

Gauss Elimination Method

OBJECTIVE:

To solve the given system of linear equation using Gauss Elimination Method.

THEORY:

Gauss elimination method is used to solve a system of linear equations. Let's recall the definition of these systems of equations. A system of linear equations is a group of linear equations with various unknown factors. As we know, unknown factors exist in multiple equations. Solving a system involves finding the value for the unknown factors to verify all the equations that make up the system. If there is a single solution that means one value for each unknown factor, then we can say that the given system is a consistent independent system. If multiple solutions exist, the system has infinitely many solutions; then we say that it is a consistent dependent system. If there is no solution for unknown factors, and this will happen if there are two or more equations that can't be verified simultaneously, then we say that it's an inconsistent system.

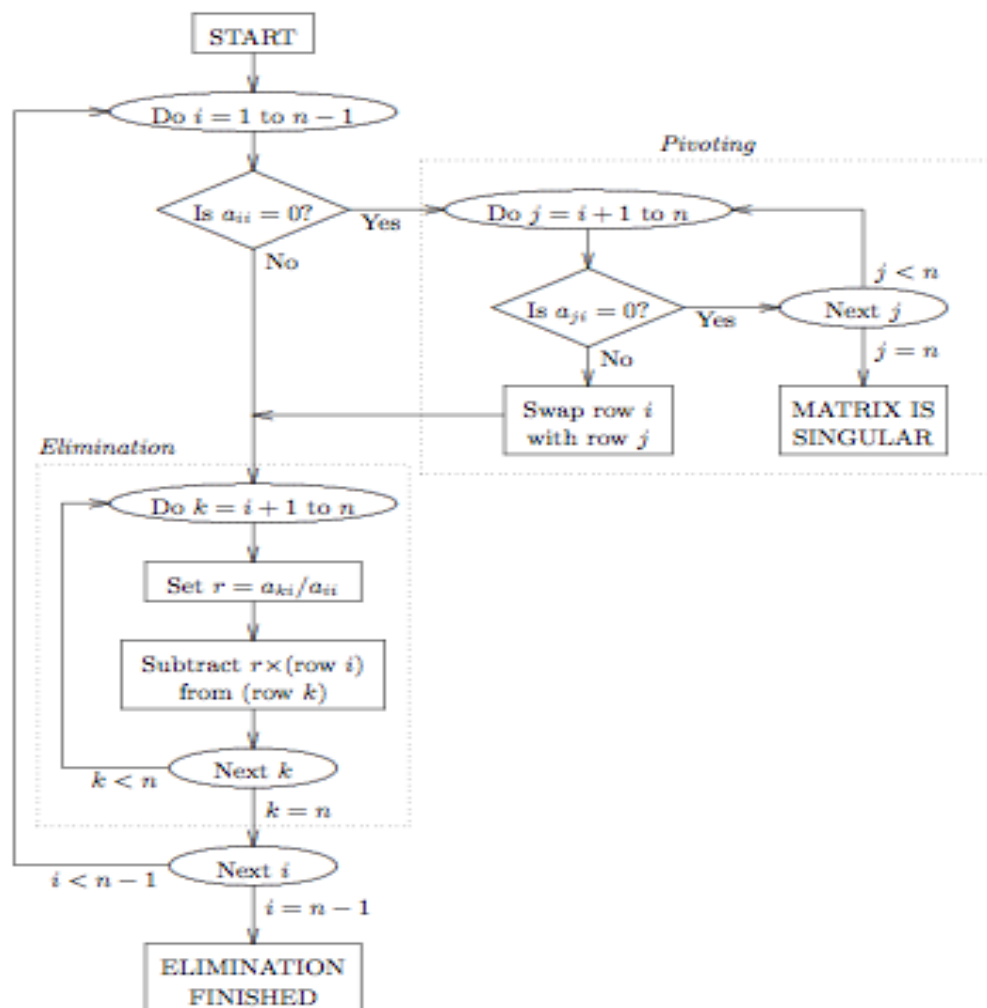
This can be summarized in a table as given below:

Name of the system of equations	Number of solutions
Consistent independent system	1
Consistent dependent system	Multiple or infinitely many
Inconsistent	0

ALGORITHM:

1. Start
2. Read Number of Unknowns: n
3. Read Augmented Matrix (A) of n by $n+1$ Size
4. Transform Augmented Matrix (A) to Upper Triangular Matrix by Row Operations.
5. Obtain Solution by Back Substitution.
6. Display Result.
7. Stop

Flowchart:



Source code:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
#define SIZE 10

int main()
{
    float a[SIZE][SIZE], x[SIZE], ratio;
    int i,j,k,n;
    printf("Aarju Mahata\t PUR077BCT004\n \n");
    printf("Enter number of unknowns: ");
    scanf("%d", &n);
    for(i=1;i<=n;i++) {
        for(j=1;j<=n+1;j++) {
            printf("a[%d][%d] = ",i,j);
            scanf("%f", &a[i][j]);
        }
    }

    for(i=1;i<=n-1;i++) {
        if(a[i][i] == 0.0) {
            printf("Mathematical Error!");
            exit(0);
        }
        for(j=i+1;j<=n;j++){
            ratio = a[j][i]/a[i][i];
            for(k=1;k<=n+1;k++) {
                a[j][k] = a[j][k] - ratio*a[i][k];
            }
        }
    }
}
```

```

    x[n] = a[n][n+1]/a[n][n];
    for(i=n-1;i>=1;i--) {
        x[i] = a[i][n+1];
        for(j=i+1;j<=n;j++)
        {
            x[i] = x[i] - a[i][j]*x[j];
        }
        x[i] = x[i]/a[i][i];
    }

    printf("\nSolution:\n");
    for(i=1;i<=n;i++) {
        printf("x[%d] = %0.3f\n",i, x[i]);
    }
    return(0);
}

```

Output:

```

17) C:\Users\Aarju Mahata\Documents\Projects\guass_elimination\guass_elimination.c
Aarju Mahata PUR077BCT004

Enter number of unknowns: 3
a[1][1] = 1
a[1][2] = 2
a[1][3] = 3
a[1][4] = 3
a[2][1] = 4
a[2][2] = 4
a[2][3] = 3
a[2][4] = 4
a[3][1] = 4
a[3][2] = 5
a[3][3] = 6
a[3][4] = 4

Solution:
x[1] = -5.000
x[2] = 8.000
x[3] = -2.667

```

DISCUSSION AND CONCLUSION:

Here, we discussed about the algorithm and source code of Guass elimination method and we practically coded the source code of the this method using

programming language (C) to test the example of solution of a problem relating to Guass Elimination method.

Lagrange's Interpolation

OBJECTIVE:

To find the value of a new data (x) for a given data (y) within the given range of discrete set of known data points.

THEORY:

Lagrange Interpolation Formula finds a polynomial called Lagrange Polynomial that takes on certain values at an arbitrary point. It is an n^{th} degree polynomial expression to the function $f(x)$. The interpolation method is used to find the new data points within the range of a discrete set of known data points.

Given few real values $x_1, x_2, x_3, \dots, x_n$ and $y_1, y_2, y_3, \dots, y_n$ and there will be a polynomial P with real coefficients satisfying the conditions $P(x_i) = y_i, \forall i = \{1, 2, 3, \dots, n\}$ and degree of polynomial P must be less than the count of real values i.e., $\text{degree}(P) < n$. The Lagrange Interpolation formula for different orders i.e., n^{th} order is given as,

Lagrange Interpolation Formula for n^{th} order is-

$$f(x) = \frac{(x-x_1)(x-x_2)\dots(x-x_n)}{(x_0-x_1)(x_0-x_2)\dots(x_0-x_n)} \times y_0 + \frac{(x-x_0)(x-x_2)\dots(x-x_n)}{(x_1-x_0)(x_1-x_2)\dots(x_1-x_n)} \times y_1 + \dots + \frac{(x-x_0)(x-x_1)\dots(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\dots(x_n-x_{n-1})} \times y_n$$

ALGORITHM:

1. Read x, n
2. for i = 0 to (n - 1) in steps of 1 do Read xi, fi endfor
3. sum = 0
4. for i = 0 to (n - 1) in steps of 1 do
5. prodfunc = 1
6. for j = 1 to (n + 1) in steps of 1 do
7. If (i # j) then
$$\text{Prodfunc} = \text{prodfunc} * (x - x_j) / (x_i - x_j)$$

endfor
- 8 sum = sum + fi × prodfunc
- 9 Write x, sum
- 10 Stop

Source code:

```
#include<stdio.h>

#include<conio.h>

#include<math.h>

int main(){

    float x[20], f[20], L[20], sum = 0, X;

    int N;

    int i,j;

    printf("Aarju Mahata\t PUR077BCT004\n \n");
```

```
printf("Enter N:\n");
scanf("%d",&N);
printf("Enter x:\n");
for(i = 0; i<N; i++)
{
    scanf("%f",&x[i]);
}
printf("Enter F:\n");
for(i = 0; i<N; i++)
{
    scanf("%f",&f[i]);
}
```

```
printf("Put X = ");
scanf("%f",&X);
for(i = 0; i<N; i++)
{
    L[i] = 1;
    for(j = 0; j<N; j++)
    {
        if(i!=j)
        {

$$L[i] = L[i]*((X-x[j])/(x[i]-x[j]));$$


```



```

    }
}

sum += f[i]*L[i];

}

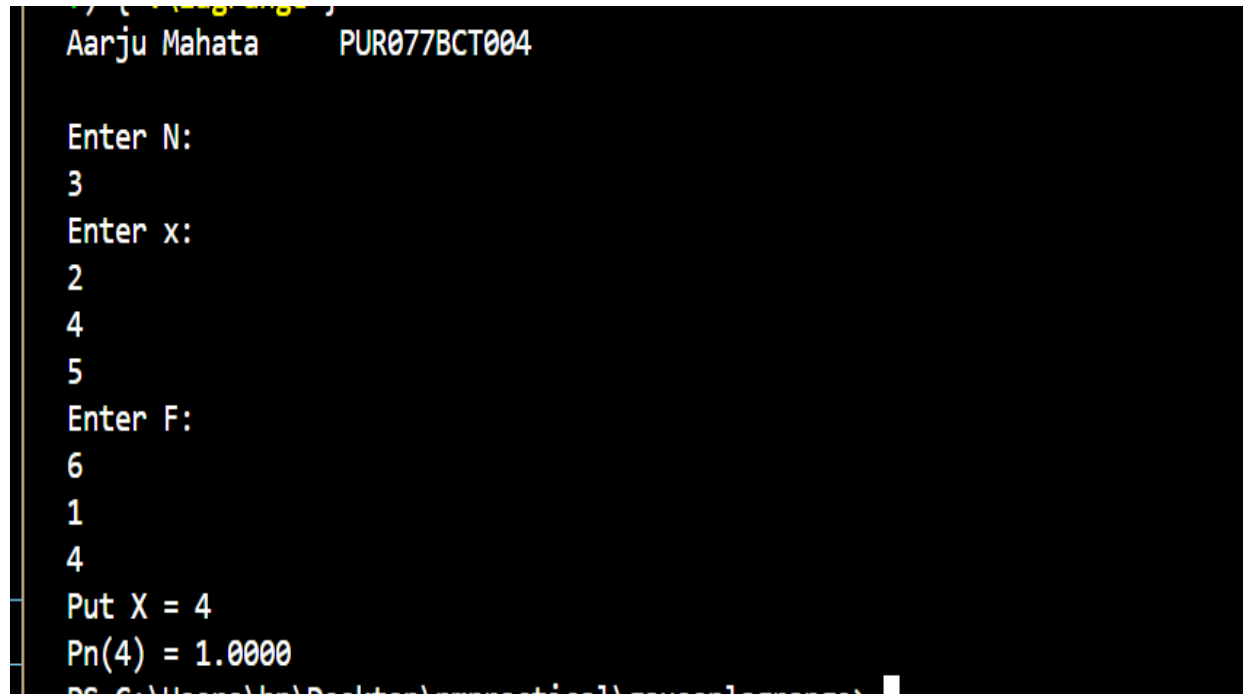
printf("Pn(%.0f) = %.4f",X,sum);

return 0;

}

```

Output:



```

Aarju Mahata  PUR077BCT004

Enter N:
3
Enter x:
2
4
5
Enter F:
6
1
4
Put X = 4
Pn(4) = 1.0000
PS C:\Users\hr\Desktop> gcc lagrange.c -o lagrange

```

DISCUSSION AND CONCLUSION:

Here, we discussed about the algorithm and source code of Lagrange's Interpolation method and we practically coded the source code of the this method using programming language (C) to test the example of solution of a problem relating to Lagrange's method.

