Name : Deepankar Sharma
Course: BCA
University Roll No: 2092014
Student Id : 20041299
Semester: 3
Date: November 24, 2021

Practical 09: Jacobi's Iteration Method

Objective: To find root of the system equation using Jacobi's Iteration method.

2. Algorithm:

```
1. Start
2. Input the matrix of equations in arr[n][n+1] where n is number of unknown variables
3. Check the validity of the equations, if valid we can apply iterative methods !!!
4. Matrix X[n] , for unknown variables.
5. Take x0=y0=z0=0
6. Convert the variables as a function of other variables.
7. Apply iteration method in loop:
        x=f(y0,z0)
        y=f(z0,x0)
        z=f(x0,y0)
        x=x0
        y=y0
        z=z0
if((z-z0)>allowed Error AND (y-y0)>allowed Error AND (x-x0)>allowed Error )
        Terminate Loop
8. Solution:
        for(i= n-1; i>=0; i--){
      X[i]= (arr[i][n])/arr[i][i];
   }
9. Print X
10. Stop
```

Code:

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define MAX 10
float arr[3][4],x[3];
int n=3;
/*
```

```
    CBNST Page No.454
    8x-3y+2z=20
    4x+11y-z=33
    6x+3y+12z=35
*/
#define x(y0, z0) (arr[0][3] - (arr[0][2]*z0 + arr[0][1]*y0)) / arr[0][0]
#define y(x0, z0) (arr[1][3] - (arr[1][2]*z0 + arr[1][0]*x0)) / arr[1][1]
#define z(y0, x0) (arr[2][3] - (arr[2][0]*x0 + arr[2][1]*y0)) / arr[2][2]

void checkValidity(){
    for (int i = 0; i < n; i++)
    {
        float sum=0;
        for(int j=0; j<n; j++){
            if(i!=j){
                sum+=fabs(arr[i][j]);
            }
        }
        if((fabs(arr[i][i])-sum)<0){
            printf("Can't apply iteration method !!!!\n\n");
            exit(1);
        }
    }
    printf("Can apply iteration method !!!!\n\n");
}

int main()
{
    // printf("Enter the number of unknown variables: ");
    // scanf("%d", &n);
    printf("Enter coefficients of Augmented Matrix:\n");
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n + 1; j++)
        {
            printf("a[%d][%d] = ", i, j);
            scanf("%f", &arr[i][j]);
        }
    }
    checkValidity();

    int iter = 1;
    float allErr, fabx, faby, fabz, deltaX, deltaY, deltaZ;
    float x0 = 0.0, y0 = 0.0, z0 = 0.0;
    printf("Enter the allowed error: ");
    scanf(" %f", &allErr);

    printf("\n\nIteration\t\tx0\t\ty0\t\tz0\t\t\tx1\t\ty1\t\tz1\n\n");
    do
    {
        /* code */
        fabx = fabs(x(y0, z0));
        deltaX = fabs(x0 - fabx);
        faby = fabs(y(x0, z0));
        deltaY = fabs(y0 - faby);
        fabz = fabs(z(y0, x0));
        deltaZ = fabs(z0 - fabz);
        printf("%d\t\t\t%f\t%f\t%f\t\t%f\t%f\t%f\n", iter, x0, y0, z0, fabx, faby, fabz);
        x0 = fabx;
        y0 = faby;
        z0 = fabz;
        iter++;

        // } while (iter<15);
    } while ((deltaX > allErr) || (deltaY > allErr) || (deltaZ > allErr));

    printf("\n\nThe roots are: \n");
    printf("x= %f\n", fabx);
    printf("y= %f\n", faby);
    printf("z= %f\n", fabz);
    return 0;
}
```

Output:

```
PS D:\01_Java\Deepankar\CCpp\CBNST\Practical-09-GuassSeidel> cd
"d:\01_Java\Deepankar\CCpp\CBNST\Practical-09-GuassSeidel\" ; if ($?) { g++ _01_guassSeidel.cpp -o
_01_guassSeidel } ; if
($?) { .\_01_guassSeidel }
Enter coefficients of Augmented Matrix:
a[0][0] = 8
a[0][1] = -3
a[0][2] = 2
a[0][3] = 20
a[1][0] = 4
a[1][1] = 11
a[1][2] = -1
a[1][3] = 33
a[2][0] = 6
a[2][1] = 3
```

```
a[2][2] = 12
a[2][3] = 35
Can apply iteration method !!!!

Enter the allowed error: 0.0001


Iteration        x0            y0         z0                   x1            y1             z1

1             0.000000      0.000000      0.000000          2.500000      3.000000      2.916667
2             2.500000      3.000000      2.916667          2.895833      2.356061      0.916667
3             2.895833      2.356061      0.916667          3.154356      2.030303      0.879735
4             3.154356      2.030303      0.879735          3.041430      1.932937      0.831913
5             3.041430      1.932937      0.831913          3.016873      1.969654      0.912717
6             3.016873      1.969654      0.912717          3.010441      1.985929      0.915817
7             3.010441      1.985929      0.915817          3.015769      1.988550      0.914964
8             3.015769      1.988550      0.914964          3.016965      1.986535      0.911644
9             3.016965      1.986535      0.911644          3.017040      1.985798      0.911550
10             3.017040      1.985798      0.911550          3.016787      1.985763      0.911697
11             3.016787      1.985763      0.911697          3.016737      1.985868      0.911833
12             3.016737      1.985868      0.911833          3.016742      1.985899      0.911831


The roots are:
x= 3.016742
y= 1.985899
z= 0.911831
```