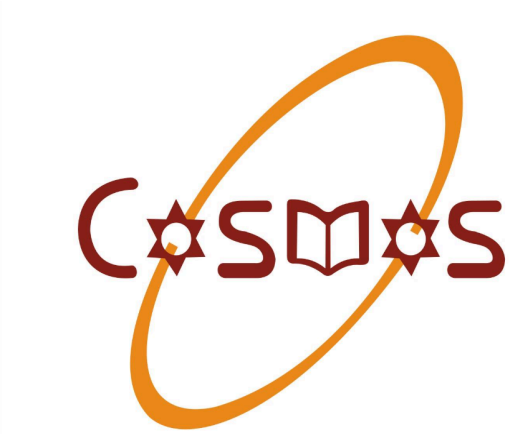
Cosmos College of Management and Technology
Affiliated to Pokhara University
Saddobato, Lalitpur



Lab Report on: Implementation of Gauss Jacobi & Gauss Seidel method

Lab report number: 06

Submitted By

Name: Aayush Karki

Roll num: 200101

Faculty: BE IT

Semester: V

Sub: Numerical Method

Group: A

Submitted To

Department of ICT

2080/10/05

.....

Lab Number: 06

Lab Title: Implementation of Gauss Jacobi & Gauss Seidel method

Lab Objective

1. Understand the concept of Gauss Jacobi & Gauss Seidel method
2. Implement the methods in C programming
3. Know the basic difference between Gauss Jacobi & Gauss Seidel method.

Theory

Gauss Jacobi method is an iterative algorithm used to solve a system of linear equations. It is named after the German mathematician Carl Gustav Jacob Jacobi. It finds the solution of a system of linear equations in multiple iterations. Here in the first iteration, the x, y and z are assumed to be zero(0) where these values are used in the equation that is formed based on the diagonal dominance of the equation. Later on from the second iteration, the value obtained in the first iteration is used and this same process continues until two of the rows have the same value till that decimal place provided in the question. The simple demonstration of Gauss Jacobi method is as follows:

Let's find the solution of the given linear equation:

20x + y - 2z = 17

3x + 20y - 1z = -18

2x - 3y + 20z = 25

Solution:

Here we can see that the equation is in diagonally dominant order. The diagonally dominant rules says that, for the first row: |d1|>|d2|+|d3|, for the second row: |d2|>|d1|+|d3|, for the third row: |d3|>|d1|+|d2|. If the equations are not in diagonal dominant order, just souffle it till we get the order.

Here

20x + y - 2z = 17	3x + 20y - 1z = -18	2x - 3y + 20z = 25
or, x = (17 - y + 2z)/2 i	or, y = (-18 - 3x + z)/20 ii	or, z = (25 - 2x + 3y)/20.....iii

S.N.	x = (17 - y + 2z)/2	y = (-18 - 3x + z)/20	z = (25 - 2x + 3y)/20
1	0.8500	-0.9000	1.2500
2	1.02	-0.9650	1.0300
3	1.0013	-1.0015	1.0033
4	1.0004	-1	0.9996
5	1	-1.0001	1
6	1	-1	1

Here we can see that the value of row 5 and 6 is the same till the third place after the decimal point, hence we got the required solution as: x=1, y=-1, z=1.

Gauss Seidel method is an iterative method used to solve a system of linear equations. It is named after the German mathematicians Carl Friedrich Gauss and Philipp Ludwig von Seidel, and is similar to the Jacobi method but the only difference is that here the value obtained after substituting is used in the same iteration if possible or if not, then in the next iteration the previous value is used.

For example: in the 1st iteration, the value of X is found out using Y=0 & Z=0 and the obtained value of X is used to find the value of Y where X=obtained value and Z=0 and at last the value of Z is found out using the obtained values of X & Y. Now the same process continues till we get our required solution.

S.N.	X1 = (58 - 2X2 + 3X3)/45	X2 = (47 + 3X1 - 2X3)/22	X3 = (67 - 5X1 - X2)/20
1	1.2889	2.3121	2.9122
2	0.9920	2.0069	3.0017
3	0.9996	1.9998	3.0001
4	1	2	3
5	1	2	3

ALGORITHM

For the Gauss Jacobi method, the algorithm to compute the solution is as follows:

1. START
2. Get the equation and find the equation of x, y and z based on which is the diagonal element of the equation.
3. Now find the value of x, y and z where for the first iteration x=y=z=0 and from the next iteration the value obtained in the previous iteration is used.
4. Continue the process of computation till we get the same value up to the required decimal place , for at least 2 rows
5. Display the obtained result
6. STOP

For the Gauss Seidel method, the simple algo is:

1. START
2. Get the equation and find the equation of x, y and z based on which is the diagonal element of the equation.
3. Now find the value of x, y and z where for the first finding of the first iteration x=y=z=0 and from the next finding, the value obtained in the previous finding is used.
4. Continue the process of computation till we get the same value up to the required decimal place , for at least 2 rows
5. Display the obtained result
6. STOP

CODE

```
// Coding the Gauss Jacobi Method
#include <stdio.h>
#include <math.h>

//Macro definition
#define f1(x, y, z) (15 - y - z) / 10
#define f2(x, y, z) (24 - x - z) / 10
#define f3(x, y, z) (15 - x - y) / 10
```

```
int main()
{
	float x0 = 0, y0 = 0, z0 = 0, x1, y1, z1, e1, e2, e3, e;
	int i = 1;
	// Till how many decimal points we want the check to be done, it is provided
	printf("Enter the error allowed: ");
	scanf("%d", &e);
	do
	{
		x1 = f1(x0, y0, z0);
		y1 = f2(x0, y0, z0);
		z1 = f3(x0, y0, z0);

		e1 = fabs(x0 - x1);
		e2 = fabs(y0 - y1);
		e3 = fabs(z0 - z1);
		i++;

		x0 = x1;
		y0 = y1;
		z0 = z1;
	} while (e1 > e || e2 > e || e3 > e);

	printf("The solution is: x=%f, y=%f, z=%f\n", x1, y1, z1);
	return 0;
}
```

Output:

```
D:\StudyMaterial\FifthSem\NumericalMethods\NM_Lab>cd "d:\StudyMaterial\FifthSem\Numeric
alMethods\NM_Lab\LabSix\" && gcc LabSixFirst.c -o LabSixFirst && "d:\StudyMaterial\Fift
hSem\NumericalMethods\NM_Lab\LabSix\"LabSixFirst
Enter the error allowed: 0.0001
The solution is: x=1.166667, y=2.166667, z=1.166667

d:\StudyMaterial\FifthSem\NumericalMethods\NM_Lab\LabSix>
```

// Coding the Gauss Seidel Method

```
#include <stdio.h>
```

```
#include <math.h>

// Function prototypes
float f1(float y, float z);
float f2(float x, float z);
float f3(float x, float y);

int main()
{
	float x0 = 0, y0 = 0, z0 = 0, x1, y1, z1, e1, e2, e3, e;
	int i = 1;

	// Till how many decimal points we want the check to be done, it is
	provided
	printf("Enter the error allowed: ");
	scanf("%f", &e);

	do
	{
	x1 = f1(y0, z0);
	y1 = f2(x1, z0);
	z1 = f3(x1, y1);

	e1 = fabs(x0 - x1);
	e2 = fabs(y0 - y1);
	e3 = fabs(z0 - z1);
	i++;

	x0 = x1;
	y0 = y1;
	z0 = z1;

	} while (e1 > e || e2 > e || e3 > e);

	printf("The solution is: x=%f, y=%f, z=%f\n", x1, y1, z1);

	return 0;
}

// Function definitions
```

```
float f1(float y, float z)
{
	return (58 - (2 * y) - (3 * z)) / 45;
}
float f2(float x, float z)
{
	return (47 + (3 * x) - (2 * z)) / 22;
}

float f3(float x, float y){
	return (67 - (5 * x) - y) / 20;
}
```

Output:

```
D:\StudyMaterial\FifthSem\NumericalMethods\NM_Lab>cd "d:\StudyMaterial\FifthSem\Numeric
alMethods\NM_Lab\LabSix\" && gcc LabSixSecond.c -o LabSixSecond && "d:\StudyMaterial\Fi
fthSem\NumericalMethods\NM_Lab\LabSix\"LabSixSecond
Enter the error allowed: 0.0001
The solution is: x=1.000000, y=2.000000, z=3.000000

d:\StudyMaterial\FifthSem\NumericalMethods\NM_Lab\LabSix>_
```

DISCUSSION

Both methods rely on the diagonal dominance of the coefficient matrix for convergence. Diagonal dominance ensures that the diagonal elements are larger than the sum of the absolute values of the other elements in the corresponding row. If the system is not diagonally dominant, it may require rearranging the equations until the dominance condition is met.

On comparing both the methods side by side, Gauss-Seidel method typically converges faster than the Gauss-Jacobi method due to its immediate use of updated values. However, it may not always converge, and the choice of method depends on the specific characteristics of the system being solved. The choice of the initial guess for the variables also affects the convergence of both methods. A good initial guess can reduce the number of iterations needed to reach a solution.

In conclusion, the implementation and exploration of Gauss-Jacobi and Gauss-Seidel methods provide valuable insights into solving linear systems iteratively. Understanding the strengths and limitations of each method is essential for selecting the most appropriate approach for a given problem.

CONCLUSION

In this lab, we implemented and explored the Gauss-Jacobi and Gauss-Seidel methods for solving systems of linear equations. These iterative numerical techniques are valuable tools for finding approximate solutions to systems of equations that might be too complex to solve algebraically.

The Gauss-Jacobi method assumes that each variable is updated using the values from the previous iteration for the other variables. It requires the system of equations to be diagonally dominant for convergence. Our implementation successfully demonstrated the convergence of the method for a given system of linear equations.

Similarly, the Gauss-Seidel method is an improvement over the Gauss-Jacobi method, as it immediately uses the updated values within the same iteration. This often leads to faster convergence. Our implementation of the Gauss-Seidel method also successfully provided a solution to the system of linear equations.