IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.12 ⁜ Vol. 10, Issue 11, November 2022

DOI: 10.17148/IJIREEICE.2022.101104

REVIEW PAPER ON SOLUTION OF LINEAR SIMULTANEOUS EQUATIONS

SAKSHI SHITOLE[1], OMKAR GOTAWALE[2], VISHAL V. MEHTRE[3]

Student of B. Tech, Department of Electrical Engineering, Bharati Vidyapeeth (Deemed To Be University) College of Engineering, Pune, Maharashtra, India[1,2]

Assistant Professor, Department of Electrical Engineering, Bharati Vidyapeeth (Deemed To Be University) College of Engineering, Pune, Maharashtra, India[3]

Abstract: The main purpose of this article is to find solutions to linear simultaneous equations using iterative methods. The iterative methods include the Jacobi and the Gauss-Seidel methods.

A brief description of the Jacobi method and Gauss-Seidel method is done followed by its derivation and programming in MATLAB. A simple code for both Jacobi and Gauss-Seidel methods is given for understanding and knowledge. A question is solved by both methods and the answer for both is verified as well.

Keywords: Jacobi iterations, Gauss Seidel iterations, Linear Simultaneous Equations

INTRODUCTION

In mathematics, a system of linear equations is a system of two linear equations in two or three variables that are solved together to find a common solution to the equations.[3] There are several ways to solve the system of linear equations: B. Elimination, substitution, graphing, etc. Iterative methods are also used to solve systems of linear equations. An iterative procedure is called convergent if the corresponding sequence of equations in the given initial approximation converges. An iterative process is a mathematical process. The iterative process generates a sequence from initial values to improve the approximate solution.[1]

The Jacobi iteration algorithm is different from the Jacobi eigenvalue algorithm. The Jacobi method is named after Carl Gustav Jacob Jacobi. The Jacobi method is an iterative algorithm for finding solutions to diagonally dominant systems of linear equations.[2]

The Gauss-Seidel method is also called the Liebmann method or the successive shift method. It is also one of the iterative methods used to solve systems of linear equations. The Gauss-Seidel method is named after the German mathematicians Carl Friedrich Gauss and Philipp Ludwig von Seidel.[4]

Jacobi iteration and Gauss-Seidel iteration are almost identical. The only difference between the Jacobi method and the Gauss-Seidel method is that the value of the variable does not change until the next iteration in the Jacobi method, whereas the value of the variable does change in the Gauss-Seidel method. When a new value is calculated, it is stored as reversed.[5] Both iterative methods are stationary methods.

DERIVATION:

Let's consider the system of equations having 4 unknowns:

$$a_{11}\ x_1 + a_{12}\ x_2 + a_{13}\ x_3 + a_{14}\ x_4 = b_1$$

$$a_{21}\ x_1 + a_{22}\ x_2 + a_{23}\ x_3 + a_{24}\ x_4 = b_2$$

$$a_{31}\ x_1 + a_{32}\ x_2 + a_{33}\ x_3 + a_{34}\ x_4 = b_3$$

$$a_{41}\ x_1 + a_{42}\ x_2 + a_{43}\ x_3 + a_{44}\ x_4 = b_4$$

These equations will be written as,

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.12 ⁜ Vol. 10, Issue 11, November 2022

DOI: 10.17148/IJIREEICE.2022.101104

$x_1 = 1/a_{11} * [b_1 - a_{12} x_2 - a_{13} x_3 - a_{14} x_4]$

$x_2 = 1/a_{22} * [b_2 - a_{21} x_1 - a_{23} x_3 - a_{24} x_4]$

$x_3 = 1/a_{33} * [b_3 - a_{31} x_1 - a_{32} x_2 - a_{34} x_4]$

$x_4 = 1/a_{44} * [b_4 - a_{41} x_1 - a_{42} x_2 - a_{44} x_3]$

Thus, the variables are expressed in terms of other variables so, Now, Let:

$x_1^{(k)}$ = Value of x_1 in 'k'th iteration;

similarly, $x_2^{(k)}$, $x_3^{(k)}$, $x_4^{(k)}$ values in 'k'th iterations.

Now let,

$x_1^{(k+1)}$, $x_2^{(k+1)}$, $x_3^{(k+1)}$, $x_4^{(k+1)}$ be the values of those variables in next

[(k+1)th] iteration, then equation gives values of the variable in next iteration.

The next iteration will be:

$x_1^{(k+1)} = 1/a_{11} * [b_1 - a_{12} x_2^{(k)} - a_{13} x_3^{(k)} - a_{14} x_4^{(k)}]$

$x_2^{(k+1)} = 1/a_{22} * [b_2 - a_{21} x_1^{(k)} - a_{23} x_3^{(k)} - a_{24} x_4^{(k)}]$

$x_3^{(k+1)} = 1/a_{33} * [b_3 - a_{31} x_1^{(k)} - a_{32} x_2^{(k)} - a_{34} x_4^{(k)}]$

$x_4^{(k+1)} = 1/a_{44} * [b_4 - a_{41} x_1^{(k)} - a_{42} x_2^{(k)} - a_{43} x_3^{(k)}]$

Normally we start at k = 0, which will be the 1st Iteration.

So, the initial values are taken as:

$x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = x_4^{(0)} = 0$

to obtain values in the next iteration.

This system can be extended to more unknowns on the same lines.

The Derivation is same for the Gauss Seidel Method, only in every iteration, the solution is calculated with the latest values.[7]

In Gauss Seidel method,

When we calculate $x2^{(k+1)}$, the value of $x1^{(k+1)}$ can be used. Similarly, to calculate $x4^{(k+1)}$ the value of x1, x2, x3 can be used from present [$(k+1)^{th}$] iteration only.

So, for Gauss Seidel Method the values will be:

$x1^{(k+1)} = 1/a_{11} * [b1 - a_{12} x_2^{(k)} - a_{13} x_3^{(k)} - a_{14} x_4^{(k)}]$

$x2^{(k+1)} = 1/a_{22} * [b2 - a_{21} x_1^{(k+1)} - a_{23} x_3^{(k)} - a_{24} x_4^{(k)}]$

$x3^{(k+1)} = 1/a_{33} * [b3 - a_{31} x_1^{(k+1)} - a_{32} x_2^{(k+1)} - a_{34} x_4^{(k)}]$

$x4^{(k+1)} = 1/a_{44} * [b4 - a_{41} x_1^{(k+1)} - a_{42} x_2^{(k+1)} - a_{43} x_3^{(k+1)}]$

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.12 ⁜ Vol. 10, Issue 11, November 2022

DOI: 10.17148/IJIREEICE.2022.101104

Before proceeding for the solution always it is checked that all the diagonal elements are dominant i.e.,

a_{11} , a_{22} , a_{33} , ..., $a_{(nm)}$ are as far as possible higher than the other elements of their respective columns. This can be said as Partial Pivoting.[8]

JACOBI METHOD IN MATLAB:

```
% Jacobian Method %
A=input ('Enter Coefficient Matrix A: ');
B=input ('Enter Matrix B:');
P=input ('Enter initial Guess Vector:');
n=input ('Enter number of iterations:');
N=length(B);
X=zeros(N,1);
for j=1: n
for i=1: N
X(i)=(B(i)/A(i,i)) - (A (i,[1: i-1, i+1: N]) *P ([1: i-1, i+1: N]))/A(i,i);
End
fprintf('Iteration No %d\n', j)
X
P=X;
end
```

OUTPUT:

```
Enter Coefficient Matrix A:
[4 1 2; 1 3 1; 1 2 5]
Enter Matrix B:
[16 ; 10; 12]
Enter initial Guess Vector:
[0; 0; 0]
Enter number of iterations:
5
Iteration No 1
X =
 4.0000
 3.3333
```

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.12 ⁜ Vol. 10, Issue 11, November 2022

DOI: 10.17148/IJIREEICE.2022.101104

2.4000

Iteration No 2

X =

1.9667
1.2000
0.2667

Iteration No 3

X =

3.5667
2.5889
1.5267

Iteration No 4

X =

2.5894
1.6356
0.6511

Iteration No 5

X =

3.2656
2.2531
1.2279

GAUSS SEIDEL METHOD IN MATLAB :

```
%Gauss-Seidel Method in MATLAB
A=input ('Enter Coefficient Matrix A: ');
B=input ('Enter Matrix B:');
P=input ('Enter initial Guess Vector:');
n=input ('Enter number of iterations:');
e=input ('Enter Your Tolerance:')
N=length(B);
X=zeros(N,1);
for j=1: n
for i=1: N
X(i)=(B(i)/A(i,i)) - (A (i,[1: i-1, i+1: N]) *P ([1: i-1, i+1: N]))/A(i,i);
P(i)=X(i);
```

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.12 ⁜ Vol. 10, Issue 11, November 2022

DOI: 10.17148/IJIREEICE.2022.101104

```
end
fprintf('Iteration No %d\n', j)
X
end
```

OUTPUT:

```
Enter Coefficient Matrix A:
[4 1 2; 1 3 1; 1 2 5]
Enter Matrix B:
[16; 10; 12]
Enter initial Guess Vector:
[ 0; 0; 0]
Enter number of iterations:
5
Enter Your Tolerance:
0.001

e =

 1.0000e-03

Iteration No 1

X =

 4.0000
 2.0000
 0.8000

Iteration No 2

X =

 3.1000
 2.0333
 0.9667

Iteration No 3

X =

 3.0083
 2.0083
 0.9950
```

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.12 ⁜ Vol. 10, Issue 11, November 2022

DOI: 10.17148/IJIREEICE.2022.101104

Iteration No 4

X =

3.0004
2.0015
0.9993

Iteration No 5

X =

3.0000
2.0002
0.9999

EXAMPLES :

Q1. Solve the following system of equations using Jacobi's Iteration method.[9]

$x_1 + 2x_2 + 5x_3 = 12$

$4x_1 + x_2 + 2x_3 = 16$

$x_1 + 3x_2 + x_3 = 10$

Solution: Here there is the need of performing Pivoting. Hence rearrange equations such that diagonal elements are dominant.

The equations arranged will be:

$4x_1 + x_2 + 2x_3 = 16$

$x_1 + 3x_2 + x_3 = 10$

$x_1 + 2x_2 + 5x_3 = 12$

NOW,

$x_1^{(k+1)} = 1/4 * [16 - x_2^{(k)} - 2x_3^{(k)}]$

$x_2^{(k+1)} = 1/3 * [10 - x_1^{(k)} - x_3^{(k)}]$

$x_3^{(k+1)} = 1/5 * [12 - x_1^{(k)} - 2x_2^{(k)}]$

ITERATION NO. 1: Let k = 0 in the iterative equations and take

$x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$ as initial solution.

Therefore, we get,

$x_1^{(1)} = 1/4 * [16 - x_2^{(0)} - 2x_3^{(0)}] = 4$

$x_2^{(1)} = 1/3 * [10 - x_1^{(0)} - x_3^{(0)}] = 10/3 = 3.3333$

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.12 ⁜ Vol. 10, Issue 11, November 2022

DOI: 10.17148/IJIREEICE.2022.101104

$x_3^{(1)} = 1/5 * [12 - x_1^{(0)} - 2x_2^{(0)}] = 12/5 = 2.4$

ITERATION NO. 2: Here k = 1 and the values obtained in ITERATION NO. 1 will be used.

Therefore, we get,

$x_1^{(2)} = 1/4 * [16 - x_2^{(1)} - 2x_3^{(1)}] = 1.9667$

$x_2^{(2)} = 1/3 * [10 - x_1^{(1)} - x_3^{(1)}] = 1.2$

$x_3^{(2)} = 1/5 * [12 - x_1^{(1)} - 2x_2^{(1)}] = 0.2667$

ITERATION NO. 3: Here k = 2 and the values obtained in ITERATION NO. 2 will be used.

Therefore, we get,

$x_1^{(3)} = 1/4 * [16 - x_2^{(2)} - 2x_3^{(2)}] = 3.56667$

$x_2^{(3)} = 1/3 * [10 - x_1^{(2)} - x_3^{(2)}] = 2.58889$

$x_3^{(3)} = 1/5 * [12 - x_1^{(2)} - 2x_2^{(2)}] = 1.526667$

ITERATION NO. 4: Here k = 3 and the values obtained in ITERATION NO. 3 will be used.

Therefore, we get,

$x_1^{(4)} = 1/4 * [16 - x_2^{(3)} - 2x_3^{(3)}] = 2.58944$

$x_2^{(4)} = 1/3 * [10 - x_1^{(3)} - x_3^{(3)}] = 1.635556$

$x_3^{(4)} = 1/5 * [12 - x_1^{(3)} - 2x_2^{(3)}] = 0.65111$

ITERATION NO. 5: Here k = 4 and the values obtained in ITERATION NO.4 will be used.

Therefore, we get,

$x_1^{(5)} = 1/4 * [16 - x_2^{(4)} - 2x_3^{(4)}] = 3.26556$

$x_2^{(5)} = 1/3 * [10 - x_1^{(4)} - x_3^{(4)}] = 2.253146$

$x_3^{(5)} = 1/5 * [12 - x_1^{(4)} - 2x_2^{(4)}] = 1.217887$

Therefore, after 5 iterations we can conclude that the approximate values which can be called the actual values of x_1, x_2, x_3 are:

$x_1 = 3$

$x_2 = 2$

$x_3 = 1$

Q2. Solve the following system of equations using Gauss Seidel Iteration method. [10]

$x_1 + 2x_2 + 5x_3 = 12$

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.12 ⁜ Vol. 10, Issue 11, November 2022

DOI: 10.17148/IJIREEICE.2022.101104

$4x_1 + x_2 + 2x_3 = 16$

$x_1 + 3x_2 + x_3 = 10$

Solution: Here there is the need of performing Pivoting. Hence rearrange equations such that diagonal elements are dominant.

The equations arranged will be:

$4x_1 + x_2 + 2x_3 = 16$

$x_1 + 3x_2 + x_3 = 10$

$x_1 + 2x_2 + 5x_3 = 12$

NOW,

$x_1^{(k+1)} = 1/4 * [16 - x_2^{(k)} - 2x_3^{(k)}]$

$x_2^{(k+1)} = 1/3 * [10 - x_1^{(k)} - x_3^{(k)}]$

$x_3^{(k+1)} = 1/5 * [12 - x_1^{(k)} - 2x_2^{(k)}]$

ITERATION NO. 1: Let k = 0 in the iterative equations and take

$x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$ as initial solution.

Therefore, for x1,

$x_1^{(1)} = 1/4 * [16 - x_2^{(0)} - 2x_3^{(0)}] = 4$

and now as it is Gauss Seidel Method, we will use x_1 value for calculating x_2 and x_1 & x_2 value for calculating x3.

So,

$x_2^{(1)} = 1/3 * [10 - x_1^{(1)} - x_3^{(0)}] = 2$

$x_3^{(1)} = 1/5 * [12 - x_1^{(1)} - 2x_2^{(1)}] = 1.3333$

ITERATION NO. 2: Here k = 1, Therefore we get,

$x_1^{(2)} = 1/4 * [16 - x_2^{(1)} - 2x_3^{(1)}] = 2.83335$

$x_2^{(2)} = 1/3 * [10 - x_1^{(2)} - x_3^{(1)}] = 1.94445$

$x_3^{(2)} = 1/5 * [12 - x_1^{(2)} - 2x_2^{(2)}] = 1.05555$

ITERATION NO. 3: Here k = 2, Therefore we get,

$x_1^{(3)} = 1/4 * [16 - x_2^{(2)} - 2x_3^{(2)}] = 2.9861125$

$x_2^{(3)} = 1/3 * [10 - x_1^{(3)} - x_3^{(2)}] = 1.9861125$

$x_3^{(3)} = 1/5 * [12 - x_1^{(3)} - 2x_2^{(3)}] = 1.0083325$

ITERATION NO. 4: Here k = 3, Therefore we get,

$x_1^{(4)} = 1/4 * [16 - x_2^{(3)} - 2x_3^{(3)}] = 2.999305625$

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.12 ⁜ Vol. 10, Issue 11, November 2022

DOI: 10.17148/IJIREEICE.2022.101104

$x_2^{(4)} = 1/3 * [10 - x_1^{(4)} - x_3^{(3)}] = 1.997453958$

$x_3^{(4)} = 1/5 * [12 - x_1^{(4)} - 2x_2^{(4)}] = 1.00116$

ITERATION NO. 5: Here k = 4, Therefore we get,

$x_1^{(5)} = 1/4 * [16 - x_2^{(4)} - 2x_3^{(4)}] = 3.000056511$

$x_2^{(5)} = 1/3 * [10 - x_1^{(5)} - x_3^{(4)}] = 2.00$

$x_3^{(5)} = 1/5 * [12 - x_1^{(5)} - 2x_2^{(5)}] = 1.00$

After Iterations we can conclude that the values of x_1 , x_2 , x_3 after rounding off to the nearest Number are:

$x_1 = 3$

$x_2 = 2$

$x_3 = 1$

Solving the same question using Jacobi iteration Method and Gauss Seidel Method, the values of x_1 , x_2 , x_3 happen to be almost the same to the very nearest decimal. Hence rounding off the values calculated by both the methods is the same.

RESULT:

Both methods are simple to solve. Codes are easier to perform in MATLAB. The values found by the OUTPUT of the program performed in MATLAB have been matched with the values calculated theoretically. The program and calculations are done up to 5 iterations to verify answers correct to one decimal point. We can perform up to 'n' number of iterations as per the choice.

The rounded off values to the nearest number after 5 iterations by the Jacobi method and the Gauss-Seidel method are:

$x_1 = 3$; $x_2 = 2$; $x_3 = 1$

We also got the same value after performing in MATLAB, hence verifying the values of x_1 , x_2 and x_3 .

Result Table (Jacobi Method):

Iteration No.	k	x_1	x_2	x_3
1	0	4	3.3333	2.4
2	1	1.9667	1.2	0.2667
3	2	3.56667	2.58889	1.526667
4	3	2.58944	1.635556	0.651111
5	4	3.26556	2.253146	1.217887

Result Table (Gauss-Seidel Method):

Iteration No.	k	x_1	x_2	x_3
1	0	4	2	1.3333
2	1	2.83335	1.94445	1.05555
3	2	2.9861125	1.9861125	1.0083325
4	3	2.999305625	1.997453958	1.00116
5	4	3.000056511	2.00	1.00

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.12 ⁜ Vol. 10, Issue 11, November 2022

DOI: 10.17148/IJIREEICE.2022.101104

CONCLUSION

The Jacobi method is based on solving every variable locally with respect to the other variables. One iteration of the method corresponds to solving for every variable once. The resulting method is easy to understand and implement, but convergence is slow.

The Gauss Seidel method is like the Jacobi method, except that it uses the latest values as soon as they are calculated. In general, the Gauss Seidel method converges faster than the Jacobi method, though still relatively slowly.

Less number of Iterations are required in Gauss Seidel Method as compared to Jacobi Method to calculate and obtain the values correct to one decimal place.

The Gauss Seidel Method gives more finite values than the Jacobi Method.

REFERENCES

[1] Amritkar, Amit; de Sturler, Eric; Świrydowicz, Katarzyna; Tafti, Danesh; Ahuja, Kapil (2015). "Recycling Krylov subspaces for CFD applications and a new hybrid recycling solver". Journal of Computational Physics.
[2] Saad, Yousef (2003). Iterative Methods for Sparse Linear Systems
[3] B.N.Datta, Numerical Linear Algebra and Applications, Pacific Grove.
[4] F.Naeimi Dafchahi, A new refinement of Jacobi method for Ax=b, vol.3, no.17, 819-827, Iran (2008).
[5] Gerald R. Morris and Viktor K. Prasanna, "An FPGA-Based Floating-Point Jacobi Iterative Solver", Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms and Networks 2005 IEEE.
[6] Huabin Ruan, 2 Xiaomeng Huang, 3 Haohuan Fu and 4 Guang Wen Yang, "Jacobi Solver: A Fast FPGA-based Engine System for Jacobi Method", Research Journal of Applied Sciences, Engineering and Technology 6(23): 4459-4463, 2013 ISSN: 2040-7459; e-ISSN: 2040-7467 © Maxwell Scientific Organization, 2013.
[7] Linear Algebra: Numerical Methods. Version: August 12, 2000
[8] Harpinder Kaur, Khushpreet Kaur, "Convergence of Jacobi and Gauss-Seidel Method and Error Reduction Factor", IOSR Journal of Mathematics (IOSRJM) ISSN: 2278-5728 Volume 2, Issue 2 (July-Aug 2012), PP 20-23.
[9] Sinan Shi, "GPU Implementation of Iterative Solvers in Numerical Weather Prediction Models", The University of Edinburgh, 2012.
[10] Mareike Schmidtobreick 1, Florian Wilhelm 1, Fabian Nowak 2, Vincent Heuveline 2, Wolfgang Karl 2, "Employing a high-level language for porting Numerical Applications to Reconfigurable Hardware", Preprint Series of the Engineering Mathematics and Computing Lab (EMCL) ISSN 2191–0693 No. 2011-13.