CAN CANKU 19011072

Yaptığım konular

1. Bisection
2. Regula-Falsi
3. Newton-Rapshon
4. NxN'lik bir matrisin tersi
5. Gauss Eleminasyon
6. Gauss Seidal
7. Sayısal Türev (merkezi, ileri ve geri)
8. Simpson yöntemi
9. Trapez yöntemi
10. Değişken dönüşümsüz Gregory Newton Enterpolasyonu

Bisection

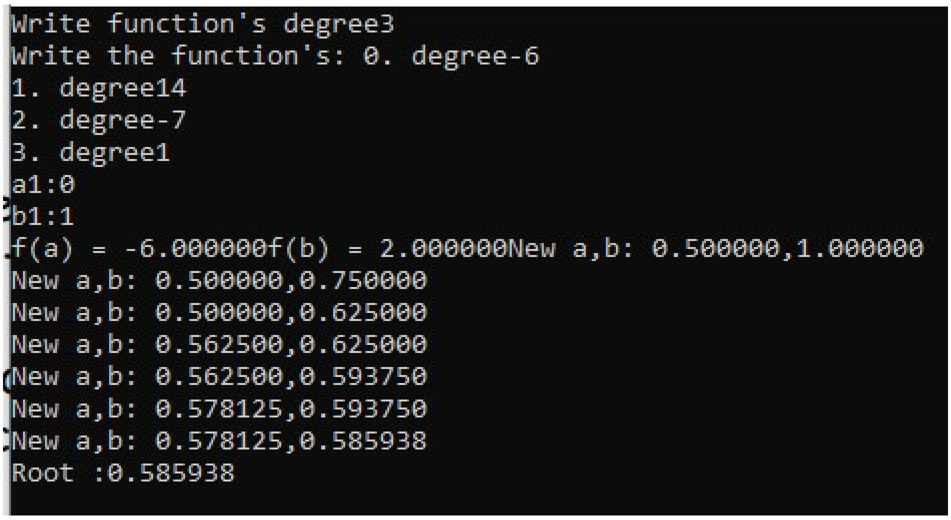
```
#include<stdio.h>


double e=0.01;
double f_result(double x,double f[],int degree){


    double result=0;
    int i,j;
    double tempfunction[degree];


    for(i=0;i<=degree;i++){
        tempfunction[i]=f[i];
    }


    for(i = 0; i <= degree; i++){
```

```
                        for(j = 0; j < i; j++){
                                tempfunction[i] = tempfunction[i] * x;
                        }
                        result = result + tempfunction[i];
                }
                return result;
}

double bisection_root(double a, double b,double f[], int degree) {

                double next;

                while((b-a) > e){
                        next = (a+b)/2;



                        if(f_result(next,f,degree) == 0){
                                return next;
                         }else if(f_result(b,f,degree)*f_result(next,f,degree) < 0){
                                printf("New a,b: %lf,%lf\n",next, b);
                                a = next;
                         }else{
                                printf("New a,b: %lf,%lf\n", a,next);
                                b = next;
                         }
                }
                return next;
}



int main(){
```

```
double a,b;
int degree;
int i;
double function[10];


printf("Write function's degree");


scanf("%d", &degree);
printf("Write the function's: ");
for(i = 0; i <= degree; i++){
    printf("%d. degree", i);
    scanf("%lf", &function[i]);
}




printf("a1:");
scanf("%lf", &a);
printf("b1:");
scanf("%lf", &b);
printf("f(a) = %lf", f_result(a,function,degree));
printf("f(b) = %lf", f_result(b,function,degree));
while((f_result(a,function,degree)) * (f_result(b,function,degree)) >= 0){


    if(f_result(a,function,degree) * f_result(b,function,degree) == 0){
        printf("A or B is root");
    }else{
        printf("f(a) * f(b) should be negative\n");
```

```
            printf("%lf, %lf", a,b);
            printf("f(a) = %lf", f_result(a,function,degree));
            printf("f(b) = %lf", f_result(b,function,degree));
            printf("a1:");
            scanf("%lf", &a);
            printf("b1:");
            scanf("%lf", &b);
        }

    }



    printf("Root :%lf\n", bisection_root(a,b,function,degree));



    return 0;
}
```

EKRAN GÖRÜNTÜSÜ



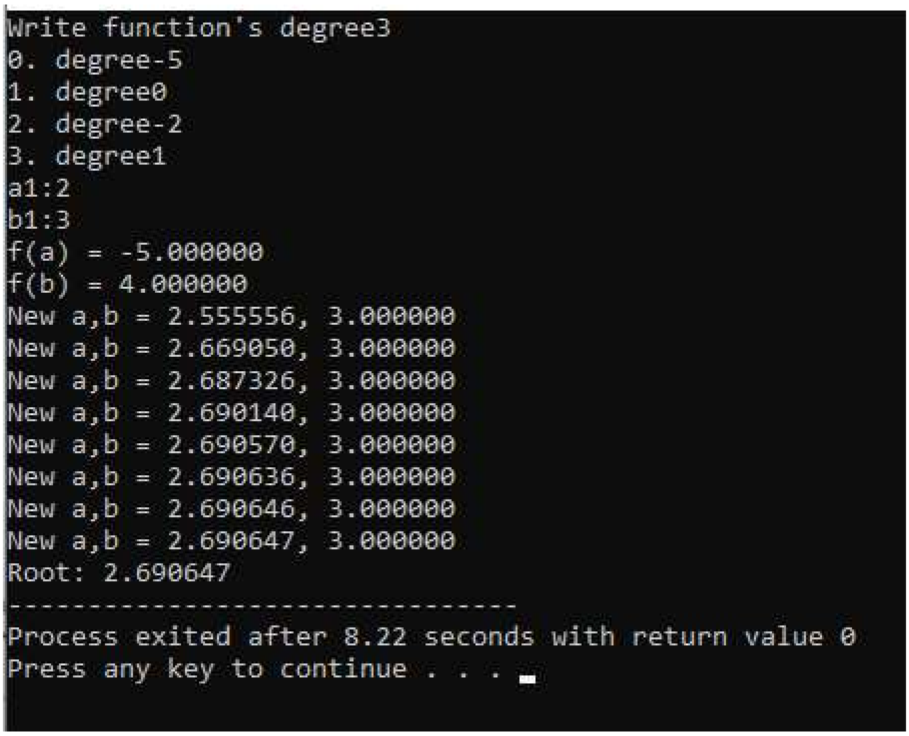
```
Write function's degree3
Write the function's: 0. degree-6
1. degree14
2. degree-7
3. degree1
a1:0
b1:1
f(a) = -6.000000f(b) = 2.000000New a,b: 0.500000,1.000000
New a,b: 0.500000,0.750000
New a,b: 0.500000,0.625000
New a,b: 0.562500,0.625000
New a,b: 0.562500,0.593750
New a,b: 0.578125,0.593750
New a,b: 0.578125,0.585938
Root :0.585938
```

REGULA-FALSİ

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

```
double f_result(double x,double function[],int degree){

        double result=0;
        int i,j;
        double tempfunction[degree];

        for(i=0;i<=degree;i++){
                tempfunction[i]=function[i];
        }

        for(i = 0; i <= degree; i++){
                for(j = 0; j < i; j++){
                        tempfunction[i] = tempfunction[i] * x;
                }
                result = result + tempfunction[i];
        }
        return result;
}

double regulafalsi(double a,double b,double function[],int degree){
        double c;
        double fa,fb,fc;
        int z = 2;
        double e=0.001;
        while((b-a)/z>e){
                fa = f_result(a,function,degree);
                fb = f_result(b,function,degree);
                if(fa*fb<0){
```

```
			c = (b*fa-a*fb)/(fa-fb);
			fc = f_result(c,function,degree);

			if((fa*fc<0)){
				b=c;
				printf("New a,b = %lf, %lf\n", a, b);
			}else if(fa*fc>0){
				a=c;
				printf("New a,b = %lf, %lf\n", a, b);
			}else{
				return c;
			}
		}
		z *= 2;
	}
	return c;
}

int main(){

	double a,b;
	int degree;
	int i;
	double function[10];


	printf("Write function's degree");

	scanf("%d", &degree);

	for(i = 0; i <= degree; i++){
        printf("%d. degree", i);
        scanf("%lf", &function[i]);
    }

    printf("a1:");
    scanf("%lf", &a);
    printf("b1:");
    scanf("%lf", &b);
    printf("f(a) = %lf\n", f_result(a,function,degree));
    printf("f(b) = %lf\n", f_result(b,function,degree));
    while((f_result(a,function,degree)) * (f_result(b,function,degree)) >= 0){

        if(f_result(a,function,degree) * f_result(b,function,degree) == 0){
            printf("A or B is root");
        }else{
            printf("f(a) * f(b) should be negative \n");
            printf("%lf, %lf", a,b);
            printf("f(a) = %lf", f_result(a,function,degree));
            printf("f(b) = %lf", f_result(b,function,degree));
            printf("a1:");
            scanf("%lf", &a);
            printf("b1:");
            scanf("%lf", &b);
        }

    }
    printf("Root: %lf", regulafalsi(a,b,function,degree));

    return 0;
}
```



```
Write function's degree3
0. degree-5
1. degree0
2. degree-2
3. degree1
a1:2
b1:3
f(a) = -5.000000
f(b) = 4.000000
New a,b = 2.555556, 3.000000
New a,b = 2.669050, 3.000000
New a,b = 2.687326, 3.000000
New a,b = 2.690140, 3.000000
New a,b = 2.690570, 3.000000
New a,b = 2.690636, 3.000000
New a,b = 2.690646, 3.000000
New a,b = 2.690647, 3.000000
Root: 2.690647
--------------------------------
Process exited after 8.22 seconds with return value 0
Press any key to continue . . .
```

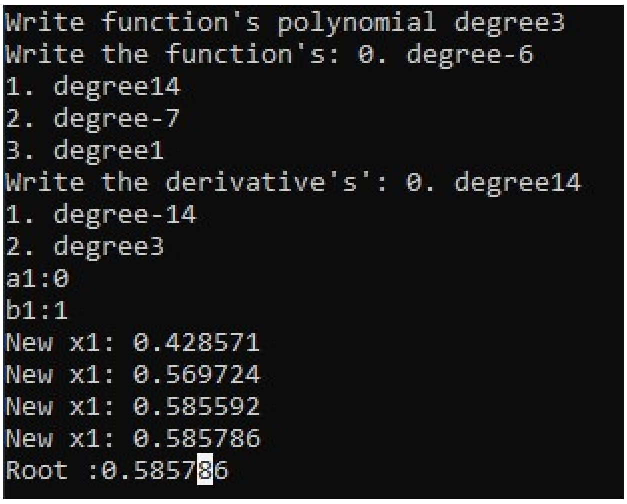
NEWTON RAPHSON YÖNTEMİ

```
#include<stdio.h>


double f_result(double x,double f[],int degree){


        double result=0;

        int i,j;

        double tempfunction[degree];


        for(i=0;i<=degree;i++){

                tempfunction[i]=f[i];

        }


        for(i = 0; i <= degree; i++){

                for(j = 0; j < i; j++){

                        tempfunction[i] = tempfunction[i] * x;

                }

                result = result + tempfunction[i];
```

```
    }
    return result;
}

double newtonraphson(double x,double y, double f[], double d[], int degree){
    double e = 0.000001;
    double x1,absolute_value,fx,fdx;
    fx = f_result(x,f,degree);
    fdx = f_result(x,d,degree-1);
    x1 = x - fx/fdx;
    absolute_value = x1-x;
    if(absolute_value < 0){
        absolute_value *= -1;
        }
    while(absolute_value > e){
        printf("New x1: %lf\n", x1);
        x = x1;
        fx = f_result(x,f,degree);
        fdx = f_result(x,d,degree-1);
        x1 = x - fx/fdx;
        absolute_value = x1-x;
        if(absolute_value < 0){
            absolute_value *= -1;
        }
    }
    return x1;
}

int main(){
```

```c
double a,b;
int degree;
int i;
double function[10];
double derivative[9];

printf("Write function's polynomial degree");

scanf("%d", &degree);

printf("Write the function's: ");
for(i = 0; i <= degree; i++){
        printf("%d. degree", i);
        scanf("%lf", &function[i]);
}

printf("Write the derivative's': ");
for(i = 0; i <= degree-1; i++){
        printf("%d. degree", i);
        scanf("%lf", &derivative[i]);
}


printf("a1:");
scanf("%lf", &a);
printf("b1:");
scanf("%lf", &b);

printf("Root :%lf\n", newtonraphson(a,b,function,derivative,degree));

return 0;
```

}



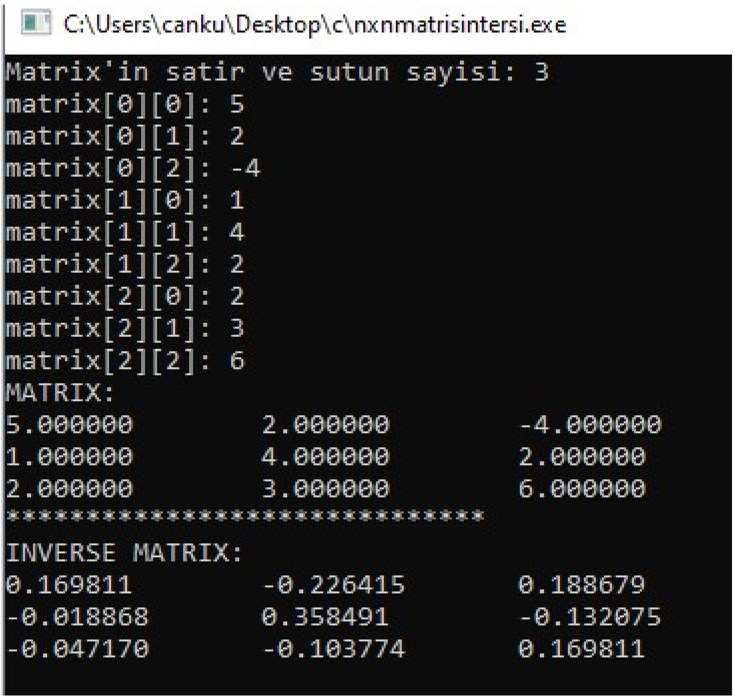
```
Write function's polynomial degree3
Write the function's: 0. degree-6
1. degree14
2. degree-7
3. degree1
Write the derivative's': 0. degree14
1. degree-14
2. degree3
a1:0
b1:1
New x1: 0.428571
New x1: 0.569724
New x1: 0.585592
New x1: 0.585786
Root :0.585786
```

NXN Matrix'in Tersi

```
#include<stdio.h>


int main(){


        double matrix [20][20];

        double inverse[20][20];

        double temp;

        int n;

        int i,j,k;




        printf("Matrix'in satir ve sutun sayisi: ");

        scanf("%d", &n);





        for(i = 0; i < n;i++){

                for(j = 0; j < n; j++){

                        printf("matrix[%d][%d]: ", i,j);

                        scanf("%lf", &matrix[i][j]);
```

```
            inverse[i][j] = 0;
        }
        inverse[i][i] = 1;
    }
    printf("MATRIX:\n");
    for(i = 0; i < n; i++){
        for(j = 0; j< n; j++){
            printf("%lf\t", matrix[i][j]);
        }
        printf("\n");
    }

    for(i = n-1; i > n; i--){//Matrix[0][0], 0 olduğunda da çalışması için
                            //satırları ilk sütunlarının büyüklüklerine göre
sıralama
        if(matrix[i-1][0] < matrix[i][0]){
            for(j = 0; j < n; j++){
                temp = matrix[i][j];
                matrix[i][j] = matrix[i-1][j];
                matrix[i-1][j] = temp;
            }
        }
    }



    for(i = 0; i < n; i++){
        if(matrix[i][i] == 0){
            printf("Diyagonelde 0 var hatali calisacak.");//Determinantı 0?
        }
        temp = matrix[i][i];
        for(j = 0; j < n; j++){// i. satırı matrix[i][i] ye böl.
```

```
            matrix[i][j] /= temp;

            inverse[i][j] /= temp;

        }

        for(j = 0; j < n; j++){//i. sütunu sıfırlamak için i j'ye eşit olmadığında

                                //j. satırın satırın i. elemanını temp
olarak al

            temp = matrix[j][i];

            for(k = 0; k < n; k++){//j. satırdan i'inci satırın temp ile çarpımını çıkar.

                if(i!=j){

                    matrix[j][k] -= matrix[i][k] * temp;

                    inverse[j][k] -= inverse[i][k] * temp;

                }

            }

        }

    }

    printf("******************************\n");

    printf("INVERSE MATRIX:\n");

    for(i = 0; i < n; i++){

        for(j = 0; j< n; j++){

            printf("%lf\t", inverse[i][j]);

        }

        printf("\n");

    }


    return 0;

}
```



C:\Users\canku\Desktop\c\nxnmatrisintersi.exe

```
Matrix'in satir ve sutun sayisi: 3
matrix[0][0]: 5
matrix[0][1]: 2
matrix[0][2]: -4
matrix[1][0]: 1
matrix[1][1]: 4
matrix[1][2]: 2
matrix[2][0]: 2
matrix[2][1]: 3
matrix[2][2]: 6
MATRIX:
5.000000        2.000000        -4.000000
1.000000        4.000000        2.000000
2.000000        3.000000        6.000000
******************************
INVERSE MATRIX:
0.169811        -0.226415       0.188679
-0.018868       0.358491        -0.132075
-0.047170       -0.103774       0.169811
```

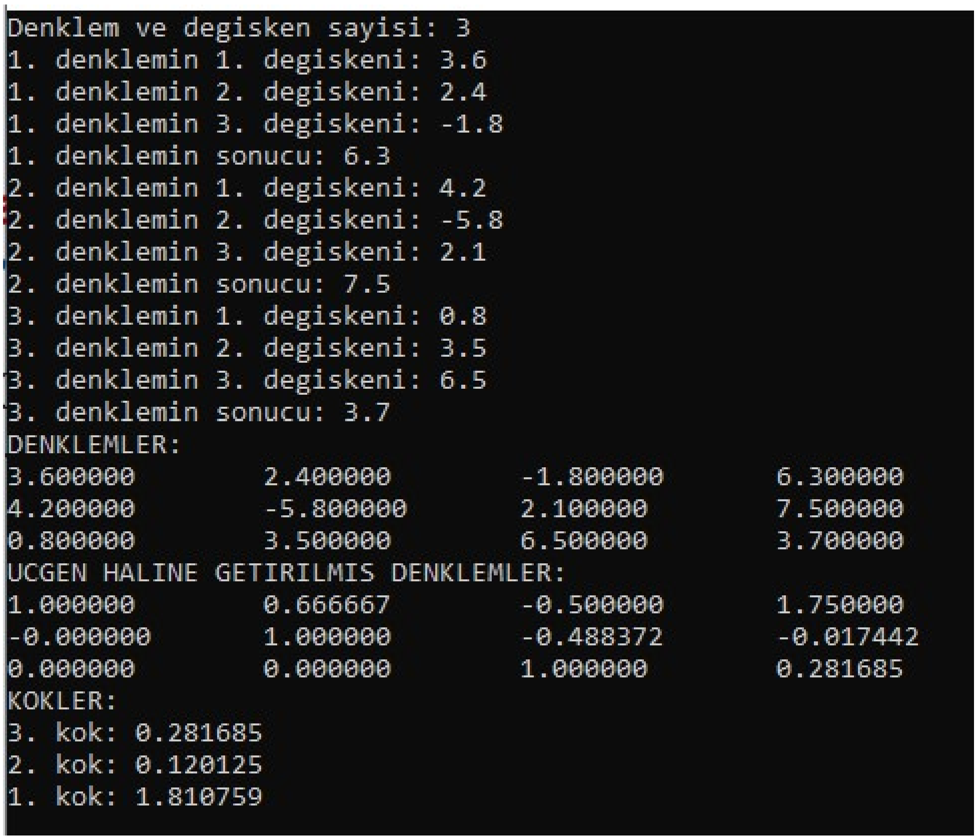
Gauss Eleminasyon

```
#include<stdio.h>



int main(){


        double matrix[20][21];

        double kokler[20];

        double temp,sigma;

        int n;

        int i,j,k;


        printf("Denklem ve degisken sayisi: ");

        scanf("%d", &n);




        for(i = 0; i < n;i++){

                for(j = 0; j < n; j++){
```

```
				printf("%d. denklemin %d. degiskeni: ", i+1,j+1);
				scanf("%lf", &matrix[i][j]);
			}
			printf("%d. denklemin sonucu: ",i+1);
			scanf("%lf", &matrix[i][j]);
		}
		printf("DENKLEMLER:\n");
		for(i = 0; i < n; i++){
			for(j = 0; j<=n; j++){
				printf("%lf\t", matrix[i][j]);
			}
			printf("\n");
		}

		for(i = n-1; i > n; i--){//Matrix[0][0], 0 olduğunda da çalışması için
										//satırları ilk sütunlarının büyüklüklerine göre
sıralama
			if(matrix[i-1][0] < matrix[i][0]){
				for(j = 0; j < n; j++){
					temp = matrix[i][j];
					matrix[i][j] = matrix[i-1][j];
					matrix[i-1][j] = temp;
				}
			}
		}

		for(i = 0; i < n; i++){
			if(matrix[i][i] == 0){
				printf("Diyagonelde 0 var hatali calisacak.");//Determinantı 0?
			}
			temp = matrix[i][i];
```

```
        for(j = 0; j <= n; j++){// i. satırı matrix[i][i] ye böl.
            matrix[i][j] /= temp;
        }
        for(j = i+1; j < n; j++){//i. sütunda i. satırın altını sıfırlamak için
                                //j. satırın satırın i. elemanını temp
olarak al
            temp = matrix[j][i];
            for(k = 0; k <= n; k++){//j. satırdan i'inci satırın temp ile çarpımını çıkar.

                matrix[j][k] -= matrix[i][k] * temp;
            }
        }
    }
    printf("UCGEN HALINE GETIRILMIS DENKLEMLER:\n");
    for(i = 0; i < n; i++){
        for(j = 0; j<=n; j++){
            printf("%lf\t", matrix[i][j]);
        }
        printf("\n");
    }

    for(i = n-1; i >= 0; i--){
        sigma = 0;
        for(j = i+1; j < n; j++){
            sigma += matrix[i][j]*kokler[j];
        }
        kokler[i] = (1/matrix[i][i])*(matrix[i][n] - sigma);
    }

    printf("KOKLER:\n");
    for(i = n-1; i >= 0; i--){
        printf("%d. kok: %lf\n",i+1, kokler[i]);
```

```
	}
	return 0;
}
```



```
Denklem ve degisken sayisi: 3
1. denklemin 1. degiskeni: 3.6
1. denklemin 2. degiskeni: 2.4
1. denklemin 3. degiskeni: -1.8
1. denklemin sonucu: 6.3
2. denklemin 1. degiskeni: 4.2
2. denklemin 2. degiskeni: -5.8
2. denklemin 3. degiskeni: 2.1
2. denklemin sonucu: 7.5
3. denklemin 1. degiskeni: 0.8
3. denklemin 2. degiskeni: 3.5
3. denklemin 3. degiskeni: 6.5
3. denklemin sonucu: 3.7
DENKLEMLER:
3.600000        2.400000        -1.800000       6.300000
4.200000        -5.800000       2.100000        7.500000
0.800000        3.500000        6.500000        3.700000
UCGEN HALINE GETIRILMIS DENKLEMLER:
1.000000        0.666667        -0.500000       1.750000
-0.000000       1.000000        -0.488372       -0.017442
0.000000        0.000000        1.000000        0.281685
KOKLER:
3. kok: 0.281685
2. kok: 0.120125
1. kok: 1.810759
```

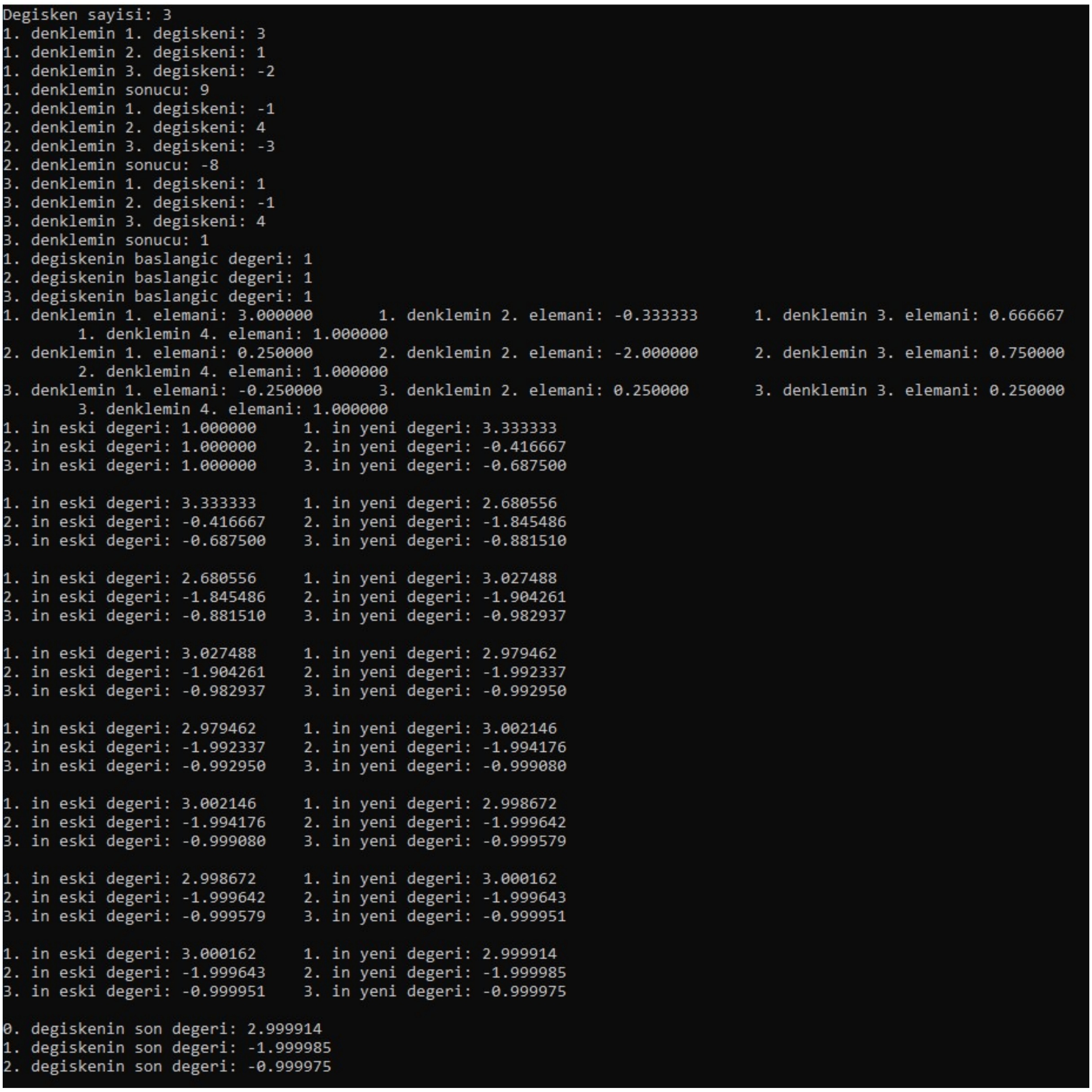
Gauss-Seidel

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

void denklemduzenleme(int n, double denklem[10][11]){//x. denklemden x. elemanı çekerek yalnız bırakma

	int x,j;
	double temp;

	for(x = 0; x < n; x++){
		temp = denklem[x][x]*-1;
		denklem[x][x] = denklem[x][n]*-1;
		denklem[x][n] = temp;
```

```
        for(j = 0; j <= n; j++){
            denklem[x][j] /= temp;
        }
    }


}
void baslangicdegerleriniyazdirma(int n,double baslangicdegerleri[]){


    int i;


    for(i = 0; i < n; i++){
        printf("%d. degisken: %lf:\n", i+1, baslangicdegerleri[i]);
    }
}


void yenidegerhesaplama(int x, int n, double denklem[10][11],double baslangicdegerleri[]){


    double result = 0;
    int i;


    printf("%d. in eski degeri: %lf\t", x+1, baslangicdegerleri[x]);


    for(i = 0; i < x; i++){
        result += baslangicdegerleri[i]*denklem[x][i];
    }
    result += denklem[x][i];
    for(i = x+1; i < n; i++){
        result += baslangicdegerleri[i]*denklem[x][i];
    }
```

```
		baslangicdegerleri[x] = result;
		printf("%d. in yeni degeri: %lf\n", x+1, baslangicdegerleri[x]);
}

void deger_esitleme(double baslangicdegerleri[], double eskidegerler[], int n){
		int i;
		for(i = 0; i < n; i++){
			eskidegerler[i] = baslangicdegerleri[i];
		}
}



void gauss_seidel(int n, double denklem[10][11],double baslangicdegerleri[],double eskidegerler[]){

		int i;
		int x=0;
		int flag = 0;
		double e = 0.001;

		deger_esitleme(baslangicdegerleri,eskidegerler,n);



		/*
		for(i = 0; i < n; i++){//İlk degiskenin degeri baslangic degerine esit oldugunda cikmasin diye ilk tur while'in disinda
			yenidegerhesaplama(i,n,denklem,baslangicdegerleri);
			eskidegerler[i] = baslangicdegerleri[i];
		}
		baslangicdegerleriniyazdirma(n,baslangicdegerleri);*/ //artik gerek yok
```

```
    while(flag == 0){
        for(i = 0; i < n; i++){
            yenidegerhesaplama(i,n,denklem,baslangicdegerleri);
        }
        baslangicdegerleriniyazdirma(n,baslangicdegerleri);
        i = 0;
        while(fabs(baslangicdegerleri[i]-eskidegerler[i]) < e && i < n){
            i++;
        }
        if(i == n) flag = 1;
        deger_esitleme(baslangicdegerleri,eskidegerler,n);

    }
}




int main(){

    int n,i,j;
    double denklem[10][11];
    double baslangicdegerleri[10];
    double eskidegerler[10];
    printf("Degisken sayisi: ");
    scanf("%d", &n);

    for(i = 0;i < n; i++){
        for(j = 0; j < n; j++){
            printf("%d. denklemin %d. degiskeni: ", i+1,j+1);
            scanf("%lf", &denklem[i][j]);
        }
        printf("%d. denklemin sonucu: ", i+1);
        scanf("%lf", &denklem[i][j]);
    }
    for(i = 0; i <n; i++){
        printf("%d. degiskenin baslangic degeri: ", i+1);
        scanf("%lf", &baslangicdegerleri[i]);
    }
    denklemduzenleme(n,denklem);


    for(i = 0; i < n; i++){
        for(j = 0; j <= n; j++){
            printf("%d. denklemin %d. elemani: %lf\t", i+1,j+1, denklem[i][j]);
        }
        printf("\n");
    }

    gauss_seidel(n,denklem,baslangicdegerleri,eskidegerler);
    for(i = 0; i < n; i++){
        printf("%d. degiskenin son degeri: %lf\n",i, baslangicdegerleri[i]);
    }

    return 0;
}
```

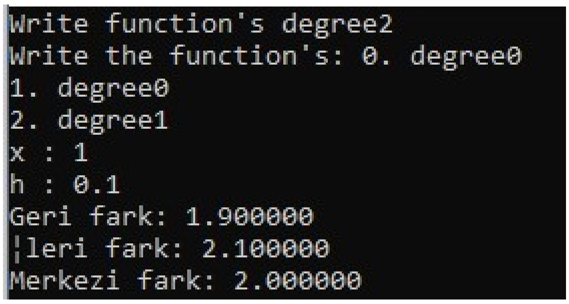


```
Degisken sayisi: 3
1. denklemin 1. degiskeni: 3
1. denklemin 2. degiskeni: 1
1. denklemin 3. degiskeni: -2
1. denklemin sonucu: 9
2. denklemin 1. degiskeni: -1
2. denklemin 2. degiskeni: 4
2. denklemin 3. degiskeni: -3
2. denklemin sonucu: -8
3. denklemin 1. degiskeni: 1
3. denklemin 2. degiskeni: -1
3. denklemin 3. degiskeni: 4
3. denklemin sonucu: 1
1. degiskenin baslangic degeri: 1
2. degiskenin baslangic degeri: 1
3. degiskenin baslangic degeri: 1
1. denklemin 1. elemani: 3.000000       1. denklemin 2. elemani: -0.333333      1. denklemin 3. elemani: 0.666667
        1. denklemin 4. elemani: 1.000000
2. denklemin 1. elemani: 0.250000       2. denklemin 2. elemani: -2.000000      2. denklemin 3. elemani: 0.750000
        2. denklemin 4. elemani: 1.000000
3. denklemin 1. elemani: -0.250000      3. denklemin 2. elemani: 0.250000       3. denklemin 3. elemani: 0.250000
        3. denklemin 4. elemani: 1.000000
1. in eski degeri: 1.000000     1. in yeni degeri: 3.333333
2. in eski degeri: 1.000000     2. in yeni degeri: -0.416667
3. in eski degeri: 1.000000     3. in yeni degeri: -0.687500

1. in eski degeri: 3.333333     1. in yeni degeri: 2.680556
2. in eski degeri: -0.416667    2. in yeni degeri: -1.845486
3. in eski degeri: -0.687500    3. in yeni degeri: -0.881510

1. in eski degeri: 2.680556     1. in yeni degeri: 3.027488
2. in eski degeri: -1.845486    2. in yeni degeri: -1.904261
3. in eski degeri: -0.881510    3. in yeni degeri: -0.982937

1. in eski degeri: 3.027488     1. in yeni degeri: 2.979462
2. in eski degeri: -1.904261    2. in yeni degeri: -1.992337
3. in eski degeri: -0.982937    3. in yeni degeri: -0.992950

1. in eski degeri: 2.979462     1. in yeni degeri: 3.002146
2. in eski degeri: -1.992337    2. in yeni degeri: -1.994176
3. in eski degeri: -0.992950    3. in yeni degeri: -0.999080

1. in eski degeri: 3.002146     1. in yeni degeri: 2.998672
2. in eski degeri: -1.994176    2. in yeni degeri: -1.999642
3. in eski degeri: -0.999080    3. in yeni degeri: -0.999579

1. in eski degeri: 2.998672     1. in yeni degeri: 3.000162
2. in eski degeri: -1.999642    2. in yeni degeri: -1.999643
3. in eski degeri: -0.999579    3. in yeni degeri: -0.999951

1. in eski degeri: 3.000162     1. in yeni degeri: 2.999914
2. in eski degeri: -1.999643    2. in yeni degeri: -1.999985
3. in eski degeri: -0.999951    3. in yeni degeri: -0.999975

0. degiskenin son degeri: 2.999914
1. degiskenin son degeri: -1.999985
2. degiskenin son degeri: -0.999975
```

Sayısal Türev

```
#include<stdio.h>




double f_result(double x,double f[],int degree){


        double result=0;

        int i,j;

        double tempfunction[degree];
        for(i=0;i<=degree;i++){
                tempfunction[i]=f[i];
        }

        for(i = 0; i <= degree; i++){
                for(j = 0; j < i; j++){
                        tempfunction[i] = tempfunction[i] * x;
                }
                result = result + tempfunction[i];
        }
        return result;
}

double gerifark(double x,double h,double f[], int degree){

        double turev;

        turev = (f_result(x,f,degree)-f_result(x-h,f,degree))/h;

        return turev;
}
double ilerifark(double x,double h,double f[], int degree){

        double turev;

        turev = (f_result(x+h,f,degree)-f_result(x,f,degree))/h;

        return turev;
}
double merkezifark(double x,double h,double f[], int degree){
        double turev;

        turev = (f_result(x+h,f,degree)-f_result(x-h,f,degree))/(2*h);

        return turev;
}
int main(){

        double x,h;
        int degree;
        int i;
        double function[10];


        printf("Write function's degree");

        scanf("%d", &degree);
        printf("Write the function's: ");
        for(i = 0; i <= degree; i++){
                printf("%d. degree", i);
                scanf("%lf", &function[i]);
        }
        printf("x : ");
        scanf("%lf", &x);
        printf("h : ");
        scanf("%lf", &h);

        printf("Geri fark: %lf\n", gerifark(x,h,function,degree));
        printf("İleri fark: %lf\n", ilerifark(x,h,function,degree));
        printf("Merkezi fark: %lf", merkezifark(x,h,function,degree));
```

```
    return 0;

}
```

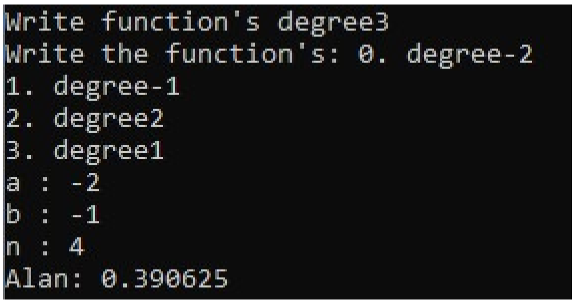


```
Write function's degree2
Write the function's: 0. degree0
1. degree0
2. degree1
x : 1
h : 0.1
Geri fark: 1.900000
¦leri fark: 2.100000
Merkezi fark: 2.000000
```

Trapez

```
#include<stdio.h>

#include<math.h>


double f_result(double x,double f[],int degree){


    double result=0;

    int i,j;

    double tempfunction[degree];


    for(i=0;i<=degree;i++){

        tempfunction[i]=f[i];

    }


    for(i = 0; i <= degree; i++){

        for(j = 0; j < i; j++){

            tempfunction[i] = tempfunction[i] * x;

        }

        result = result + tempfunction[i];

    }

    return result;
```

```
}

double trapez(double a, double b, int n, double f[],int degree){

    double result;
    double i;
    double h = (b-a)/n;

    result = (fabs((f_result(a,f,degree))) + fabs(f_result(b,f,degree)))/2;

    for(i = a+h; i <= b-h; i += h){
        result += fabs(f_result(i,f,degree));
    }
    result *= fabs(h);

    return result;
}

int main(){

    double a,b;
    int degree;
    int i;
    int n;
    double function[10];


    printf("Write function's degree");

    scanf("%d", &degree);
    printf("Write the function's: ");
    for(i = 0; i <= degree; i++){
        printf("%d. degree", i);
        scanf("%lf", &function[i]);
    }
    printf("a : ");
    scanf("%lf", &a);
    printf("b : ");
    scanf("%lf", &b);
    printf("n : ");
    scanf("%d", &n);

    printf("Alan: %lf", trapez(a,b,n,function,degree));


    return 0;
}
```

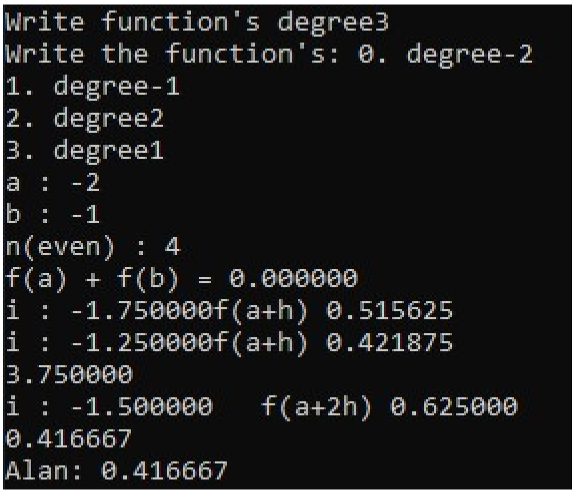


```
Write function's degree3
Write the function's: 0. degree-2
1. degree-1
2. degree2
3. degree1
a : -2
b : -1
n : 4
Alan: 0.390625
```

Simpson Yöntemi(1/3)

```
#include<stdio.h>



double f_result(double x,double f[],int degree){

    double result=0;
    int i,j;
    double tempfunction[degree];
    for(i=0;i<=degree;i++){
        tempfunction[i]=f[i];
    }

    for(i = 0; i <= degree; i++){
        for(j = 0; j < i; j++){
            tempfunction[i] = tempfunction[i] * x;
        }
        result = result + tempfunction[i];
    }
    return result;
}

double simpson(double a, double b, int n, double f[],int degree){

    double h = (b-a)/n;
    double result;
    double i,temp;

    temp = f_result(a,f,degree);

        if(temp < 0){
            temp *= -1;
        }

    result = temp;

    temp = f_result(b,f,degree);
    if(temp < 0){
            temp *= -1;
        }

result += temp;

printf("f(a) + f(b) = %lf\n", result);
for(i = a+h; i <= b-h; i += 2*h){

        temp = f_result(i,f,degree);

        if(temp < 0){
                temp *= -1;
        }
        printf("i : %lf", i);
        printf("f(a+h) %lf\n", temp);


        result += 4*temp;
}
printf("%lf\n", result);
for(i = a+2*h; i <= b-2*h; i+= 2*h){

        temp = f_result(i,f,degree);

        if(temp < 0){
                temp *= -1;
        }
        printf("i : %lf\t", i);
        printf("f(a+2h) %lf\n", temp);


        result += 2*temp;
```

```
	}

	result *= h/3;
	printf("%lf\n", result);

	return result;
}

int main(){

	double a,b;
	int degree;
	int i;
	int n;
	double function[10];


	printf("Write function's degree");

	scanf("%d", &degree);
	printf("Write the function's: ");
	for(i = 0; i <= degree; i++){
		printf("%d. degree", i);
		scanf("%lf", &function[i]);
	}
	printf("a : ");
	scanf("%lf", &a);
	printf("b : ");
	scanf("%lf", &b);
	printf("n(even) : ");
	scanf("%d", &n);
	while(n%2 == 1){
		printf("n must be even number, new n: ");
		scanf("%d", &n);
	}




	printf("Alan: %lf", simpson(a,b,n,function,degree));

	return 0;
}
```

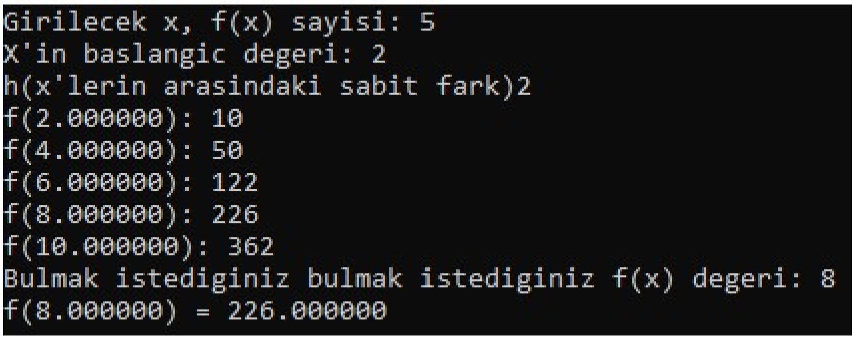


```
Write function's degree3
Write the function's: 0. degree-2
1. degree-1
2. degree2
3. degree1
a : -2
b : -1
n(even) : 4
f(a) + f(b) = 0.000000
i : -1.750000f(a+h) 0.515625
i : -1.250000f(a+h) 0.421875
3.750000
i : -1.500000   f(a+2h) 0.625000
0.416667
Alan: 0.416667
```

Gregory Newton Enterpolasyonu

```
#include<stdio.h>


double faktoriyel(int n){
	int f = 1;
	if(n == 0){
		return 1;
	}
	int i;
	for(i = 2; i <= n; i++){
		f *= i;
	}
	return f;
}

double k(double xi, double x0, double h,int i){
	if(i == 0) return 1;
	double k = (xi-x0)/h;
	double temp = k;
	int j;
	for(j = 1; j < i; j++){
		k *= (temp-j);
	}
	return k;
}

double gregorynewton(double fonksiyon[10][10],int n,double h,double x_baslangic,double bulunacak_x){

	int i,j;

	for(i = 1; i < n; i++){
		for(j = 0; j < n-i; j++){
			fonksiyon[j][i+1] = fonksiyon[j+1][i] - fonksiyon[j][i];
		}
	}

	double gregorynewton=0;
	for(i = 0; i < n; i++){
		gregorynewton += fonksiyon[0][i+1]*k(bulunacak_x,x_baslangic,h,i)/faktoriyel(i);
	}
	return gregorynewton;
```

```
}



int main(){

        int i,n;
        double x_baslangic,h;
        double bulunacak_x;
        double fonksiyon[10][10];

        printf("Girilecek x, f(x) sayisi: ");
        scanf("%d", &n);

        printf("X'in baslangic degeri: ");
        scanf("%lf", &x_baslangic);

        printf("h(x'lerin arasindaki sabit fark)");
        scanf("%lf", &h);

        for(i = 0; i < n; i++){
                fonksiyon[i][0] = x_baslangic+i*h;
                printf("f(%lf): ", x_baslangic+i*h);
                scanf("%lf", &fonksiyon[i][1]);
        }

        printf("Bulmak istediginiz bulmak istediginiz f(x) degeri: ");
        scanf("%lf", &bulunacak_x);

        printf("f(%lf) = %lf", bulunacak_x, gregorynewton(fonksiyon,n,h,x_baslangic,bulunacak_x));
        return 0;
}
```



```
Girilecek x, f(x) sayisi: 5
X'in baslangic degeri: 2
h(x'lerin arasindaki sabit fark)2
f(2.000000): 10
f(4.000000): 50
f(6.000000): 122
f(8.000000): 226
f(10.000000): 362
Bulmak istediginiz bulmak istediginiz f(x) degeri: 8
f(8.000000) = 226.000000
```