

**DEHRADUN CAMPUS**

# PRACTICAL FILE / TERM WORK

**CBNST LAB**

**PMA-502**

**B.Tech CSE**

**V**

**2022-23**

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**GRAPHIC ERA HILL UNIVERSITY, DEHRADUN**

**SUBMITTED TO SUBMITTED BY**

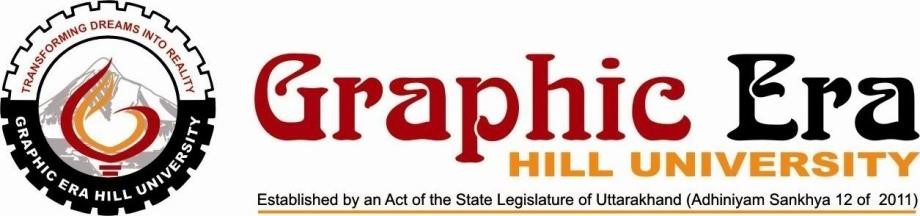
Mr. PURUSHOTTAM DAS NAME: AMANDEEP SINGH

ASST. PROFESSOR EXAMINATION ROLLNO.: 2018143

DEPARTMENT OF COMPUTER COURSE/SEM: B.TECH/5

SCIENCE & ENGG.

**COLLEGE ROLL NO.\_\_\_\_\_\_\_\_ EXAMINATION ROLL NO.\_\_\_\_\_\_\_**



## DEHRADUN CAMPUS

THIS IS TO CERTIFY THAT Mr. / Ms. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HAS SATISFACTORILY COMPLETED ALL THE EXPERIMENTS IN THE LABORATORY OFTHIS

COLLEGE. THE COURSE OF THE EXPERIMENTS / TERM WORK

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ IN PARTIAL FULLFILLMENT OF THE REQUIREMENT IN \_\_\_\_\_\_\_\_\_\_\_ SEMESTER OF B.TECH (CSE) / M.TECH( ) / BCA / MCA /

BBA / MBA DEGREE COURSE PRESCRIBED BY GRAPHIC ERA HILL UNIVERSITY, DEHRADUN DURING THE YEAR \_\_\_\_\_\_\_ - \_\_\_\_\_\_\_

CONCERNED FACULTY HEAD OF DEPARTMENT

NAME OF EXAMINER:

SIGNATURE OF EXAMINER:

Annexure-A



## Department of Computer Science & Application Lab Details

**Name of the Lab: -** CBNST Lab

**Lab Code: -** PMA-502

**Subject Credit: -** 2

**Course: -** B.Tech

**Branch: -** CSE

**Semester: -** V

**Section: -** E

**Number of students enrolled: -** 67

**Name of the Faculty: -** Mr. Purushottam Das

**Name of Lab Instructor: -**

**Lab Number:-** LAB-5

## Lab Time Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Lab Session** | **Day** | **Lecture Number** | **Timing** |
| 1 | Wednesday | 01 and 02 | 8:00 AM – 9:50 AM |

Annexure- B



**Department of Computer Science & Application**

## List of Practical’s

**Subject Code: PMA-502 Subject Name: CBNST Lab**

**Course : B.Tech CSE Branch & Sem:-V**

1. Write a program in C to find absolute, relative and percentage error for roundoff and truncation cases.
2. Write a program in C to find the roots of any polynomial / transcendental equation using bisection method correct up to three decimal places.
3. Write a program in C to find the solution of any transcendental equation using Regula-Falsi method correct up to three decimal places.
4. Write a program in C to find the solution of any non-polynomial equation using Newton-Raphson method correct up to four decimal places.
5. Write a program in C to find the roots of any non-polynomial equation using Iteration method correct up to four decimal places.
6. Write a program in C to solve the system of linear equations using Gauss Elimination method.
7. Write a program in C to solve the homogeneous system of linear equations using Gauss Jordan method.
8. Write a program in C to solve given system of linear equations using Gauss Siedal iterative method.
9. Write a program in C to interpolate using Newton’s forward difference formula for the stated values.
10. Write a program in C to implement Newton’s backward difference formula.
11. Write a program in C to interpolate using Gauss forward Interpolation formula for given values.
12. Write a program in C to implement Lagrange’s Interpolation formula for unequal intervals.
13. Write a program in C to integrate given values using Trapezoidal rule.
14. Write a program in C to integrate using Simpson’s 1/3 rule for the stated values.
15. Write a program in C to implement Simpson’s 3/8 rule.
16. Write a C Program to implement Euler’s method.
17. Write a C Program to implement Runge-Kutta’s Method.
18. Write a C Program to implement curve fitting for a straight line.
19. Write a C Program to implement parabolic curve fitting.
20. Write a C Program to implement regression lines.

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

**STUDENT LAB REPORT SHEET**

**CBNST LAB (PMA-502)**

**Name of Student ……………………………………...…………. Mo. No………………….……….…………..**

**Address Permanent …….…………………………………………………………………………..……………..**

**Father’s Name ………………………………..……………… Mo No ………………………**

**Mother’s Name ……………………………………………… Mo No…………..…………..**

**Section ……..……….Branch……………………Semester…………….. Class Roll No……………..**

**Local Address……………………………………………………Email……………. ……Grade A B C**

**Marks**  5 3 1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.**  **No.** | **Name of the Experiment** | **D.O.P.** | **Date of Submiss-**  **ion** | **Grade (Viva)** | **Grade**  **(Report**  **File)** | **Total**  **Marks**  **(out of**  **10)** | **Student’s Signature** | **Teacher’s Signature** |
| **1** | Write a program in C to find absolute, relative and percentage error for round-off and truncation cases. |  |  |  |  |  |  |  |
| **2** | Write a program in C to find the roots of any polynomial / transcendental equation using bisection method correct up to three decimal places. |  |  |  |  |  |  |  |
| **3** | Write a program in C to find the solution of any transcendental equation using Regula-Falsi method correct up to three decimal places. |  |  |  |  |  |  |  |
| **4** | Write a program in C to find the solution of any non-polynomial equation using Newton-Raphson method correct up to four decimal places. |  |  |  |  |  |  |  |
| **5** | Write a program in C to find the roots of any nonpolynomial equation |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | using Iteration method correct up to four decimal places. |  |  |  |  |  |  |  |
| **6** | Write a program in C to solve the system of linear equations using Gauss Elimination method. |  |  |  |  |  |  |  |
| **7** | Write a program in C to solve the homogeneous system of linear equations using Gauss  Jordan method. |  |  |  |  |  |  |  |
| **8** | Write a program in C to solve given system of linear equations using Gauss-Siedal iterative method. |  |  |  |  |  |  |  |
| **9** | Write a program in C to interpolate using Newton’s forward difference formula for the stated values. |  |  |  |  |  |  |  |
| **10** | Write a program in C to implement Newton’s backward difference formula. |  |  |  |  |  |  |  |
| **11** | Write a program in C to interpolate using Gauss forward Interpolation formula for given values. |  |  |  |  |  |  |  |
| **12** | Write a program in C to implement Lagrange’s Interpolation formula for unequal intervals. |  |  |  |  |  |  |  |
| **13** | Write a program in C to integrate given values using Trapezoidal rule. |  |  |  |  |  |  |  |
| **14** | Write a program in C to integrate using Simpson’s 1/3 rule for the stated values. |  |  |  |  |  |  |  |
| **15** | Write a program in C to implement Simpson’s 3/8 |  |  |  |  |  |  |  |
|  | rule. |  |  |  |  |  |  |  |
| **16** | Write a C Program to implement Euler’s method. |  |  |  |  |  |  |  |
| **17** | Write a C Program to implement RungeKutta’s Method. |  |  |  |  |  |  |  |
| **18** | Write a C Program to implement curve fitting for a straight line. |  |  |  |  |  |  |  |
| **19** | Write a C Program to implement parabolic curve fitting. |  |  |  |  |  |  |  |
| **20** | Write a C Program to implement regression lines. |  |  |  |  |  |  |  |

**Total No of Practical allotted: ………………………………………**

**Total No of Practical completed: ………………………………….**

**Percentage Attendance of Practical: …………………………….**

# ACKNOWLEDGEMENT

I would like to particularly thank my Computer Based Numerical and Statistical Techniques Lab Faculty **Mr. Purushottam Das** for his patience, support, and encouragement throughout the completion of this Term work.

At last, but not the least I greatly indebted to all other persons who directly or indirectly helped me during this course.

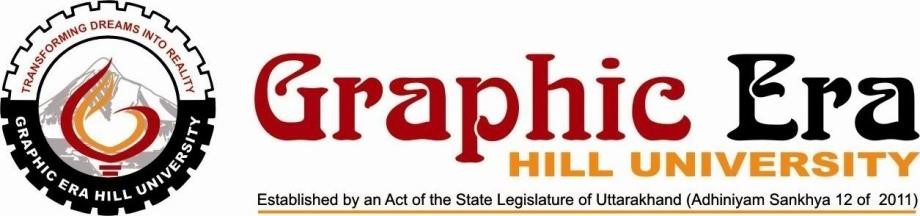
**Amandeep Singh**

**University Rollno. 2018143**

**B. Tech CSE-I-III Sem**

**Session: 2021-22**

**GEHU, Dehradun**



**DEHRADUN CAMPUS**

## INDEX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EXP. NO.** | **DATE** | **EXPERIMENT** | **PAGE NO.** | **REMARKS** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

INDEX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EXP. NO.** | **DATE** | **EXPERIMENT** | **PAGE NO.** | **REMARKS** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**PROGRAM NO.1**

**NAME:** Amandeep Singh

**COURSE:** B.TECH

**BRANCH/SEMESTER:** CSE/5

**ROLL NO:** 09

**DATE:**

1. **OBJECTIVE**: To find absolute, relative and percentage error for roundoff and truncation cases

1. **METHOD**:

Step 1 – Enter the value of x.

Step 2 – Enter number of digits upto which truncation and roundoff to be done.

Step 3 – Find the truncated and roundoff value of x upto entered number of digits.

Step 4 – Calculate the Absolute Error using |x-x’| for both roundoff and truncation.

Step 5 – Calculate the Relative Error using | (x-x’) / ( x) | for both roundoff and truncation.

Step 6 – Calculate the Percentage Error using ( | (x-x’) / ( x) | ) \* 100 for both roundoff and truncation.

1. **PROGRAM**:

#include <stdio.h>

#include <math.h>

int main(){

double x;

printf("Enter x: ");

scanf("%lf", &x);

int a;

printf("Enter decimal places upto which truncation & roundoff to be done: ");

scanf("%d", &a);

printf("For Truncation case - \n");

long long int b = x \* pow(10,a);

double y = (double)b/pow(10,a);

printf("Truncated value: %lf\n", y);

double A\_E = fabs(x-y);

printf("Absolute Error: %lf\n", A\_E);

double R\_E = fabs(A\_E/x);

printf("Relative Error: %lf\n", R\_E);

double P\_E = R\_E\*100;

printf("Percentage Error: %lf\n", P\_E);

printf("For Roundoff Case - \n");

b = x\*pow(10,a+1); //146749

int c = b%10;

b= b/(float)10;

if (c>5){

b++;

}

y = (float)b/pow(10,a);

printf("Roundoff: %lf\n",y);

A\_E = fabs(x-y);

printf("Absolute Error: %lf\n", A\_E);

R\_E = fabs(A\_E/x);

printf("Relative Error: %lf\n", R\_E);

P\_E = R\_E\*100;

printf("Percentage Error: %lf\n", P\_E);

return 0;

}

1. **OUTPUT**:

Enter x: 1.46749134

Enter decimal places upto which truncation & roundoff to be done: 4

For Truncation case -

Truncated value: 1.467400

Absolute Error: 0.000091

Relative Error: 0.000062

Percentage Error: 0.006224

For Roundoff Case -

Roundoff: 1.467500

Absolute Error: 0.000009

Relative Error: 0.000006

Percentage Error: 0.000590

**PROGRAM NO.2**

**NAME:** Amandeep Singh

**COURSE:** B.TECH

**BRANCH/SEMESTER:** CSE/5

**ROLL NO:** 09

**DATE:**

1. **OBJECTIVE**: To find the roots of any polynomial / transcendental equation using bisection method correct up to three decimal places.

1. **METHOD**:

Step 1 - Define the function f(x)

Step 2 - Begin by choosing the two initial guesses. Let the two guesses be x0 and x1 such that f(x0)f(x1) < 0

Step 3 - Choose pre-specified tolerable error

Step 4 - Calculate the root obtained as x2 = (x0 + x1)/2

Step 5 - Calculate f(x0)f(x2);

* 1. if f(x0)f(x2) < 0 then x0 = x0 and x1 = x2
  2. if f(x0)f(x2) > 0 then x0 = x2 and x1 = x1
  3. if f(x0)f(x2) = 0 then follow the next step

Step 6 - if |f(x2)| > e then go to step 5 else go to the next step

1. **PROGRAM**:

#include <stdio.h>

#include <math.h>

double function(double x){

return x\*x\*x-4\*x-9;

}

int main(){

double a, b;

for (int i = -5; i <= 5; i++){

a = i;

b = i+1;

if (function(a)\*function(b) < 0){

break;

}

}

printf("a = %lf , b = %lf\n", a,b);

double fx,fa,fb,x;

int steps = 0;

do{

fa = function(a);

fb = function(b);

x = (a+b)/2;

fx = function(x);

printf("Interation %d, x = %.4lf\n", ++steps, x);

if (fx\*fb < 0){

a = x;

}

else if (fx\*fa < 0){

b = x;

}

}while(fabs(x-a) >= 0.0001 || fabs(x-b) >= 0.0001);

printf("Root is %.4lf\n", x);

}

1. **OUTPUT**:

a = 2.000000 , b = 3.000000

Interation 1, x = 2.5000

Interation 2, x = 2.7500

Interation 3, x = 2.6250

Interation 4, x = 2.6875

Interation 5, x = 2.7188

Interation 6, x = 2.7031

Interation 7, x = 2.7109

Interation 8, x = 2.7070

Interation 9, x = 2.7051

Interation 10, x = 2.7061

Interation 11, x = 2.7065

Interation 12, x = 2.7063

Interation 13, x = 2.7064

Interation 14, x = 2.7065

Root is 2.7065

**PROGRAM NO.3**

**NAME:** Amandeep Singh

**COURSE:** B.TECH

**BRANCH/SEMESTER:** CSE/5

**ROLL NO:** 09

**DATE:**

1. **OBJECTIVE**: To find the solution of any transcendental equation using Regula-Falsi method correct up to three decimal places.

1. **METHOD**:

Step 1 – Define function f(x).

Step 2 – Choose initial guesses x0 and x1 such that f(x0)f(x1) < 0.

Step 3 – Choose pre-specified tolerable error e.

Step 4 – Calculate new approximated root as: x2 = x0 - ((x0-x1) \* f(x0))/(f(x0) - f(x1)).

Step 5 – Calculate f(x0)\*f(x2)

a. if f(x0)\*f(x2) < 0 then x0 = x0 and x1 = x2

b. if f(x0)\*f(x2) > 0 then x0 = x2 and x1 = x1

c. if f(x0)\*f(x2) = 0 then goto (8)

Step 6 – if |f(x2)|>e then goto (5) otherwise goto (7).

Step 7 – Display x2 as root.

1. **PROGRAM**:

#include <stdio.h>

#include <math.h>

double function(double x){

return x\*x\*x-2\*x-5;

}

int main(){

double a, b;

for (int i = -5; i <= 5; i++){

a = i;

b = i+1;

if (function(a)\*function(b) < 0){

break;

}

}

printf("a : %lf, b : %lf\n",a,b);

double fx, fa, fb, x, x1;

int step = 0;

do{

fa = function(a);

fb = function(b);

x = (a\*fb - b\*fa)/(fb-fa);

fx = function(x);

printf("Iteration %d, x = %lf\n", ++step, x);

if (fx < 0){

x1 = a;

a = x;

}

else if(fx > 0){

x1 = b;

b = x;

}

}while(fabs(x-x1) >= 0.0001);

printf("x : %lf\n", x);

}

1. **OUTPUT**:

a : 2.000000, b : 3.000000

Iteration 1, x = 2.058824

Iteration 2, x = 2.081264

Iteration 3, x = 2.089639

Iteration 4, x = 2.092740

Iteration 5, x = 2.093884

Iteration 6, x = 2.094305

Iteration 7, x = 2.094461

Iteration 8, x = 2.094518

x : 2.094518

**PROGRAM NO.4**

**NAME:** Amandeep Singh

**COURSE:** B.TECH

**BRANCH/SEMESTER:** CSE/5

**ROLL NO:** 09

**DATE:**

1. **OBJECTIVE**: To find the solution of any non-polynomial equation using Newton-Raphson method correct up to four decimal places.
2. **METHOD**:

Step 1 – Define function as f(x)

Step 2 – Define first derivative of f(x) as g(x)

Step 3 – Input initial guess (x0), tolerable error (e) and maximum iteration (N)

Step 4 – Initialize iteration counter i = 1

Step 5 – If g(x0) = 0 then print "Mathematical Error" and goto (12) otherwise goto (7)

Step 6 – Calcualte x1 = x0 - f(x0) / g(x0)

Step 7 – Increment iteration counter i = i + 1

Step 8 – If i >= N then print "Not Convergent" and goto (12) otherwise goto (

Step 9 – If |f(x1)| > e then set x0 = x1 and goto (6) otherwise goto (11)

Step 10 – Print root as x1

1. **PROGRAM**:

#include <stdio.h>

#include <math.h>

double function(double x){

return x\*x\*x - 3\*x -5;

}

double derivative(double x){

return 3\*x\*x - 3;

}

double ex(double x){

return (x - (function(x)/derivative(x)));

}

int main(){

double a, b;

for (int i = 0; i <= 5; i++){

a = i;

b = i+1;

if (function(a)\*function(b) < 0){

break;

}

}

printf("a : %lf , b : %lf\n", a, b);

double x = (a+b)/2;

double x1;

int step = 0;

do{

x1 = x;

x = ex(x);

printf("Interation %d, x : %lf\n", ++step, x);

}while(fabs(x-x1) >= 0.001);

printf("x : %lf\n", x);

}

1. **OUTPUT**:

a : 2.000000 , b : 3.000000

Interation 1, x : 2.301587

Interation 2, x : 2.279291

Interation 3, x : 2.279019

x : 2.279019

**PROGRAM NO.5**

**NAME:** Amandeep Singh

**COURSE:** B.TECH

**BRANCH/SEMESTER:** CSE/5

**ROLL NO:** 09

**DATE:**

1. **OBJECTIVE**: To find the roots of any non-polynomial equation using Iteration method correct up to four decimal places.
2. **METHOD**:

Step 1 – Read values of x0 and e.

Step 2 – Calculate x1 = g(x0)

Step 3 – If |x1 – x0| <= e, goto step 6.

Step 4 – Else, assign x0 = x1 and goto step 3.

Step 5 – Display x1 as the root.

1. **PROGRAM**:

#include <stdio.h>

#include <math.h>

double function(double x){

return (x\*x\*x - 2\*x\*x - 4);

}

double ex(double x){

return cbrt(2\*x\*x + 4);

}

double derivative(double x){

return ((4\*x)/(3\*cbrt((2\*x\*x + 4)\*(2\*x\*x + 4))));

}

int main(){

double a, b;

for (int i = -5; i <= 5; i++){

a = i;

b = i+1;

if (function(a)\*function(b) < 0){

break;

}

}

double x = (a+b)/2;

double x1;

int step = 0;

do{

x1 = x;

x = ex(x);

printf("Interation %d, x : %lf\n", ++step, x);

}while(fabs(x-x1) >= 0.001);

printf("x : %lf\n", x);

}

1. **OUTPUT**:

Interation 1, x : 2.545822

Interation 2, x : 2.569385

Interation 3, x : 2.581500

Interation 4, x : 2.587727

Interation 5, x : 2.590928

Interation 6, x : 2.592573

Interation 7, x : 2.593419

x : 2.593419

**PROGRAM NO.6**

**NAME:** Amandeep Singh

**COURSE:** B.TECH

**BRANCH/SEMESTER:** CSE/5

**ROLL NO:** 09

**DATE:**

1. **OBJECTIVE**: To solve the homogeneous system of linear equations using Gauss Jordan method

1. **METHOD**:

Step 1 – Read Number of Unknowns: n

Step 2 – Read Augmented Matrix (A) of n by n+1 Size

Step 3 – Transform Augmented Matrix (A) to Upper Triangular Matrix by Row Operations.

Step 4 – Obtain Solution by Back Substitution.

Step 5 – Display Result.

1. **PROGRAM**:

#include <stdio.h>

int main(){

int n;

printf("Enter order of matrix: ");

scanf("%d", &n);

float arr[n][n+1];

for (int i = 0; i < n; i++){

for (int j = 0; j < n+1; j++){

printf("Enter arr[%d][%d] : ",i+1,j+1);

scanf("%f",&arr[i][j]);

}

}

for (int i = 0; i < n; i++){

for (int j = 0; j < n+1; j++){

printf("%f ", arr[i][j]);

}

printf("\n");

}

float a;

for (int i = 0; i<n-1; i++){

if (arr[i][i] == 0.0){

printf("Error\n");

return 0;

}

for (int j = i+1; j < n; j++){

a = arr[j][i]/arr[i][i];

for (int k = 0; k < n+1;k++){

arr[j][k] = arr[j][k] - a\*arr[i][k];

}

}

}

float ans[n];

ans[n-1] = arr[n-1][n]/arr[n-1][n-1];

for (int i = n-2; i >=0; i--){

ans[i] = arr[i][n];

for (int j = i+1; j < n; j++){

ans[i] = ans[i] - arr[i][j]\*ans[j];

}

ans[i] = ans[i]/arr[i][i];

}

for (int i = 0; i < n; i++){

printf("x%d : %f\n", i+1, ans[i]);

}

}

1. **OUTPUT**:

Enter order of matrix: 3

Enter arr[1][1] : 10

Enter arr[1][2] : 1

Enter arr[1][3] : 1

Enter arr[1][4] : 12

Enter arr[2][1] : 1

Enter arr[2][2] : 10

Enter arr[2][3] : 1

Enter arr[2][4] : 12

Enter arr[3][1] : 1

Enter arr[3][2] : 1

Enter arr[3][3] : 10

Enter arr[3][4] : 12

10.000000 1.000000 1.000000 12.000000

1.000000 10.000000 1.000000 12.000000

1.000000 1.000000 10.000000 12.000000

x1 : 1.000000

x2 : 1.000000

x3 : 1.000000

**PROGRAM NO.7**

**NAME:** Amandeep Singh

**COURSE:** B.TECH

**BRANCH/SEMESTER:** CSE/5

**ROLL NO:** 09

**DATE:**

1. **OBJECTIVE**: To solve given system of linear equations using Gauss Siedal iterative method.

1. **METHOD**:

Step 1 – Read Number of Unknowns: n

Step 2 – Read Augmented Matrix (A) of n by n+1 Size

Step 3 – Transform Augmented Matrix (A) to Diagonal Matrix by Row Operations.

Step 4 – Obtain Solution by Making All Diagonal Elements to 1.

Step 5 – Display Result.

1. **PROGRAM**:

#include<stdio.h>

int main(){

int i,j,k,n;

printf("Enter the size of matrix: ");

scanf("%d",&n);

printf("Enter the elements\n");

float A[n][n+1],c,x[n];

for(i=0; i<n; i++){

for(j=0; j<n+1; j++){

printf("A[%d][%d]:", i+1,j+1);

scanf("%f",&A[i][j]);

}

}

for(j=0; j<n; j++){

for(i=0; i<n; i++){

if(i!=j){

c=A[i][j]/A[j][j];

for(k=0; k<n+1; k++){

A[i][k]=A[i][k]-c\*A[j][k];

}

}

}

}

for(i=0; i<n; i++){

x[i]=A[i][n]/A[i][i];

printf("x%d = %f\n",i,x[i]);

}

}

1. **OUTPUT**

Enter the size of matrix: 3

Enter the elements

A[1][1]:10

A[1][2]:1

A[1][3]:1

A[1][4]:12

A[2][1]:1

A[2][2]:10

A[2][3]:1

A[2][4]:12

A[3][1]:1

A[3][2]:1

A[3][3]:10

A[3][4]:12

x0 = 1.000000

x1 = 1.000000

x2 = 1.000000

**PROGRAM NO.8**

**NAME:** Amandeep Singh

**COURSE:** B.TECH

**BRANCH/SEMESTER:** CSE/5

**ROLL NO:** 09

**DATE:**

1. **OBJECTIVE**: To solve given system of linear equations using Gauss Siedal iterative method.

1. **METHOD**:

Step 1 – Arrange given system of linear equations in diagonally dominant form

Step 2 – Read tolerable error (e)

Step 3 – Convert the first equation in terms of first variable, second equation in terms of second variable and so on.

Step 4 – Set initial guesses for x0, y0, z0 and so on

Step 5 – Substitute value of y0, z0 ... from step 5 in first equation obtained from step 4 to calculate

new value of x1. Use x1, z0, u0 .... in second equation obtained from step 4 to caluclate new value of y1. Similarly, use x1, y1, u0... to find new z1 and so on.

Step 6 – If| x0 - x1| > e and | y0 - y1| > e and | z0 - z1| > e and so on then goto step 9

Step 7 – Set x0=x1, y0=y1, z0=z1 and so on and goto step 6

Step 8 – Print value of x1, y1, z1 and so on

1. **PROGRAM**:

#include <stdio.h>

int main(){

int n;

printf("Enter order of matrix: ");

scanf("%d", &n);

float arr[n][n+1];

float ans[n];

for (int i = 0; i < n; i++){

for (int j = 0; j < n+1; j++){

printf("Enter arr[%d][%d] : ",i+1,j+1);

scanf("%f",&arr[i][j]);

}

}

for (int i = 0; i < n; i++){

for (int j = 0; j < n+1; j++){

printf("%f ", arr[i][j]);

}

printf("\n");

}

int iteration;

printf("Enter iteration: ");

scanf("%d",&iteration);

float x1,y1,z1;

x1=y1=z1=0.0;

int i = 0;

while(--iteration){

x1 = (arr[0][3] - arr[0][2]\*z1 - arr[0][1]\*y1)/arr[0][0];

y1 = (arr[1][3] - arr[1][2]\*z1 - arr[1][0]\*x1)/arr[1][1];

z1 = (arr[2][3] - arr[2][0]\*x1 - arr[2][1]\*y1)/arr[2][2];

printf("Iteration %d\n",++i);

printf("x: %f\n",x1);

printf("y: %f\n",y1);

printf("z: %f\n",z1);

}

printf("Final Values: \n");

printf("x: %f\n",x1);

printf("y: %f\n",y1);

printf("z: %f\n",z1);

}

1. **OUTPUT**

Enter order of matrix: 3

Enter arr[1][1] : 8

Enter arr[1][2] : -1

Enter arr[1][3] : 1

Enter arr[1][4] : 18

Enter arr[2][1] : 2

Enter arr[2][2] : 5

Enter arr[2][3] : -2

Enter arr[2][4] : 3

Enter arr[3][1] : 1

Enter arr[3][2] : 1

Enter arr[3][3] : -3

Enter arr[3][4] : -16

8.000000 -1.000000 1.000000 18.000000

2.000000 5.000000 -2.000000 3.000000

1.000000 1.000000 -3.000000 -16.000000

Enter iteration: 7

Iteration 1

x: 2.250000

y: -0.300000

z: 5.983333

Iteration 2

x: 1.464583

y: 2.407500

z: 6.624028

Iteration 3

x: 1.722934

y: 2.560437

z: 6.761124

Iteration 4

x: 1.724914

y: 2.614484

z: 6.779799

Iteration 5

x: 1.729336

y: 2.620186

z: 6.783174

Iteration 6

x: 1.729627

y: 2.621419

z: 6.783682

Final Values:

x: 1.729627

y: 2.621419

z: 6.783682