of ϵ . The stiffness matrix is much more diagonally dominant for the case of higher values of epsilon.
- In both methods, for very low values of ϵ , the centered scheme does not converge. In general, the hybrid scheme converged with very less iteration and the centered scheme converged but with large number of iteration and with high oscillation before convergence.

- III. Convergence area of the ILU preconditioned BiCGSTAB method is superior to the FIM method. However, the speed of convergence of the FIM is the best.
- IV. In both methods, for $\varepsilon \in (.01, 0.9)$, the hybrid scheme does not converge. However, the centered scheme converged.
- V. For higher value of ε , both the scheme converges. However, the error in case of centered scheme is much less as compared to the hybrid scheme.

Next, by $\varepsilon = 4$, $X = Y = 4$, Tolerance= $1e-20$ and number of mesh size on both direction= 20 , we show the residual reduction plots for some iterative methods and hybrid discrization scheme (see *Tables 11-14*).

Table 11. Residual plot of the ILU method [12].

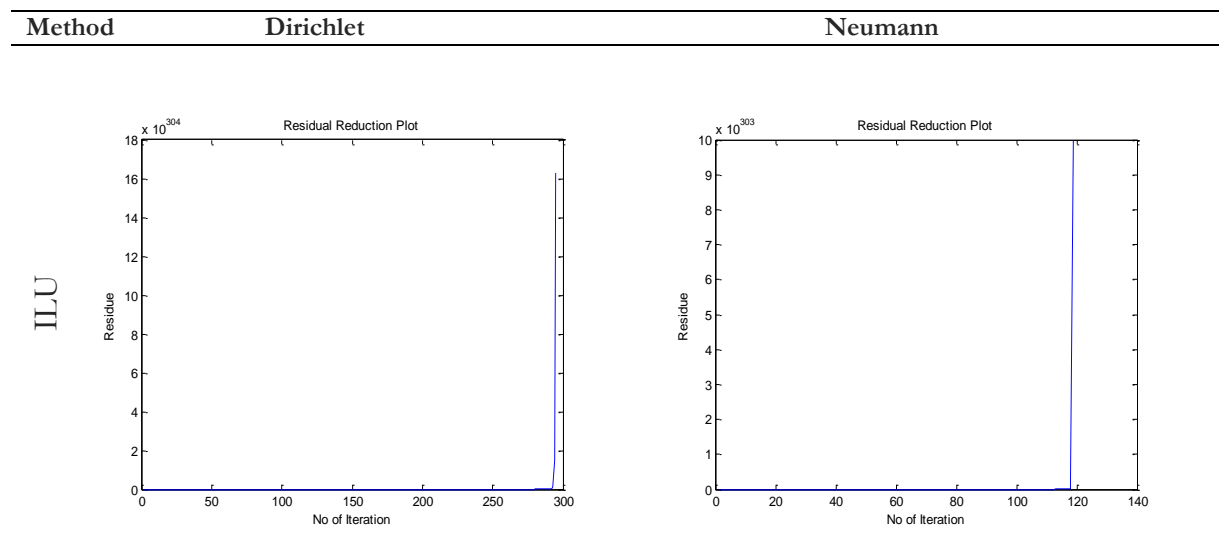


Table 12. Residual plot of the BiCGSTAB method.

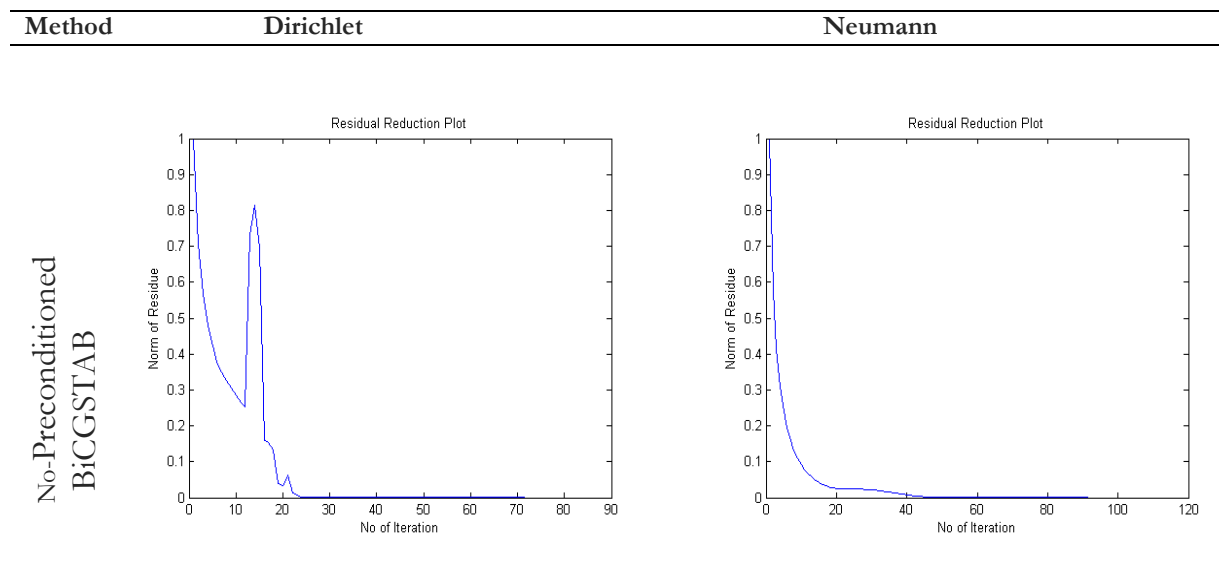


Table 13. Residual plot of the Pre-BiCGSTAB method.

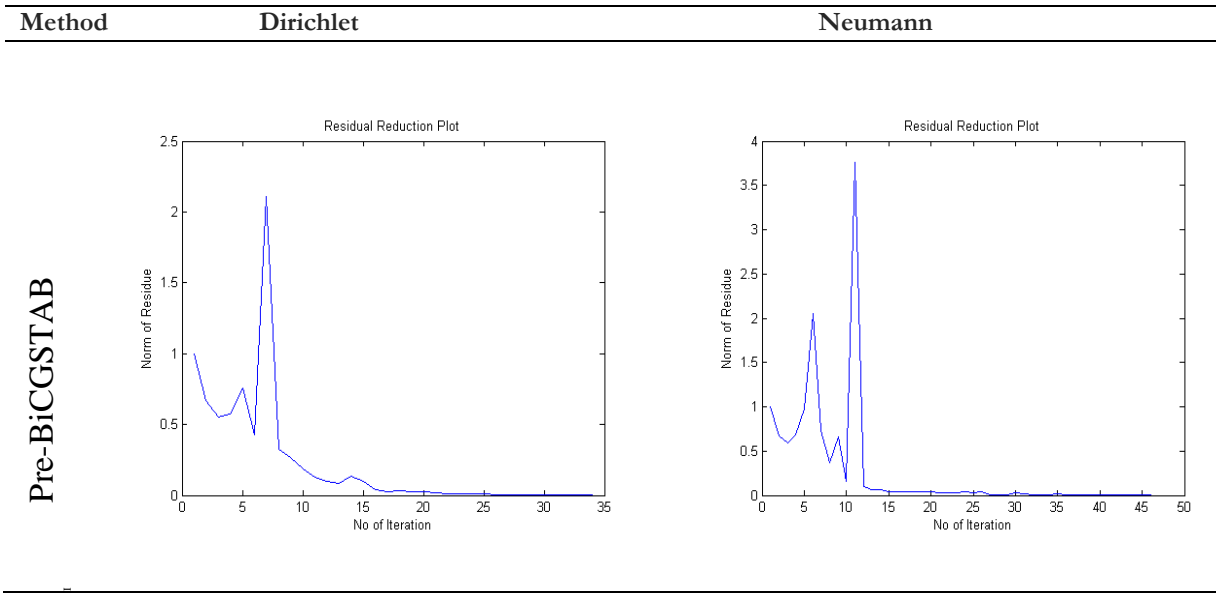
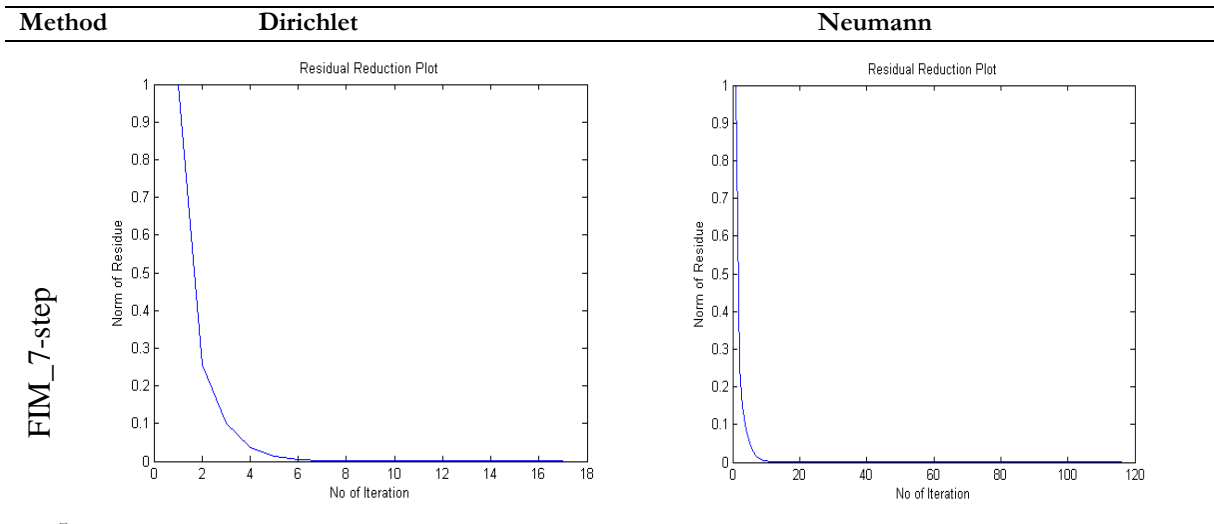


Table 14. Residual plot of the FIM method with $k=-5.5$.



Next, we compare the methods based on amplification factor plot (flow parameter is same as before).

The amplification factor is computed as: $\frac{\|r^{k+1}\|}{\|r^k\|}$, where the index k denotes the number of iterations.

The amplification factor is expected to converge to a value less than 1 (see Tables 15-18).

Table 12. Amplification factor plot of the ILU method [12].

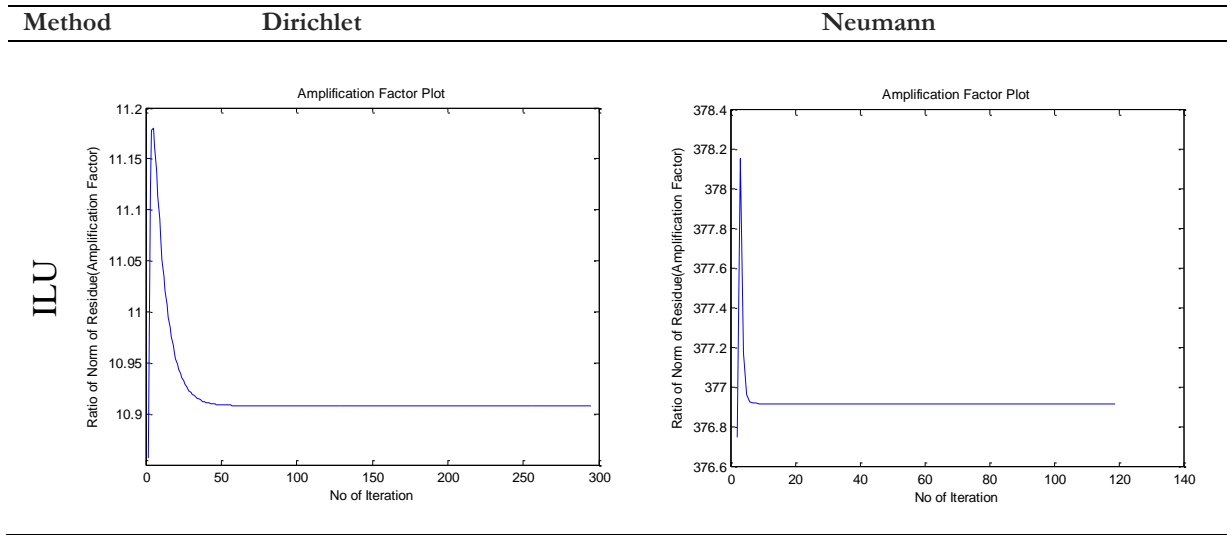


Table 13. Amplification factor plot of the BiCGSTAB method.

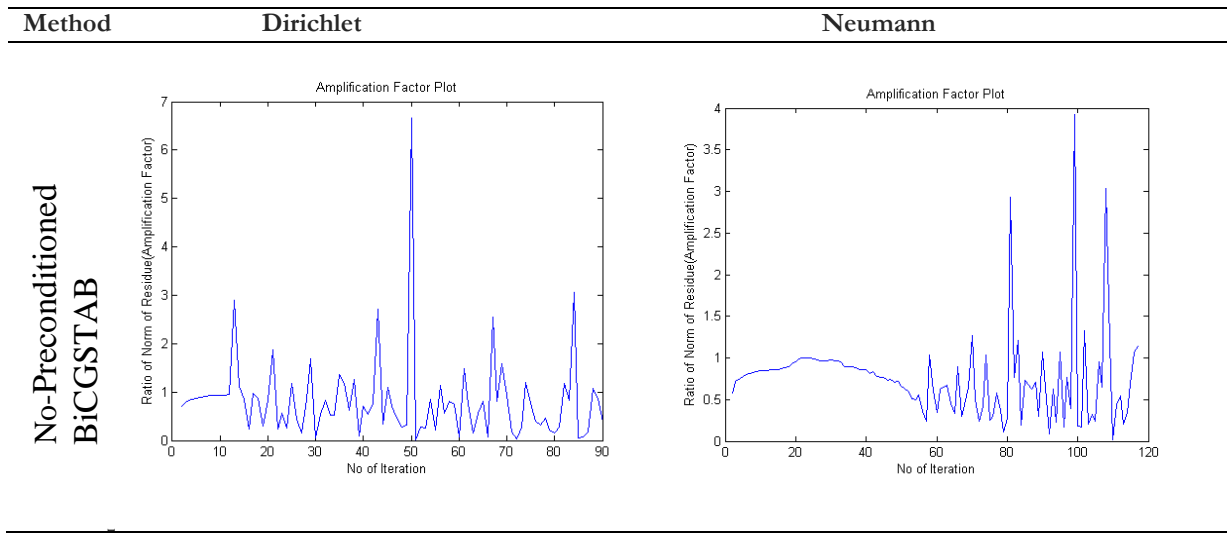


Table 17. Amplification factor plot of the Pre-BiCGSTAB method.

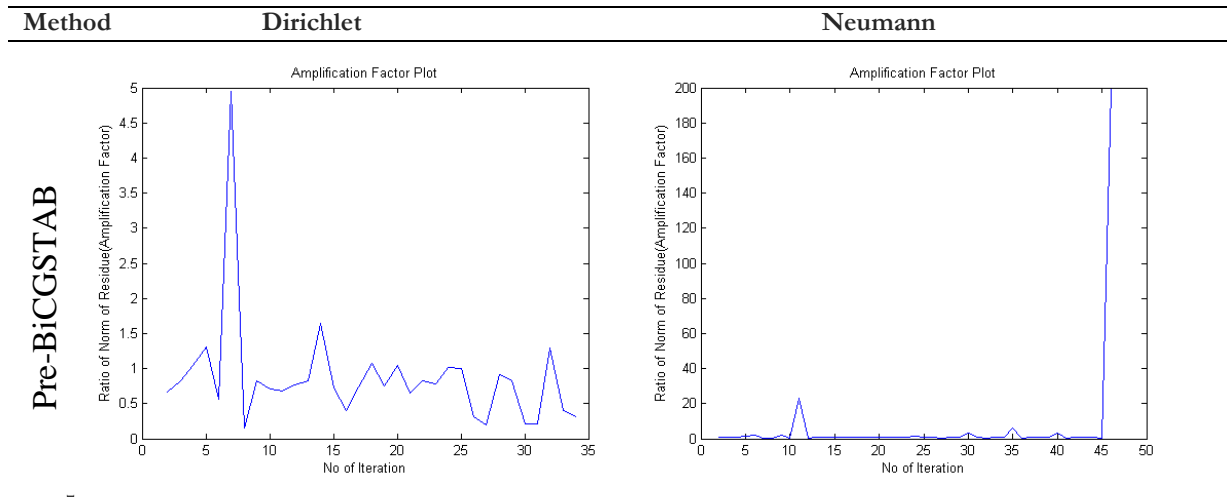


Table 18. Amplification factor plot of the FIM method with $k=-5.5$.

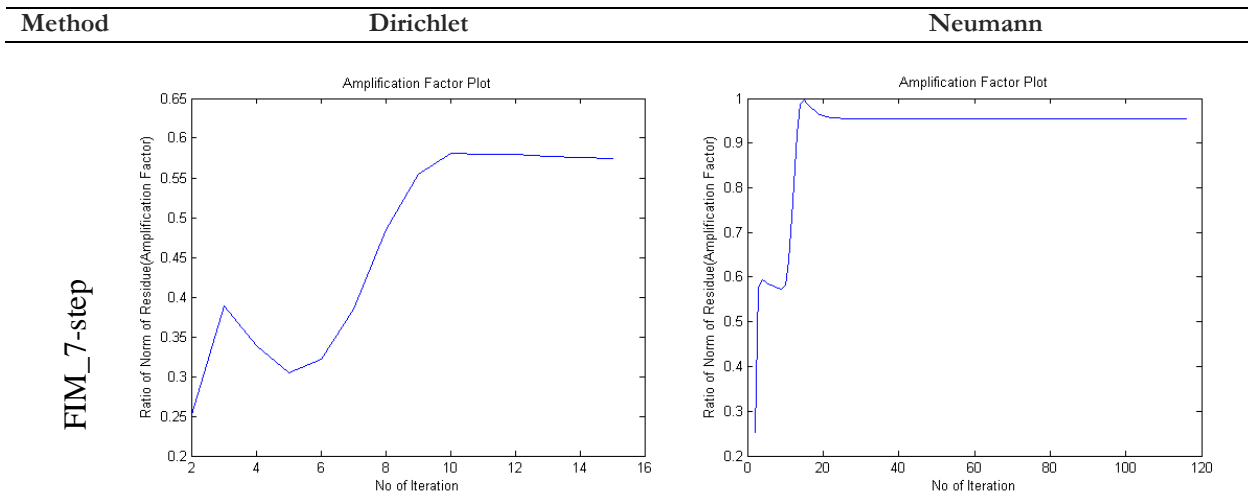


Table 19, shows a total comparison between two models.

Table 14. Brief statistics.

		ILU[7]	BiCGSTAB	Pre-BiCGSATB	FIM_5-step(K=-3.7)
No. of Iteration Required	Dirichlet	NC	91	35	23
	Neumann	NC	118 (with cpu time= 7.5998)	47 (with cpu time= 3.8977)	128 (with cpu time= 0.0118)
Maximum Error	Dirichlet	NC	0.0073	0.0073	0.0073
	Neumann	NC	0.0110	0.0110	0.0113

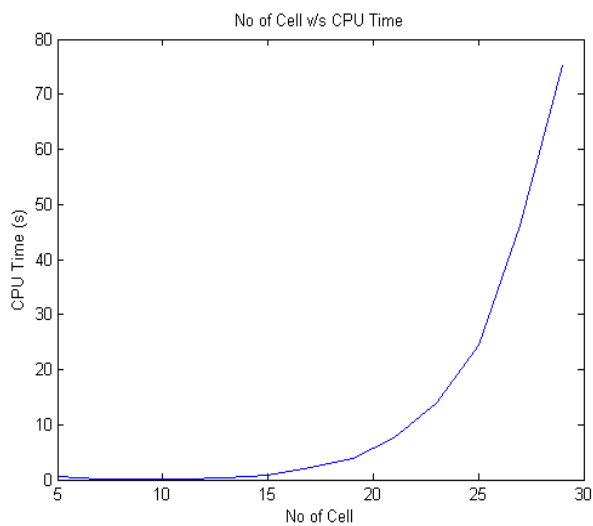


Fig. 1.CPU time for non-preconditioned BiCGSTAB (Hybrid scheme).

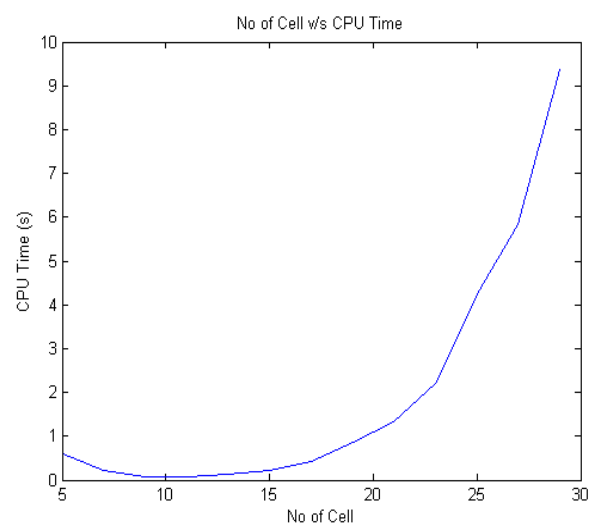


Fig. 2 .CPU time for preconditioned BiCGSTAB (Hybrid scheme).

In all of above results the number of mesh size on both direction is 20. Here, we study the results based on different number of mesh size. In *Figs. 9-12* we show CPU time of different methods with different cell number for hybrid discrization scheme and Dirichlet boundary condition. In *Figs. 13-16* we show CPU time of different methods with different cell number for centered discrization scheme and Dirichlet boundary condition.

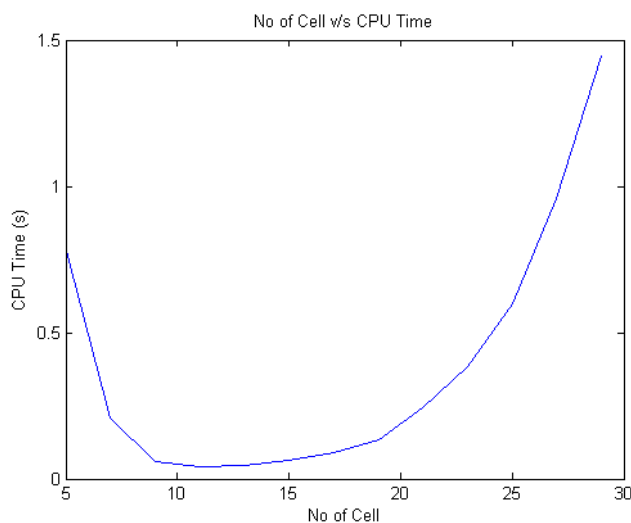


Fig. 3. CPU time for FIM_5-step (Hybrid scheme).

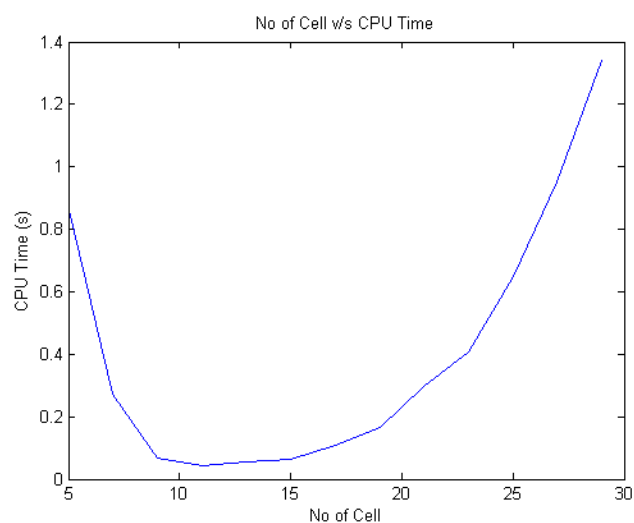


Fig. 4. CPU time for FIM_7 step(k=-5.5) (Hybrid scheme).

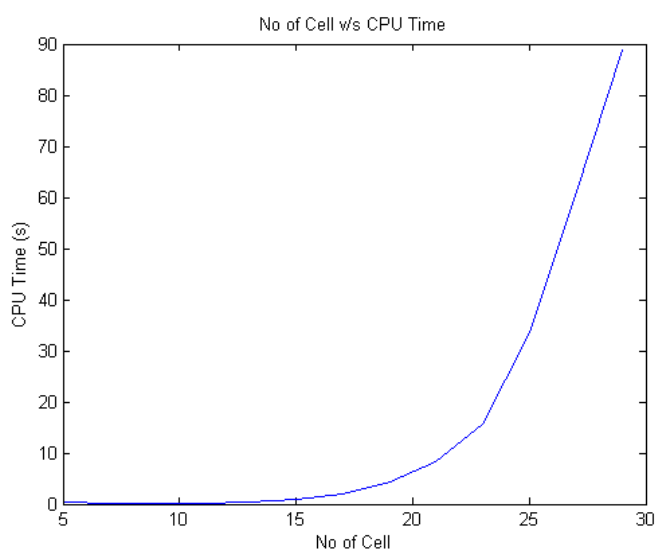


Fig. 5. CPU time for non-preconditioned BiCGSTAB (Centered scheme).

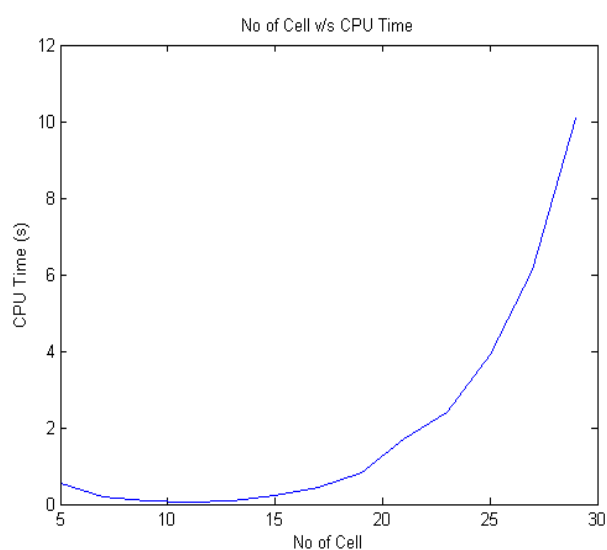


Fig. 6. CPU time for preconditioned BiCGSTAB (Centered scheme).

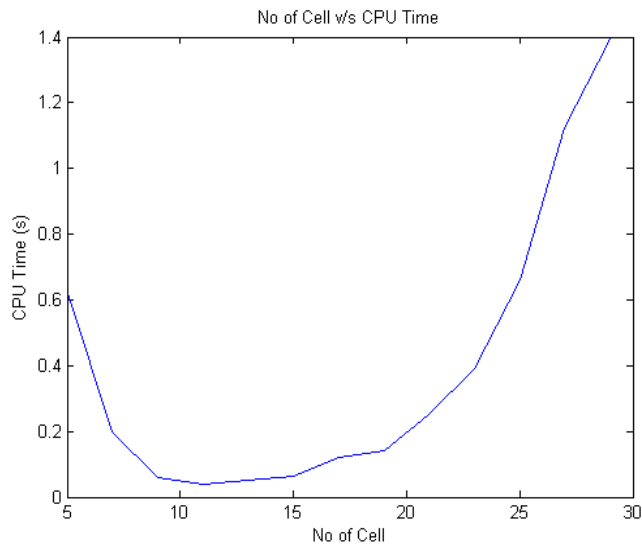


Fig. 7. CPU time for FIM_5-step ($k=-3.7$) (Centered scheme).

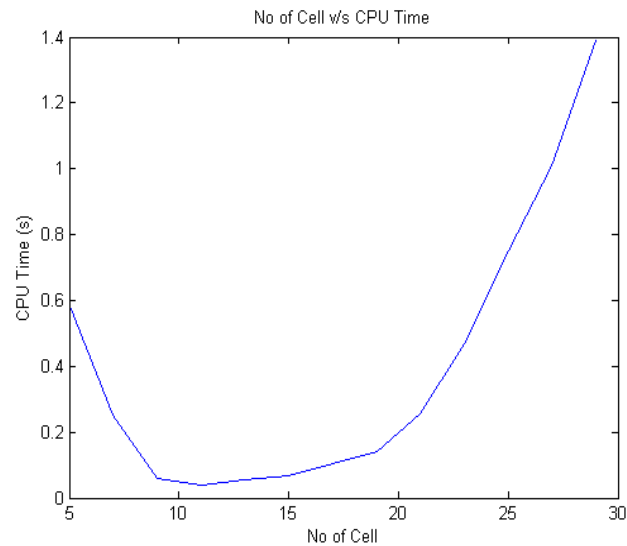


Fig. 8. CPU time for FIM_7 step ($k=-5.5$) (Centered scheme).

From that above figures we can see that:

- I. The time required for case of FIM_7-step is less as compared to all other cases.
- II. The maximum time required is for the case of BiCGSTAB.
- III. Preconditioned BiCGSTAB takes less time as compared to the No preconditioned BiCGSTAB.
- IV. FIM_7-step takes less time as compared to the FIM_5step.
- V. Generally, hybrid discretization scheme takes less time as compared to the centered discretization scheme.

4 | Conclusion

In this paper, we investigated the comparison of two iterative methods for solving the elliptic partial differential equations. Our experiments can be explained as follow:

- I. Among all the methods discussed, FIM method plays well as per as CPU time and convergence and other features are considered.
- II. Large number of cells gives better result. But it takes more time to converge.
- III. Preconditioned used also plays nice role in the CPU time consumption and the converges of the solution is considered.
- IV. The value of ϵ plays a crucial role as depending on the ϵ , the contribution of the diffusion term will determine.

Acknowledgments

The author is most grateful to the anonymous referees for the very constructive criticism on a previous version of this work which significantly improved the quality of the present paper.

Funding

No funding applied for this paper.

Conflicts of Interest

All co-authors have seen and agree with the contents of the manuscript and there is no financial interest to report. We certify that the submission is original work and is not under review at any other publication.

References

- [1] Paige, C. C., & Saunders, M. A. (1975). Solution of sparse indefinite systems of linear equations. *SIAM journal on numerical analysis*, 12(4), 617-629.
- [2] Björck, Å. (1996). *Numerical methods for least squares problems*. Society for Industrial and Applied Mathematics.
- [3] Wang, P., Gao, Y., Yu, N., Ren, W., Lian, J., & Wu, D. (2022). Distributed and communication-efficient solutions to linear equations with special sparse structure. *Systems & control letters*, 160, 105065.
- [4] Edalatpanah, S. A. (2020). Systems of neutrosophic linear equations. *Neutrosophic sets and systems*, 33(1), 92-104.
- [5] Axelsson, O. (1996). *Iterative solution methods*. Cambridge university press.
- [6] Varga, R. S. (1962). *Iterative analysis*. Berlin: Springer.
- [7] Young, D. M. (2014). *Iterative solution of large linear systems*. Elsevier.
- [8] Hadjidimos, A. (1978). Accelerated overrelaxation method. *Mathematics of computation*, 32(141), 149-157.
- [9] Saberi Najafi, H., & Edalatpanah, S. A. (2011). Fast iterative method-FIM. Application to the convection–diffusion equation. *Journal of information and computing science*, 6(4), 303-313.
- [10] Argyros, I. K., & Szidarovszky, F. (2018). *The theory and applications of iteration methods*. CRC Press.
- [11] Concus, P., & Golub, G. H. (1976). A generalized conjugate gradient method for nonsymmetric systems of linear equations. In *Computing methods in applied sciences and engineering* (pp. 56-65). Springer, Berlin, Heidelberg.
- [12] Saad, Y. (2003). *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics.
- [13] Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., ... & Van der Vorst, H. (1994). *Templates for the solution of linear systems: building blocks for iterative methods*. Society for Industrial and Applied Mathematics.
- [14] Saad, Y., & Schultz, M. H. (1986). GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3), 856-869.
- [15] Freund, R. W., & Nachtigal, N. M. (1994). An implementation of the QMR method based on coupled two-term recurrences. *SIAM Journal on scientific computing*, 15(2), 313-337.
- [16] Sonneveld, P. (1989). CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM journal on scientific and statistical computing*, 10(1), 36-52.
- [17] Starke, G., Alternating direction implicit preconditioning for nonsymmetric systems of linear equations, *SIAM J. Sci. Statist. Comput.*, 15(1994) 369–384.
- [18] Meyer, P. D., Valocchi, A. J., Ashby, S. F., & Saylor, P. E. (1989). A numerical investigation of the conjugate gradient method as applied to three-dimensional groundwater flow problems in randomly heterogeneous porous media. *Water resources research*, 25(6), 1440-1446.
- [19] Pini, G., & Gambolati, G. (1990). Is a simple diagonal scaling the best preconditioner for conjugate gradients on supercomputers? *Advances in water resources*, 13(3), 147-153.
- [20] Bruaset, A. M. (2018). *A survey of preconditioned iterative methods*. Routledge.
- [21] Sundar, S., & Bhagavan, B. K. (1999). Comparison of Krylov subspace methods with preconditioning techniques for solving boundary value problems. *Computers & mathematics with applications*, 38(11-12), 197-206.
- [22] Van der Vorst, H. A. (2003). *Iterative Krylov methods for large linear systems* (No. 13). Cambridge University Press.
- [23] Meijerink, J. A., & Van Der Vorst, H. A. (1977). An iterative solution method for linear systems of which the coefficient matrix is a symmetric M -matrix. *Mathematics of computation*, 31(137), 148-162.

- [24] Peaceman, D. W., & Rachford, Jr, H. H. (1955). The numerical solution of parabolic and elliptic differential equations. *Journal of the society for industrial and applied mathematics*, 3(1), 28-41.
- [25] Starke, G. (1994). Alternating direction preconditioning for nonsymmetric systems of linear equations. *SIAM journal on scientific computing*, 15(2), 369-384.
- [26] Milaszewicz, J. P. (1987). Improving jacobi and gauss-seidel iterations. *Linear algebra and its applications*, 93, 161-170.
- [27] Gunawardena, A. D., Jain, S. K., & Snyder, L. (1991). Modified iterative methods for consistent linear systems. *Linear algebra and its applications*, 154, 123-143.
- [28] Kohno, T., Kotakemori, H., Niki, H., & Usui, M. (1997). Improving the modified gauss-seidel method for Z-matrices. *Linear algebra and its applications*, 267, 113-123.
- [29] Usui, M., Niki, H., & Kohno, T. (1994). Adaptive gauss-seidel method for linear systems. *International journal of computer mathematics*, 51(1-2), 119-125.
- [30] Kotakemori, H., Harada, K., Morimoto, M., & Niki, H. (2002). A comparison theorem for the iterative method with the preconditioner (I+ Smax). *Journal of computational and applied mathematics*, 145(2), 373-378.
- [31] Hirano, H., & Niki, H. (2001). Application of a preconditioning iterative method to the computation of fluid flow. *Numerical functional analysis and optimization*, 22(3-4), 405-417.
- [32] Hadjidimos, A., Noutsos, D., & Tzoumas, M. (2003). More on modifications and improvements of classical iterative schemes for M-matrices. *Linear algebra and its applications*, 364, 253-279.
- [33] Wang, L., & Song, Y. (2009). Preconditioned AOR iterative methods for M-matrices. *Journal of computational and applied mathematics*, 226(1), 114-124.
- [34] Saberi Najafi, H., & Edalatpanah, S. A. (2012). New model for preconditioning techniques with application to the boundary value problems. *Journal of advanced research in computer engineering: an international journal*, 6(2), 107-114.
- [35] Wu, M., Wang, L., & Song, Y. (2007). Preconditioned AOR iterative method for linear systems. *Applied numerical mathematics*, 57(5-7), 672-685.
- [36] Yuan, J. Y., & Zontini, D. D. (2012). Comparison theorems of preconditioned Gauss-Seidel methods for M-matrices. *Applied mathematics and computation*, 219(4), 1947-1957.
- [37] Saberi Najafi, H., & Edalatpanah, S. A. (2013). Comparison analysis for improving preconditioned SOR-type iterative method. *Numerical analysis and applications*, 6(1), 62-70.
- [38] Najafi, H. S., & Edalatpanah, S. A. (2013). A collection of new preconditioners for solving linear systems. *Scientific research and essays*, 8(31), 1522-1531.
- [39] Saberi Najafi, H., Edalatpanah, S. A., & Refahisheikhani, A. H. (2018). An analytical method as a preconditioning modeling for systems of linear equations. *Computational and applied mathematics*, 37(2), 922-931.
- [40] Tian, Z., Li, X., Dong, Y., & Liu, Z. (2021). Some relaxed iteration methods for solving matrix equation $AXB = C$. *Applied mathematics and computation*, 403, 126189.
- [41] Liu, Z., Li, Z., Ferreira, C., & Zhang, Y. (2020). Stationary splitting iterative methods for the matrix equation $AXB = C$. *Applied mathematics and computation*, 378, 125195.
- [42] Cui, L. B., Zhang, X. Q., & Wu, S. L. (2020). A new preconditioner of the tensor splitting iterative method for solving multi-linear systems with M-tensors. *Computational and applied mathematics*, 39(3), 1-16.
- [43] Edalatpanah, S. A. (2020). On the preconditioned projective iterative methods for the linear complementarity problems. *RAIRO-operations research*, 54(2), 341-349.
- [44] Mao, X., Wang, X., Edalatpanah, S. A., & Fallah, M. (2019). The monomial preconditioned SSOR method for linear complementarity problem. *IEEE Access*, 7, 73649-73655.