YILDIZ TEKNİK ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ

SAYISAL ANALİZ DERSİ DÖNEM PROJESİ

18011613 – Mehmet Emre Gül

YÖNTEMLER (4 Tane)

Bisection Yöntemi

Regula Falsi

Newton Raphson

Gauss Eliminasyon Yöntemi

Haziran, 2021

Bisection Yöntemi

```
G:\YTU\Say²sal Analiz\Proje Kodlar²\BisectionMethod_v2.exe
What is the degree of the polynomial : 3
Enter the coefficient of x^3  1
Enter the coefficient of x^2  0
Enter the coefficient of x^1  -1
Enter the coefficient of x^0  -2
polynomial :
1.00*(x^3) + 0.00*(x^2) + -1.00*(x^1) + -2.00*(x^0)
Enter the [a, b] :
a : 1

b : 2

iteration    a          b          c          previous-c     f(c)          Error          EpsilonError
1       1.000000        2.000000        1.500000        0.000000        -0.125000       1.500000        0.010000
2       1.500000        2.000000        1.750000        1.500000        1.609375        0.250000        0.010000
3       1.500000        1.750000        1.625000        1.750000        0.666016        0.125000        0.010000
4       1.500000        1.625000        1.562500        1.625000        0.252197        0.062500        0.010000
5       1.500000        1.562500        1.531250        1.562500        0.059113        0.031250        0.010000
6       1.500000        1.531250        1.515625        1.531250        -0.034054       0.015625        0.010000
7       1.515625        1.531250        1.523438        1.515625        0.012250        0.007813        0.010000

error is smaller than epsilonError
root found : 1.523438
program is ending....

--------------------------------
Process exited after 22.34 seconds with return value 0
Press any key to continue . . .
```

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

double bisectionFunction(double a, double b);
double f(double a);
int degree;
float coefficients[20];

int main(){
    int i;
    printf("What is the degree of the polynomial : ");
    scanf("%d", &degree);

    for(i=degree; i>-1; i--){
        printf("Enter the coefficient of x^%d  ",i);
        scanf("%f",&coefficients[i]);
    }
    printf("polynomial : \n");

    for(i=degree; i>-1; i--){
        printf("%.2f*(x^%d) ", coefficients[i], i);
        if (i != 0) printf("+ ");
    }

    double a,b;
    printf("\nEnter the [a, b] : ");
```

```
    printf("\na : ");
    scanf("%lf", &a);
    printf("\nb : ");
    scanf("%lf", &b);
    double result = bisectionFunction(a, b);
    printf("root found : %lf\n", result);
    printf("program is ending....\n");
}

double bisectionFunction(double x, double y){
    double errorEpsilon = 0.01;
    double a = x;
    double b = y;
    double c = 0;
    int iterationCounter = 0;
    double lastValue = 0;
    double lastRoot = 0;
    printf("\niteration    a\t\tb  \t\tc \t\tprevious-c \tf(c)  \t\tError
\t\tEpsilonError\n");
    if(f(a) == 0) return a;

    if(f(b) == 0) return b;

    if ( f(a) * f(b) < 0){
        while (1){//&& equationResult < 0){
            iterationCounter++;
            c = (a+b) / 2;

            printf("%d \t%lf \t%lf \t%lf \t%lf \t%lf \t%lf\n",
iterationCounter,a, b, c, lastRoot, f(c), fabs(lastRoot - c),
errorEpsilon);
            if (f(c) == 0){
                printf("\nresult is found \n");
                return c;
            }
            if( fabs(lastRoot - c)  < errorEpsilon ){
                printf("\nerror is smaller than epsilonError\n");
                return c;
            }
            if (f(c) * f(a) > 0){ a = c; }
            else if(f(c) * f(b) > 0){ b = c; }
            lastValue = f(c);
            lastRoot = c;
        }
    }
    printf("f(a) * f(b) > 0   \n");
    printf("a : %lf\n", a);
    printf("b : %lf\n", b);
    return c;
}

double f(double a){
    int i;
    double result = 0;
    for(i=degree; i>-1; i--){
        result = result + coefficients[i] * pow(a, i);
    }
    return result;
}
```

Regula Falsi

```
G:\YTU\Say²sal Analiz\Proje Kodlar²\RegulaFalsi_v2.exe
What is the degree of the polynomial : 3
Enter the coefficient of x^3  1
Enter the coefficient of x^2  0
Enter the coefficient of x^1  -2
Enter the coefficient of x^0  -5
polynomial :
1.00*(x^3) + 0.00*(x^2) + -2.00*(x^1) + -5.00*(x^0)
Enter the [a, b] :
a : 2

b : 3

iteration    a          b          c          previous-c     f(c)          Error          EpsilonError
1       2.000000   3.000000   2.058824   0.000000   -0.390800   2.058824   0.010000
2       2.058824   3.000000   2.081264   2.058824   -0.147204   0.022440   0.010000
3       2.081264   3.000000   2.089639   2.081264   -0.054677   0.008376   0.010000

Root is found !

Root found : 2.089639

--------------------------------
Process exited after 76.89 seconds with return value 0
Press any key to continue . . .
```

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


double regulafalsiFunction(double a, double b);
double f(double a);
double calculateC(double a, double b);
int degree;
float coefficients[20];

int main(){
    int i;
    printf("What is the degree of the polynomial : ");
    scanf("%d", &degree);

    for(i=degree; i>-1; i--){
        printf("Enter the coefficient of x^%d  ",i);
        scanf("%f",&coefficients[i]);
    }
    printf("polynomial : \n");

    for(i=degree; i>-1; i--){
        printf("%.2f*(x^%d) ", coefficients[i], i);
```

```
        if (i != 0) printf("+ ");
    }
    double a,b;
    printf("\nEnter the [a, b] : ");
    printf("\na : ");
    scanf("%lf", &a);
    printf("\nb : ");
    scanf("%lf", &b);
    double result = regulafalsiFunction(a, b);
    printf("\nRoot found : %lf\n", result);
    return 0;
}

double regulafalsiFunction(double x, double y){
    double errorEpsilon = 0.01;
    double a,b,c;
    a = x;
    b = y;
    double lastValue = 0;
    double lastRoot = 0;
    int iterationCounter = 0;
    printf("\niteration    a\t\tb  \t\tc \t\tprevious-c \tf(c)  \t\tError
\t\tEpsilonError\n");

    if(f(a) == 0) return a;

    if(f(b) == 0) return b;

    if(f(a) * f(b) < 0){
        while(1){
            iterationCounter++;
            c = calculateC(a,b);
            printf("%d \t%lf \t%lf \t%lf \t%lf \t%lf \t%lf \t%lf\n",
iterationCounter, a, b, c, lastRoot, f(c), fabs(lastRoot - c),
errorEpsilon);
            if( fabs(c - lastRoot) < errorEpsilon){
                printf("\nRoot is found !\n");
                return c;
            }
            if(f(a)*f(c) < 0) b=c;
            if(f(a)*f(c) > 0) a=c;
            lastValue = f(c);
            lastRoot = c;
        }
    }
    else{
        printf("Wrong initials \n");
    }
}

double f(double a){
    int i;
    double result = 0;
    for(i=degree; i>-1; i--){
        result = result + coefficients[i] * pow(a, i);
    }
    return result;
}

double calculateC(double a, double b){
    return ( b*f(a) - a*f(b) ) / (f(a)-f(b));
}
```

Newton Raphson

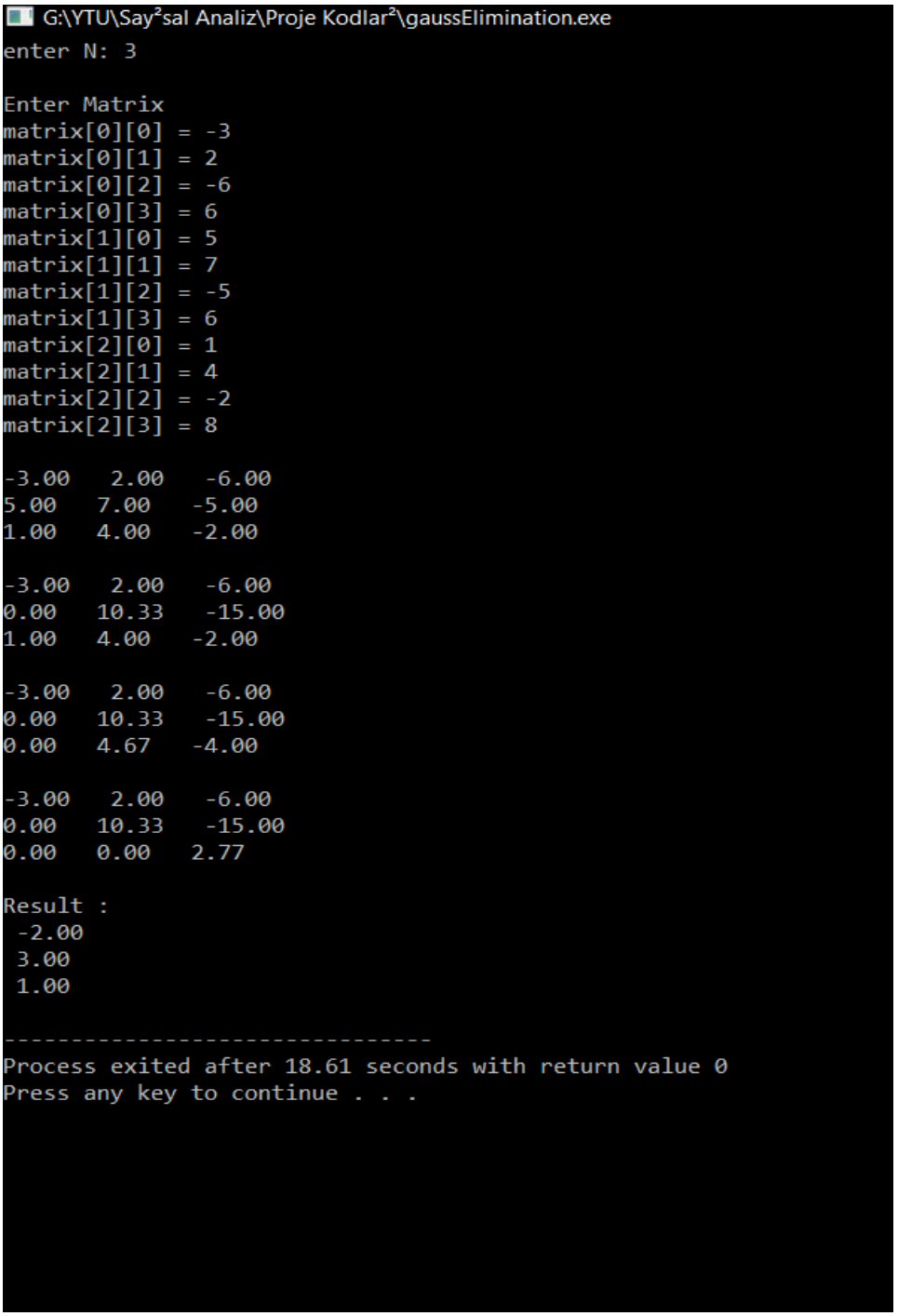
```
G:\YTU\Say²sal Analiz\Proje Kodlar²\newtonRaphson_v2.exe
What is the degree of the polynomial : 3
Enter the coefficient of x^3  1
Enter the coefficient of x^2  0
Enter the coefficient of x^1  -1
Enter the coefficient of x^0  -1
polynomial :
1.00*(x^3) + 0.00*(x^2) + -1.00*(x^1) + -1.00*(x^0)
derivative :
3.00*(x^2) + 0.00*(x^1) + -1.00*(x^0)
Enter the [a] :
a : 1.5

iteration       x               previous-x      f(x)            Error           EpsilonError
    1           1.347826        0.000000        0.100682        1.347826        0.010000
    2           1.325200        1.347826        0.002058        0.022626        0.010000
    3           1.324718        1.325200        0.000001        0.000482        0.010000

Root is found !

Root found : 1.324718

--------------------------------
Process exited after 30.3 seconds with return value 0
Press any key to continue . . .
```

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

double newtonRaphson(double a1);
double f(double a);
double f_derivative(double a);
double calculateX(double a);
int degree;
float coefficients[20];
int degreeD;
float coefficientsD[20];

int main(){
    int i;
    printf("What is the degree of the polynomial : ");
    scanf("%d", &degree);
    for(i=degree; i>-1; i--){
        printf("Enter the coefficient of x^%d  ",i);
        scanf("%f",&coefficients[i]);
    }
    printf("polynomial : \n");
    for(i=degree; i>-1; i--){
```

```
        printf("%.2f*(x^%d) ", coefficients[i], i);
        if (i != 0) printf("+ ");
    }

    degreeD = degree + -1;
    for(i=degreeD; i>-1; i--){
        coefficientsD[i] = coefficients[i+1] * (i+1);
    }
    printf("\nderivative : \n");
    for(i=degreeD; i>-1; i--){
        printf("%.2f*(x^%d) ", coefficientsD[i], i);
        if (i != 0) printf("+ ");
    }
    double a,b;
    printf("\nEnter the [a] : ");
    printf("\na : ");
    scanf("%lf", &a);

    double result = newtonRaphson(a);
    printf("\nRoot found : %lf\n", result);
    return 0;
}

double newtonRaphson(double a1){
    double errorEpsilon = 0.01;
    double a,x;
    int iterationCounter = 0;
    double lastRoot = 0;
    a = a1;
    printf("\niteration  \tx \t\tprevious-x \tf(x)  \t\tError
\t\tEpsilonError\n");
    x = a;
    if(f(x) == 0) return x;
    while(1){
        iterationCounter++;
        x = calculateX(x);
        printf("    %d \t\t%lf  \t%lf \t%lf \t%lf \t%lf\n",
iterationCounter, x , lastRoot, f(x),  fabs(x - lastRoot)  ,errorEpsilon);

        if( fabs(x - lastRoot) < errorEpsilon){
            printf("\nRoot is found !\n");
            return x;
        }
        lastRoot = x;
        if(iterationCounter > 15) return x;
    }
    return 0;
}

double calculateX(double a){
    return a - ( f(a) / f_derivative(a) );
}

double f(double a){
    int i;
    double result = 0;
    for(i=degree; i>-1; i--){
        result = result + coefficients[i] * pow(a, i);
    }
    return result;
}
```

```
double f_derivative(double a){
    int i;
    double result = 0;
    for(i=degreeD; i>-1; i--){
        result = result + coefficientsD[i] * pow(a, i);
    }
    return result;
}
```

Gauss Eliminasyon Yöntemi



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

void gaussElimination(int n, float matrix[n][n+1]);
void print_matrix(int n, float matrix[25][25]);

int main(){
    int x1,y1;
    int i = 0;
    int j = 0;
    int x = 0;
    float oran;
    int n;
    int k = 0;
    printf("enter N: ");
    scanf("%d", &n);
    float matrix_v2[25] = {0};
    printf("\nEnter Matrix\n");
    float matrix[25][25];
    for(i=0;i<n;i++){
          for(j=0;j<n+1;j++){
               printf("matrix[%d][%d] = ",i,j);
               scanf("%f", &matrix[i][j]);
          }
    }

    printf("\n");
    print_matrix(n, matrix);

    for (i=0 ; i<n ; i++){
        if (matrix[i][i] == 0){
            printf("There is error / divided by zero ! ");
        }
        for(j=i+1 ; j<n ; j++){
            oran = matrix[j][i] / matrix[i][i];
            for(x=0 ; x<n+1 ; x++){
                matrix[j][x] = matrix[j][x] - oran * matrix[i][x];
            }
            print_matrix(n, matrix);

        }
    }

    matrix_v2[n-1] = matrix[n-1][n] / matrix[n-1][n-1];

    for (i=n-2 ; i>-1 ; i--){
        matrix_v2[i] = matrix[i][n];
        for(j=i+1 ; j<n ; j++){
            matrix_v2[i] = matrix_v2[i] - matrix[i][j]*matrix_v2[j];
        }
        matrix_v2[i] = matrix_v2[i] / matrix[i][i];
    }

    printf("Result : \n");
    for(k=0 ; k<n ; k++){
        printf(" %.2f\n", matrix_v2[k]);
    }


    return 0;
}
```

```
void print_matrix(int n, float matrix[25][25]){
    int x1,y1;
    for (x1=0 ; x1<n ; x1++){
        for(y1=0; y1<n ; y1++){
            printf("%.2f   ", matrix[x1][y1]);
        }
        printf("\n");
    }
    printf("\n");
}
```