

Jahangirnagar University
Department of Computer Science and Engineering

LAB REPORT
ON
CSE-206 (NUMERICAL METHODS LAB)

EXPERIMENT No.: 05

EXPERIMENT NAME: Determining the solution of system of Linear Equations using Jacobi Iteration Method.



SUBMITTED BY Sumaita Binte Shorif Class Roll: 357 Exam Roll: 191338 Submission Date: Sept 04, 2021			SUBMITTED TO Dr. Md. Golam Moazzam Professor Dept. of Computer Science & Engineering Jahangirnagar University
---	--	--	--

Experiment No: 05

Name of the Experiment: Determining the solution of system of Linear Equations using Jacobi Iteration Method.

Objectives:

- ✚ Understanding the process of solving system of linear equations using Jacobi Iteration Method.
- ✚ Executing the implementation of the algorithm of Jacobi Iteration Method.
- ✚ To achieve the accurate root and expected output of given system of linear equations.
- ✚ To be able to interpret the advantages and disadvantages of Jacobi Iteration Method.

Theory:

In numerical mathematics, the Jacobi-iteration method is an iterative algorithm for determining the solutions of a strictly diagonally dominant system of linear equations. It is one of simplest iterative algorithms to find roots of linear system of equations. Each diagonal element is solved for, and an approximate value is plugged in. The process is then iterated until it converges.

Let us consider a system of n equations in n unknowns.

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2$$

.

.

.

$$a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n$$

We rewrite the original system as,

$$x_1 = \frac{b_1 - (a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n)}{a_{11}}$$

$$x_2 = \frac{b_2 - (a_{21}x_1 + a_{23}x_3 + \dots + a_{2n}x_n)}{a_{22}}$$

.

.

.

$$x_n = \frac{b_n - (a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_{n-1})}{a_{nn}}$$

Now we can compute x_1, x_2, \dots, x_n by using initial guesses for these values. These new values are again used to compute the next set of x values. The process can continue till we obtain a desired level of accuracy in the x values.

Algorithm:

Jacobi Iteration Method:

1. Obtain n , a_{ij} and b_i values.
2. Set $x_{0i} = b_i / a_{ii}$ for $i = 1, 2, \dots, n$
3. Set $key = 0$.
4. For $i = 1, 2, \dots, n$ i.
 - i. Set $sum = b_i$.
 - ii. For $j = 1, 2, \dots, n$ ($j \neq i$) Set $sum = sum - a_{ij} x_{0j}$ Repeat j
 - iii. Set $x_i = sum / a_{ii}$
 - iv. If $key = 0$ then,
 - If $|(x_i - x_{0i}) / x_i| > error$,
 - Then Set $key = 1$;
- Repeat i
5. If $key = 1$ then Set $x_{0i} = x_i$ Go to step 3
6. Write results.

Coding in C:

```
#include<stdio.h>
#include<math.h>
#define EPS 0.000001
#define MAXIT 200

void Jacobi(int n,float a[10][10],float b[10], float x[10], int
           *count, int *status)
{
    int i,j,key;
    float sum, x0[10];

    for(i=1; i<=n; i++)
        x0[i]=b[i]/(a[i][i]*1.0);

    *count=1;
    while(1)
    {
        key=0;
        /*Computing values of x[i]
        -----
        x1=(b1-a12 x2-a13 x3-.....a1n xn)/ a11
        x2=(b2-a21 x1-a23 x3-.....a2n xn)/a22
        */
        for(i=1; i<=n; i++)
        {
            sum=b[i];
            for(j=1; j<=n; j++)
            {
                if(i==j)
                    continue;
                sum=sum-a[i][j]*x0[j];
            }
            x[i]=sum/(a[i][i]*1.0);
            if(key==0)
            {
                //Testing for accuracy
                if(fabs((x[i]-x0[i])/x[i])>EPS)
                    key=1;
            }
        }

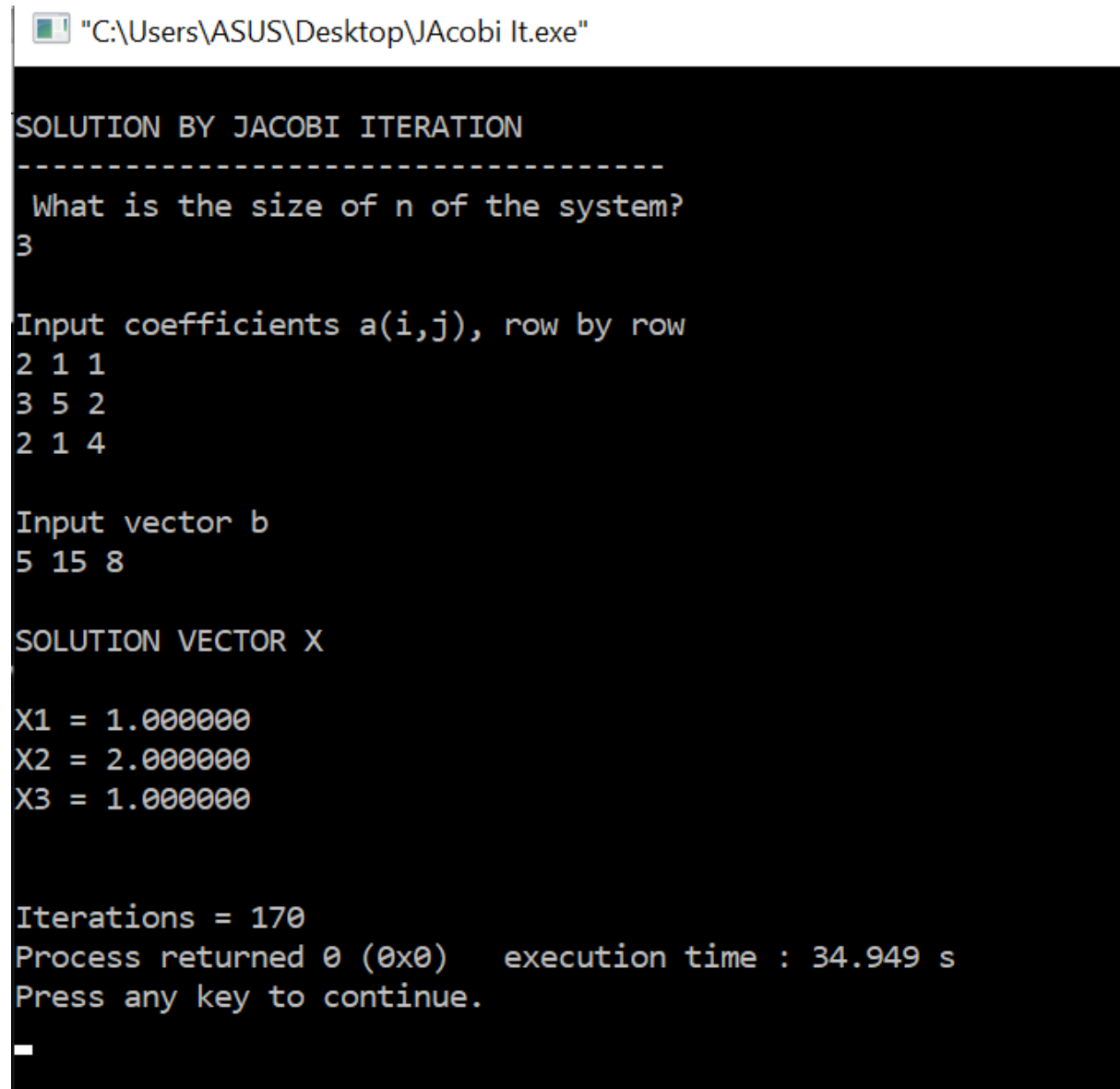
        if(key==1)
        {
            if(*count==MAXIT)
            {
                *status=2;
                return;
            }
            else
            {
                *status=1;
                for(i=1; i<=n; i++)
                {
                    x0[i]=x[i];
                }
            }
        }
    }
}
```

```

        }
        *count=*count+1;
    }
    else
    {
        break;
    }
}
return;
}
int main()
{
    int i,j,n,count,status;
    float a[10][10],b[10],x[10];
    printf("\nSOLUTION BY JACOBI ITERATION \n");
    printf("-----");
    printf("\n What is the size of n of the system? \n");
    scanf("%d",&n);
    printf("\nInput coefficients a(i,j), row by row \n");
    for(i=1; i<=n; i++)
    {
        for(j=1; j<=n; j++)
        {
            scanf("%f",&a[i][j]);
        }
    }
    printf("\nInput vector b\n");
    for(i=1; i<=n; i++)
    {
        scanf("%f",&b[i]);
    }
    Jacobi(n,a,b,x,&count,&status);
    if(status==2)
    {
        printf("\nNo convergence in %d iterations",MAXIT);
        printf("\n\n");
    }
    else
    {
        printf("\nSOLUTION VECTOR X\n\n");
        for(i=1; i<=n; i++)
        {
            printf("X%d = %.6f\n",i,x[i]);
        }
        printf("\n\nIterations = %d ",count);
    }
    return 0;
}

```

Output:



```
"C:\Users\ASUS\Desktop\Jacobi It.exe"

SOLUTION BY JACOBI ITERATION
-----
What is the size of n of the system?
3

Input coefficients a(i,j), row by row
2 1 1
3 5 2
2 1 4

Input vector b
5 15 8

SOLUTION VECTOR X

X1 = 1.000000
X2 = 2.000000
X3 = 1.000000

Iterations = 170
Process returned 0 (0x0)   execution time : 34.949 s
Press any key to continue.
■
```

Fig 01: Output Obtained.

Discussion:

As we can see from above experiment, the more the iterations, the more the solution is accurate. Jacobi Iteration method can solve any n by n system of linear equations. This method takes too many iterations and gives the correct output.

Note that the simplicity of this method has both advantages and disadvantages. The method is preferred because it is relatively easy to understand and thus is a good first taste of iterative methods.
