Yıldız Teknik Üniversitesi
Elektrik-Elektronik Fakültesi
Bilgisayar Mühendisliği Bölümü
BLM1022-Sayısal Analiz
Grup 2
Öğr. Gör. Dr. Ahmet ELBİR
Dönem Projesi

Fatıma Bera Çolak

23011111

bera.colak@std.yildiz.edu.tr

İçindekiler

Ön Bilgi

Program, 10 tane belirli işlemi yerine getirebilmek için tasarlanmıştır. Bu işlemler sırasıyla şöyledir:
1. Bisection yöntemi
2. Regula-Falsi yöntemi
3. Newton-Rapshon yöntemi
4. NxN'lik bir matrisin tersi
5. Gauss eliminasyon yöntemi
6. Gauss-Seidel yöntemi
7. Sayısal Türev
8. Simpson yöntemi
9. Trapez yöntemi
10. Değişken dönüşümsüz Gregory-Newton enterpolasyonu

Yöntemlerin çalışıp çalışmaması + ve - ile belirtilmiştir									
1	2	3	4	5	6	7	8	9	10
+	+	+	+	+	+	+	+	+	+

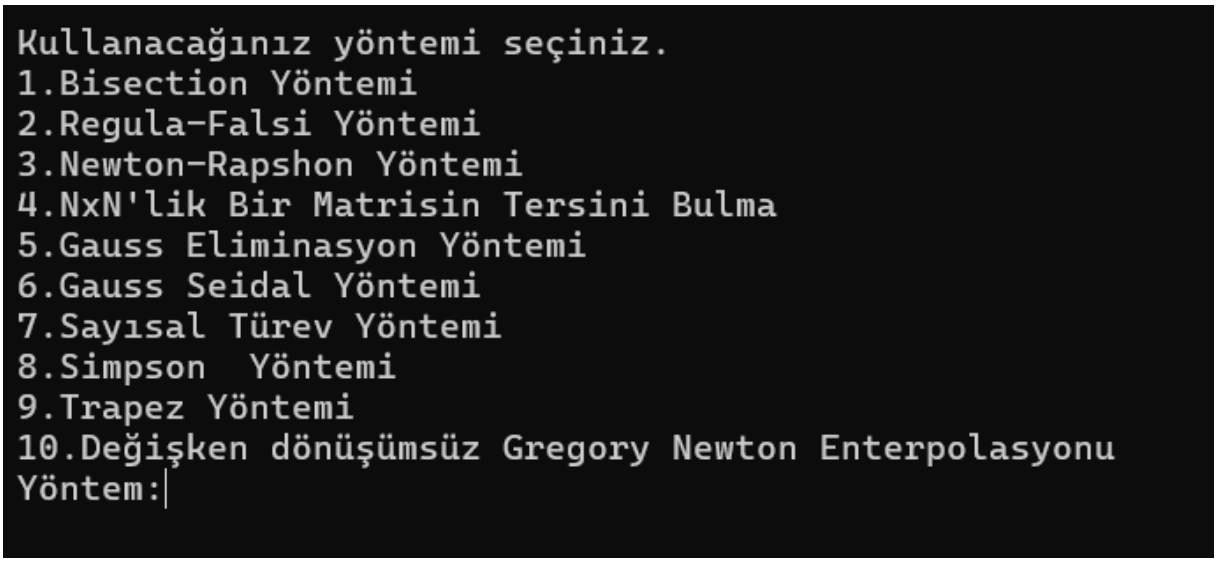
Desteklenen Fonksiyonlar

Program; (, ^, /, *, -, +,) ifadeleri ile oluşturulan her türlü fonksiyonu hesaplayarak yöntemi uygulayabilir, 'e' sayısını destekler. Üstel fonksiyon ve polinom hesabı yapabilir.

2x^3+x^2-4
(e^(x^2-5x)+1) / 2x
1/(1+x^4)

Ana Menü

Çalıştırılmak istenilen işlem program çalıştırıldıktan sonra numarası girilip gereken parametrelerin verilmesiyle çalışır.



Enterpolasyon için Parametre Alımı

Enterpolasyonda parametreler (x, y) değerleri ve hesaplama yapılmak istenen değerdir. X ve y değerlerini tutan struct yapısı ve bunları almak için bir fonksiyonum var.

```
struct Data { //structto keep inputs (x and y values)
    double x;
    double y;
};
// Function to get inputs (x and y values)
void getInputsforGN(struct Data data[], int *n, double * xi) {
    printf("Kaç tane veri gireceğinizi belirtiniz: ");
    scanf("%d", n);
int i;
    printf("\nDeğerleri giriniz:\n");
    for ( i = 0; i < *n; i++) {
        printf("x[%d] = ", i);
        scanf("%lf", &data[i].x);
        printf("y[%d] = ", i);
        scanf("%lf", &data[i].y);
    }
    printf("Enterpolasyon hesaplaması yapılacak değeri giriniz:");
        scanf("%lf", xi);
}
```

Matris Girişi

Matrisin tersi (4) yöntemi için ilk istenilen parametre NxN'lik bir kare matris için N değeridir. Lineer denklem çözümü yöntemleri (5, 6) için ise bu değer N tane fonksiyonun olduğunu belirtir. Nx(N+1)'lik bir matris için yer açılır. N değeri girildikten sonra matrisin elemanları satır satır alınır.

```
Kaç tane deklem gireceğinizi yazınız.
2
 Denklem katsayılarını ve fonksiyonun eşitlendiği değeri sırasıyla giriniz:

Matrisin [][3]. elemanı çok değişkenli fonksiyonun eşitlendiği değerdir.
Matrisin elemanlarını giriniz:
[1][1]elemanı giriniz: 1
[1][2]elemanı giriniz: 4
[1][3]elemanı giriniz: 5
[2][1]elemanı giriniz: 5
[2][2]elemanı giriniz: 6
[2][3]elemanı giriniz: 2
```

```
Kaç tane deklem gireceğinizi yazınız.
5
Matrisin elemanlarını giriniz:
[1][1]elemanı giriniz:
22
[1][2]elemanı giriniz: 5
[1][3]elemanı giriniz: 4
[1][4]elemanı giriniz: 7
[1][5]elemanı giriniz: -8
[1][6]elemanı giriniz: 45
[2][1]elemanı giriniz: 78
[2][2]elemanı giriniz: 4
[2][3]elemanı giriniz: 1
[2][4]elemanı giriniz: 1
[2][5]elemanı giriniz: 25
[2][6]elemanı giriniz: 7
[3][1]elemanı giriniz: 7
[3][2]elemanı giriniz: 88
[3][3]elemanı giriniz: 9
[3][4]elemanı giriniz: -47
[3][5]elemanı giriniz: 45
[3][6]elemanı giriniz: 12
[4][1]elemanı giriniz: 15
[4][2]elemanı giriniz: 77
[4][3]elemanı giriniz: 2
[4][4]elemanı giriniz: 80
[4][5]elemanı giriniz: 45
[4][6]elemanı giriniz: 180
[5][1]elemanı giriniz: 4
[5][2]elemanı giriniz: 5
[5][3]elemanı giriniz: 7
[5][4]elemanı giriniz: 8
[5][5]elemanı giriniz: -5
[5][6]elemanı giriniz: 46
```

```
Matris boyutunu girin: 3
Matrisin elemanlarını giriniz:
[1][1]elemanı giriniz: 1
[1][2]elemanı giriniz: 2
[1][3]elemanı giriniz: 3
[2][1]elemanı giriniz: 4
[2][2]elemanı giriniz: 5
[2][3]elemanı giriniz: 6
[3][1]elemanı giriniz: 4
[3][2]elemanı giriniz: 5
[3][3]elemanı giriniz: 7
Verilen Matris:
1,000000 2,000000 3,000000
4,000000 5,000000 6,000000
4,000000 5,000000 7,000000
```

Fonksiyon Girişi

Diğer bütün yöntemlerde (enterpolasyon hariç) fonksiyon kullanıcıdan string olarak alınır. Girdi girilirken parantez kullanımına ve boşluk bırakmamaya dikkat edilmesi gerekir. Polinomlarda katsayıların çarpma işlemiyle belirtilmesi gerekir. Negatif katsayı ile başlanacaksa başına 0 gelmesi gerekir.

ÖRNEKLER:

Fonksiyon = -(1/3)x^3 - (6/5)x^2 + 1/x^2
Girilmesi gereken string = 0-(1/3)*x^3-(6/5)*x^2+1/x^2

Fonksiyon = e^(x^2) - 5ex + e^5
Girilmesi gereken string = e^(x^2)-5*e*x+e^5

Fonksiyon Girişi Olan Yöntemler için Parametre Alımı

Newton Raphson açık bir yöntem olduğu için tek değer alınırken kapalı yöntemlerde aralık alınıyor, integral hesaplarken de hata değeri yerine kaç parçaya ayrılmak istendiği parametre olarak alınıyor. Sayısal türevde de yöntem seçimi var. Bu yüzden dört farklı degerAl fonksiyonu oluşturdum.

```
void degerAl(double* a, double* b, char* fonksiyon, double* hata ){
  printf("Kök bulmak istediiniz fonksiyonu giriniz.\n");
        scanf("%s",fonksiyon);
            printf("Kökün bulunduğu aralığın değerlerini giriniz.\n");
            scanf("%lf %lf", a, b);
            printf("Hata değerini giriniz:");
            scanf("%lf", hata);
}
void NRdegerAl(double* a, char* fonksiyon, double* hata ){
  printf("Kök bulmak istediğiniz fonksiyonu giriniz.\n");
            scanf("%s",fonksiyon);
            printf("Başlangıç değerini giriniz.\n");
            scanf("%lf", a);
            getchar();
            printf("Hata değerini giriniz:");
            scanf("%lf", hata);

}
void integralDegerAl(double* a, double* b,int* n, char* fonksiyon){
        printf("İntegralini almak istediğiniz fonksiyonu giriniz.\n");
        scanf("%s", fonksiyon);
        printf("İntegralini almak istediğiniz aralığı (a ve b olarak)giriniz .\n");
        scanf("%lf %lf", a, b);
        printf("Kaç parçayla hesap yapmak istediğinizi giriniz.\nSimpson 1/3 için çift sayı olmalıdır.\n");
        scanf("%d", n);
void getInputforST(char *fonksiyon, double *x, double *h, int *method) {
    printf("Türevini bulmak istediğiniz fonksiyonu ve değerlerini giriniz:\n");
    printf("Türevini bulmak istediğiniz fonksiyonu giriniz\n");
    scanf("%s", fonksiyon);
    getchar(); //  scanf sonraki adımda doğru çalışabilsin diye

    printf("Türevini bulmak istediğiniz değerini giriniz:");
    scanf("%lf", x);
    getchar(); //  scanf sonraki adımda doğru çalışabilsin diye

    printf("h değerini giriniz: ");
    scanf("%lf", h);
    getchar(); //  scanf sonraki adımda doğru çalışabilsin diye

    printf("Hangi yöntemi kullanacağınızı seçiniz.\n");
    printf("1. İleri farkla\n");
    printf("2. Geri farkla\n");
    printf("3. Merkezi farkla\n");
    scanf("%d", method);
    getchar(); // scanf sonraki adımda doğru çalışabilsin diye
}
```

Bisection Yöntemi

Parametreler:

Fonksiyon= x^3 – 7x^2 + 14x – 6

Kökün bulunduğu aralık: (0,1)

Hata payı: 0,01

```
Yöntem:1
Kök bulmak istediiniz fonksiyonu giriniz.
x^3-7*x^2+14*x-6
Kökün bulunduğu aralığın değerlerini giriniz.
0
1
Hata değerini giriniz:0,01

Sinirlarimiz : [0,000000, 1,000000]
Orta Noktamiz : 0,500000
Hata=1,000000

Sinirlarimiz : [0,500000, 1,000000]
Orta Noktamiz : 0,750000
Hata=0,250000

Sinirlarimiz : [0,500000, 0,750000]
Orta Noktamiz : 0,625000
Hata=0,062500

Sinirlarimiz : [0,500000, 0,625000]
Orta Noktamiz : 0,562500
Hata=0,015625
İterasyon sayisi: 4
Bulunan yaklaşık kök değeri= 0,5625
```

```c
double bisectionMethod(char* postfix, double a, double b, double hata){
    double c;     // orta nokta
    int iter = 0;
    if (evaluateExpression(postfix, b) * evaluateExpression(postfix, a) > 0) {
        printf("Başlangıç değerleri uygun değil. f(a) ve f(b) aynı işaretli olmamalıdır.\n");
        return NAN; // Not-A-Number (NaN) değeri döndürme
    }

    while (((b - a) / pow(2, iter) >= hata ) && iter<MAX_ITER) { //durma kosulu
        c = (a + b) / 2.0;       //orta nokta sayılır
        double c_degeri = evaluateExpression(postfix, c);
        printf("\nSinirlarimiz : [%lf, %lf]", a, b);
        printf("\nOrta Noktamiz : %lf\n", c);
        printf("Hata=%lf\n", (b - a) / pow(2, iter));

        if (c_degeri == 0.0) {
            printf("Kök bulundu.\n");
            return c;
        } else if (evaluateExpression(postfix, b) * c_degeri > 0) {
            b = c;
        } else {
            a = c;
        }
        iter++;
    }
    printf("İterasyon sayisi: %d\n", iter);
    return c;
}
```

Regula-Falsi Yöntemi

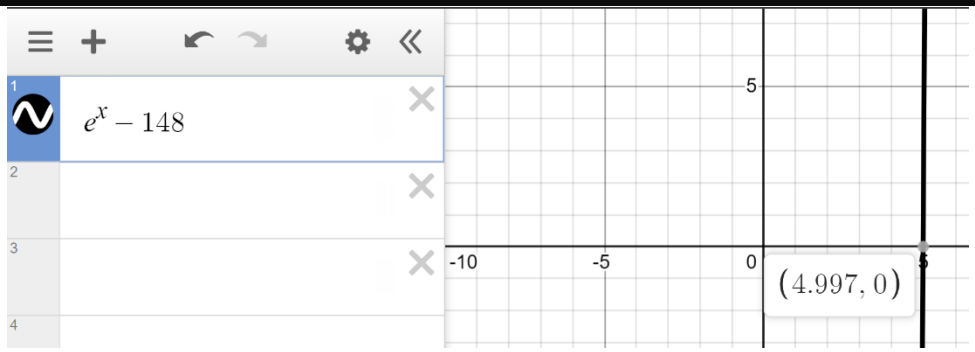
Parametreler:

Fonksiyon= e^x-148

Kökün bulunduğu aralık: (4,5)

Hata payı: 0,01

```
Yöntem:2
Kök bulmak istediiniz fonksiyonu giriniz.
e^x-148
Kökün bulunduğu aralığın değerlerini giriniz.
4
5
Hata değerini giriniz:0,01
Bulunan yaklaşık kök değeri= 4,9956
--------------------------------
Process exited after 16.07 seconds with return value 0
Press any key to continue . . .
```




```
double regulaFalsi(char* postfix, double a, double b, double hata){
    double c;
     int iter = 0;
     if (evaluateExpression(postfix, b) * evaluateExpression(postfix, a) > 0) {
         printf("Başlangıç değerleri uygun değil. f(a) ve f(b) aynı işaretli olmamalıdır.\n");
         return NAN; // Not-A-Number (NaN) değeri döndürme
    }
    double fa=1;
    double fc,fb=1;
     while ((b - a) / pow(2, iter) >= hata && (fa*fb )!=0){
         fa=evaluateExpression(postfix,a);
         fb=evaluateExpression(postfix,b);
         c= ((b*fa)-(a*fb))/ (fa-fb);
         fc=evaluateExpression(postfix,c);
          if(fa*fc<0){
              b=c;
           }else {
              a=c;
           }
         iter++;
     }
    if((fa*fb)!=0){
         return c;
    }else if(fa==0){
         return a;
    }else return b;
}
```

Newton Raphson Yöntemi

Parametreler:

Fonksiyon= x^3 – 7x^2 + 14x – 6

Başlangıç değeri: 0

Hata payı: 0,01

```
Yöntem:3
Kök bulmak istediğiniz fonksiyonu giriniz.
x^3-7*x^2+14*x-6
Başlangıç değerini giriniz.
0
Hata değerini giriniz:0,01
Bulunan yaklaşık kök değeri= 0,5858
--------------------------------
```

```c
double newtonRaphson(char* postfix, double a, double hata){
    double x = a;
    double fx, dfx;
    double x1;
    int flag=1;
    while (flag) {
    fx = evaluateExpression(postfix, x);
    double h = 1e-7; // küçük bir sayi
    double fxh1 = evaluateExpression(postfix, x + h); //sayisalTurev fonksiyonumu cagırmaya calistim olmadi
        double fxh2 = evaluateExpression(postfix, x - h);
        dfx = (fxh1 - fxh2) / (2 * h); // merkezi fark hesabi
        x1 = x - fx / dfx;                // formul uygulaması
        if (fabs(x1 - x) < hata) {        //durma kosulu
            flag=0;
        }
     x = x1;
    }
    free(postfix);
    return x1;
}
```

Matrisle ilgili Fonksiyonlar

```c
double** yerAc(int n, int m){
    int i;
    double **matrix = (double **)calloc(n ,sizeof(double *));
    if (matrix == NULL) {
        printf("Bellek tahsis edilemedi.\n");
        exit(1);
    }
    for ( i = 0; i < n; i++) {
        matrix[i] = (double *)calloc(m ,sizeof(double));
        if (matrix[i] == NULL) {
            printf("Bellek tahsis edilemedi.\n");
            exit(1);
        }
    }
    return matrix;
}
```

```c
void freeMemory(double **matrix, int m){
            int j;
        for ( j = 0; j < m; j++) {
                free(matrix[j]);
            }
            free(matrix);
}
```

```c
void getInputMatrix(double **matrix, int n, int m) {
    int i,j;
    printf("Matrisin elemanlarını giriniz:\n");
    for ( i = 0; i < n; i++) {
        for ( j = 0; j < m; j++) {
            printf("[%d][%d]elemanı giriniz: ", i+1, j+1);
            scanf("%lf", &matrix[i][j]);
        }
    }
}// matrisin elemanlarını almak için fonksiyon
void birimMatris(double **matrix, int n){
    int i,j;
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            if(i==j){          //köşegen mi kontrolü
                *(*(matrix+i)+j)=1;
            }else *(*(matrix+i)+j)=0;
        }
    }
}// birimMatris oluşturmak için fonksiyon
void printMatrix(double **matrix, int n, int m){
        int i,j;
    for(i=0;i<n;i++){
        for(j=0;j<m;j++){
            printf("%lf ", matrix[i][j]);
        }printf("\n");
}
```

```c
void swapRows(double **matrix, int row1, int row2, int n) {
    int i;
    for ( i = 0; i < n; i++) {
        double temp = matrix[row1][i];
        matrix[row1][i] = matrix[row2][i];
        matrix[row2][i] = temp;
    }
}
void multiplyRow(double **matrix, int row, double scalar, int n) {
    int i;
    for ( i = 0; i < n; i++)
        matrix[row][i] *= scalar;
}
void addMultipleOfRow(double **matrix, int sourceRow, int destinationRow, double multiple, int n) {
   int i;
    for ( i = 0; i < n; i++)
        matrix[destinationRow][i] += multiple * matrix[sourceRow][i];
}
double** findMaxRow(double **mat,int n){
    int i,j;
    double temp;
    double max;
    for(i=0;i<n;i++){
        max=mat[i][i];
        int row=i;
        for(j=i+1;j<n;j++){
            temp=mat[j][i];
            if (findMax(temp,max)){
                row=j;}
        } swapRows(mat, i, row, n+1);
    }
    return mat;
}// Gauss Seidelde max katsayıları olanları köşegene almak için
```

NxN'lik Bir Matrisin Tersini Bulma

```
Yöntem:4
Matris boyutunu girin: 3
Matrisin elemanlarını giriniz:
[1][1]elemanı giriniz: 1
[1][2]elemanı giriniz: 0
[1][3]elemanı giriniz: 5
[2][1]elemanı giriniz: 1
[2][2]elemanı giriniz: 1
[2][3]elemanı giriniz: 1
[3][1]elemanı giriniz: 5
[3][2]elemanı giriniz: 8
[3][3]elemanı giriniz: 4
Verilen Matris:
1,000000 0,000000 5,000000
1,000000 1,000000 1,000000
5,000000 8,000000 4,000000
Matrisin Tersi:
-0,363636 3,636364 -0,454545
0,090909 -1,909091 0,363636
0,272727 -0,727273 0,090909
```

```c
double **matrisTersi(double **matrix, int n) {
    //birim matris için yer açma ve olusturma
    double **inverseMatrix = yerAc(n,n);
    birimMatris(inverseMatrix, n);
    // Girilen matrisi birim matrise cevirmek icin elementer satir islemleri kullanma
    int singular = 0;    // tersi var mi diye kontrol etmek icin
    int i = 0;
    while (i < n && !singular) {
        // Find the pivot
        int foundPivot = 0;
        int j = i;
        while (j < n && matrix[j][i] == 0) {// satırda 0 olmayan ilk elemanı buluyoruz
            j++;
        }
        if (j < n) {     //satirdaki hepsi 0 ise else statementa gider singüler matristir
            if (j != i) {        // satirda 0 olmayan ilk elemani bulduk
                swapRows(matrix, i, j, n);       // birim matrise benzesin diye satir degisiyor
                swapRows(inverseMatrix, i, j, n);// ayni islemi hem verilen matrise hem de birim matrise yapariz
            }
            // kosegendeki elemanı 1 yapalım
            double scalar = 1.0f / matrix[i][i];
            multiplyRow(matrix, i, scalar, n);
            multiplyRow(inverseMatrix, i, scalar, n);
            for (j = 0; j < n; j++) {  // Köşegenin altında ve üstünde kalan tüm elemanlar 0'A eşitlenir
                if (j != i) {
                    double multiple = -matrix[j][i];
                    addMultipleOfRow(matrix, i, j, multiple, n);
                    addMultipleOfRow(inverseMatrix, i, j, multiple, n);
                }
            }
        } else { //satirdaki hepsi 0 ise matrisin determinant 0 olur tersi yoktur
            singular = 1;
        }
        i++;
    }
    if (singular) {
        printf("Matris singülerdir, tersi yoktur.\n");
        return NULL;
    }
    return inverseMatrix;
}
```

Gauss Eliminasyon Yöntemi

```
Yöntem:5
Kaç tane deklem gireceğinizi yazınız.
5
Matrisin elemanlarını giriniz:
[1][1]elemanı giriniz:
22
[1][2]elemanı giriniz: 5
[1][3]elemanı giriniz: 4
[1][4]elemanı giriniz: 7
[1][5]elemanı giriniz: -8
[1][6]elemanı giriniz: 45
[2][1]elemanı giriniz: 78
[2][2]elemanı giriniz: 4
[2][3]elemanı giriniz: 1
[2][4]elemanı giriniz: 1
[2][5]elemanı giriniz: 25
[2][6]elemanı giriniz: 7
[3][1]elemanı giriniz: 7
[3][2]elemanı giriniz: 88
[3][3]elemanı giriniz: 9
[3][4]elemanı giriniz: -47
[3][5]elemanı giriniz: 45
[3][6]elemanı giriniz: 12
[4][1]elemanı giriniz: 15
[4][2]elemanı giriniz: 77
[4][3]elemanı giriniz: 2
[4][4]elemanı giriniz: 80
[4][5]elemanı giriniz: 45
[4][6]elemanı giriniz: 180
[5][1]elemanı giriniz: 4
[5][2]elemanı giriniz: 5
[5][3]elemanı giriniz: 7
[5][4]elemanı giriniz: 8
[5][5]elemanı giriniz: -5
[5][6]elemanı giriniz: 46
1,000000 0,227273 0,181818 0,318182 -0,363636 2,045455
-0,000000 1,000000 0,960265 1,735099 -3,887417 11,112583
-0,000000 -0,000000 1,000000 2,646645 -5,095842 12,791595
0,000000 0,000000 0,000000 1,000000 -0,199859 1,793310
0,000000 0,000000 0,000000 0,000000 1,000000 -1,123906
Çözüm:
x1: 0,330
x2: 1,225
x3: 2,913
x4: 1,569
x5: -1,124
```

$$R_4 \leftarrow R_4 + 0.199859 \times R_5$$

$$= \left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & 0.329707 \\ 0 & 1 & 0 & 0 & 0 & 1.224805 \\ 0 & 0 & 1 & 0 & 0 & 2.912587 \\ 0 & 0 & 0 & 1 & 0 & 1.568687 \\ 0 & 0 & 0 & 0 & 1 & -1.123906 \end{array}\right]$$

i. e.

$a = 0.329707$

$b = 1.224805$

$c = 2.912587$

$d = 1.568687$

$e = -1.123906$

Solution By Gauss jordan elimination method
$a = 0.329707, b = 1.224805, c = 2.912587, d = 1.568687$ and $e = -1.123906$

```
void gaussEliminasyon(double **matrix, int n, double*cevap) {
    // Girilen matrisi ust ucgen matrise cevirmek icin elementer satir islemleri kullanma
    int i = 0,j;
    while (i < n) {
            // kosegendeki elemanı 1 yapalım
            double scalar = 1.0f / matrix[i][i];
            multiplyRow(matrix, i, scalar, n+1);
            // Eliminate non-zero elements below the diagonal
            for (j = i+1; j < n; j++) {
                if (matrix[j][i]!=0) {
                    double multiple = (-1)*matrix[j][i];
                    addMultipleOfRow(matrix, i, j, multiple, n+1);
                }
            }
        i++;
    }
    printMatrix(matrix, n, n+1);
     for (i = n - 1; i >= 0; i--) {
        cevap[i] = matrix[i][n];
        for (j = i + 1; j < n; j++) {
            cevap[i] -= matrix[i][j] * cevap[j];
        }

    }
    sonucYazdir(cevap, n);
}
```

Gauss Seidel Yöntemi

```
Kullanacağınız yöntemi seçiniz.
1.Bisection Yöntemi
2.Regula-Falsi Yöntemi
3.Newton-Rapshon Yöntemi
4.NxN'lik Bir Matrisin Tersini Bulma
5.Gauss Eliminasyon Yöntemi
6.Gauss Seidal Yöntemi
7.Sayısal Türev Yöntemi
8.Simpson  Yöntemi
9.Trapez Yöntemi
10.Değişken dönüşümsüz Gregory Newton Enterpolasyonu
Yöntem:6
Kaç tane deklem gireceğinizi yazınız.
2
 Denklem katsayılarını ve fonksiyonun eşitlendiği değeri sırasıyla giriniz:

Matrisin [][3]. elemanı çok değişkenli fonksiyonun eşitlendiği değerdir.
Matrisin elemanlarını giriniz:
[1][1]elemanı giriniz: 1
[1][2]elemanı giriniz: 4
[1][3]elemanı giriniz: 5
[2][1]elemanı giriniz: 5
[2][2]elemanı giriniz: 6
[2][3]elemanı giriniz: 2
Hata değerini girin: 0,001
Başlangıç değerlerini girin:
x1: 1
x2: 1
İterasyon 1, Hata = 2,250000
İterasyon 2, Hata = 0,675000
İterasyon 3, Hata = 0,202500
İterasyon 4, Hata = 0,060750
İterasyon 5, Hata = 0,018225
İterasyon 6, Hata = 0,005467
İterasyon 7, Hata = 0,001640
İterasyon 8, Hata = 0,000492
Gauss-Seidel yöntemiyle 8 iterasyonda sonuca ulaşıldı.
Çözüm:
x1: -1,571
x2: 1,643
```

```c
void gaussSeidel(double **mat, int n, double epsilon) {
    double *x = (double *)malloc(n * sizeof(double));
    double *prevX = (double *)malloc(n * sizeof(double));
    int i,j;
    printf("Başlangıç değerlerini girin:\n");
    for ( i = 0; i < n; i++) {
        printf("x%d: ", i + 1);
        scanf("%lf", &x[i]);
        prevX[i] = x[i];
    }
    double error;
    int iter = 0;
    do {
        for ( i = 0; i < n; i++) {
            prevX[i] = x[i];          // hata hesabı için eski değerleri saklıyoruz
        }
        for ( i = 0; i < n; i++) {
            double sum = 0.0;
            for ( j = 0; j < n; j++) {
                if (j != i) {
                    sum += mat[i][j] * x[j];     // x1 hesaplanırkenmesela a2*x2+a3*x3 hesabı yapılıyor
                }
            }
            x[i] = (mat[i][n] - sum) / mat[i][i];// bulunan toplam istenilen xnin katsayısına bölünüyor xn hesabı bitiyor
        }
        error = 0.0;
        for ( i = 0; i < n; i++) {
            error += fabs(x[i] - prevX[i]); //tüm xnler için hata oranı toplanır
        }
        iter++;

        printf("İterasyon %d, Hata = %lf\n", iter, error); // Her iterasyon için hata değerini yazdır
    } while (error > epsilon && iter < MAX_ITER);

    if (iter == MAX_ITER) {
        printf("Yeterli yakınsama sağlanamadı.\n");
    }else{
        printf("Gauss-Seidel yöntemiyle %d iterasyonda sonuca ulaşıldı.\n", iter);
        sonucYazdir(x,n);
    }
    free(x);
    free(prevX);
}
```

Sayısal Türev Hesabı

$f(x) = e^{x^2} - 5ex + e^5$ and $h = 0.001$, estimate $f'(1)$,
using Two point Forward difference formula
Also find exact value of f', f'' and error for each estimation

Solution:

Equation is $f(x) = e^{x^2} - 5ex + e^5$.

$$\therefore f'(x) = 2e^{x^2}x - 5e$$

$$\therefore f''(x) = 4e^{x^2}x^2 + 2e^{x^2}$$

The value of table for x and y

x	0.999	1	1.001
y	137.5482	137.54	137.5319

Two-point FDF (Forward difference formula)

$$f'(x) = \frac{f(x+h) - f(x)}{h}$$

$$f'(1) = \frac{f(1+0.001) - f(1)}{0.001}$$

$$f'(1) = \frac{f(1.001) - f(1)}{0.001}$$

$$f'(1) = \frac{137.5319 - 137.54}{0.001}$$

$$f'(1) = -8.1467$$

```
Türevini bulmak istediğiniz fonksiyonu ve değerlerini giriniz:
Türevini bulmak istediğiniz fonksiyonu giriniz
e^(x^2)-5*x*e+e^5
Türevini bulmak istediğiniz değerini giriniz:1
h değerini giriniz: 0,001
Hangi yöntemi kullanacağınızı seçiniz.
1. İleri farkla
2. Geri farkla
3. Merkezi farkla
1
1,0000 değeri için türev -8,1467 bulundu.
```

İLERİ FARK

$f(x) = 2x^3 + x^2 - \frac{4}{2}x$ and $h = 0.01$, estimate $f'(1)$,
using Two point Backward difference formula
Also find exact value of f', f'' and error for each

Solution:

Equation is $f(x) = \frac{2x^3 + x^2 - 4}{2x}$.

$$\therefore f'(x) = \frac{4x^3 + x^2 + 4}{2(x^2)}$$

$$\therefore f''(x) = \frac{4x^4 - 8x}{2(x^4)}$$

The value of table for x and y

x	0.99	1	1.01
y	-0.5451	-0.5	-0.4551

Two-point BDF (Backward difference formula)

$$f'(x) = \frac{f(x) - f(x-h)}{h}$$

$$f'(1) = \frac{f(1) - f(1 - 0.01)}{0.01}$$

$$f'(1) = \frac{f(1) - f(0.99)}{0.01}$$

$$f'(1) = \frac{-0.5 - -0.5451}{0.01}$$

$$f'(1) = 4.5102$$

```
Türevini bulmak istediğiniz fonksiyonu ve değerlerini giriniz:
Türevini bulmak istediğiniz fonksiyonu giriniz
x^3-525
Türevini bulmak istediğiniz değerini giriniz:100
h değerini giriniz: 0,01
Hangi yöntemi kullanacağınızı seçiniz.
1. İleri farkla
2. Geri farkla
3. Merkezi farkla
3
100,0000 değeri için türev 30000,0001 bulundu.
```

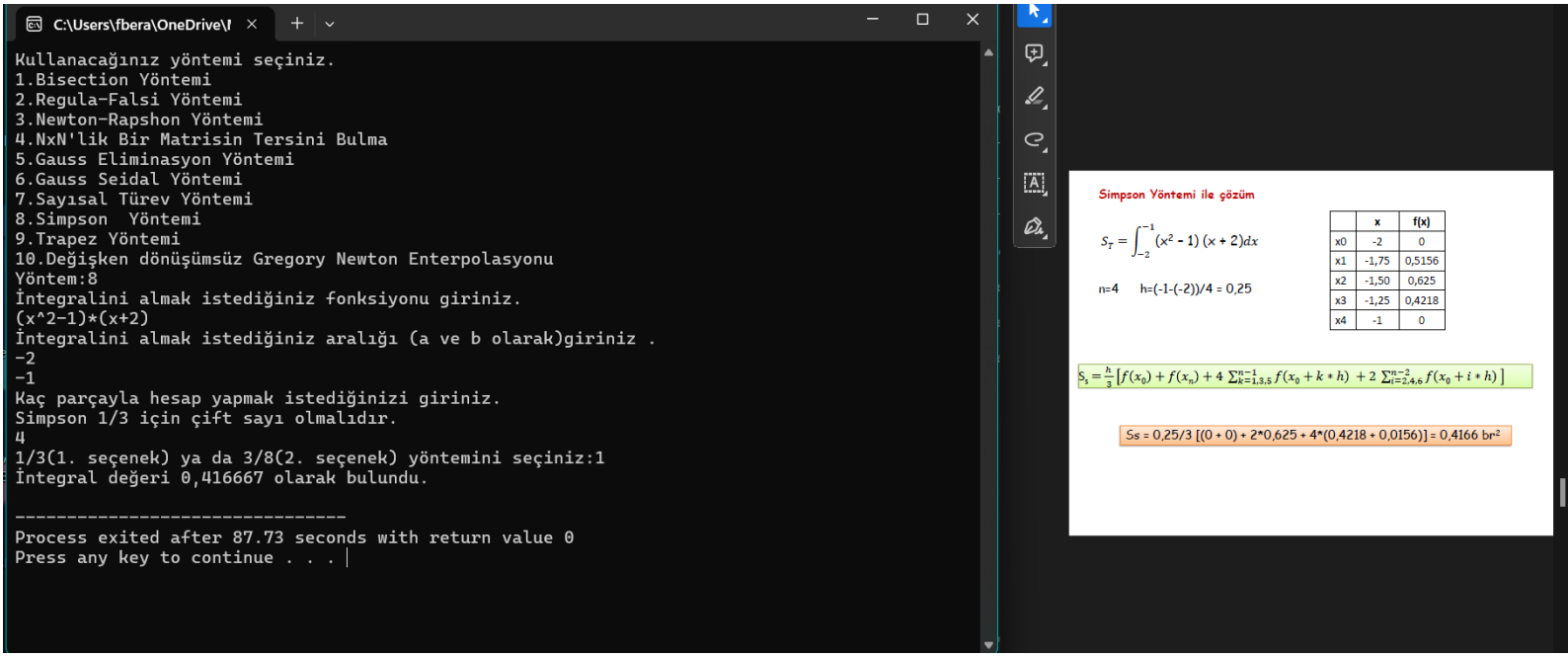
MERKEZİ FARK

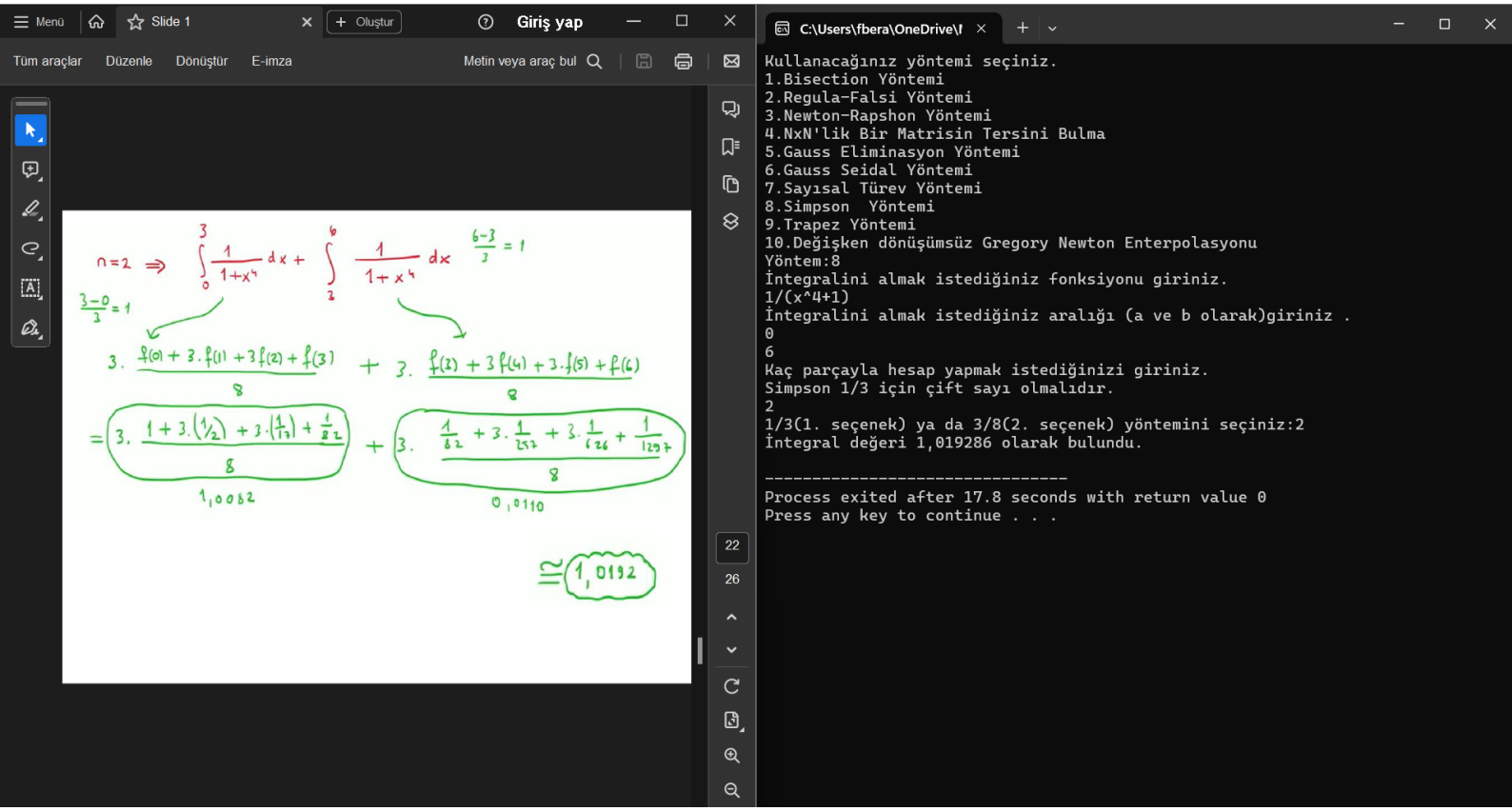
```
Türevini bulmak istediğiniz fonksiyonu ve değerlerini giriniz:
Türevini bulmak istediğiniz fonksiyonu giriniz
(2*x^3+x^2-4)/(2*x)
Türevini bulmak istediğiniz değerini giriniz:1
h değerini giriniz: 0,01
Hangi yöntemi kullanacağınızı seçiniz.
1. İleri farkla
2. Geri farkla
3. Merkezi farkla
2
1,0000 değeri için türev 4,5102 bulundu.
```

GERİ FARK

Simpson Yöntemleri

Örnekler





Kod Parçası

```
double simpson_1 (char* postfix, double a, double b,int n){
     double h= (b-a)/n;
     double c=a; int i;
     double fc, fa= evaluateExpression(postfix, a);
     double fb= evaluateExpression(postfix,b);
     double sum=fa+fb;
     for(i=1;i<n;i++){
          c+=h;
          fc=evaluateExpression(postfix,c);
          if(i%2==1){
               fc*=4;
          }else{
               fc*=2;
          }
          sum+=fc;
     }
     sum*= (h/3);
      free(postfix);
     return sum;
}
double simpson_2 (char* postfix, double a, double b,int n){
     double h=(b-a)/n; // h=6-0 /2=3
     double h2=h/3;          // h2=1
     int i,j;
     double a1=a;
     double b1,c;
     double sum1=0.0,sum=0.0;
     for(i=0;i<n;i++){
          a1=a1+ i*h; // a1=0        a1=3
          b1=a1+h;      //b1=3        b1=6
     double fa= evaluateExpression(postfix, a1);       //0       3
     double fc= evaluateExpression(postfix, a1+h2);    //1       4
     double fd= evaluateExpression(postfix, b1-h2);    //2       5
     double fb= evaluateExpression(postfix,b1);        //3       6
      sum=fa+3*fc+3*fd+fb;
      sum1+=sum;
     }
     sum1= sum1*(h/8);
     free(postfix);
     return sum1;

}
```

Trapez Yöntemi

```
Yöntem:9
İntegralini almak istediğiniz fonksiyonu giriniz.
0-(1/3)*x^3-(6/5)*x^2
İntegralini almak istediğiniz aralığı (a ve b olarak)giriniz .
1
2
Kaç parçayla hesap yapmak istediğinizi giriniz.
Simpson 1/3 için çift sayı olmalıdır.
6
İntegral değeri -4,062500 olarak bulundu.
```

Method-2:
Using Trapezoidal Rule

$$\int y dx = \frac{h}{2}\left[y_0 + 2y_1 + 2y_2 + 2y_3 + 2y_4 + 2y_5 + y_6\right]$$

$$y_0 = f(x_0) = f(1) = -1.53333333$$

$$2y_1 = 2f(x_1) = 2f(1.16666667) = 2 \cdot -2.16265432 = -4.32530864$$

$$2y_2 = 2f(x_2) = 2f(1.33333333) = 2 \cdot -2.92345679 = -5.84691358$$

$$2y_3 = 2f(x_3) = 2f(1.5) = 2 \cdot -3.825 = -7.65$$

$$2y_4 = 2f(x_4) = 2f(1.66666667) = 2 \cdot -4.87654321 = -9.75308642$$

$$2y_5 = 2f(x_5) = 2f(1.83333333) = 2 \cdot -6.08734568 = -12.17469136$$

$$y_6 = f(x_6) = f(2) = -7.46666667$$

$$\int y dx = \frac{0.16666667}{2} \cdot ((-1.53333333) + (-4.32530864) + (-5.84691358) + (-7.65) + (-9.75308642) + (-12.17469136) + (-7.46666667))$$

$$\int y dx = \frac{0.16666667}{2} \cdot (-48.75)$$

$$\int y dx = -4.0625$$

Solution by Trapezoidal Rule is -4.0625

```
double Trapez(char *postfix, double a,double b, double n){
     double h= (b-a)/n;
     int i;
     double fa= evaluateExpression(postfix,a);
     double fb= evaluateExpression(postfix,b);
     double sum=fa+fb;
     double c=a;
     for(i=1;i<n;i++){
          c+=h;
          sum+= 2*(evaluateExpression(postfix,c));
     }
     sum*= (h/2);
     free(postfix);
     return sum;
}
```

Değişken Dönüşümsüz Gregory-Newton Enterpolasyonu

Farkları hesaplamak için ayrı bir fonksiyonum var. Recursive bir şekilde çalışıyor. İstenen değeri bulurken de bu fonksiyonu çağırıyorum,her x değerinde (xi-x) ile bulunan fark (term)çarpılıyor ve sum değerine ekleniyor.

```c
// Farklari hesaplar
double calculateDifference(struct Data data[], int n, int i, int j) {
    if (j == 0)
        return data[i].y;
    else
        return (calculateDifference(data, n, i + 1, j - 1) - calculateDifference(data, n, i, j - 1)) / (data[i + j].x - data[i].x);
}
void gregoryNewton(struct Data data[], int n, double xi) {
    double result = 0;
    int i,j;
    for (i = 0; i < n; i++) {
        double term = calculateDifference(data, n, 0, i);
        for ( j = 0; j < i; j++) {
            term *= (xi - data[j].x);
        }
        result += term;
    }
    printf("%.2lf için hesaplanan enterpolasyon değeri= %.4lf", xi,result);
}
```

```
Yöntem:10
Kaç tane veri gireceğinizi belirtiniz: 5

Değerleri giriniz:
x[0] = 1
y[0] = 3
x[1] = 2
y[1] = 6
x[2] = 3
y[2] = 5
x[3] = 4
y[3] = 7
x[4] = 5
y[4] = 9
Enterpolasyon hesaplaması yapılacak değeri giriniz:3,5
3,50 için hesaplanan enterpolasyon değeri= 5,5781
```

InfixtoPostfix Fonksiyonu

Shunting Yard algoritmasını kullanarak stringi hesaplanabilir hale getiriyorum. Stack yapısına ihtiyacım vardı c dilinde oldığu için fonksiyon yazmak gerekti :)

Ayrıca işlemlerin önceliğini bulmak ve kullanabilmek için precedence fonksiyonu, değişken yani x olduğunu kontrol etmek için isOperand, sayı olup olmadığını kontrol için de isInteger fonksiyonları var. Parametre olarak char alıyorlar.

```c
struct Stack // Structure to
{
    int top;
    int totalSize;
    char *arrayStack;
};
```

```c
void initializeStack(struct Stack *stack, int size) {        //infixtopostfix için
    stack->totalSize = size;
    stack->top = -1;
    stack->arrayStack = (char *)malloc(stack->totalSize * sizeof(char));
}
void destroyStack(struct Stack *stack) {
    free(stack->arrayStack);
}
int isEmpty(struct Stack *stack) {
    return stack->top == -1;
}
int isFull(struct Stack *stack) {
    return stack->top == stack->totalSize - 1;
}
char peek(struct Stack *stack) {
    if (isEmpty(stack))
        return -1;
    return stack->arrayStack[stack->top];
}
char pop(struct Stack *stack) {
    if (isEmpty(stack))
        return -1;
    return stack->arrayStack[stack->top--];
}
void push(char element, struct Stack *stack) {
    if (isFull(stack)) {
        printf("Stack is already Full.");
        return;
    }
    stack->arrayStack[++stack->top] = element;
}
int precedence(char ch) {
    switch (ch) {
    case '+':
    case '-':
        return 1;
    case '*':
    case '/':
        return 2;
    case '^':
        return 3;
    }
    return -1;
}
int isOperand(char element) {
    return (element >= 'A' && element <= 'Z') || (element >= 'a' && element <= 'z');
}
int isInteger(char element){
    return (element >= '0' && element <= '9');
}
```

```c
char* infixToPostfix(char *expressionArray, struct Stack *stack) {
    initializeStack(stack, strlen(expressionArray));  // Initialize stack
    int postfixSize = strlen(expressionArray) * 2;     // Initial estimate of postfix size
    char *postfix = (char *)malloc(postfixSize * sizeof(char)); // Allocate memory for postfix expression
    int postfixIndex = 0; // Index for postfix expression
    int currIndex = 0;

    while (expressionArray[currIndex] != '\0') {
        if (isOperand(expressionArray[currIndex]) || isInteger(expressionArray[currIndex])){
            postfix[postfixIndex++] = expressionArray[currIndex++];
            while (isOperand(expressionArray[currIndex]) || isInteger(expressionArray[currIndex])) {
                postfix[postfixIndex++] = expressionArray[currIndex++];
            }
            postfix[postfixIndex++] = ' ';     // Insert space after each operand
        } else if (expressionArray[currIndex] == '(') {
            push(expressionArray[currIndex], stack);
            currIndex++;
        } else if (expressionArray[currIndex] == ')') {
            while (peek(stack) != '('){
                postfix[postfixIndex++] = pop(stack);
                postfix[postfixIndex++]=' ';
            }

            pop(stack); // Pop '('
            currIndex++;
        } else {
            while (!isEmpty(stack) && precedence(peek(stack)) >= precedence(expressionArray[currIndex])){
                postfix[postfixIndex++]= pop(stack);
                postfix[postfixIndex++]=' ';
            }

            push(expressionArray[currIndex], stack);
            currIndex++;
        }
    }

    // Pop remaining operators from stack
    while (!isEmpty(stack)){
        postfix[postfixIndex++] = pop(stack);
        postfix[postfixIndex++]=' ';
    }

    postfix[postfixIndex] = '\0'; // Null-terminate the string
    destroyStack(stack); // Deallocate memory used by stack
    return postfix;
}
```

```
Fonksiyonu giriniz
x^3-5*x+1/(5*x)-2
Enter the value of x: 2
Postfix Expression: x 3 ^ 5 x * - 1 5 x * / + 2 -
Result for 2,000000= -3,900000
```

Fonksiyonun İşleyişi

1. **Yığın Başlatma:** İfade uzunluğuna bağlı olarak yığın başlatılır.
2. **Postfix Belleği Ayırma:** Postfix ifadesi için tahmini bir boyut belirlenerek bellek ayırılır.
3. **İfade İşleme:** İfade karakter karakter işlenir:
    - Operand (x) veya sayı ise, postfix ifadesine eklenir.
    - **HER POSTFİXE EKLENEN ELEMANDAN SONRA BİR BOŞLUK BIRAKILIR.**
    - Parantezler açılıp kapanırken uygun işlemler yapılır.
        - İfadedeki bir açma parantezi '(' bulunduğunda, bu parantez yığına eklenir.
        - Bir kapalı parantez ')' bulunduğunda, bu parantez yığındaki en yakın açılış parantezi '(' ile eşleşir.
        - Bu durumda, yığındaki açma parantezine kadar olan tüm operatörler yığından çıkarılır ve postfix ifadesine eklenir.
        - Açma parantezi '(' yığından çıkarılır ancak postfix ifadesine eklenmez.
    - Operatörler yığına eklenir veya yığından postfix ifadesine aktarılır.
4. **Yığının Boşaltılması:** İfadeyi işledikten sonra yığında kalan operatörler postfix ifadesine eklenir.
5. **Bellek Serbest Bırakma:** Yığın için ayrılan bellek serbest bırakılır.

```c
typedef struct {
    int top;
    int totalSize;
    double *stack;
} stack2;

void initializeStack2(stack2 *s, int size) {     //evaluation için
    s->top = -1;
    s->totalSize = size;
    s->stack = (double *)calloc(s->totalSize, sizeof(double));
    if (s->stack == NULL) {
        printf("Memory allocation failed\n");
        exit(EXIT_FAILURE);
    }
}
void push2(stack2 *s, double item) {
    if (s->top >= s->totalSize - 1) {
        printf("Stack Overflow\n");
        exit(EXIT_FAILURE);
    }
    s->top++;
    s->stack[s->top] = item;
}
double pop2(stack2 *s) {
    if (s->top < 0) {
        printf("Stack Underflow\n");
        exit(EXIT_FAILURE);
    }
    double item = s->stack[s->top];
    s->top--;
    return item;
}
int is_operator(char symbol) {
    return (symbol == '+' || symbol == '-' || symbol == '*' || symbol == '/' || symbol == '^');
}
```

Hesaplama yaparken de yığın yapısına ihtiyacım var. Bu yığında array double değerleri tutuyor sadece. İşlemleri stringde rastlayınca hemen hesaplıyoruz, yığında tutmaya gerek yok. Ama işlemleri kontrol etmek için is_operator diye bir fonksiyon kullandık.

Önceki sayfada boşluk bıraktığımızı özellikle belirtmiştim, çünkü burada fonksiyon postfix stringi strtok fonksiyonu kullanarak ayırıyor. Boşluk görene kadarki stringi token'e atadıktan sonra gerekli kontroller yapılıp sayıysa float değerine çevirip yığına koyarken işlem ise yığındaki son iki float değerini pop edip hesaplama yapıp bulunan değeri yine yığına push ediyoruz.

```c
double evaluate(char* expression, double x) {
    stack2 s;
    initializeStack2(&s, strlen(expression));
    double operand1, operand2, result;
    char *token = strtok(expression, " ");
    while (token!= NULL) {
        if (isdigit(*token)) {
            // If it's a number, push it onto the stack
            push2(&s, atof(token));
        } else if (*token == 'x') {
            // If it's 'x', push the value of x onto the stack
            push2(&s, x);
        } else if(*token == 'e'){// if it is e  push the yaklaşık değer
            push2(&s, 2.718281828459);
        }   else if (is_operator(*token)) {
            // If it's an operator, pop operands from the stack, perform the operation, and push the result onto the stack
            operand2 = pop2(&s);
            operand1 = pop2(&s);
            switch (*token) {
                case '+': result = operand1 + operand2; break;
                case '-': result = operand1 - operand2; break;
                case '*': result = operand1 * operand2; break;
                case '/': if (operand2 == 0) {
                                printf("Division by zero error\n");
                                exit(EXIT_FAILURE);
                            }
                            result = operand1 / operand2; break;
                case '^': result = pow(operand1, operand2); break;
                default: printf("Unknown operator: %c\n", *token); exit(EXIT_FAILURE);
            }
            push2(&s, result);
        } else {
            printf("Unknown token: %s\n", token);
            exit(EXIT_FAILURE);
        }
        token = strtok(NULL, " ");
    }
    return result;
}
double evaluateExpression(char* postfix, double x) {
    char* postfix_copy = strdup(postfix); // Make a copy of postfix cuz strtok ruins the string
    if (!postfix_copy) {
        printf("Memory allocation failed\n");
        exit(EXIT_FAILURE);
    }
    double result = evaluate(postfix_copy, x);
    free(postfix_copy); // Free the copied postfix expression
    return result;
}
```

Strtok fonksiyonu kullanıldığı için string yapısı bozuluyor o yüzden her seferinde postfix kopyalayıp evaluate fonksiyonunu evaluateExpression içinde hesaplama yapmamız lazım.