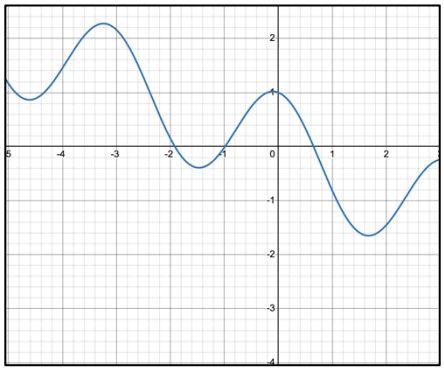
**Student Name: Mina Gerges**
**ID: 70040441**
**Computational Physics_ PHYS624**
**HW3: Finding Roots of nonlinear equations**

UAEU كلية العلوم College of Science
جامعة الإمارات العربية المتحدة United Arab Emirates University

## Finding roots of nonlinear equations

**a)Write a program that calculates the roots of the following non-linear fuction using Newton and Secant methods. Also apply brute force method together with the bisection method for following sub intervals.**

f(x) = Cos (2x) - 0.4x



**Answer:**

**Steps followed in the script:**

1. Graph the function using Desmos.

2. Defining a double function to be solved.
3. Defining the derivative( double) for the newtonian method.
4. Defining a function for the bisection method in the interval [-2,-1] as follows:
   a. Checking if {f(a) * f(b) >= 0}, so, initial guesses are incorrect.
   b. Passing the tolerance to the while loop
   c. Calculate the value of midpoint c.
   d. Check if f(c) = 0 , so the root is c.
   e. Checing if {f(c) * f(a) < 0} so , b=c and else is a=c.
   ( for the bisection method you to change the interval used for the calculation as this function has multiple roots )
5. Defining the newton function with iteration of 1000 and tolerance of 1e-6 the three initial guesses are [-2,-1,0.5].
   Note that: the initial guess is defined as a list of three guesses obtained from Desoms graphing, where they are automatically selected within the loop to get the three roots of the function.
6. Defining a double function for the secant method.
   ( for the secant method you to change the interval used for the calculation as this function has multiple roots )
7. Defining a function for th Brute Force Method with bisection together to scan the whole interval [-2,1].
8. Printing out the results and saving them to output file including iterations and calculated roots.

**Used intervals and initial guesses:**

- Enter the interval [a, b] for the Bisection Method: -2 -1
- Enter three initial guesses for Newton's Method: -2 -1 0.5
- Enter two initial guesses for Secant Method (x0 and x1): -2 -1
- For the Brute Force Method the start point is (-2) and the end point is (1) the step is (0.01)

*The script requires user input for values.

**Student Name: Mina Gerges**
**ID: 70040441**
**Computational Physics_ PHYS624**
**HW3: Finding Roots of nonlinear equations**

UAEU كلية العلوم College of Science
جامعة الإمارات العربية المتحدة United Arab Emirates University

**C++ script:**

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <fstream>

using namespace std;

// Define the function
double f(double x) {
    double k;
    k = cos(2 * x) - 0.4 * x;
    return k;
}
// Define the derivative
double df(double x) {
    double D;
    D = -2 * sin(2 * x) - 0.4;
    return D;
}
// Bisection Method
void bisection(double a, double b, double tol, int& iter, ofstream& outfile) {
    if (f(a) * f(b) >= 0) {
        outfile << "Bisection method: Incorrect initial guesses." << endl;
        return;
    }

    double c = a;
    iter = 0;
    outfile << "Iteration\tCurrent Root Estimate" << endl;

    while ((b - a) >= tol) {

        c = (a + b) / 2;
        iter++;

        if (f(c) == 0.0) {
            break;
        }
        else if (f(c) * f(a) < 0) {
            b = c;
        } else {
            a = c;
        }

        outfile << iter << "\t\t" << c << endl; // Output current estimate
    }

    outfile << "Bisection method root: " << c << endl;
}
```

**Student Name: Mina Gerges**
**ID: 70040441**
**Computational Physics_ PHYS624**
**HW3: Finding Roots of nonlinear equations**

UAEU كلية العلوم College of Science
جامعة الإمارات العربية المتحدة United Arab Emirates University

```
// Newton's Method
void newton(double x0, double tol, int max_iter, ofstream& outfile) {
    double h = f(x0) / df(x0);
    int iter = 0;
    outfile << "Iteration\tCurrent Root Estimate" << endl;

    while (abs(h) >= tol && iter < max_iter) {
        h = f(x0) / df(x0);
        x0 = x0 - h;
        outfile << iter + 1 << "\t\t" << x0 << endl;
        iter++;
    }

    if (iter == max_iter)
        outfile << "Newton's method did not converge within the maximum iterations." << endl;
    else
        outfile << "Newton's method root: " << x0 << endl;
}

// Secant Method
void secant(double x0, double x1, double tol, int max_iter, ofstream& outfile) {
    double x2;
    int iter = 0;
    outfile << "Iteration\tCurrent Root Estimate" << endl;

    while (iter < max_iter) {
        if (f(x0) == f(x1)) {
            outfile << "Secant method: Division by zero error." << endl;
            return;
        }

        // Apply Secant formula
        x2 = x1 - (f(x1) * (x1 - x0)) / (f(x1) - f(x0));

        if (abs(x2 - x1) < tol) {
            outfile << "Secant method root: " << x2 << endl;
            return;
        }

        // Update x0 and x1
        x0 = x1;
        x1 = x2;
        iter++;
        outfile << iter << "\t\t" << x2 << endl;
    }
    outfile << "Secant method did not converge within the maximum iterations." << endl;
}

// Brute Force Method
void bruteForce(double start, double end, double step, ofstream& outfile) {
    double x1 = start;
    double f1 = f(x1);

    for (double x2 = start + step; x2 <= end; x2 += step) {
        double f2 = f(x2);
```

```
        // Check for sign change
        if (f1 * f2 < 0) {

            double root = (x1 + x2) / 2.0;
            cout << "Root found: x = " << setprecision(6) << root << endl;
            outfile << "Root found: x = "<< "\t\t" << root<< endl;
        }

        // Update x1 and f1
        x1 = x2;
        f1 = f2;
    }
}

// Main function
int main() {
    double a, b;
    double tol = 1e-6;
    int max_iter = 1000;
    double start = -2, end = 1, step = 0.01;
    int iter = 0;

    ofstream outfile("HW3_Q1_Results.txt");
    if (!outfile) {
        cerr << "Error opening file!" << endl;
        return 1;
    }

    outfile << fixed << setprecision(6);
    outfile << "Enter the interval [a, b] for the Bisection Method: ";
    cout << "Enter the interval [a, b] for the Bisection Method: ";
    cin >> a >> b;

    // Apply Bisection Method
    bisection(a, b, tol, iter, outfile);

    // Apply Newton's Method
    double x0[3];
    cout << "Enter three initial guesses for Newton's Method: ";
    outfile << "Enter three initial guesses for Newton's Method: ";
    for (int i = 0; i < 3; ++i) {
        cin >> x0[i];
        outfile << "Initial guess " << i + 1 << ": " << x0[i] << endl;
    }
    for (int i = 0; i < 3; ++i) {
        newton(x0[i], tol, max_iter, outfile);
    }

    // Apply Secant Method
    double x1;
    cout << "Enter two initial guesses for Secant Method (x0 and x1): ";
    outfile << "Enter two initial guesses for Secant Method (x0 and x1): ";
    cin >> x0[0] >> x1;
    secant(x0[0], x1, tol, max_iter, outfile);
```

```
    // Apply Brute Force Method
    outfile << "Brute force method in interval [" << start << ", " << end << "] with step size
" << step << ":" << endl;
    bruteForce(start, end, step, outfile);

    outfile.close();
    return 0;
}
```

**Code results:**

Enter the interval [a, b] for the Bisection Method:

| Iteration | Current Root Estimate |
|---|---|
| 1 | -1.500000 |
| 2 | -1.750000 |
| 3 | -1.875000 |
| 4 | -1.937500 |
| 5 | -1.906250 |
| 6 | -1.921875 |
| 7 | -1.914062 |
| 8 | -1.917969 |
| 9 | -1.919922 |
| 10 | -1.918945 |
| 11 | -1.918457 |
| 12 | -1.918701 |
| 13 | -1.918823 |
| 14 | -1.918762 |
| 15 | -1.918732 |
| 16 | -1.918747 |
| 17 | -1.918739 |
| 18 | -1.918736 |
| 19 | -1.918734 |
| 20 | -1.918733 |

Bisection method root: -1.918733

Enter three initial guesses for Newton's Method: Initial guess 1: -2.000000

Initial guess 2: -1.000000

Initial guess 3: 0.500000

| Iteration | Current Root Estimate |
|---|---|
| 1 | -1.923518 |
| 2 | -1.918754 |
| 3 | -1.918734 |
| 4 | -1.918734 |

Newton's method root: -1.918734

| Iteration | Current Root Estimate |
|---|---|
| 1 | -0.988618 |
| 2 | -0.988692 |
| 3 | -0.988692 |

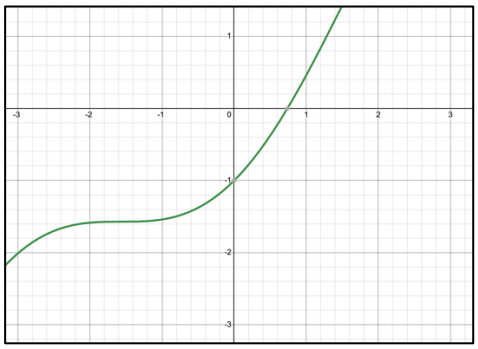
Newton's method root: -0.988692

| Iteration | Current Root Estimate |
|---|---|
| 1 | 0.663376 |

```
2               0.653242
3               0.653220
4               0.653220
Newton's method root: 0.653220
Enter two initial guesses for Secant Method (x0 and x1):
Iteration       Current Root Estimate
1               -1.099363
2               -0.987807
3               -0.988759
4               -0.988692
Secant method root: -0.988692
Brute force method in interval [-2.000000, 1.000000] with step size 0.010000:
Root found: x =         -1.915000
Root found: x =         -0.985000
Root found: x =         0.655000
```

**Student Name: Mina Gerges**
**ID: 70040441**
**Computational Physics_ PHYS624**
**HW3: Finding Roots of nonlinear equations**

**b) Write a program which finds the roots of the following function. Compare the results using bisection and false position methods. Please take interval of [0,4] and tolerance of 1.0e-6**



**f(x) =X - Cos(x)**

**Answer:**

**Steps followed to solve the problem::**

1. Graph the function using Desmos.
2. Defining a double function to be solved
3. Defining a double function for the bisection method as explained in the previous question in the interval [0,4].
4. Defining a double function for the False Position method.
5. The root is calculated through an iterative process until the tolerance is achieved.
6. Results are saved to output file.

**C++ Script**

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <fstream>

using namespace std;

// Define the function
double f(double x) {
    double k;
    k = x - cos(x);
    return k;
}

// Bisection Method
void bisection(double a, double b, double tol, int& iter, ofstream& outfile) {
    if (f(a) * f(b) >= 0) {
        outfile << "Bisection method: Incorrect initial guesses." << endl;
        return;
    }

    double c;
    iter = 0;
    outfile << "Iteration\tCurrent Root Estimate" << endl;

    while ((b - a) >= tol) {
        c = (a + b) / 2.0;
        iter++;

        if (f(c) == 0.0) {
            break;
```

```
        } else if (f(c) * f(a) < 0) {
            b = c;
        } else {
            a = c;
        }

        outfile << iter << "\t\t\t\t" << c << endl;
    }

    outfile << "Bisection method root: " << c << endl;
}

// False Position Method
void False_P(double a, double b, double tol, int max_iter, ofstream& outfile) {
    double x2;
    int iter = 0;
    outfile << "Iteration\tCurrent Root Estimate" << endl;

    while (iter < max_iter) {
        if (f(a) == f(b)) {
            outfile << "False_Position method: Division by zero error." << endl;
            return;
        }

        // Apply False position formula
        x2 = b - (f(b) * (b - a)) / (f(b) - f(a));

        if (fabs(x2 - b) < tol) {
            outfile << "False_Position method root: " << x2 << endl;
            return;
        }

        // Update a and b
        a = b;
        b = x2;
        iter++;
        outfile << iter << "\t\t\t\t" << x2 << endl;
    }

    outfile << "False_Position method did not converge within the maximum iterations." <<
endl;
}

// Main function
int main() {
    double a, b;
    double tol = 1e-6;
    int max_iter = 1000;
    int iter = 0;

    ofstream outfile("HW3_Q2_Results.txt");
    if (!outfile) {
```

```
        cerr << "Error opening file!" << endl;
        return 1;
    }

    outfile << fixed << setprecision(6);

    // Input interval [a, b]
    cout << "Enter the interval [a, b] for both methods: ";
    cin >> a >> b;

    // Apply Bisection Method
    outfile << "Calculating the root with bisection_Method.........." << endl;

    bisection(a, b, tol, iter, outfile);



    // Apply Fals Position Method
    outfile << "Calculating the root with false_position_Method.........." << endl;
    False_P(a, b, tol, max_iter, outfile);

    outfile.close();
    return 0;
}
```

**Code results:**

Calculating the root with bisection_Method.....

| Iteration | Current Root Estimate |
|---|---|
| 1 | 2.0000000000 |
| 2 | 1.0000000000 |
| 3 | 0.5000000000 |
| 4 | 0.7500000000 |
| 5 | 0.6250000000 |
| 6 | 0.6875000000 |
| 7 | 0.7187500000 |
| 8 | 0.7343750000 |
| 9 | 0.7421875000 |
| 10 | 0.7382812500 |
| 11 | 0.7402343750 |
| 12 | 0.7392578125 |
| 13 | 0.7387695312 |
| 14 | 0.7390136719 |
| 15 | 0.7391357422 |
| 16 | 0.7390747070 |
| 17 | 0.7391052246 |
| 18 | 0.7390899658 |
| 19 | 0.7390823364 |
| 20 | 0.7390861511 |
| 21 | 0.7390842438 |
| 22 | 0.7390851974 |

Bisection method root: 0.7390851974

**Student Name: Mina Gerges**
**ID: 70040441**
**Computational Physics_ PHYS624**
**HW3: Finding Roots of nonlinear equations**



Calculating the root with false_position_Method.....

| Iteration | Current Root Estimate |
|---|---|
| 1 | 0.7075083377 |
| 2 | 0.7442210930 |
| 3 | 0.7390488245 |
| 4 | 0.7390850921 |

False_Position method root: 0.7390851332