- **Dersin Adı** :SAYISAL ANALİZ
- **Grup** :1
- **Eğitmen** :PROF. DR. BANU DİRİ
- **Sunum tarihi** :09.06.2021
- **Teslim tarihi** :15.06.2021
- **Öğrenci Numarası** :20011023
- **Öğrenci Adı ve Soyadı** :MEHMET ALPEREN ÖLÇER

- **Sunulan Yöntemler** :Simpson ve Gregory Newton Enterpolasyon Yöntemleri
- **Yapılan Yöntem Sayısı** :10 + 2 (listede olmayanlar, grafik ve secant yöntemleri)

- **İmza** :

Sunulan Yöntemlerin Algoritmaları ve Ekran Görüntüleri

8 - Simpson Yöntemi

```
#include <stdio.h>
#define MAX 20

float us_al(float x, int y);
float denklem(float x, float carpanlar[MAX], int derece);
void girdi_alma(float *a, float *b, float *n);
float cozum(float a, float b, float n, float carpanlar[MAX], int derece);

int main()
{
   float carpanlar[MAX] = {0}, a, b, n;
   int derece, i;

   printf("Turevi alinacak denklemin derecesi : ");
   scanf("%d", &derece);
   for ( i = 0; i <= derece; i++)
   {
      printf("x uzeri %d nin carpanini giriniz : ", i);
      scanf("%f",  &carpanlar[i]);
   }
   printf("\n----------------------\n\n");
   girdi_alma(&a, &b, &n);
   printf("Denklemin %.2f  -  %.2f araligindaki integralinin alani : %f\n", a, b, cozum(a, b, n,
carpanlar, derece));

   return 0;
}

float us_al(float x, int y)
{
   if (y == 0)
      return 1;
   else
      return us_al(x, y-1)*x;
}

float denklem(float x, float carpanlar[MAX], int derece)
{
   int i;
   float res = 0;
   for ( i = 0; i <= derece; i++)
      res += carpanlar[i] * us_al(x, i);
   return res;
}

void girdi_alma(float *a, float *b, float *n)
{
```

```
    printf("Integral alinacak araligin baslangic degeri (a): ");
    scanf("%f", a);
    printf("Integral alinacak araligin bitis degeri (b) : ");
    scanf("%f", b);
    printf("n degerini giriniz (a-b arasinda kac parabol olacak) (n) : ");
    scanf("%f", n);
}

float cozum(float a, float b, float n, float carpanlar[MAX], int derece)
{
    float h = (b - a) / n;
    float sonuc = h/3.0;
    float toplam = (denklem(a, carpanlar, derece) + denklem(b, carpanlar, derece));
    float carpan = 4;
    while (a != b)
    {
        a += h;
        toplam += carpan*denklem(a, carpanlar, derece);
        if (carpan == 4)
            carpan = 2;
        else
            carpan = 4;
    }

    // alan pozitif olmali
    if (toplam < 0)
        toplam *= -1;

    return sonuc*toplam;
}
```

```
Integrali alinacak denklemin derecesi : 3
x uzeri 0 nin carpanini giriniz : -2
x uzeri 1 nin carpanini giriniz : -1
x uzeri 2 nin carpanini giriniz : 2
x uzeri 3 nin carpanini giriniz : 1

----------------------

Integral alinacak araligin baslangic degeri (a): -2
Integral alinacak araligin bitis degeri (b) : -1
n degerini giriniz (a-b arasinda kac parabol olacak) (n) : 4
Denklemin -2.00  -  -1.00 araligindaki integralinin alani : 0.416667
```

10-Degisken Donusumsuz Gregory Newton Enterpolasyonu

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

void cozum(float x[MAX], float y[MAX], float istenilen_x);
int elemanlar_ayni_mi(float dizi[MAX]);
float islem(float x[MAX], float delta_f, float istenilen_x, int count, float h);
float us_al(float x, int y);
int fact_al(int x);

int main()
{
   float x[MAX], y[MAX], temp, istenilen_x;
   int i=0;
   printf("\nBilinen x ve y degerlerini giriniz. Deger girmeyi bitimek icin x'e 99 giriniz.");
   printf("\nNOT : x ler arasi artim miktari (h) sabit olmalidir.");
   printf("\nx : ");
   scanf("%f", &x[i]);
   while (x[i] != 99)
   {
      printf("f(%f) : ", x[i]);
      scanf("%f", &y[i]);
      i ++;
      printf("\nx : ");
      scanf("%f", &x[i]);
   }
   printf("Girdiginiz degerler ile uydurulacak fonksiyonda bakilacak x degerini giriniz : ");
   scanf("%f", &istenilen_x);
   cozum(x, y, istenilen_x);
   return 0;
}

void cozum(float x[MAX], float y[MAX], float istenilen_x)
{
   float carpanlar[MAX], delta_f[MAX], h, result=y[0], delta_f_0[MAX];
   int i=0, count=1;
   h = x[1] - x[0];
   while (x[i] != 99)
   {
      delta_f[i] = y[i];
      i ++;
   }
   delta_f[i] = 99;
   delta_f_0[0] = delta_f[0];
   while (!elemanlar_ayni_mi(delta_f))
   {
      i=0;
      while (delta_f[i+1] != 99)
      {
         delta_f[i] = delta_f[i+1] - delta_f[i];
         i ++;
```

```
        }
        delta_f[i] = 99;
        i=0;
        while (delta_f[i] != 99)
        {
            printf("%f ", delta_f[i]);
            i ++;
        }
        printf("\n");
        delta_f_0[count] = delta_f[0];
        result += islem(x, delta_f_0[count], istenilen_x, count, h);
        count ++;
    }
    printf("F(%f) : %f\n", istenilen_x, result);
}

int elemanlar_ayni_mi(float dizi[MAX])
{
    int i=0;
    while (dizi[i] != 99 && dizi[0] == dizi[i])
        i ++;
    if (dizi[i] == 99)
        return 1;
    else
        return 0;
}

float islem(float x[MAX], float delta_f, float istenilen_x, int count, float h)
{
    float ust=1, alt;
    alt = fact_al(count)*us_al(h, count);
    ust *= delta_f;
    count --;
    for (; count >= 0; count--)
        ust *= istenilen_x-x[count];
    return ust/alt;
}

float us_al(float x, int y)
{
    if (y == 0)
        return 1;
    else
        return us_al(x, y-1)*x;
}

int fact_al(int x)
{
    if (x==1)
        return 1;
    return x*fact_al(x-1);
}
```

```
Bilinen x ve y degerlerini giriniz. Deger girmeyi bitimek icin x'e 99 giriniz.
NOT : x ler arasi artim miktari (h) sabit olmalidir.
x : 2
f(2.000000) : 10

x : 4
f(4.000000) : 50

x : 6
f(6.000000) : 122

x : 8
f(8.000000) : 226

x : 10
f(10.000000) : 362

x : 99
Girdiginiz degerler ile uydurulacak fonksiyonda bakilacak x degerini giriniz : 8
40.000000 72.000000 104.000000 136.000000
32.000000 32.000000 32.000000
F(8.000000) : 226.000000
```

NOT : Yapılan Yöntemler Kısmına Simpson ve Gregory Newton yöntemleri tekrardan konulmamıştır.

Yapılan Yöntemler

1-Bisection Yöntemi

```
// durma kosullari
// iterasyon sayisi
// gercek kok (P) var ise |P-xn| < hata
// gercek kok yok ise |x(n+1)-xn| < hata

#include <stdio.h>

float mutlak_deger(float x);
float us_al(float x, int y);
float denklem(float x, float carpanlar[15], int derece);
void iterasyon_solution(float x0, float x1, float iterasyon_sayisi, float carpanlar[15], int derece);
void gercek_kokle_solution(float x0, float x1, float istenilen_hata_miktari, float gercek_deger, float carpanlar[15], int derece);
void gercek_koksuz_solution(float x0, float x1, float istenilen_hata_miktari, float carpanlar[15], int derece);

int main()
{
   float carpanlar[15] = {0};
   int derece, i;
   printf("Girilecek denklemin derecesi : ");
   scanf("%d", &derece);
   for ( i = 0; i <= derece; i++)
   {
      printf("x uzeri %d nin carpanini giriniz : ", i);
      scanf("%f",  &carpanlar[i]);
   }

   // gerekli girdileri alma
   float istenilen, x0, x1, gercek_deger;
   printf("\nIstenilen hata miktari sinirini veya iterasyon sayisini giriniz (iterasyon sayisi 2 den buyuk olmalidir) : ");
   scanf("%f", &istenilen);
   printf("\nBakilacak x degerleri aralagi (alt sinir ve ust siniri aralarinda bosluk olacak sekilde giriniz.) : ");
   scanf("%f %f", &x0, &x1);

   // durma kosuluna gore fonksiyona gonderme
   if (istenilen < 3.0)
   {
      printf("\nGercek kokun x degerini giriniz (bilinmiyorsa 999 giriniz.): ");
      scanf("%f", &gercek_deger);
      if (gercek_deger == 999)
         gercek_koksuz_solution(x0, x1, istenilen, carpanlar, derece);
      else
```

```
            gercek_kokle_solution(x0, x1, istenilen, gercek_deger, carpanlar, derece);
    }
    else
        iterasyon_solution(x0, x1, istenilen, carpanlar, derece);

    return 0;
}

float mutlak_deger(float x)
{
    if (x < 0)
        x *= -1;
    return x;
}

float us_al(float x, int y)
{
    if (y == 0)
        return 1;
    else
        return us_al(x, y-1)*x;
}

float denklem(float x, float carpanlar[15], int derece)
{
    int i;
    float res = 0;
    for ( i = 0; i <= derece; i++)
        res += carpanlar[i] * us_al(x, i);
    return res;
}

void iterasyon_solution(float x0, float x1, float iterasyon_sayisi, float carpanlar[15], int derece)
{
    int i;
    float alt_deger, orta_deger, ust_deger, c;
    for ( i = 0; i < iterasyon_sayisi; i++)
    {
        c = (x0+x1) / 2;
        alt_deger = denklem(x0, carpanlar, derece);
        orta_deger = denklem(c, carpanlar, derece);
        ust_deger = denklem(x1, carpanlar, derece);

        if (orta_deger*ust_deger < 0)
            x0 = c;
        else if (orta_deger != c)
            x1 = c;

        printf("\n%d iterasyon sonrasinda bulunan sonuc : f(%f) = %f \n", i+1, c, orta_deger);
    }
}
```

```
void gercek_kokle_solution(float x0, float x1, float istenilen_hata_miktari, float gercek_deger, float carpanlar[15], int derece)
{
   int i=0;
   float alt_deger, orta_deger, ust_deger, c, hata_miktari=istenilen_hata_miktari+2;
   while (hata_miktari > istenilen_hata_miktari)
   {
      c = (x0+x1) / 2;
      alt_deger = denklem(x0, carpanlar, derece);
      orta_deger = denklem(c, carpanlar, derece);
      ust_deger = denklem(x1, carpanlar, derece);

      if (orta_deger*ust_deger < 0)
         x0 = c;
      else
         x1 = c;

      hata_miktari = mutlak_deger( c - gercek_deger );
      i++;
      printf("\n Hata miktari %f iken bulunan sonuc : f(%f) = %f \n", hata_miktari, c, orta_deger);
   }
}

void gercek_koksuz_solution(float x0, float x1, float istenilen_hata_miktari, float carpanlar[15], int derece)
{
   int i=0;
   float alt_deger, orta_deger, ust_deger, c, hata_miktari=istenilen_hata_miktari+2;
   while (hata_miktari > istenilen_hata_miktari)
   {
      c = (x0+x1) / 2;
      alt_deger = denklem(x0, carpanlar, derece);
      orta_deger = denklem(c, carpanlar, derece);
      ust_deger = denklem(x1, carpanlar, derece);

      if (orta_deger*ust_deger < 0)
      {
         x0 = c;
         hata_miktari = mutlak_deger( x1 - c );
      }
      else
      {
         x1 = c;
         hata_miktari = mutlak_deger( c - x0 );
      }
      i++;
      printf("\n Hata miktari %f iken bulunan sonuc : f(%f) = %f \n", hata_miktari, c, orta_deger);
   }
}
```

2 - Regula Falsi Yöntemi

```
// durma kosullari
// iterasyon sayisi
// gercek kok (P) var ise |P-xn| < hata
// gercek kok yok ise |x(n+1)-xn| < hata

#include <stdio.h>

float mutlak_deger(float x);
float us_al(float x, int y);
float denklem(float x, float carpanlar[15], int derece);
void iterasyon_solution(float x0, float x1, float iterasyon_sayisi, float carpanlar[15], int derece);
void gercek_kokle_solution(float x0, float x1, float istenilen_hata_miktari, float gercek_deger, float carpanlar[15], int derece);
void gercek_koksuz_solution(float x0, float x1, float istenilen_hata_miktari, float carpanlar[15], int derece);

int main()
{
   float carpanlar[15] = {0};
   int derece, i;
   printf("Girilecek denklemin derecesi : ");
   scanf("%d", &derece);
   for ( i = 0; i <= derece; i++)
   {
      printf("x uzeri %d nin carpanini giriniz : ", i);
      scanf("%f",  &carpanlar[i]);
   }

   // gerekli girdileri alma
   float istenilen, x0, x1, gercek_deger;
   printf("\nIstenilen hata miktari sinirini veya iterasyon sayisini giriniz (iterasyon sayisi 2 den buyuk olmalidir) : ");
   scanf("%f", &istenilen);
   printf("\nBakilacak x degerleri aralagi (alt sinir ve ust siniri aralarinda bosluk olacak sekilde giriniz.) : ");
   scanf("%f %f", &x0, &x1);

   // durma kosuluna gore fonksiyona gonderme
   if (istenilen < 3.0)
   {
      printf("\nGercek kokun x degerini giriniz (bilinmiyorsa 999 giriniz.): ");
      scanf("%f", &gercek_deger);
      if (gercek_deger == 999)
         gercek_koksuz_solution(x0, x1, istenilen, carpanlar, derece);
      else
         gercek_kokle_solution(x0, x1, istenilen, gercek_deger, carpanlar, derece);
   }
   else
      iterasyon_solution(x0, x1, istenilen, carpanlar, derece);
```

```
    return 0;
}

float mutlak_deger(float x)
{
    if (x < 0)
        x *= -1;
    return x;
}

float us_al(float x, int y)
{
    if (y == 0)
        return 1;
    else
        return us_al(x, y-1)*x;
}

float denklem(float x, float carpanlar[15], int derece)
{
    int i;
    float res = 0;
    for ( i = 0; i <= derece; i++)
        res += carpanlar[i] * us_al(x, i);
    return res;
}

void iterasyon_solution(float x0, float x1, float iterasyon_sayisi, float carpanlar[15], int derece)
{
    int i;
    float alt_deger, orta_deger, ust_deger, c;
    for ( i = 0; i < iterasyon_sayisi; i++)
    {
        alt_deger = denklem(x0, carpanlar, derece);
        ust_deger = denklem(x1, carpanlar, derece);

        c = ( x1*alt_deger - x0*ust_deger ) / ( alt_deger-ust_deger );
        orta_deger = denklem(c, carpanlar, derece);
        if (alt_deger*orta_deger < 0)
            x1 = c;
        else
            x0 = c;

        printf("\n%d iterasyon sonrasinda bulunan sonuc : f(%f) = %f \n", i+1, c, orta_deger);
    }
}

void gercek_kokle_solution(float x0, float x1, float istenilen_hata_miktari, float gercek_deger, float
carpanlar[15], int derece)
{
    float alt_deger, orta_deger, ust_deger, c, hata_miktari=istenilen_hata_miktari+2;
    while (hata_miktari > istenilen_hata_miktari)
```

```
    {
        alt_deger = denklem(x0, carpanlar, derece);
        ust_deger = denklem(x1, carpanlar, derece);

        c = ( x1*alt_deger - x0*ust_deger ) / ( alt_deger-ust_deger );
        orta_deger = denklem(c, carpanlar, derece);
        if (alt_deger*orta_deger < 0)
            x1 = c;
        else
            x0 = c;

        hata_miktari = mutlak_deger( c - gercek_deger );
        printf("\n Hata miktari %f iken bulunan sonuc : f(%f) = %f \n", hata_miktari, c, orta_deger);
    }
}

void gercek_koksuz_solution(float x0, float x1, float istenilen_hata_miktari, float carpanlar[15], int
derece)
{
    float alt_deger, orta_deger, ust_deger, c, hata_miktari=istenilen_hata_miktari+2;
    while (hata_miktari > istenilen_hata_miktari)
    {
        alt_deger = denklem(x0, carpanlar, derece);
        ust_deger = denklem(x1, carpanlar, derece);

        c = ( x1*alt_deger - x0*ust_deger ) / ( alt_deger-ust_deger );
        orta_deger = denklem(c, carpanlar, derece);
        if (alt_deger*orta_deger < 0)
        {
            hata_miktari = mutlak_deger( c - x1 );
            x1 = c;
        }
        else
        {
            hata_miktari = mutlak_deger( c - x0 );
            x0 = c;
        }

        printf("\n Hata miktari %f iken bulunan sonuc : f(%f) = %f \n", hata_miktari, c, orta_deger);
    }
}
```

3 – Newton RapshonYöntemi

```
// durma kosullari
// iterasyon sayisi
// gercek kok (P) var ise |P-xn| < hata
// gercek kok yok ise |x(n+1)-xn| < hata

#include <stdio.h>

float mutlak_deger(float x);
float us_al(float x, int y);
float denklem(float x, float carpanlar[15], int derece);
float denklemin_turevi(float x, float carpanlar2[15], int derece);
void iterasyon_solution(float x0, float iterasyon_sayisi, float carpanlar[15], float carpanlar2[15], int
derece);
void gercek_kokle_solution(float x0, float istenilen_hata_miktari, float gercek_deger, float
carpanlar[15], float carpanlar2[15], int derece);
void gercek_koksuz_solution(float x0, float istenilen_hata_miktari, float carpanlar[15], float
carpanlar2[15], int derece);

int main()
{
   float carpanlar[15] = {0}, carpanlar2[15] = {0};
   int derece, i;
   printf("Girilecek denklemin derecesi : ");
   scanf("%d", &derece);
   for ( i = 0; i <= derece; i++)
   {
      printf("x uzeri %d nin carpanini giriniz : ", i);
      scanf("%f",  &carpanlar[i]);
   }

   printf("\n Turev denklemini de giriniz. \n");
   for ( i = 0; i <= derece-1; i++)
   {
      printf("x uzeri %d nin carpanini giriniz : ", i);
      scanf("%f",  &carpanlar2[i]);
   }
   // gerekli girdileri alma
   float istenilen, x0, gercek_deger;
   printf("\nIstenilen hata miktari sinirini veya iterasyon sayisini giriniz (iterasyon sayisi 2 den
buyuk olmalidir) : ");
   scanf("%f", &istenilen);
   printf("\nBakilacak ilk x degerini giriniz : ");
   scanf("%f", &x0); // 0

   // durma kosuluna gore fonksiyona gonderme
   if (istenilen < 3.0)
   {
      printf("\nGercek kokun x degerini giriniz (bilinmiyorsa 999 giriniz.): ");
      scanf("%f", &gercek_deger); // 0.585786438
      if (gercek_deger == 999)
```

```
            gercek_koksuz_solution(x0, istenilen, carpanlar, carpanlar2, derece);
        else
            gercek_kokle_solution(x0, istenilen, gercek_deger, carpanlar, carpanlar2, derece);
    }
    else
        iterasyon_solution(x0, istenilen, carpanlar, carpanlar2, derece);

    return 0;
}

float mutlak_deger(float x)
{
    if (x < 0)
        x *= -1;
    return x;
}

float us_al(float x, int y)
{
    if (y == 0)
        return 1;
    else
        return us_al(x, y-1)*x;
}

float denklem(float x, float carpanlar[15], int derece)
{
    int i;
    float res = 0;
    for ( i = 0; i <= derece; i++)
        res += carpanlar[i] * us_al(x, i);
    return res;
}

float denklemin_turevi(float x, float carpanlar2[15], int derece)
{
    int i;
    float res = 0;
    for ( i = 0; i <= derece-1; i++)
        res += carpanlar2[i] * us_al(x, i);
    return res;
}

void iterasyon_solution(float x0, float iterasyon_sayisi, float carpanlar[15], float carpanlar2[15], int
derece)
{
    int i;
    float x1_degeri, x1;
    for ( i = 0; i < iterasyon_sayisi; i++)
    {
        x1 = x0 - (denklem(x0, carpanlar, derece) / denklemin_turevi(x0, carpanlar2, derece));
        x1_degeri = denklem(x1, carpanlar, derece);
```

```
        printf("\n%d iterasyon sonrasinda bulunan sonuc : f(%f) = %f \n", i+1, x1, x1_degeri);
        x0 = x1;
    }
}

void gercek_kokle_solution(float x0, float istenilen_hata_miktari, float gercek_deger, float
carpanlar[15], float carpanlar2[15], int derece)
{
    float x1_degeri, x1, hata_miktari=99;
    while ( hata_miktari > istenilen_hata_miktari )
    {
        x1 = x0 - (denklem(x0, carpanlar, derece) / denklemin_turevi(x0, carpanlar2, derece));
        x1_degeri = denklem(x1, carpanlar, derece);
        hata_miktari =  mutlak_deger( x1 - gercek_deger );
        printf("\n Hata miktari %f iken bulunan sonuc : f(%f) = %f \n", hata_miktari, x1, x1_degeri);

        x0 = x1;
    }
}

void gercek_koksuz_solution(float x0, float istenilen_hata_miktari, float carpanlar[15], float
carpanlar2[15], int derece)
{
    float x1_degeri, x1, hata_miktari=99;
    while ( hata_miktari > istenilen_hata_miktari )
    {
        x1 = x0 - (denklem(x0, carpanlar, derece) / denklemin_turevi(x0, carpanlar2, derece));
        x1_degeri = denklem(x1, carpanlar, derece);
        hata_miktari =  mutlak_deger(x1 - x0);
        printf("\n Hata miktari %f iken bulunan sonuc : f(%f) = %f \n", hata_miktari, x1, x1_degeri);

        x0 = x1;
    }
}
```

4 – NxN lik Matrisin Tersi

```
#include <stdio.h>
#include<time.h>
#include<stdlib.h>
#define MAX 20

void matris_yazdir(float matris[MAX][MAX], int boyut);
void matris_olustur(float matris[MAX][MAX], int *boyut);
void cozum(float matris[MAX][MAX], float ters_matris[MAX][MAX], int boyut);

int main()
{
   srand(time(NULL));
   float matris[MAX][MAX], ters_matris[MAX][MAX];
   int boyut;
   matris_olustur(matris, &boyut);
   printf("\nMatris\n------------------\n");
   matris_yazdir(matris, boyut);
   cozum(matris, ters_matris, boyut);
   printf("\nMatrisin tersi\n------------------\n");
   matris_yazdir(ters_matris, boyut);
   printf("\n");
   return 0;
}

void matris_yazdir(float matris[MAX][MAX], int boyut)
{
   int i, j;
   for ( i = 0; i < boyut; i++)
   {
      for ( j = 0; j < boyut; j++)
      {
         printf("%.2f ", matris[i][j]);
      }
      printf("\n");
   }
}

void matris_olustur(float matris[MAX][MAX], int *boyut)
{
   int i, j;
   do
   {
      printf("\nMatrisin boyutunu giriniz : ");
      scanf("%d", boyut);

   } while (*boyut<1);


   for ( i = 0; i < *boyut; i++)
   {
```

```
        for ( j = 0; j < *boyut; j++)
        {
            printf("\n%d. sutun %d. satirdaki elemani giriniz : ", (i+1), (j+1));
            scanf("%f", &matris[i][j]);
            // matris[i][j] = rand() % 20;
        }
    }
}

void cozum(float matris[MAX][MAX], float ters_matris[MAX][MAX], int boyut)
{
    int i, j, k;
    float temp, temp2;

    for ( i = 0; i < boyut; i++)
    {
        for ( j = 0; j < boyut; j++)
        {
            if (i == j)
                ters_matris[i][j] = 1;
            else
                ters_matris[i][j] = 0;
        }
    }
    for ( i = 0; i < boyut; i++)
    {
        temp = matris[i][i];
        for ( j = 0; j < boyut; j++)
        {
            matris[i][j] /= temp;
            ters_matris[i][j] /= temp;
        }
        for ( k = 0; k < boyut; k++)
        {
            if (k != i)
            {
                temp2 = matris[k][i];
                for ( j = 0; j < boyut; j++)
                {
                    matris[k][j] -= matris[i][j]*temp2;
                    ters_matris[k][j] -= ters_matris[i][j]*temp2;
                }
            }
        }
    }
}
```

5 – Gauus Eleminasyon Yöntemi

```
#include <stdio.h>
#define MAX 20

void matris_yazdir(float A[MAX][MAX], float B[MAX], int boyut);
void matris_olustur(float A[MAX][MAX], float B[MAX], int *boyut);
void cozum(float A[MAX][MAX], float B[MAX], int boyut);

int main()
{
   float A[MAX][MAX], B[MAX];
   int boyut;
   matris_olustur(A, B, &boyut);
   matris_yazdir(A, B, boyut);
   cozum(A, B, boyut);
   printf("\n");
   return 0;
}

void matris_yazdir(float A[MAX][MAX], float B[MAX], int boyut)
{
   int i, j;
   printf("\n[A:B] matrisi\n------------------\n");
   for ( i = 0; i < boyut; i++)
   {
      for ( j = 0; j < boyut; j++)
      {
         printf("%f ", A[i][j]);
      }
      printf("%f\n", B[i]);
   }
}

void matris_olustur(float A[MAX][MAX], float B[MAX], int *boyut)
{
   int i, j;
   printf("\n[A][X] = [B] denklem sisteminde X'i bulmak icin once A matrisini sonra B matrisini
giriniz.");
   printf("\nOrnek:\n-x + 4y – 3z = -8\n3x + y - 2z = 9\nx - y + 4z = 1\ndenklemleri icin ornek
girdi : ");
   printf("\n\nA:\n-1  4 -3\n 3  1 -2\n 1 -1  4\n\nB:\n-8  9  1\n-----------------------------------");

   *boyut = 0;
   while (*boyut<1)
   {
      printf("\nA Matrisinin boyutunu giriniz : ");
      scanf("%d", boyut);
   }
   printf("\n");
   for ( i = 0; i < *boyut; i++)
   {
```

```
        for ( j = 0; j < *boyut; j++)
        {
            printf("A matrisinin %d. sutun %d. satirdaki elemani giriniz : ", (i+1), (j+1));
            scanf("%f", &A[i][j]);
        }
    }
    printf("\n");
    for ( i = 0; i < *boyut; i++)
    {
        printf("B matrisinin %d. elemanini giriniz : ", i+1);
        scanf("%f", &B[i]);
    }
}

void cozum(float A[MAX][MAX], float B[MAX], int boyut)
{
    float x[MAX];
    int i, j, k;
    float c, sum=0.0;

    // matrisi guncelleme
    for(i=0; i<boyut; i++)
    {
        for(j=0; j<boyut; j++)
        {
            if(j>i)
            {
                c = A[j][i] / A[i][i];
                for(k=0; k<boyut; k++)
                {
                    A[j][k] -= c * A[i][k];
                }
                B[j] -= c * B[i];
            }
        }
    }

    // B yi olusturma
    x[boyut-1] = B[boyut-1] / A[boyut-1][boyut-1];
    for(i=boyut-2; i>=0; i--)
    {
        sum=0;
        for(j=i+1; j<boyut; j++)
            sum += A[i][j] * x[j];

        x[i] = (B[i]-sum) / A[i][i];
    }
    printf("\nCOZUM : \n");
    for(i=0; i<boyut; i++)
        printf("\nx%d = %f\t", i+1, x[i]);
}
```

6 – Gauus Seidal Yöntemi

```
#include <stdio.h>
#define MAX 20

void matris_yazdir(float A[MAX][MAX], float B[MAX], int boyut);
void matris_olustur(float A[MAX][MAX], float B[MAX], int *boyut);
void matris_duzenleme(float A[MAX][MAX], float B[MAX], int boyut);
float mutlak_deger(float x);
float denklem(float sonuclar[MAX], float carpanlar[20], float b, int index, int boyut);
void cozum(float A[MAX][MAX], float B[MAX], int boyut);

int main()
{
   float A[MAX][MAX], B[MAX];
   int boyut;
   matris_olustur(A, B, &boyut);
   printf("\n[A:B] Matrisi (Girilen sekilde)\n------------------\n");
   matris_yazdir(A, B, boyut);
   matris_duzenleme(A, B, boyut);
   printf("\n[A:B] Matrisi (Duzenlendikten sonra)\n------------------\n");
   matris_yazdir(A, B, boyut);
   cozum(A, B, boyut);
   printf("\n");
   return 0;
}

void matris_yazdir(float A[MAX][MAX], float B[MAX], int boyut)
{
   int i, j;
   for ( i = 0; i < boyut; i++)
   {
      for ( j = 0; j < boyut; j++)
         printf("%f ", A[i][j]);
      printf("%f\n", B[i]);
   }
}

void matris_olustur(float A[MAX][MAX], float B[MAX], int *boyut)
{
   int i, j;
   printf("\n[A][X] = [B] denklem sisteminde X'i bulmak icin once A matrisini sonra B matrisini
giriniz.");
   printf("\nOrnek:\n-x + 4y - 3z = -8\n3x + y - 2z = 9\nx - y + 4z = 1\ndenklemleri icin ornek
girdi : ");
   printf("\n\nA:\n-1  4 -3\n 3  1 -2\n 1 -1  4\n\nB:\n-8  9  1\n-----------------------------------");

   printf("\nA Matrisinin boyutunu giriniz : ");
   scanf("%d", boyut);
   printf("\n");
   for ( i = 0; i < *boyut; i++)
   {
```

```
        for ( j = 0; j < *boyut; j++)
        {
            printf("A matrisinin %d. sutun %d. satirdaki elemani giriniz : ", (i+1), (j+1));
            scanf("%f", &A[i][j]);
        }
    }
    printf("\n");
    for ( i = 0; i < *boyut; i++)
    {
        printf("B matrisinin %d. elemanini giriniz : ", i+1);
        scanf("%f", &B[i]);
    }
}

void matris_duzenleme(float A[MAX][MAX], float B[MAX], int boyut)
{
    int i, j, k, index=0;
    float temp, max;
    for ( j = 0; j < boyut; j++)
    {
        max=-9999;
        for ( i = j; i < boyut; i++)
        {
            if (max < A[i][j])
            {
                max = A[i][j];
                index = i;
            }
        }
        if (j != index)
        {
            for ( k = 0; k < boyut; k++)
            {
                temp = A[j][k];
                A[j][k] = A[index][k];
                A[index][k] = temp;
            }
            temp = B[j];
            B[j] = B[index];
            B[index] = temp;
        }
    }
}

float mutlak_deger(float x)
{
    if (x < 0)
        x *= -1;
    return x;
}

float denklem(float sonuclar[MAX], float carpanlar[20], float b, int index, int boyut)
```

```
{
   float bol = carpanlar[index];
   int i;
   float res;
   for ( i = 0; i < boyut; i++)
      if (i != index)
         b -= carpanlar[i]*sonuclar[i];
   res = b / bol;
   return res;
}

void cozum(float A[MAX][MAX], float B[MAX], int boyut)
{
   float sonuc[MAX], sonuc_eski[MAX], hata=999, istenilen_hata;
   int i, j, ite=0;
   printf("\n");
   for ( i = 0; i < boyut; i++)
   {
      printf("x%d icin baslama degerini giriniz : ", (i+1));
      scanf("%f", &sonuc[i]);
   }
   printf("\nHata miktarini giriniz : ");
   scanf("%f", &istenilen_hata);

   printf("\nIterasyon");
   for ( i = 0; i < boyut; i++)
   {
      if (i != 0)
         printf("\t");
      printf("\tx%d", i+1);
   }
   printf("\n");
   do
   {
      ite ++;
      for ( i = 0; i < boyut; i++)
      {
         sonuc_eski[i] = sonuc[i];
         sonuc[i] = denklem(sonuc, A[i], B[i], i, boyut);
      }
      printf("%d\t\t", ite);
      for ( i = 0; i < boyut; i++)
         printf("%f\t", sonuc[i]);
      printf("\n");
      for ( i = 0; i < boyut; i++)
      {
         if (hata > mutlak_deger(sonuc[i]-sonuc_eski[i]))
            hata = mutlak_deger(sonuc[i]-sonuc_eski[i]);
      }
   }while(hata > istenilen_hata);
}
```

7 – Sayısal Türev (İleri-Geri-Merkezi)

```
#include <stdio.h>

float us_al(float x, int y);
float denklem(float x, float carpanlar[15], int derece);
float ileri_fark(float x, float h, float carpanlar[15], int derece);
float geri_fark(float x, float h, float carpanlar[15], int derece);
float merkezi_fark(float x, float h, float carpanlar[15], int derece);

int main()
{
   float carpanlar[15] = {0}, h, x;
   int derece, i;
   printf("Turevi alinacak denklemin derecesi : ");
   scanf("%d", &derece);
   for ( i = 0; i <= derece; i++)
   {
      printf("x uzeri %d nin carpanini giriniz : ", i);
      scanf("%f",  &carpanlar[i]);
   }
   printf("Fark (h) : ");
   scanf("%f", &h);
   printf("Turevine bakilacak nokta (x) : ");
   scanf("%f", &x);
   printf("\n----------------------\n\n");
   printf("Ileri farklar ile sayisal turev sonucu f'(%f) = %f\n", x, ileri_fark(x, h, carpanlar, derece));
   printf("Geri farklar ile sayisal turev sonucu f'(%f) = %f\n", x, geri_fark(x, h, carpanlar, derece));
   printf("Merkezi farklar ile sayisal turev sonucu f'(%f) = %f\n", x, merkezi_fark(x, h, carpanlar,
derece));

   return 0;
}

float us_al(float x, int y)
{
   if (y == 0)
      return 1;
   else
      return us_al(x, y-1)*x;
}

float denklem(float x, float carpanlar[15], int derece)
{
   int i;
   float res = 0;
   for ( i = 0; i <= derece; i++)
      res += carpanlar[i] * us_al(x, i);
   return res;
}

float ileri_fark(float x, float h, float carpanlar[15], int derece)
```

```
{
   float res;
   res = denklem(x+h, carpanlar, derece) - denklem(x, carpanlar, derece);
   res /= h;
   return res;
}

float geri_fark(float x, float h, float carpanlar[15], int derece)
{
   float res;
   res = denklem(x, carpanlar, derece) - denklem(x-h, carpanlar, derece);
   res /= h;
   return res;
}

float merkezi_fark(float x, float h, float carpanlar[15], int derece)
{
   float res;
   res = denklem(x+h, carpanlar, derece) - denklem(x-h, carpanlar, derece);
   res /= 2*h;
   return res;
}
```

8 – Simpson Yontemi
ve
10 – Degisken Donusumsuz Gregory Newton Enterpolasyonu
Sunulduğu için pdf’in en başındalar.

9 – Trapez Yöntemi

```
#include <stdio.h>
#define MAX 20

void girdi_alma(float *a, float *b, float *n);
float us_al(float x, int y);
float denklem(float x, float carpanlar[MAX], int derece);
float cozum(float a, float b, float n, float carpanlar[MAX], int derece);

int main()
{
   float carpanlar[MAX] = {0}, a, b, n;
   int derece, i;

   printf("Turevi alinacak denklemin derecesi : ");
   scanf("%d", &derece);
   for ( i = 0; i <= derece; i++)
   {
      printf("x uzeri %d nin carpanini giriniz : ", i);
      scanf("%f",  &carpanlar[i]);
   }
   printf("\n----------------------\n\n");
   girdi_alma(&a, &b, &n);
   printf("Denklemin %.2f  -  %.2f araligindaki integralinin alani : %f\n", a, b, cozum(a, b, n,
carpanlar, derece));

   return 0;
}

void girdi_alma(float *a, float *b, float *n)
{
   printf("Integral alinacak araligin baslangic degeri (a): ");
   scanf("%f", a);
   printf("Integral alinacak araligin bitis degeri (b) : ");
   scanf("%f", b);
   printf("n degerini giriniz (a-b arasinda kac yamuk olacak) (n) : ");
   scanf("%f", n);
}

float us_al(float x, int y)
{
   if (y == 0)
      return 1;
   else
      return us_al(x, y-1)*x;
}

float denklem(float x, float carpanlar[MAX], int derece)
{
   int i;
   float res = 0;
```

```
    for ( i = 0; i <= derece; i++)
        res += carpanlar[i] * us_al(x, i);
    return res;
}

float cozum(float a, float b, float n, float carpanlar[MAX], int derece)
{
    float h = (b - a) / n;
    float sonuc = h;
    float toplam = (denklem(a, carpanlar, derece) + denklem(b, carpanlar, derece)) / 2.0;
    while (a != b)
    {
        a += h;
        toplam += denklem(a, carpanlar, derece);
    }

    // alan pozitif olmali
    if (toplam < 0)
        toplam *= -1;

    return sonuc*toplam;
}
```

8 – Simpson Yontemi
ve
10 – Degisken Donusumsuz Gregory Newton Enterpolasyonu
Sunulduğu için pdf’in en başındalar.

11 – Grafik Yontemi

```
// durma kosullari
// iterasyon sayisi
// gercek kok (P) var ise |P-xn| < hata
// gercek kok yok ise |x(n+1)-xn| < hata

#include <stdio.h>

float mutlak_deger(float x);
float us_al(float x, int y);
float denklem(float x, float carpanlar[15], int derece);
void iterasyon_solution(float x0, float x1, float iterasyon_sayisi, float carpanlar[15], int derece);
void gercek_kokle_solution(float x0, float x1, float istenilen_hata_miktari, float gercek_deger, float
carpanlar[15], int derece);
void gercek_koksuz_solution(float x0, float x1, float istenilen_hata_miktari, float carpanlar[15], int
derece);

int main()
{
   float carpanlar[15] = {0};
   int derece, i;
   printf("Girilecek denklemin derecesi : ");
   scanf("%d", &derece);
   for ( i = 0; i <= derece; i++)
   {
      printf("x uzeri %d nin carpanini giriniz : ", i);
      scanf("%f",  &carpanlar[i]);
   }

   // gerekli girdileri alma
   float istenilen, x0, delta_x, gercek_deger;
   printf("\nIstenilen hata miktari sinirini veya iterasyon sayisini giriniz (iterasyon sayisi 2 den
buyuk olmalidir) : ");
   scanf("%f", &istenilen);
   printf("\nIlk bakilacak x degeri ve ilk bakilacak aralik genisligini bir bosluk ile giriniz. : ");
   scanf("%f %f", &x0, &delta_x);

   // durma kosuluna gore fonksiyona gonderme
   if (istenilen < 3.0)
   {
      printf("\nGercek kokun x degerini giriniz (bilinmiyorsa 999 giriniz.): ");
      scanf("%f", &gercek_deger);
      if (gercek_deger == 999)
         gercek_koksuz_solution(x0, delta_x, istenilen, carpanlar, derece);
      else
         gercek_kokle_solution(x0, delta_x, istenilen, gercek_deger, carpanlar, derece);
   }
   else
      iterasyon_solution(x0, delta_x, istenilen, carpanlar, derece);

   return 0;
```

```
}

float mutlak_deger(float x)
{
   if (x < 0)
      x *= -1;
   return x;
}

float us_al(float x, int y)
{
   if (y == 0)
      return 1;
   else
      return us_al(x, y-1)*x;
}

float denklem(float x, float carpanlar[15], int derece)
{
   int i;
   float res = 0;
   for ( i = 0; i <= derece; i++)
      res += carpanlar[i] * us_al(x, i);
   return res;
}

void iterasyon_solution(float x0, float delta_x, float iterasyon_sayisi, float carpanlar[15], int derece)
{
   int i;
   float alt_deger, ust_deger;
   for ( i = 0; i < iterasyon_sayisi; i++)
   {
      do
      {
         alt_deger = denklem(x0, carpanlar, derece);
         ust_deger = denklem((x0+delta_x), carpanlar, derece);
         x0 += delta_x;
      } while (alt_deger * ust_deger > 0);
      x0 -= delta_x;
      delta_x /= 2.0;
      printf("\n%d iterasyon sonrasinda bulunan sonuc : f(%f) = %f \n", i+1, x0, alt_deger);
   }
}

void gercek_kokle_solution(float x0, float delta_x, float istenilen_hata_miktari, float gercek_deger,
float carpanlar[15], int derece)
{
   int i=0;
   float alt_deger, ust_deger, hata_miktari=istenilen_hata_miktari+1;
   while (hata_miktari > istenilen_hata_miktari)
   {
      i++;
```

```
        do
        {
            alt_deger = denklem(x0, carpanlar, derece);
            ust_deger = denklem((x0+delta_x), carpanlar, derece);
            x0 += delta_x;
        } while (alt_deger * ust_deger > 0);
        x0 -= delta_x;
        delta_x /= 2;
        hata_miktari = mutlak_deger(gercek_deger - x0);
        printf("\n%d ) Hata miktari %f iken bulunan : f(%f) = %f \n", i, hata_miktari, x0, alt_deger);
    }
}

void gercek_koksuz_solution(float x0, float delta_x, float istenilen_hata_miktari, float
carpanlar[15], int derece)
{
    int i=0;
    float alt_deger, ust_deger;
    while (delta_x > istenilen_hata_miktari)
    {
        i++;
        do
        {
            alt_deger = denklem(x0, carpanlar, derece);
            ust_deger = denklem((x0+delta_x), carpanlar, derece);
            x0 += delta_x;
        } while (alt_deger * ust_deger > 0);
        x0 -= delta_x;
        delta_x /= 2;
        printf("\n %d ) Hata miktari (delta x) %f iken bulunan : f(%f) = %f \n", i, delta_x, x0,
alt_deger);
    }
}
```

12 – Secant Yontemi

```
// durma kosullari
// iterasyon sayisi
// gercek kok (P) var ise |P-xn| < hata
// gercek kok yok ise |x(n+1)-xn| < hata

#include <stdio.h>

float mutlak_deger(float x);
float us_al(float x, int y);
float denklem(float x, float carpanlar[15], int derece);
float denklemin_turevi(float x0, float x1, float carpanlar[15], int derece);
void iterasyon_solution(float x0, float x1, float iterasyon_sayisi, float carpanlar[15], int derece);
void gercek_kokle_solution(float x0, float x1, float istenilen_hata_miktari, float gercek_deger, float
carpanlar[15], int derece);
void gercek_koksuz_solution(float x0, float x1, float istenilen_hata_miktari, float carpanlar[15], int
derece);

int main()
{
   float carpanlar[15] = {0};
   int derece, i;
   printf("Girilecek denklemin derecesi : ");
   scanf("%d", &derece);
   for ( i = 0; i <= derece; i++)
   {
      printf("x uzeri %d nin carpanini giriniz : ", i);
      scanf("%f",  &carpanlar[i]);
   }

   // gerekli girdileri alma
   float istenilen, x0, x1, gercek_deger;
   printf("\nIstenilen hata miktari sinirini veya iterasyon sayisini giriniz (iterasyon sayisi 2 den
buyuk olmalidir) : ");
   scanf("%f", &istenilen);
   printf("\n x1 ve x0 degerlerini bir bosluk birakarak giriniz : ");
   scanf("%f %f", &x1, &x0); // -5, -4  ,  14, 15

   // durma kosuluna gore fonksiyona gonderme
   if (istenilen < 3.0)
   {
      printf("\nGercek kokun x degerini giriniz (bilinmiyorsa 999 giriniz.): ");
      scanf("%f", &gercek_deger); // 0.585786438 , 3.414213
      if (gercek_deger == 999)
         gercek_koksuz_solution(x0, x1, istenilen, carpanlar, derece);
      else
         gercek_kokle_solution(x0, x1, istenilen, gercek_deger, carpanlar, derece);
   }
   else
      iterasyon_solution(x0, x1, istenilen, carpanlar, derece);
```

```
    return 0;
}

float mutlak_deger(float x)
{
    if (x < 0)
        x *= -1;
    return x;
}

float us_al(float x, int y)
{
    if (y == 0)
        return 1;
    else
        return us_al(x, y-1)*x;
}

float denklem(float x, float carpanlar[15], int derece)
{
    int i;
    float res = 0;
    for ( i = 0; i <= derece; i++)
        res += carpanlar[i] * us_al(x, i);
    return res;
}

float denklemin_turevi(float x0, float x1, float carpanlar[15], int derece)
{
    return ( denklem(x0, carpanlar, derece) - denklem(x1, carpanlar, derece) ) / (x0 - x1);
}

void iterasyon_solution(float x0, float x1, float iterasyon_sayisi, float carpanlar[15], int derece)
{
    int i;
    float x2_degeri, x2;
    for ( i = 0; i < iterasyon_sayisi; i++)
    {
        x2 = x0 - (denklem(x0, carpanlar, derece) / denklemin_turevi(x0, x1, carpanlar, derece));
        x2_degeri = denklem(x2, carpanlar, derece);

        printf("\n%d iterasyon sonrasinda bulunan sonuc : f(%f) = %f \n", i+1, x2, x2_degeri);
        x0 = x1;
        x1 = x2;
    }
}

void gercek_kokle_solution(float x0, float x1, float istenilen_hata_miktari, float gercek_deger, float
carpanlar[15], int derece)
{
    float x2_degeri, x2, hata_miktari=99;
    while ( hata_miktari > istenilen_hata_miktari )
```

```
    {
        x2 = x0 - (denklem(x0, carpanlar, derece) / denklemin_turevi(x0, x1, carpanlar, derece));
        x2_degeri = denklem(x2, carpanlar, derece);
        hata_miktari = mutlak_deger(x2 - gercek_deger);
        printf("\n Hata miktari %f iken bulunan sonuc : f(%f) = %f \n", hata_miktari, x2, x2_degeri);
        x0 = x1;
        x1 = x2;
    }
}

void gercek_koksuz_solution(float x0, float x1, float istenilen_hata_miktari, float carpanlar[15], int
derece)
{
    float x2_degeri, x2, hata_miktari=99;
    while ( hata_miktari > istenilen_hata_miktari )
    {
        x2 = x0 - (denklem(x0, carpanlar, derece) / denklemin_turevi(x0, x1, carpanlar, derece));
        x2_degeri = denklem(x2, carpanlar, derece);
        hata_miktari =  mutlak_deger(x2 - x1);
        printf("\n Hata miktari %f iken bulunan sonuc : f(%f) = %f \n", hata_miktari, x2, x2_degeri);
        x0 = x1;
        x1 = x2;
    }
}
```