

## DEHRADUN CAMPUS

**PRACTICAL FILE / TERM WORK**

**CBNST LAB PMA-502**

**B.Tech CSE V**

**2023-24**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GRAPHIC ERA HILL UNIVERSITY, DEHRADUN**

**SUBMITTED TO SUBMITTED BY**

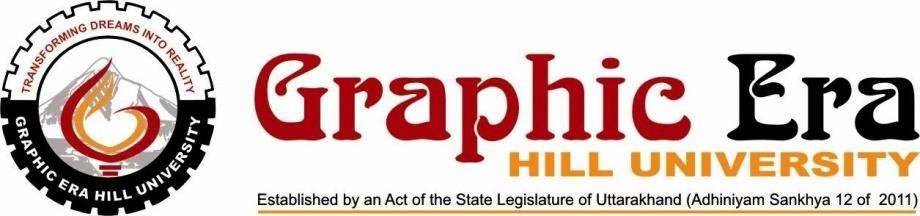
Mr. PURUSHOTTAM DAS NAME: Samay Maurya

ASST. PROFESSOR Examination Roll No.: 2119101

DEPARTMENT OF COMPUTER Course/Sem: B.Tech/5 sem

SCIENCE & ENGG.

**COLLEGE ROLL NO. 54 EXAMINATION ROLL NO. 2119101**



## DEHRADUN CAMPUS

THIS IS TO CERTIFY THAT Mr. / Ms. SAMAY MAURYA 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 5 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: -** D

# Number of students enrolled: -

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

# Name of Lab Instructor: - Lab Number:-

**Lab Time Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Lab Session** | **Day** | **Lecture Number** | **Timing** |
| 1 | Tuesday | 7 - 8 | 4:10 – 6:00 PM |

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 D**

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 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 SAMAY MAURYA Mo. No: 9118605875**

**Address Permanent : DLW Varanasi 221106**

**Father’s Name Rajesh Kumar Maurya Mo No 7985021184**

**Mother’s Name Kusum Maurya Mo No 904222703**

**Section: D Branch : CSE Semester 5th Class Roll No 54**

**Local AddressC-13 Clemet Town Dehradun** [**Email: mauryasamay4@gmail.com**](mailto:Email:%20mauryasamay4@gmail.com) **Grade A B C**

**Marks** 5 3 1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.** | **Name of the Experiment** | **D.O.P.** | **Date of** | **Grade** | **Grade** | **Total** | **Student’s** | **Teacher’s** |
| **No.** |  |  | **Submiss-** | **(Viva)** | **(Report** | **Marks** | **Signature** | **Signature** |
|  |  |  | **ion** |  | **File)** | **(out of** |  |  |
|  |  |  |  |  |  | **10)** |  |  |
| **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 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. |  |  |  |  |  |  |  |

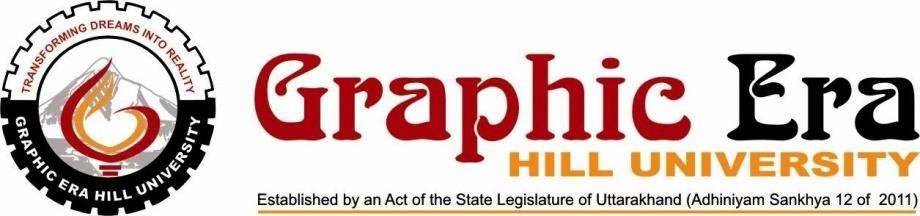
**Total No of Practical allotted: ………………………………………**

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

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

ACKNOWLEDGEMENT

Name of Student



**DEHRADUN CAMPUS**

INDEX

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

INDEX

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

**Page No.1**

## PROGRAM NO.1

**NAME:**

**COURSE:**

**BRANCH/SEMESTER: ROLL NO:**

**DATE:**

1. **OBJECTIVE**:
2. **METHOD**:
3. **PROGRAM**:
4. **OUTPUT**:

**Program 1**

**OBJECTIVE**:

**Write a program in C to find absolute, relative and percentage error for round- off and truncation cases.**

**METHOD**:

**Source Code:**

#include <stdio.h>

#include <math.h>

int main() {

double trueValue, approxValue;

printf("Enter the true value: ");

scanf("%lf", &trueValue);

printf("Enter the approximated value: ");

scanf("%lf", &approxValue);

double absoluteError = fabs(trueValue - approxValue);

// double absoluteError = (trueValue > approxValue) ? (trueValue - approxValue) : (approxValue - trueValue);

double relativeError = absoluteError / fabs(trueValue);

// double relativeError = absoluteError / ((trueValue != 0.0) ? trueValue : 1.0);

double percentageError = relativeError \* 100.0;

// double percentageError = relativeError \* 100.0;

printf("True Value: %.6lf\n", trueValue);

printf("Approximated Value: %.6lf\n", approxValue);

printf("Absolute Error: %.6lf\n", absoluteError);

printf("Relative Error: %.6lf\n", relativeError);

printf("Percentage Error: %.2lf%%\n", percentageError);

return 0;

}

**Output:**

samaymaurya@MSI:~$ cd Desktop

samaymaurya@MSI:~/Desktop$ cd cbnstlab

samaymaurya@MSI:~/Desktop/cbnstlab$ gcc q1.c

samaymaurya@MSI:~/Desktop/cbnstlab$ ./a.out

Enter the true value: 10.5

Enter the approximated value: 10.2

True Value: 10.500000

Approximated Value: 10.200000

Absolute Error: 0.300000

Relative Error: 0.028571

Percentage Error: 2.86%

**Program 2**

**OBJECTIVE**:

**Write a program in C to find the roots of any polynomial / transcendental equation using bisection method correct up to three decimal places.**

**METHOD**:

**Source Code:**

#include<stdio.h>

#include<math.h>

/

double f(double x){

return x\*x\*x-4.0\*x-9.0;

}

+

double bisection(double a, double b, double tolereance){

double c;

if (f(a)\*f(b)>=0)

{

printf("Bisection mehtod may not converge\n");

return -1.0;

}

int iteration =0;

while ((b-a)>=tolereance)

{

c=(a+b)/2.0;

if (f(c)==0.0)

{

return c;

}

else if (f(c)\*f(a)<0)

{

b=c;

}

else

{

a=c;

}

iteration++;

}

return c;

}

int main()

{

double a,b,tolerance,root;

printf("Enter the left Interval(a)");

scanf("%lf",&a);

printf("Enter the right Interval(b)");

scanf("%lf",&b);

printf("Enter the left tolerance in 0.001 types");

scanf("%lf",&tolerance);

root = bisection(a,b,tolerance);

if (root!=-1){

printf("Approx root corret up to three decimal places: %.3lf\n",root);

}

else{

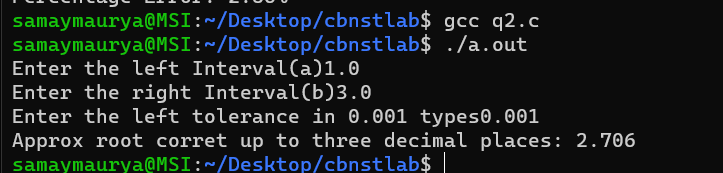
printf("Bisection methhod did not coverage within the given interval.\n");

}

return 0;

}

**Output:**

****

**Program 3**

**OBJECTIVE**:

**Write a program in C to find the solution of any transcendental equation using Regula-Falsi method correct up to three decimal places.**

**METHOD**:

**Source Code:**

/\*

The Regula-Falsi method, also known as the False Position method, is a numerical technique used to find the root of a transcendental equation.

It is similar to the bisection method but uses linear interpolation to approximate the root. Here's how the method works:

1. \*\*Initialization\*\*: Choose two initial guesses, \(x\_0\) and \(x\_1\), such that \(f(x\_0)\) and \(f(x\_1)\) have opposite signs,

indicating that a root lies between \(x\_0\) and \(x\_1\).

2. \*\*Interpolation\*\*: Compute the next approximation \(x\_2\) using linear interpolation based on the points

\((x\_0, f(x\_0))\) and \((x\_1, f(x\_1))\). The formula for \(x\_2\) is:

\[x\_2 = x\_1 - \frac{f(x\_1) \cdot (x\_1 - x\_0)}{f(x\_1) - f(x\_0)}\]

3. \*\*Convergence Check\*\*: Calculate \(f(x\_2)\). If \(f(x\_2)\) is very close to zero (i.e., less than a

specified tolerance), or if \(x\_2\) is very close to either \(x\_0\) or \(x\_1\), consider \(x\_2\) as

the root and stop. Otherwise, proceed to the next step.

4. \*\*Update Intervals\*\*: Determine whether the root lies between \(x\_0\) and \(x\_2\) or between

\(x\_1\) and \(x\_2\) based on the sign of \(f(x\_2)\). Replace \(x\_0\) or \(x\_1\) with \(x\_2\) accordingly.

5. \*\*Repeat\*\*: Repeat steps 2-4 until \(f(x\_2)\) is sufficiently close to zero or until the desired accuracy is achieved.

//

\*/

#include <stdio.h>

#include <math.h>

double f(double x) {

return x \* x \* x - 4.0 \* x - 9.0;

}

double df(double x) {

return 3.0 \* x \* x - 4.0;

}

double newtonRaphson(double x0, double tolerance) {

double x1, f0;

while (1) {

f0 = f(x0);

if (fabs(f0) < tolerance) {

return x0;

}

x1 = x0 - f0 / df(x0);

x0 = x1;

}

}

int main() {

double x0, tolerance, solution;

printf("Enter the initial guess (x0): ");

scanf("%lf", &x0);

printf("Enter the tolerance (e.g., 0.0001 for four decimal places): ");

scanf("%lf", &tolerance);

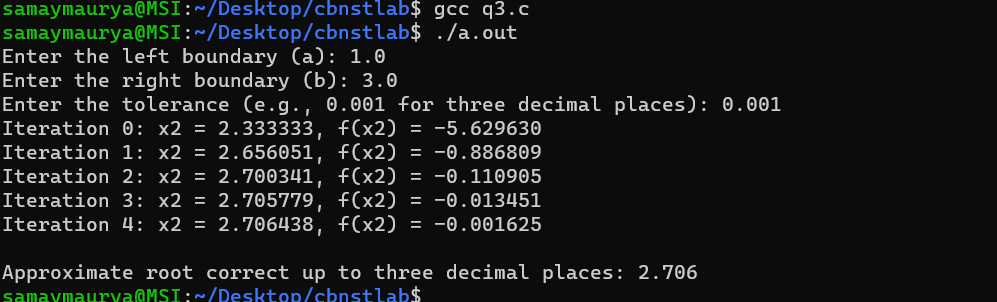
solution = newtonRaphson(x0, tolerance);

printf("\nApproximate solution correct up to four decimal places: %.4lf\n", solution);

return 0;

}

**Output:**

****

**Program 3**

**OBJECTIVE**:

Write a program in C to find the solution of any non-polynomial equation using Newton-Raphson method correct up to four decimal places.

METHOD:

Step 1: Define the Equation and Its Derivative

* First, define the equation you want to find the solution for by implementing a function that calculates the equation's value for a given x.
* Implement a separate function for the derivative of the equation. The derivative is needed for the Newton-Raphson method.

Step 2: Implement the Newton-Raphson Method

* Write a function that performs the Newton-Raphson method to find the solution of the equation.
* The function should take parameters such as the initial guess (x0) and the desired tolerance (how close to zero the result should be).
* Within this function, use a loop to iteratively perform the Newton-Raphson process.
* Calculate the next approximation (x1) using the formula: x1=x0−f(x0)/f(x0)*x*1​.
* Continue iterating until the equation value at x1 is close enough to zero (within the specified tolerance).

Step 3: Get User Input and Display Results

* In the main function, get user input for the initial guess (x0) and the tolerance.
* Call the Newton-Raphson method with these input values to find the solution.
* Display the approximate solution, correct up to four decimal places, to the user

**Source Code:**

#include <stdio.h>

#include <math.h>

double f(double x) {

return x \* x \* x - 4.0 \* x - 9.0;

}

double df(double x) {

return 3.0 \* x \* x - 4.0;

}

double newtonRaphson(double x0, double tolerance) {

double x1, f0;

while (1) {

f0 = f(x0);

if (fabs(f0) < tolerance) {

return x0;

}

x1 = x0 - f0 / df(x0);

x0 = x1;

}

}

int main() {

double x0, tolerance, solution;

printf("Enter the initial guess (x0): ");

scanf("%lf", &x0);

printf("Enter the tolerance (e.g., 0.0001 for four decimal places): ");

scanf("%lf", &tolerance);

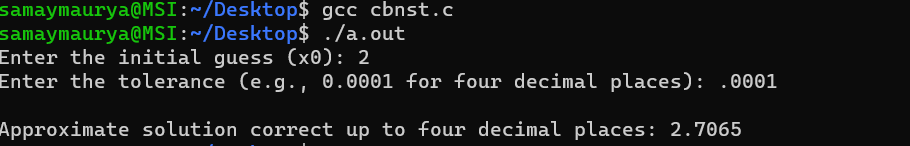
solution = newtonRaphson(x0, tolerance);

printf("\nApproximate solution correct up to four decimal places: %.4lf\n", solution);

return 0;

}

**OUTPUT**

****