YILDIZ TEKNİK ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

SAYISAL ANALİZ

DÖNEM PROJE RAPORU

Yusuf Mert ÇELİKARSLAN

19011042

l1119042@std.yildiz.edu.tr

Eğitmen : Prof. Dr. Banu DİRİ

İçindekiler:

1. Bisection(sunumda gösterildi)…….....3
    a. Ekran çıktısı…….6
2. Regula Falsi…….…...7
3. Newton-Raphson……....9
4. Matrisin İnversi……12
5. Gauss Eleminasyon……..16
6. Gauss Seidal……19
7. Sayısal Türev……23
8. Simpson……....25
9. Trapez……...27
10. Gregory Newton Enterpolasyonu(sunumda gösterildi) …….....29
    a. Ekran çıktısı……33

Bisection Metodu

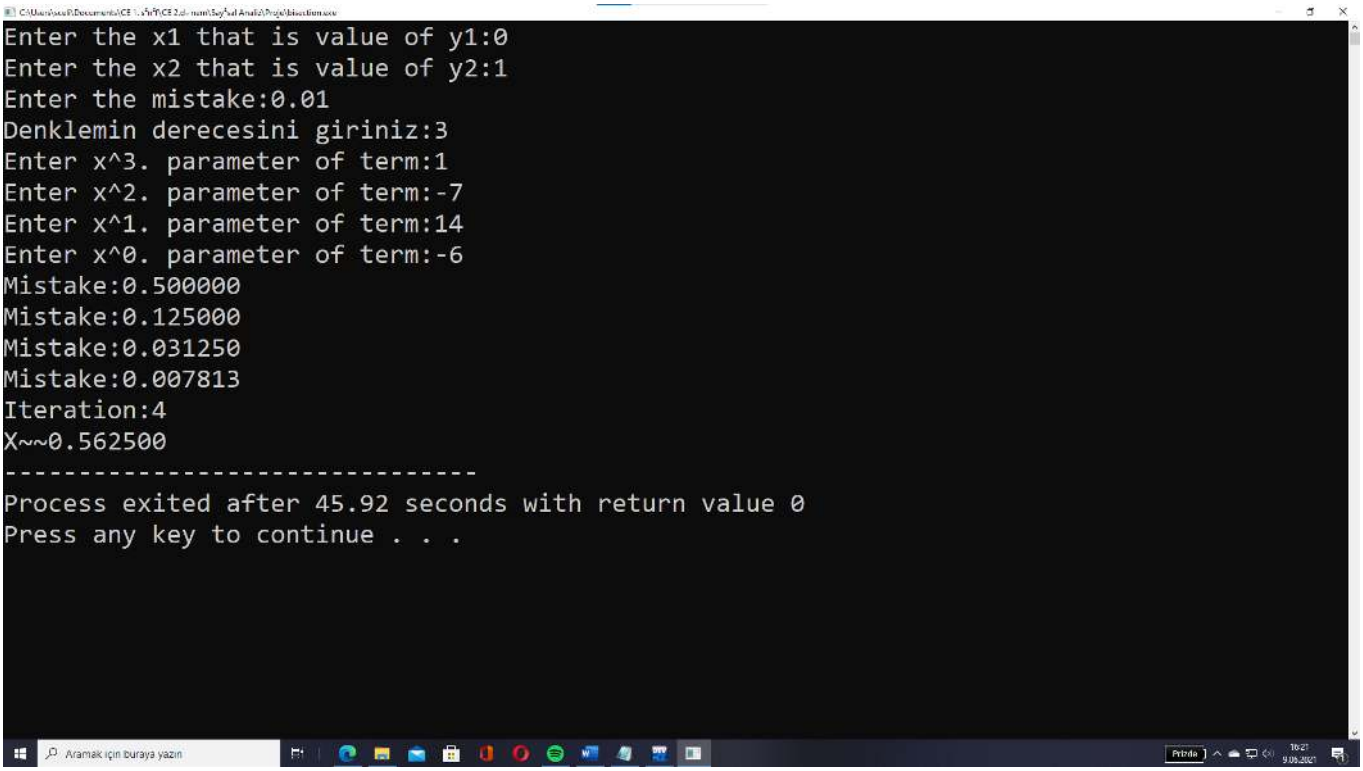
```
#include<stdio.h>
#include<math.h>

float bisection(float ,float ,float );
float equat(float ,int [],int );
int main(){
	float x1,x2,kok,mistake;
	printf("Enter the x1 that is value of y1:");
	scanf("%f",&x1);
	printf("Enter the x2 that is value of y2:");
	scanf("%f",&x2);
	printf("Enter the mistake:");
	scanf("%f",&mistake);
	kok=bisection(x1,x2,mistake);
	printf("X~~%f",kok);

	return 0;
}

float bisection(float a,float b,float k){
	float Hata=k+1,fa,fb,fc,c;
	int derece,kat[100],i,iteration=0;
	printf("Denklemin derecesini giriniz:");
	scanf("%d",&derece);
```

```
for(i=derece;i>=0;i--){
	printf("Enter x^%d. parameter of term:",i);
	scanf("%d",&kat[i]);
}
fa=equat(a,kat,derece);
fb=equat(b,kat,derece);
if((fa*fb)>0){
	printf("You entered wrong index!");
	return -1;
}
else if(fa==0){
	return a;
}
else if(fb==0){
	return b;
}
while((fa*fb)<0 && Hata>k){
	iteration++;
	Hata=fabs(a-b)/pow(2,iteration);
	//Hata=fabs(b-a);
	printf("Mistake:%f\n",Hata);
	c=(a+b)/2;
	fc=equat(c,kat,derece);
	if(fa*fc<0){
		b=c;
		fb=fc;
	}
```

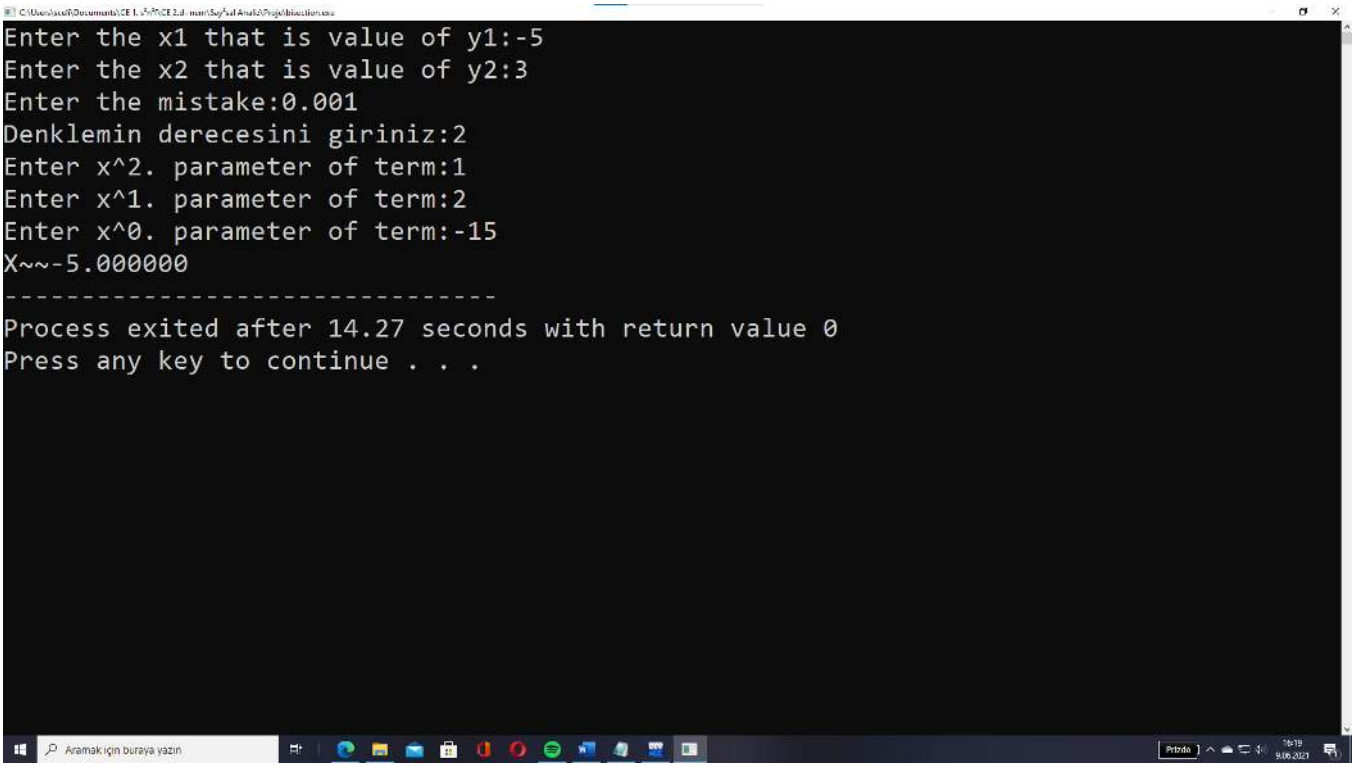
```
            else if(fa*fc>0){
                a=c;
                fa=fc;
            }
            else{
                printf("Iteration:%d\n",iteration);
                return c;
            }
        }
        printf("Iteration:%d\n",iteration);
        if(Hata<=k){
            return c;
        }
}
float equat(float x,int kat[],int derece){
        float denklem=0;
        int i;
        for(i=derece;i>=0;i--){
            denklem=denklem+kat[i]*pow(x,i);
        }
        return denklem;
}
```

Bisection Ekran Çıktısı

$x^3-7x^2+14x-6=0$



$x^2+2x-15=0$



Regula-Falsi Metodu

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

float regula_falsi(float a,float b,float k);
float equat(float ,int [],int );
int main(){
	float x1,x2,kok,mistake;
	printf("Enter the x1 that is value of y1:");
	scanf("%f",&x1);
	printf("Enter the x2 that is value of y2:");
	scanf("%f",&x2);
	printf("Enter the mistake:");
	scanf("%f",&mistake);
	kok=regula_falsi(x1,x2,mistake);
	printf("X~~%f",kok);

	return 0;
}

float regula_falsi(float a,float b,float k){
	float Hata=k+1,fa,fb,fc,c;
	int derece,kat[100],i,iteration=0;
	printf("Denklemin derecesini giriniz:");
```

```
scanf("%d",&derece);
for(i=derece;i>=0;i--){
        printf("Enter x^%d. parameter of term:",i);
        scanf("%d",&kat[i]);
}
fa=equat(a,kat,derece);
fb=equat(b,kat,derece);
if((fa*fb)>=0){
        printf("You entered wrong index!---");
        return -1;
}
while((fa*fb)<0 && Hata>k){
        iteration++;
        Hata=fabs(b-a)/pow(2,iteration);
        printf("Mistake:%f\n",Hata);
        c=(b*fa-a*fb)/(fa-fb);
        fc=equat(c,kat,derece);
        if(fa*fc<0){
                b=c;
                fb=fc;
        }
        else if(fa*fc>0){
                a=c;
                fa=fc;
        }
        else{
                printf("Iteration:%d\n",iteration);
```

```
                return c;
            }
        }
        if(Hata<=k){
            return c;
        }
}
float equat(float x,int kat[],int derece){
        float denklem=0;
        int i;
        for(i=derece;i>=0;i--){
            denklem=denklem+kat[i]*pow(x,i);
        }
        return denklem;
}
```

Newton-Raphson Metodu

```
#include<stdio.h>
#include<math.h>
float newton_raphson(float a,float k);
float derivative(float x,int kat[],int derece);
float equat(float x,int kat[],int derece);
int main(){
        float x1,x2,kok,initiate,mistake;
        char join;
```

```
	printf("Enter the x1 that is value of y1:");
	scanf("%f",&x1);
	printf("Enter the x2 that is value of y2:");
	scanf("%f",&x2);
	printf("Do you want to assign an initial value:(Y/N)");
	scanf(" %c",&join);



	printf("Enter the mistake:");
	scanf("%f",&mistake);
	if(join=='Y' || join=='y'){
		printf("Initial Value:");
		scanf("%f",&initiate);
		kok=newton_raphson(initiate,mistake);
	}
	else if(x1<x2)
		kok=newton_raphson(x1,mistake);
	else
		kok=newton_raphson(x2,mistake);
	printf("X~~%f",kok);
	return 0;
}
float newton_raphson(float a,float k){
	float Hata=k+1,fa,ga,c;
	int derece,kat[100],i;
	printf("Denklemin derecesini giriniz:");
	scanf("%d",&derece);
```

```
for(i=derece;i>=0;i--){
	printf("Enter x^%d. parameter of term:",i);
	scanf("%d",&kat[i]);
}
fa=equat(a,kat,derece);
ga=derivative(a,kat,derece);
if(ga==0){
	a+=k;
	ga=derivative(a,kat,derece);
}
printf("denklem=%f --- turev=%f\n",fa,ga);
while(Hata>k){
	c=a-(fa/ga);
	printf("C:%f\n",c);
	Hata=fabs(c-a);//x1-x0
	printf("Mistake:%f\n",Hata);
	a=c; //a=x1
	fa=equat(a,kat,derece);
	ga=derivative(a,kat,derece);
	if(ga==0){
		a+=k;
		ga=derivative(a,kat,derece);
	}
}
//printf("Iteration:%d\n",iteration);
if(Hata<=k){
	return c;
```

```
        }
}
float equat(float x,int kat[],int derece){
        float denklem=0,deriv=0;
        int i;
        for(i=derece;i>=0;i--){
                denklem=denklem+kat[i]*pow(x,i);
        }
        return denklem;
}
float derivative(float x,int kat[],int derece){
        float deriv=0;
        int i;
        for(i=derece;i>0;i--){
                deriv=deriv+(kat[i]*i*pow(x,i-1));
        }
        return deriv;
}
```

Matris Inversi

```
#include<stdio.h>
#define MAX 100
int main(){
        float matrix[MAX][MAX];
        int row,column,i,j;
```

```
        printf("ROW LENGTH:");
        scanf("%d",&row);
        printf("COLUMN LENGTH:");
        scanf("%d",&column);
        scan(matrix,row,column);
        for(i=0;i<row;i++){
                for(j=column;j<column*2;j++){
                        if(i==(j-column)){
                                matrix[i][j]=1;
                        }
                        else{
                                matrix[i][j]=0;
                        }
                }
        }
        print(matrix,row,column*2);
        eliminasyon(matrix,row,column*2);
        printf("Invers A matrix:\n");
        for(i=0;i<row;i++){
                for(j=column;j<column*2;j++){
                        printf("%-0.3f ",matrix[i][j]);
                }
                printf("\n");
        }
}

void eliminasyon(float a[][MAX],int r,int c){
```

```
int i,j,p,k;
float hold,tmp;
i=0;
while(i<r){//1.satır
	p=i+1;
	while(a[i][i]==0 && p<r){
		for(k=0;k<c;k++){
			tmp=a[p][k];
			a[p][k]=a[i][k];
			a[i][k]=tmp;
		}
		p++;
	}
	hold=a[i][i];
	for(k=i;k<c;k++){
		a[i][k]/=hold;
	}
	for(j=0;j<r;j++){
		hold=a[j][i];
		if(j!=i){
			for(p=i;p<c;p++){
				a[j][p]=a[j][p]-(hold*a[i][p]);
			}
		}
		printf("\n");
	}
	print(a,r,c);
```

```
            i++;
        }
}

void print(float a[][MAX],int r,int c){
        int i,j;
        for(i=0;i<r;i++){
            for(j=0;j<c;j++){
                printf("%0.3f ",a[i][j]);
            }
            printf("\n");
        }
        printf("\n");
}
void scan(float a[][MAX],int r,int c){
        int i,j;
        for(i=0;i<r;i++){
            for(j=0;j<c;j++){
                printf("[%d][%d]=",i,j);
                scanf("%f",&a[i][j]);
            }
        }
}
```

Gauss Eleminasyon

```
#include<stdio.h>
#define MAX 100
int main(){
	float extcoefmatrix[MAX][MAX],variables[MAX]={0};
	int row,column,i;
	printf("Denklem sayisini giriniz:");
	scanf("%d",&row);
	printf("Degisken sayisini giriniz:");
	scanf("%d",&column);
	column++;
	scan(extcoefmatrix,row,column);
	print(extcoefmatrix,row,column);
	eliminasyon(extcoefmatrix,row,column);
	print(extcoefmatrix,row,column);
	variable(extcoefmatrix,variables,row,column);
	for(i=0;i<column-1;i++){
		printf("x%d= %0.3f\t",i+1,variables[i]);
	}
}

void eliminasyon(float a[][MAX],int r,int c){
	int i,j,p,k;
	float hold,tmp;
```

```
        i=0;
        while(i<r){//1.satır
                p=i+1;
                while(a[i][i]==0 && p<r){
                        for(k=0;k<c;k++){
                                tmp=a[p][k];
                                a[p][k]=a[i][k];
                                a[i][k]=tmp;
                        }
                        p++;
                }
                hold=a[i][i];
                for(k=i;k<c;k++){
                        a[i][k]/=hold;
                }
                for(j=i+1;j<r;j++){
                        hold=a[j][i];
                        for(p=i;p<c;p++){
                                a[j][p]=a[j][p]-(hold*a[i][p]);
                        }
                }
                printf("\n\n\n");
                print(a,r,c);
                i++;
        }
}
```

```
void variable(float a[][MAX],float arr[],int r,int c){
	int i,j;
	arr[c-2]=a[r-1][c-1];
	for(i=r-2;i>=0;i--){
		for(j=c-2;j>i;j--){
			arr[i]=arr[i]+(arr[j]*a[i][j]);
		}
		arr[i]=a[i][c-1]-arr[i];
	}
}

void print(float a[][MAX],int r,int c){
	int i,j;
	for(i=0;i<r;i++){
		for(j=0;j<c;j++){
			printf("%0.2f ",a[i][j]);
		}
		printf("\n");
	}
	printf("\n");
}

void scan(float a[][MAX],int r,int c){
	int i,j;
	for(i=0;i<r;i++){
		printf("%d. denklemi giriniz:\n",(i+1));
		for(j=0;j<c;j++){
```

```
                if(j==c-1){
                        printf("C%d'i giriniz:",i+1);
                }
                else
                        printf("x%d:",j+1);
                scanf("%f",&a[i][j]);
        }
    }
}
```

Gauss Seidal

```
#include<stdio.h>
#include<math.h>
#define MAX 100
int mistake(float hata[],float E,int n);
void max_diagonal(float a[][MAX],int m,int n,float c[]);
void find_root(float hata[],float a[][MAX],float x[],float c[],int m,int n,float E);
int main(){
    float variables[MAX],katsayi[MAX][MAX],result[MAX],delta[MAX];
    float hata;
    int row,column,i,j;
    printf("Degisken sayisini giriniz:");
    scanf("%d",&column);
    printf("Denklem sayisini giriniz:");
```

```
scanf("%d",&row);
for(i=0;i<row;i++){
	printf("%d.denklemin katsayilarini giriniz:",i+1);
	for(j=0;j<column;j++){
		scanf("%f",&katsayi[i][j]);
	}
	printf("%d.denklemin sabit terimini giriniz:",i+1);
	scanf("%f",&result[i]);
}
printf("Degiskenlerin ilk degerlerini giriniz:");
for(i=0;i<column;i++){
	scanf("%f",&variables[i]);
}
printf("Hatayi giriniz:");
scanf("%f",&hata);
for(i=0;i<column;i++){
	delta[i]=hata+1000;
}
max_diagonal(katsayi,row,column,result);
for(i=0;i<column;i++){
	printf("    x%d\t\t",i+1);
}
printf("\n");
do{
	find_root(delta,katsayi,variables,result,row,column,hata);
}while(mistake(delta,hata,column));
for(i=0;i<column;i++){
```

```
            printf("x%d = %0.2f\t",i+1,variables[i]);
        }
}

int mistake(float hata[],float E,int n){
        int i=0;
        while(hata[i]<=E && i<n){
            i++;
        }
        if(i==n){
            return 0;
        }
        return 1;
}

void find_root(float hata[],float a[][MAX],float x[],float c[],int m,int n,float
E){
        int i,j;
        float sum,hold;
        for(i=0;i<m;i++){
            sum=0;
            if(hata[i]>E){
            for(j=0;j<n;j++){
                if(i!=j){
                    sum+=(a[i][j]*x[j]);
                }
            }
```

```
            hold=x[i];
            x[i]=1.0*(c[i]-sum)/a[i][i];
            hata[i]=fabs(x[i]-hold);
            printf("%0.3f ",x[i]);
            printf("%0.3f\t",hata[i]);
            }
        }
        printf("\n");
}

void max_diagonal(float a[][MAX],int m,int n,float c[]){
        float prod=1,tmp;
        int max,i,j,k;
        for(i=0;i<n;i++){
            max=0;
            for(j=1;j<m;j++){
                if(fabs(a[j][i])>fabs(a[max][i])){
                    max=j;
                }
            }
            tmp=c[i];
            c[i]=c[max];
            c[max]=tmp;
            for(k=0;k<n;k++){
                tmp=a[i][k];
                a[i][k]=a[max][k];
                a[max][k]=tmp;
```

```
            }
        }
}
```

Sayısal Türev

```
#include<stdio.h>
#include<math.h>
#define MAX 100
float turev(float x,float h,float kat[],int mode,int mode2,int mertebe);
float equation(float t,float kat[],int derece);
int main(){
        float kat[MAX]={0},value,h,deriv;
        int mode,mertebe,i,mode2;
        do{
                printf("Turev hesaplama metodunu seciniz:\n(1)Ileri fark\n(2)Geri fark\n(3)Merkezi fark\nMode:");
                scanf("%d",&mode);
        }while(mode<1 || mode>3);
        printf("Deger ve hassasiyet miktarini giriniz:");
```

```
	scanf("%f%f",&value,&h);
	deriv=turev(value,h,kat,mode,mode2,mertebe);
	printf("TUREV:%0.4f",deriv);
}

float turev(float x,float h,float kat[],int mode,int mode2,int mertebe){
	float f1,fxi,fx0,fx1,fx2,fx3;
	int derece,i;
	printf("Derece:");
	scanf("%d",&derece);
	for(i=derece;i>=0;i--){
		printf("x^%d. terimin katsayisini giriniz:",i);
		scanf("%f",&kat[i]);
	}
	switch(mode){
		case 1://ileri fark
			fxi=equation(x+h,kat,derece);
			fx0=equation(x,kat,derece);
			f1=(fxi-fx0)/h*1.0;
			break;
		case 2:// geri fark
			fxi=equation(x-h,kat,derece);
			fx0=equation(x,kat,derece);
			f1=(fx0-fxi)/h*1.0;
			break;
		case 3:// merkezi fark
			fxi=equation(x+h,kat,derece);
```

```
                fx0=equation(x-h,kat,derece);
                f1=(fxi-fx0)/(h*2.0);
                break;
        default:
                printf("Mode is not correct.");
    }
    return f1;
}

float equation(float t,float kat[],int derece){
    int i;
    float denk=0;
    for(i=derece;i>=0;i--){
        denk+=(pow((double)t,(double)i)*kat[i]);
    }
    return denk;
}
```

Simpson Metodu

```
#include<stdio.h>
#include<math.h>
#define MAX 100
float integral(float kat[],float xn,float x0,int n,int ,int);
float equation(float t,float kat[],int derece,int );
int main(){
```

```
	float kat[MAX],a,b;
	int N,derece,i;
	printf("Limit degerlerini kucukten buyuge giriniz:");
	scanf("%f%f",&a,&b);
	printf("N:");
	scanf("%d",&N);
	printf("Derece:");
	scanf("%d",&derece);
	for(i=0;i<=derece;i++){
		printf("x^%d. terimin katsayisini giriniz:",i);
		scanf("%f",&kat[i]);
	}
	printf("Integral|%0.f to %0.f =
%0.2f\n",a,b,integral(kat,b,a,N,1,derece));
	printf("Area|%0.f to %0.f = %0.2f",a,b,integral(kat,b,a,N,-1,derece));
}

float integral(float kat[],float xn,float x0,int n,int sign,int derece){
	float h,S=0;
	int i;
	h=fabs(1.0*(xn-x0)/n);
	S=equation(x0,kat,derece,sign)+equation(xn,kat,derece,sign);
	for(i=1;i<=n-1;i+=2){
		//printf("%f\n",S);
		S=S+4*equation(x0+i*h,kat,derece,sign);
	}
	for(i=2;i<=n-2;i+=2){
```

```
            //printf("%f\n",S);
            S=S+2*equation(x0+i*h,kat,derece,sign);
        }
        S=(h/3)*S;
        return S;
}
float equation(float t,float kat[],int derece,int sign){
        int i;
        float denk=0;
        for(i=0;i<=derece;i++){
            denk+=(pow((double)t,(double)i)*kat[i]);
        }
        if(denk<0){
            denk=denk*sign;
        }
        return denk;
}
```

Trapez Metodu

```
#include<stdio.h>
#include<math.h>
#define MAX 100

float integral(float kat[],float xn,float x0,int n,int ,int);
float equation(float t,float kat[],int derece,int );
int main(){
```

```
	float kat[MAX],a,b;
	int N,derece,i;
	printf("Limit degerlerini giriniz:");
	scanf("%f%f",&a,&b);
	printf("N:");
	scanf("%d",&N);
	printf("Derece:");
	scanf("%d",&derece);
	for(i=0;i<=derece;i++){
		printf("x^%d. terimin katsayisini giriniz:",i);
		scanf("%f",&kat[i]);
	}
    printf("Integral|%0.f to %0.f = %0.2f\n",a,b,integral(kat,b,a,N,1,derece));
    printf("Area|%0.f to %0.f = %0.2f",a,b,integral(kat,b,a,N,-1,derece));
}



float integral(float kat[],float xn,float x0,int n,int sign,int derece){
	float h,S=0;
	int i;
	h=fabs(1.0*(xn-x0)/n);
	S=(equation(x0,kat,derece,sign)+equation(xn,kat,derece,sign))/2;
	for(i=1;i<n;i++){
		//printf("%f\n",S);
		S=S+equation(x0+i*h,kat,derece,sign);
	}
```

```
        S=h*S;
        return S;
}


float equation(float t,float kat[],int derece,int sign){
        int i;
        float denk=0;
        for(i=0;i<=derece;i++){
                denk+=(pow((double)t,(double)i)*kat[i]);
        }
        if(denk<0){
                denk=denk*sign;
        }
        return denk;
}
```

Gregory-Newton Entepolasyonu

```
#include<stdio.h>
#define MAX 100
void gregory_newton(float a[][2],int size,float x,int);
float pow(float a,int b);
int fakt(int f);
int ileri_fark(float a[][2],int size);
```

```
int esitmi(float dizi[],int a,int n);
float kok(float x,int kere,float a[][2]);
int main(){
	float deger[MAX][2];
	int i,j,cozum,exit;
	float value;
	printf("Gireceginiz ornek sayisini yaziniz:");
	scanf("%d",&cozum);
	for(i=0;i<cozum;i++){
			printf("%d. index degerini giriniz:",i+1);
			scanf("%f",&deger[i][0]);
			printf("%d. fonksiyon degerini giriniz:",i+1);
			scanf("%f",&deger[i][1]);
	}
	i=0;
	do{
		printf("f(x) fonksiyonu icin x degerini giriniz:");
		scanf("%f",&value);
		gregory_newton(deger,cozum,value,i);
		printf("(1)Tekrar deger girmek icin 1'e basiniz\n(2)Cikmak icin
0'a basiniz\nEnter:");
		scanf("%d",&exit);
		i++;
	}while(exit);

	return 0;
}
```

```
void gregory_newton(float a[][2],int size,float x,int sayi){
	float h=a[1][0]-a[0][0];
	static int sinir;
	int i;
	float denk=a[0][1];
	if(sayi==0){
		sinir=ileri_fark(a,size);
	}
	for(i=1;i<=sinir;i++){
		denk+=((a[i][1]*kok(x,i,a))/(pow(h,i)*1.0*fakt(i)));
	}
	printf("f(%.2f) = %f\n",x,denk);
}

float kok(float x,int kere,float a[][2]){
	int i;
	float kuad=1;
	for(i=0;i<kere;i++){
		kuad*=(x-a[i][0]);
	}
	return kuad;
}

float pow(float a,int b){
	int k;
	if(b==0){
		return 1;
```

```
        }
        else{
                k=pow(a,b/2);
                if(b%2==1)
                        return a*k*k;
                else
                        return k*k;
        }
}
int fakt(int f){
        if(f==1 || f==0)
                return 1;
        else
                return f*fakt(f-1);
}
int ileri_fark(float a[][2],int size){
        int i,j=0;
        float gecici[size];
        do{
                for(i=j;i<size;i++){
                        gecici[i+1]=a[i+1][1]-a[i][1];
                }
                printf("%d. ileri fark:\n",j+1);
                for(i=j+1;i<size;i++){
                        a[i][1]=gecici[i];
                        printf("%.f\n",a[i][1]);
                }
```

```
            j++;
        }while(esitmi(gecici,j,size));
        return j;
}
int esitmi(float dizi[],int a,int n){
        int i=a;
        while(i<n-1 && dizi[i]==dizi[i+1]){
            i++;
        }
        if(i==(n-1))
            return 0;
        else
            return 1;
}
```

Gregory-Newton Entepolasyonu Ekran Çıktısı