# MA202 LAB3

## Name: Dipean Dasgupta ID:202151188

Q1. Write a program to find the roots of the polynomial $(x-4)(x-2)(x+3)(x+7)=0$. Using all three methods: Bisection, False position, Secant. You can use the random no. generator to feed in the initial guesses as and when required. Comment upon the no. of iterations needed to find the root with error $\leq 10^{-6}$, in each case.

Solution code:

**Method 1: Bisection**

```
 #include <stdio.h>
#include <math.h>
#include <stdlib.h>
float func(float x)
{
return ((x - 4)*(x - 2)*(x + 3)*(x + 7));
}
void bisection(float *x, float a, float b)
{
*x = (a + b) / 2;
}
int main()
{
int itr = 0, maxmitr;
float x, a, b, x1, c;
float accuracy;
 printf("\nEnter the values of a, b, maximum iterations and accuracy: ");
scanf("%f %f %d %f", &a, &b, &maxmitr, &accuracy);
srand(time(0));
x = (rand() % (int)(b - a)) + a;
printf("\nInitial value of x = %f", x);
do
{
itr++;
c = x;
bisection(&x1, a, b);
if (func(c) * func(x1) < 0)
b = x1;
else
a = x1;
x = x1;
printf("\nAfter iteration no. %d, x = %f", itr, x);
} while (fabs(a - b) > accuracy && itr < maxmitr);
printf("\nAfter %d iterations, root = %f", itr, x);
```

```
return 0;
}
```

Output:

```
Enter the values of a, b, maximum iterations and accuracy: 2 3.5 20 .000001

Initial value of x = 2.000000
After iteration no. 1, x = 2.750000
After iteration no. 2, x = 3.125000
After iteration no. 3, x = 3.312500
After iteration no. 4, x = 3.406250
After iteration no. 5, x = 3.453125
After iteration no. 6, x = 3.476563
After iteration no. 7, x = 3.488281
After iteration no. 8, x = 3.494141
After iteration no. 9, x = 3.497070
After iteration no. 10, x = 3.498535
After iteration no. 11, x = 3.499268
After iteration no. 12, x = 3.499634
After iteration no. 13, x = 3.499817
After iteration no. 14, x = 3.499908
After iteration no. 15, x = 3.499954
After iteration no. 16, x = 3.499977
After iteration no. 17, x = 3.499989
After iteration no. 18, x = 3.499994
After iteration no. 19, x = 3.499997
After iteration no. 20, x = 3.499999
After 20 iterations, root = 3.499999
```

**Method 2: Secant**

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

float func(float x)
{
return ((x - 4)*(x - 2)*(x + 3)*(x + 7));
}
void secant(float *x, float x1, float x2)
{
*x = x2 - (func(x2)*(x2-x1))/(func(x2)-func(x1));
}
int main()
{
int itr = 0, maxmitr;
float x, x1, x2, c;
float accuracy;
printf("\nEnter the values of x1, x2, maximum iterations and accuracy: ");
scanf("%f %f %d %f", &x1, &x2, &maxmitr, &accuracy);
srand(time(0));
x = (rand() % (int)(x2 - x1)) + x1;
printf("\nInitial value of x = %f", x);
do
{
itr++;
c = x;
secant(&x, x1, x2);
x2 = x1;
```

```
x1 = x;
x = x1;
printf("\nAfter iteration no. %d, x = %f", itr, x);
} while (fabs(c - x) > accuracy && itr < maxmitr);
printf("\nAfter %d iterations, root = %f", itr, x);
return 0;
}
```

```
Enter the values of x1, x2, maximum iterations and accuracy: 2.5 3.5 15 .000001

Initial value of x = 2.500000
After iteration no. 1, x = -0.765625
After iteration no. 2, x = 1.925577
After iteration no. 3, x = 2.028887
After iteration no. 4, x = 2.000390
After iteration no. 5, x = 1.999998
After iteration no. 6, x = 2.000000
After iteration no. 7, x = 2.000000
After 7 iterations, root = 2.000000
```

## Method 3: False position

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
float func(float x)
{
return ((x - 4)*(x - 2)*(x + 3)*(x + 7));
}
void falseposition(float *x, float a, float b)
{
*x = (a*func(b) - b*func(a))/(func(b) - func(a));
}
int main()
{
int itr = 0, maxmitr;
float x, a, b, x1, c;
float accuracy;
printf("\nEnter the values of a, b, maximum iterations and accuracy: ");
scanf("%f %f %d %f", &a, &b, &maxmitr, &accuracy);
srand(time(0));
x = (rand() % (int)(b - a)) + a;
printf("\nInitial value of x = %f", x);
do
{
itr++;
c = x;
falseposition(&x1, a, b);
if (func(c) * func(x1) < 0)
b = x1;
else
```

```
a = x1;
x = x1;
printf("\nAfter iteration no. %d, x = %f", itr, x);
} while (fabs(a - b) > accuracy && itr < maxmitr);
printf("\nAfter %d iterations, root = %f", itr, x);
return 0;
}
```

## Output:

```
Enter the values of a, b, maximum iterations and accuracy: 1.325 3.458 15 0.000001

Initial value of x = 2.325000
After iteration no. 1, x = 2.496380
After iteration no. 2, x = -0.102687
After iteration no. 3, x = 2.017306
After iteration no. 4, x = 1.997424
After iteration no. 5, x = 2.000377
After iteration no. 6, x = 1.999945
After iteration no. 7, x = 2.000008
After iteration no. 8, x = 1.999999
After iteration no. 9, x = 2.000000
After iteration no. 10, x = 2.000000
After 10 iterations, root = 2.000000

...Program finished with exit code 0
Press ENTER to exit console.
```

Q2. Write a program to find all the roots of the equation $\lambda x = tan(x)$ in the region $0 \leq x \leq 20$, using any method. You can use the random no. generator to feed in the initial guesses. Take $\lambda = 1$ to begin with. What caution is to be taken to employ this method ?

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
int main()
{
int k ;
float x, y;
printf("enter value of k ");
scanf("%d" , &k);
printf("enter value of x ");
scanf("%f", &x);
y = k * x;
while (fabs(y - tan(x)) > 0.00001)
{
x = x - ((y - tan(x)) / k);
y = k * x;
}
printf("Root of the equation is %f", x);
return 0;
```

}

```
enter value of k 3
enter value of x 21.4
Root of the equation is -0.000004
D:\Cprogramming>
```

Q3. Write a program which contains a subroutine to evaluate the determinant of any given $3 \times 3$ matrix $A$. Find the determinant of the matrix $A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ using the program.

```
#include <stdio.h>

// Function to calculate determinant of a 3x3 matrix
int determinant(int mat[3][3]) {
    int result = 0;
    result = mat[0][0] * (mat[1][1] * mat[2][2] - mat[2][1] * mat[1][2]);
    result -= mat[0][1] * (mat[1][0] * mat[2][2] - mat[2][0] * mat[1][2]);
    result += mat[0][2] * (mat[1][0] * mat[2][1] - mat[2][0] * mat[1][1]);
    return result;
}

int main() {
    int mat[3][3];
    printf("Enter the elements of the 3x3 matrix: \n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            scanf("%d", &mat[i][j]);
        }
    }
    int det = determinant(mat);
    printf("The determinant of the matrix is: %d\n", det);
    return 0;
}
```

```
Enter the elements of the 3x3 matrix:
1 0 0
0 0 1
0 1 0
The determinant of the matrix is: -1
```

Q4. Now use the above program to evaluate the determinant of the matrix $A - \lambda 1$ for any given value of $\lambda$.

```
#include <stdio.h>
int determinant(int mat[3][3], int k) {
    int result = 0;
    result = (mat[0][0] - k) * (mat[1][1] * mat[2][2] - mat[2][1] * mat[1][2]);
    result -= mat[0][1] * (mat[1][0] * mat[2][2] - mat[2][0] * mat[1][2]);
    result += mat[0][2] * (mat[1][0] * mat[2][1] - mat[2][0] * (mat[1][1] - k));
    return result;
}
int main() {
    int mat[3][3];
    int k;
    printf("Enter the elements of the 3x3 matrix: \n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            scanf("%d", &mat[i][j]);
        }    }
    printf("Enter the value of k: \n");
    scanf("%d", &k);
    int det = determinant(mat, k);
    printf("The determinant of the matrix A - %dI is: %d\n", k,det);
    return 0;
}
```

```
Enter the elements of the 3x3 matrix:
4 7 3
8 1 3
5 2 6
Enter the value of k:
4
The determinant of the matrix A - 4I is: -138
```

Q5. Now define a function $f(\lambda) = Det(A - \lambda 1)$, and use the Bisection, Secant or False Positioning Method to find the roots of this function in the region $-2 \leq \lambda \leq 2$.

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

// function to calculate the determinant of a 3x3 matrix
```

```
int determinant(int mat[3][3], int k) {
   int result = 0;
   result = (mat[0][0] - k) * (mat[1][1] * mat[2][2] - mat[2][1] * mat[1][2]);
   result -= mat[0][1] * (mat[1][0] * mat[2][2] - mat[2][0] * mat[1][2]);
   result += mat[0][2] * (mat[1][0] * mat[2][1] - mat[2][0] * (mat[1][1] - k));
   return result;
}
// function to evaluate f(k)
double f(double k)
{
   int mat[3][3];
   for (int i = 0; i < 3; i++)
      for (int j = 0; j < 3; j++)
         mat[i][j] = A[i][j] - k;
   return determinant(mat);
}

// Bisection Method to find the root of f(k)
double bisection(double a, double b, double epsilon)
{
   double c = (a + b) / 2;
   double fa = f(a);
   double fb = f(b);
   double fc = f(c);
   while (fabs(fc) >= epsilon) {
      if (fa * fc < 0)
         b = c;
      else
         a = c;
      c = (a + b) / 2;
      fa = f(a);
      fb = f(b);
      fc = f(c);
   }
   return c;
}
int main()
{
   double k;
   printf("Enter the value of k: ");
   scanf("%lf", &k);
   int A[3][3];
   printf("Enter the elements of the 3x3 matrix: \n");
   for (int i = 0; i < 3; i++) {
      for (int j = 0; j < 3; j++) {
         scanf("%d", &A[i][j]);
      }
   }
```

```
    int mat[3][3];
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            mat[i][j] = A[i][j] - k;

    double epsilon = 1e-6;

    double root_bisection = bisection(-2, 2, epsilon);

    printf("Root (Bisection method) = %lf\n", root_bisection);

    return 0;
}
```