

## PROJE KONU BAŞLIKLARI:

(Tamamlanan yöntemler tiklenmiştir)

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 ✓

## SUNUMU YAPILAN YÖNTEMLERİN KODU VE EKRAN GÖRÜNTÜSÜ:

```
Gregory-Newton Enterpolasyon Yontemi
(Degisken Donusumsuz)

N,x0,h degerlerini giriniz:

N (x degerleri adedi max 20): 5
x0 (baslangic x degeri): 0
h (x degerleri arasindaki sabit fark): 1

f(x) degerlerini giriniz:

f(0) = 0
f(1) = 4
f(2) = 10
f(3) = 18
f(4) = 28

X (F(x)'in bulunmasini istediginiz x degeri): 1.5

Enterpolasyon ile bulunan F(1.5) = 6.8
```

```
Simpson 1/3 Yontemi
Fonksiyonun derecesini giriniz(max 10): 2

(x uzeri 2)'in katsayisini giriniz: 1
(x uzeri 1)'in katsayisini giriniz: 2
(x uzeri 0)'in katsayisini giriniz: 3

Integral araligini [a,b] giriniz:
a(alt): 1
b(ust): 2

N degerini(aralik sayisi) giriniz(Cift olmalı):8

Hesaplanan yaklasik integral= 8.333334
```

```
void enterpolasyon(int x0,int h,int x_deg[20],float fonk_deg[20],int n){
    float x,ileriFark[20][20];
    int i=0,j;
    while(i >= 0){
        if(i == 0){ //1.dereceden ileri farki bulmak icin
            for(j=0;j<n-1;j++){
                ileriFark[i][j]=fonk_deg[j+1]-fonk_deg[j];
            }
        }
        else{ //Diger ileri farklari bulmak icin
            for(j=0;j<n-1-i;j++){
                ileriFark[i][j]=ileriFark[i-1][j+1]-ileriFark[i-1][j];
            }
        }
        if(ileriFark[i][0] == ileriFark[i][1]) i*=-1;
        else i++;
    }
    i*=-1; //i degerini geri almak icin
    printf("\nX (F(x)'in bulunmasini istediginiz x degeri): ");
    scanf("%f",&x);
    float temp,toplam=fonk_deg[0];
    int k,faktoriyel;
    for(j=1;j<=i+1;j++){
        faktoriyel=1;
        for(k=1;k<=j;k++) faktoriyel *= k;
        temp=1;
        for(k=0;k<j;k++) temp *= x-x_deg[k];
        toplam += (temp/us(h,j))*(ileriFark[j-1][0]/faktoriyel);
    }
    printf("\n\tEnterpolasyon ile bulunan F(%.1f) = %.1f\n",x,toplam);
}
```

```
//Simpson 1/3 Metodu:
float h = (aralik_bit-aralik_bas) / n,toplam_tek=0,toplam_cift=0;
for(i=1;i<=n-1; i += 2){
    toplam_tek += fonk(aralik_bas+(i*h),derece,katsayilar);
    // teklerin f(x0+i*h) toplami
}
for(i=2;i<=n-2; i += 2){
    toplam_cift += fonk(aralik_bas+(i*h),derece,katsayilar);
    // ciftlerin f(x0+i*h) toplami
}
float S =
h/3 * (fonk(aralik_bas,derece,katsayilar)+fonk(aralik_bit,derece,katsayilar)+4*toplam_tek+2*toplam_cift);
printf("\n\tHesaplanan yaklasik integral= %f",S);
```

Not: Kullanıcıdan gerekli değerleri aldığımız main() fonksiyonları dahil değildir.

BÜTÜN YÖNTEMLERİN KODLARI:

Not: Kullanıcıdan değer aldığımız main() fonksiyonları dahil değildir.

Açıklama: Bu üç C fonksiyonunu kendim yazıp yöntemlerin içinde ortak olarak kullandım.

f(x) fonk(x,derece,katsayilar) = f(x)      f'(x) fonk\_t(x,derece,katsayilar) = f'(x)      x^i us(x,i) = x^i

<pre>void bisection(float hata,float aralik_bas, float aralik_bit, float derece, float katsayilar[10]){  int iterasyon=1; float hata_temp,aralik_temp;  hata_temp = (aralik_bit-aralik_bas) / us(2,iterasyon); while(hata_temp &gt; hata){      aralik_temp = (aralik_bit+aralik_bas) / 2.0;      if ( ( fonk(aralik_temp,derece,katsayilar)*fonk(aralik_bas,derece,katsayilar) ) &lt; 0){         aralik_bit = aralik_temp;         hata_temp = (aralik_bit-aralik_bas) / us(2,iterasyon);     }      else if ( ( fonk(aralik_temp,derece,katsayilar)*fonk(aralik_bit,derece,katsayilar) ) &lt; 0){         aralik_bas = aralik_temp;         hata_temp = (aralik_bit-aralik_bas) / us(2,iterasyon);     }      else if( fonk(aralik_temp,derece,katsayilar) == 0 ) {         hata_temp = hata;     }      iterasyon++; } printf("Bulunan kok = %f", aralik_temp); }</pre>	<p><b>Bisection Yöntemi</b></p>	<pre>float us(float taban, int ust){ /*<b>Us alma fonksiyonu</b>*/ int i; float temp=1; for(i=0;i&lt;ust;i++){     temp *= taban; } return temp; }  /*<b>Fonksiyon fonksiyonu:</b>*/ float fonk(float x , int derece , float katsayilar[10]){ int i,j; float sonuc=0;  for(i=0;i&lt;=derece;i++){     sonuc += us(x,i)*katsayilar[i]; } return sonuc; }  /*<b>Turev fonksiyonu fonksiyonu:</b>*/ float fonk_t(float x , int derece , float katsayilar_t[10]){ int i,j; float sonuc=0;  for(i=0;i&lt;=derece-1;i++){     sonuc += us(x,i)*katsayilar_t[i]; } return sonuc; }</pre>	<pre>void matrix_tersini_al(float matrix[10][10],float birim_matrix[10][10],int n){ float temp; int i,j,k;  for(i=0;i&lt;n;i++){     temp=matrix[i][i]; //m[i][i] ilerdeki islemlerde degisecegi icin      for(j=0;j&lt;n;j++){ //1. adim: i. satiri m[i][i]'ye bol         matrix[i][j] /= temp;         birim_matrix[i][j] /= temp;     }      for(j=0;j&lt;n;j++){ //2.adim: i. olmayan satirlardan (m[satir][i]*i.satir) cikar         if(j!=i){ //i. olmayan satirlari bulmak icin             temp=matrix[j][i]; //m[satir][i] ilerdeki islemlerde degisecegi icin             for(k=0;k&lt;n;k++){                 matrix[j][k] -= temp*matrix[i][k];                 birim_matrix[j][k] -= temp*birim_matrix[i][k];             }         }     }      printf("\nTers Matris:\n\n");     matrix_yazdir(birim_matrix,n); }</pre>	<p><b>Gauss Eleminasyon ile Matrisin Tersini Alma</b></p>
<pre>void regula_falsi(float hata,float aralik_bas, float aralik_bit, float derece, float katsayilar[10]){  int iterasyon=1; float hata_temp,aralik_temp;  hata_temp = (aralik_bit-aralik_bas) / us(2,iterasyon); while(hata_temp &gt; hata){      aralik_temp =     (( aralik_bit * fonk(aralik_bas,derece,katsayilar) ) - ( aralik_bas * fonk(aralik_bit,derece,katsayilar) ))     /     (fonk(aralik_bas,derece,katsayilar) - fonk(aralik_bit,derece,katsayilar));      if ( ( fonk(aralik_temp,derece,katsayilar)*fonk(aralik_bas,derece,katsayilar) ) &lt; 0){         aralik_bit = aralik_temp;         hata_temp = (aralik_bit-aralik_bas) / us(2,iterasyon);     }      else if ( ( fonk(aralik_temp,derece,katsayilar)*fonk(aralik_bit,derece,katsayilar) ) &lt; 0){         aralik_bas = aralik_temp;         hata_temp = (aralik_bit-aralik_bas) / us(2,iterasyon);     }      else if( fonk(aralik_temp,derece,katsayilar) == 0 ) {         hata_temp = hata;     }      iterasyon++; } printf("Bulunan kok = %f", aralik_temp); }</pre>	<p><b>Regula-Falsi Yöntemi</b></p>	<pre>void newton_raphson(float x[2],int derece,float katsayilar[10],float hata,float katsayilar_t[10]){  float hata_temp=hata+1; //donguye girmesini saglamak icin while(hata_temp &gt; hata){      x[1] = x[0] - ( fonk(x[0],derece,katsayilar) / fonk_t(x[0],derece,katsayilar_t) );      if(fonk(x[1],derece,katsayilar) == 0) hata_temp = hata; // f(x1)=0 ise x1 koktur.      hata_temp=x[1]-x[0];     if(hata_temp &lt; 0) hata_temp *= -1; //mutlak deger      if( hata_temp &gt; 128 ) hata_temp = -1; // lraksiyor ise devam etmemesi icin.      x[0] = x[1]; }  if( hata_temp == -1 ) printf("\n\t Cozum lraksar."); else printf("\n\tBulunan kok= %f",x[1]); }</pre>	<p><b>Newton_Raphson Yöntemi</b></p>	
		<p><b>Sayısal Türev</b></p>		
		<pre>//Sayisal turev:  printf("\n\tGeri farklar ile turev = %f\n", ( fonk(x,derece,katsayilar)-fonk(x-h,derece,katsayilar) )/h);  printf("\t\tleri farklar ile turev = %f\n", ( fonk(x+h,derece,katsayilar)-fonk(x,derece,katsayilar) )/h);  printf("\t\tMerkezi farklar ile turev = %f\n", ( fonk(x+h,derece,katsayilar)-fonk(x-h,derece,katsayilar) )/(2*h));</pre>		

```
void gauss_eleminasyon(float katsayilar[10][10],float sonuclar[10],int n){
    int i,j,k;
    float temp,x[10];

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

        temp=katsayilar[i][i]; //katsayilar[i][i] degisecegi icin
        for(j=0;j<n;j++){ //i. satiri katsayilar[i][i]'ye bol
            katsayilar[i][j] /= temp;
        }
        sonuclar[i] /= temp;

        for(j=0;j<n;j++){ //i.satirdan sonraki satirlari isleme sokar
            if(j>i){ //i'den sonraki satirlar

                temp=katsayilar[j][i]; //katsayilar[j][i] ilerde degisecegi icin

                //1-) Eldeki satiri katsayilar[satir][i]'ye bol
                for(k=0;k<n;k++){
                    katsayilar[j][k] /= temp;
                }
                sonuclar[j] /= temp;

                //2-)Eldeki satirdan i. satiri cikar
                for(k=0;k<n;k++){
                    katsayilar[j][k] -= katsayilar[i][k];
                }
                sonuclar[j] -= sonuclar[i];

                //3-) Eldeki satiri katsayilar[satir][i] ile carp
                for(k=0;k<n;k++){
                    katsayilar[j][k] *= temp;
                }
                sonuclar[j] *= temp;
            }
        }
    }
}
```

## Gauss Eleminasyon Yöntemi

```
printf("\n\tBulunan kokler:\n");

for(k=0;k<n;k++){ //Matrixten kokleri bulmak icin. (PDF'teki formulun koda gecirilmis hali)
    i=n-k-1;
    temp=0;
    for(j=i+1;j<n;j++){
        temp+=katsayilar[i][j]*x[j];

        x[i]= (1/katsayilar[i][i])*(sonuclar[i]-temp);
    }

    for(i=0;i<n;i++){ //Kokleri yazdirir
        printf("\n\tx%d = %.3f",i+1,x[i]);
    }
}
```

## Trapez Yöntemi

```
//Trapez Metodu:
float h = (aralik_bit-aralik_bas) / n , toplam=0; //aralik miktarı

for(i=1;i<=n-1;i++){
    toplam += fonk(aralik_bas+(i*h),derece,katsayilar); // k=1 -> n-1 f(x0+k*h) toplami
}

float S =
h*(((fonk(aralik_bas,derece,katsayilar) + fonk(aralik_bit,derece,katsayilar))/2) + toplam);
//      f(x0)                        f(xn)                        E

printf("\n\tHesaplanan yaklasik integral= %f",S);
```

```
void gauss_seidel(int n,float x[10],float katsayilar[10][10],float sonuclar[10],float hata){
    float temp,x_temp,hata_temp;
    int i,m=0,hata_count=0;
```

## Gauss-Seidel Yöntemi

```
while(hata_count != n){
    temp=0;
    for(i=0;i<n;i++){
        temp += katsayilar[m][i]*x[i]; //Aranan x dahil hepsini toplar
    }
    temp -= katsayilar[m][m]*x[m]; //Aranan x'i cikarir

    x_temp = x[m]; //hata hesaplamak icin
    x[m] = ( sonuclar[m]-(temp) ) / katsayilar[m][m]; //genel formül

    hata_temp = x[m] - x_temp;
    if(hata_temp < 0) hata_temp *= -1; //mutlak deger
    if( hata_temp > 128 ) hata_temp = -1; // lraksiyor ise devam etmemesi icin.

    //Butun degiskenlerin hatasi istenilen duzeye gelinceye kadar devam etmesi icin:
    if(hata_temp <= hata) hata_count++;

    m++;
    if(m == n){ //tam tur
        m = 0; //x'ler arasinda tur atmasi icin
        if(hata_count != n) hata_count=0; //hata sayacini sifirlamak icin
        // ^^ sonsuz while engellemek icin
    }

    if( hata_temp == -1 ) printf("\n\t Cozum lraksar.");
    else{
        printf("\n\tBulunan kokler:\n");
        for(i=0;i<n;i++) printf("x%d = %f\n",i+1,x[i]);
    }
}
```

```
//Simpson 1/3 Metodu:
float h = (aralik_bit-aralik_bas) / n,toplam_tek=0,toplam_cift=0;

for(i=1;i<=n-1; i += 2){
    toplam_tek += fonk(aralik_bas+(i*h),derece,katsayilar);
    //      teklerin f(x0+i*h) toplami
}

for(i=2;i<=n-2; i += 2){
    toplam_cift += fonk(aralik_bas+(i*h),derece,katsayilar);
    //      ciftlerin f(x0+i*h) toplami
}

float S =
h/3 * (fonk(aralik_bas,derece,katsayilar)+fonk(aralik_bit,derece,katsayilar)+4*toplam_tek+2*toplam_cift);

printf("\n\tHesaplanan yaklasik integral= %f",S);
```

## Simpson 1/3 Metodu

```
void enterpolasyon(int x0,int h,int x_deg[20],float fonk_deg[20],int n){
    float x,ileriFark[20][20];
    int i=0,j;

    while(i >= 0){
        if(i == 0){ //1.dereceden ileri farki bulmak icin
            for(j=0;j<n-1;j++){
                ileriFark[i][j]=fonk_deg[j+1]-fonk_deg[j];
            }
        }
        else{ //Diger ileri farklari bulmak icin
            for(j=0;j<n-1-i;j++){
                ileriFark[i][j]=ileriFark[i-1][j+1]-ileriFark[i-1][j];
            }

            if(ileriFark[i][0] == ileriFark[i][1]) i*=-1;
            else i++;
        }
        i*=-1; //i degerini geri almak için

        printf("\nX (F(x)'in bulunmasini istediginiz x degeri): ");
        scanf("%f",&x);

        float temp,toplam=fonk_deg[0];
        int k,faktoriyel;
        for(j=1;j<=i+1;j++){
            faktoriyel=1;
            for(k=1;k<=j;k++)    faktoriyel *= k;

            temp=1;
            for(k=0;k<j;k++)    temp *= x-x_deg[k];

            toplam += (temp/us(h,j))*(ileriFark[j-1][0]/faktoriyel);
        }
        printf("\n\tEnterpolasyon ile bulunan F(%.1f) = %.1f\n",x,toplam);
    }
}
```

## Gregory-Newton Enterpolasyon Yontemi