LAB REPORT ON

"Solution of Non-Linear Equations"

Submitted By:

Name: Aditya Tharu

Date:23 Aug, 2024

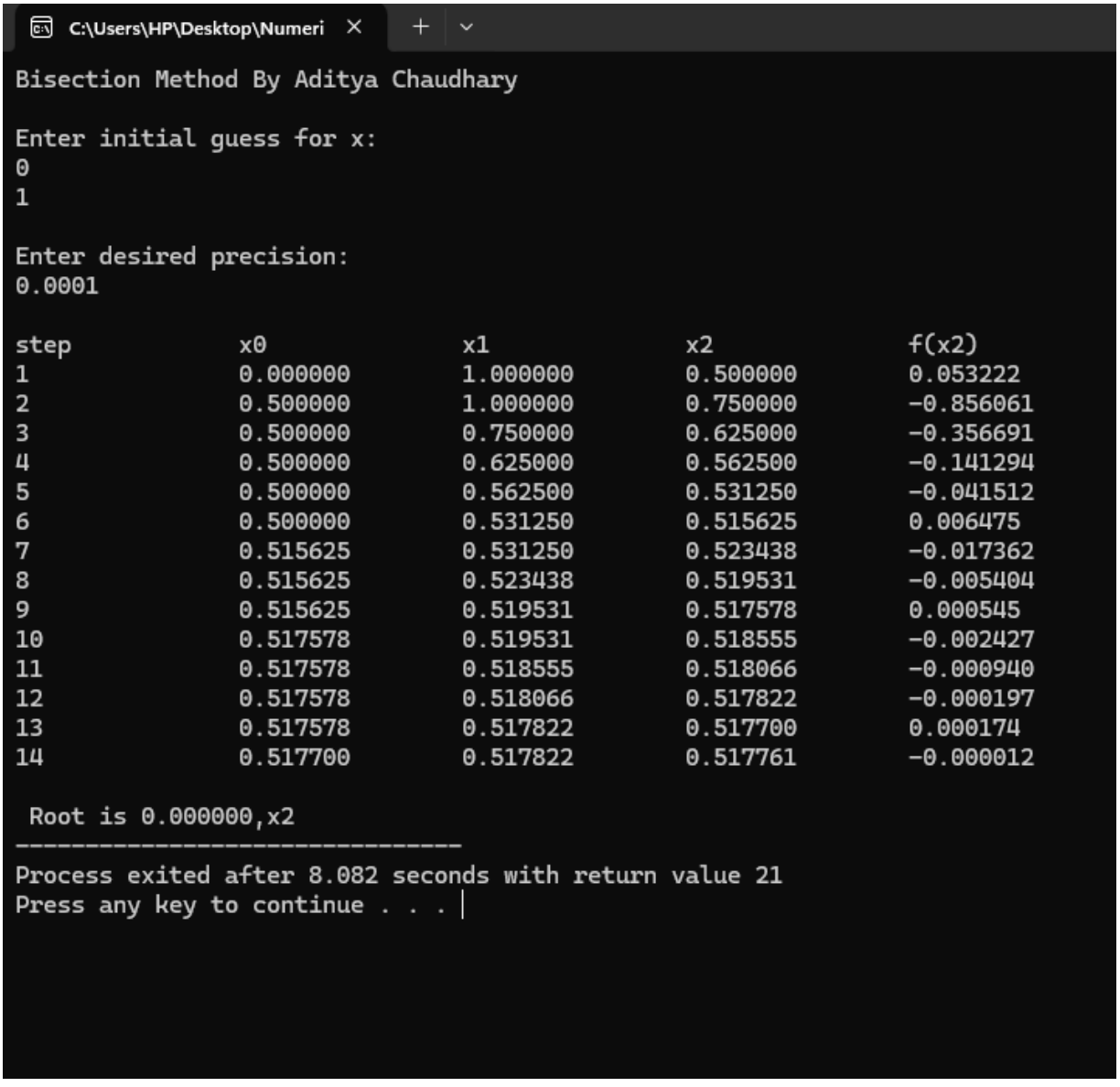
BISECTION METHOD

CODE:

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

/* Defining the equation to be solved. */
#define f(x) (cos(x) - (x) * exp(x))

int main() {
    float x0, x1, x2, f0, f1, f2, e;
    int step = 1;

    // Inputs
    do {
        printf("Enter two initial guesses: ");
        scanf("%f%f", &x0, &x1);
        printf("\nEnter tolerable error: ");
        scanf("%f", &e);

        // Calculating functional values
        f0 = f(x0);
        f1 = f(x1);

        // Checking whether the given guesses bracket the root or not
```

```
        if (f0 * f1 > 0.0) {
    printf("Incorrect initial guesses. The function values at the guesses must
have opposite signs.\n");
        }
    } while (f0 * f1 > 0.0);

    // Implementing Bisection Method
    printf("\nStep\t\tx0\t\tx1\t\tx2\t\tf(x2)\n");

    do {
        x2 = (x0 + x1) / 2;
        f2 = f(x2);

        printf("%d\t\t%f\t%f\t%f\t%f\n", step, x0, x1, x2, f2);
        if (f0 * f2 < 0) {
            x1 = x2;
            f1 = f2;
        } else {
            x0 = x2;
            f0 = f2;
        }
        step += 1;

    } while (fabs(f2) > e);
    printf("\nRoot is: %f\n", x2);
    return 0;
}
```

OUTPUT:


C:\Users\HP\Desktop\Numeri
Bisection Method By Aditya Chaudhary
Enter initial guess for x:
0
1
Enter desired precision:
0.0001
step x0 x1 x2 f(x2)
1 0.000000 1.000000 0.500000 0.053222
2 0.500000 1.000000 0.750000 -0.856061
3 0.500000 0.750000 0.625000 -0.356691
4 0.500000 0.625000 0.562500 -0.141294
5 0.500000 0.562500 0.531250 -0.041512
6 0.500000 0.531250 0.515625 0.006475
7 0.515625 0.531250 0.523438 -0.017362
8 0.515625 0.523438 0.519531 -0.005404
9 0.515625 0.519531 0.517578 0.000545
10 0.517578 0.519531 0.518555 -0.002427
11 0.517578 0.518555 0.518066 -0.000940
12 0.517578 0.518066 0.517822 -0.000197
13 0.517578 0.517822 0.517700 0.000174
14 0.517700 0.517822 0.517761 -0.000012
Root is 0.000000,x2
Process exited after 8.082 seconds with return value 21
Press any key to continue . . .

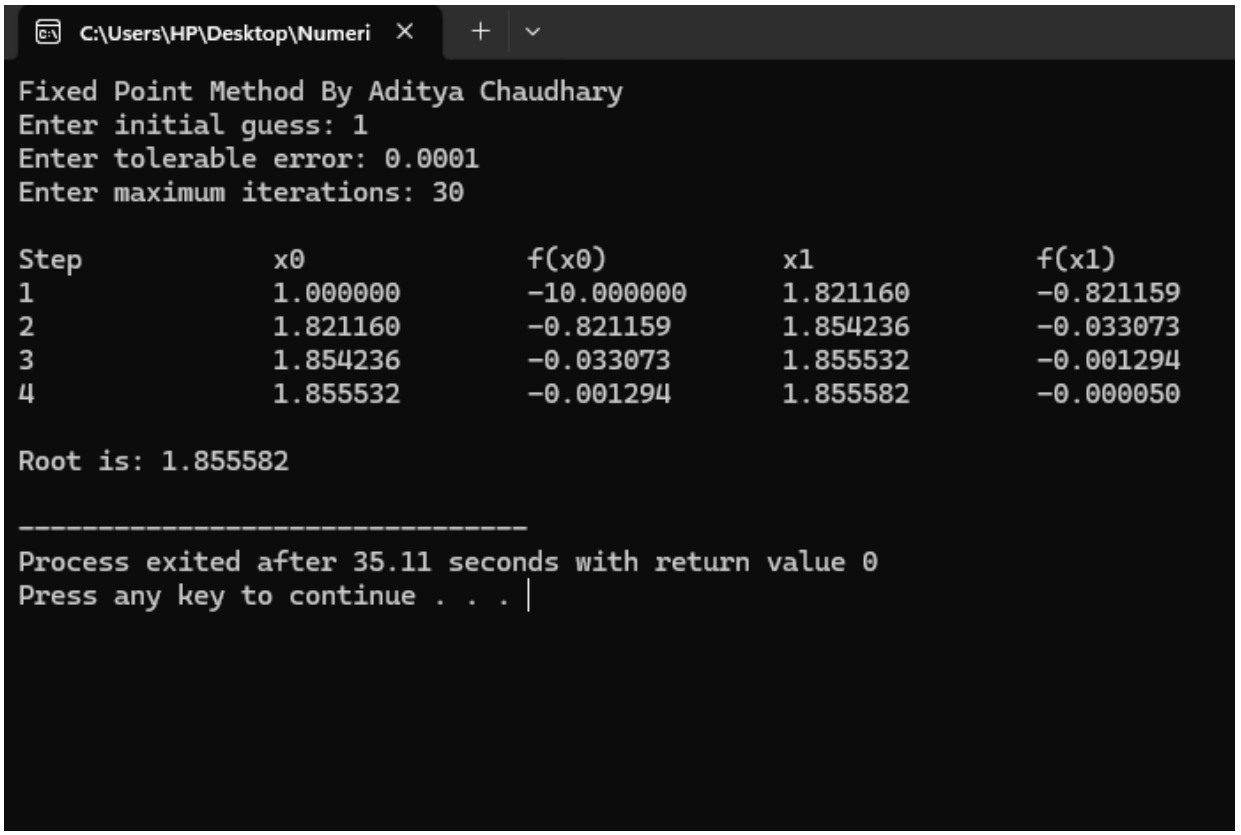
FIXED POINT METHOD

CODE:

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

/* Define the function f(x) to be solved */
#define f(x) ((x)*(x)*(x)*(x) - (x) - 10)

/* Write f(x) as x = g(x) and define g(x) here */
#define g(x) (pow((x) + 10, 0.25))

int main() {
    int step = 1, N;
    float x0, x1, e;

    // Inputs
    printf("Enter initial guess: ");
    scanf("%f", &x0);
    printf("Enter tolerable error: ");
    scanf("%f", &e);
    printf("Enter maximum iterations: ");
    scanf("%d", &N);

    // Implementing Fixed Point Iteration
```

```
    printf("\nStep\t\tx0\t\tf(x0)\t\tx1\t\tf(x1)\n");

    do {
        x1 = g(x0);
        printf("%d\t\t%f\t%f\t%f\t%f\n", step, x0, f(x0), x1, f(x1));

        // Check for convergence
        if (fabs(x1 - x0) < e) {
            printf("\nRoot is: %f\n", x1);
            return 0; // Exit successfully
        }

        x0 = x1;
        step += 1;

        // Check for maximum iterations
        if (step > N) {
            printf("Not Convergent.\n");
            return 0; // Exit with non-convergence message
        }

    } while (1);

    return 0;
}
```

OUTPUT:


C:\Users\HP\Desktop\Numeri
Fixed Point Method By Aditya Chaudhary
Enter initial guess: 1
Enter tolerable error: 0.0001
Enter maximum iterations: 30
Step x0 f(x0) x1 f(x1)
1 1.000000 -10.000000 1.821160 -0.821159
2 1.821160 -0.821159 1.854236 -0.033073
3 1.854236 -0.033073 1.855532 -0.001294
4 1.855532 -0.001294 1.855582 -0.000050
Root is: 1.855582
Process exited after 35.11 seconds with return value 0
Press any key to continue . . .

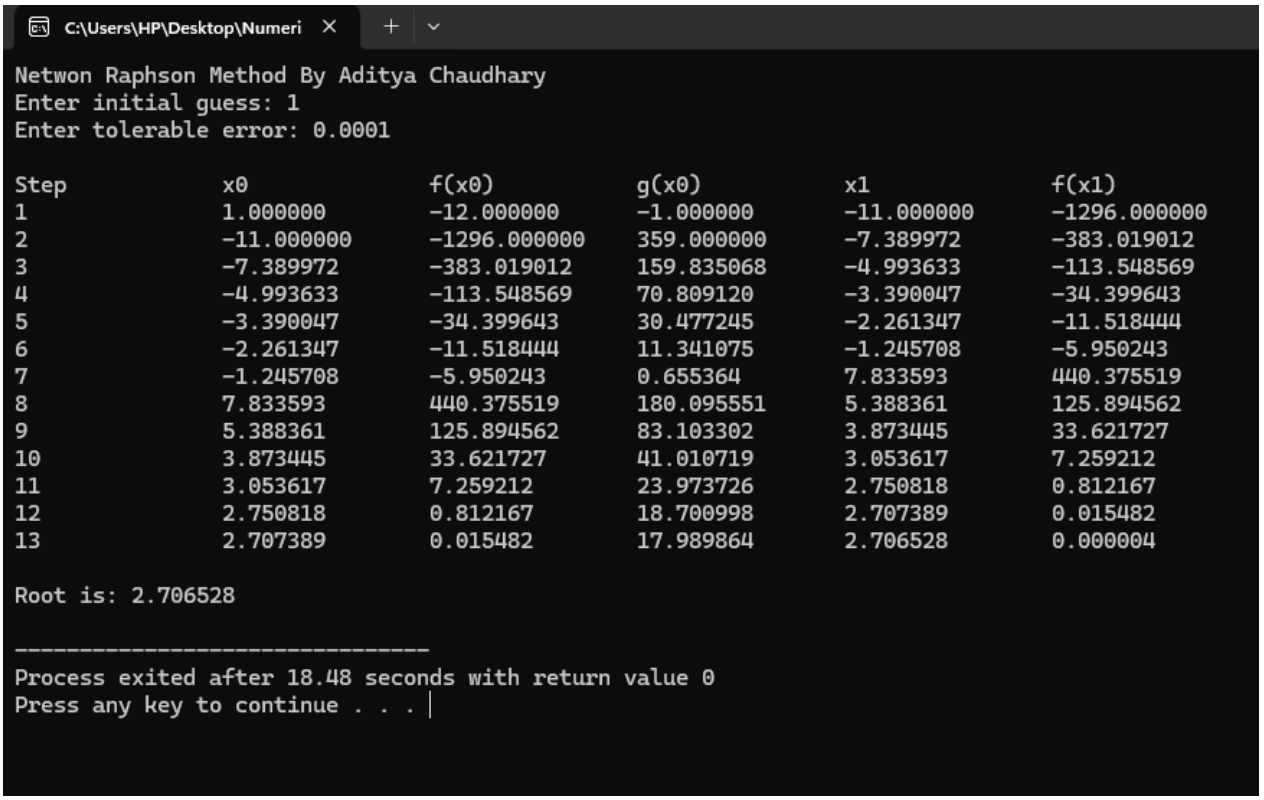
NEWTON RAPHSON METHOD

CODE:

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

/* Defining the equation to be solved.
  Change this equation to solve another problem. */
#define f(x) (x*x*x - 4*x9)

/* Defining the derivative of the equation.
  Change this equation as per the problem. */
#define g(x) (3*x*x4)

int main() {
   float x0, x1, f0, g0, e;
    int step = 1;

    // Inputs
   up:
    printf("Enter initial guess: ");
    scanf("%f", &x0);
    printf("Enter tolerable error: ");
    scanf("%f", &e);
```

```
    // Implementing Newton-Raphson Method
    printf("\nStep\t\tx0\t\tf(x0)\t\tg(x0)\t\tx1\t\tf(x1)\n");

    do {
        g0 = g(x0);
        f0 = f(x0);

        if (g0 == 0.0) {
            printf("Mathematical error: derivative is zero.\n");
           goto up;
        }

        x1 = x0 - f0/g0;
        printf("%d\t\t%f\t%f\t%f\t%f\t%f\n", step, x0, f0, g0, x1, f(x1));

        x0 = x1;
        step += 1;

    } while (fabs(f(x1)) > e);

    printf("\nRoot is: %f\n", x1);
    return 0;
}
```

OUTPUT:


C:\Users\HP\Desktop\Numeri
Netwon Raphson Method By Aditya Chaudhary
Enter initial guess: 1
Enter tolerable error: 0.0001
Step x0 f(x0) g(x0) x1 f(x1)
1 1.000000 -12.000000 -1.000000 -11.000000 -1296.000000
2 -11.000000 -1296.000000 359.000000 -7.389972 -383.019012
3 -7.389972 -383.019012 159.835068 -4.993633 -113.548569
4 -4.993633 -113.548569 70.809120 -3.390047 -34.399643
5 -3.390047 -34.399643 30.477245 -2.261347 -11.518444
6 -2.261347 -11.518444 11.341075 -1.245708 -5.950243
7 -1.245708 -5.950243 0.655364 7.833593 440.375519
8 7.833593 440.375519 180.095551 5.388361 125.894562
9 5.388361 125.894562 83.103302 3.873445 33.621727
10 3.873445 33.621727 41.010719 3.053617 7.259212
11 3.053617 7.259212 23.973726 2.750818 0.812167
12 2.750818 0.812167 18.700998 2.707389 0.015482
13 2.707389 0.015482 17.989864 2.706528 0.000004
Root is: 2.706528
Process exited after 18.48 seconds with return value 0
Press any key to continue . . .

REGULA FALSI METHOD

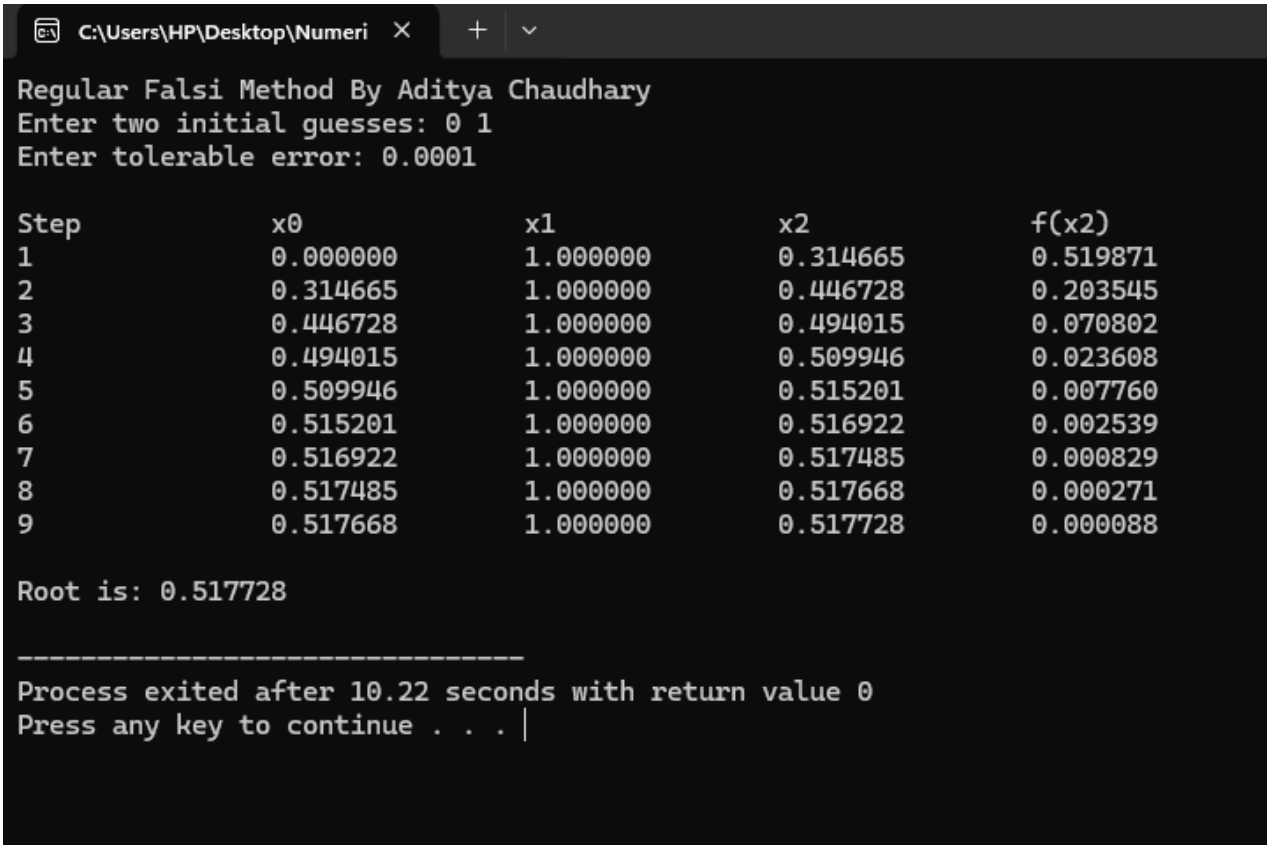
CODE:

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

/* Defining the equation to be solved. */
#define f(x) (cos(x) - (x) * exp(x))

int main() {
    float x0, x1, x2, f0, f1, f2, e;
    int step = 1;

    // Inputs
   up:
    printf("Enter two initial guesses: ");
    scanf("%f%f", &x0, &x1);
    printf("Enter tolerable error: ");
    scanf("%f", &e);

    // Calculating functional values
    f0 = f(x0);
    f1 = f(x1);

    // Checking whether the given guesses bracket the root or not
```

```
    if (f0 * f1 > 0.0) {
        printf("Incorrect initial guesses. The function values at the guesses must have opposite signs.\n");
        goto up;
    }

    // Implementing Regula Falsi (False Position) Method
    printf("\nStep\t\tx0\t\tx1\t\tx2\t\tf(x2)\n");

    do {
        // Avoid division by zero
        if (f0 == f1) {
            printf("Mathematical error: Division by zero.\n");
            return 1;
        }

        x2 = x0 - (x0 - x1) * f0 / (f0 - f1);
        f2 = f(x2);

        printf("%d\t\t%f\t%f\t%f\t%f\n", step, x0, x1, x2, f2);
        if (f0 * f2 < 0) {
            x1 = x2;
            f1 = f2;
        } else {
            x0 = x2;
            f0 = f2;
        }
```

```
        step += 1;

        if (fabs(x1 - x0) < e) {  // Improved stopping condition
            printf("\nRoot is: %f\n", x2);
            return 0;
        }

    } while (fabs(f2) > e);

    printf("\nRoot is: %f\n", x2);
    return 0;
}
```

OUTPUT:



SECANT METHOD

CODE:

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

/* Defining the equation to be solved. */
#define f(x) ((x)*(x)*(x) - 4*(x)-6)

int main() {
    float x0, x1, x2, f0, f1, f2, e;
    int step = 1, N;

    // Inputs
   up:
    printf("Enter initial guesses: ");
    scanf("%f%f", &x0, &x1);
    printf("Enter tolerable error: ");
    scanf("%f", &e);
    printf("Enter maximum iterations: ");
    scanf("%d", &N);

    // Implementing Secant Method
    printf("\nStep\t\tx0\t\tx1\t\tx2\t\tf(x2)\n");
```

```
do {
    f0 = f(x0);
    f1 = f(x1);

    // Handle division by zero
    if (f1 - f0 == 0.0) {
        printf("Mathematical error: f(x0) and f(x1) are equal.\n");
        goto up; // Re-prompt for new guesses
    }

    x2 = x1 - (x1 - x0) * f1 / (f1 - f0);
    f2 = f(x2);

    printf("%d\t\t%f\t%f\t%f\t%f\n", step, x0, x1, x2, f2);

    // Update values for the next iteration
    x0 = x1;
    x1 = x2;
    step += 1;

    if (step > N) {
        printf("Not Convergent.\n");
        return -1; // Exit with error code
    }

} while (fabs(f2) > e);
```

```
    printf("\nRoot is: %f\n", x2);
    return 0; // Exit successfully
}
```

OUTPUT: