YILDIZ TEKNİK ÜNİVERSİTESİ

ELEKTRİK-ELEKTRONİK FAKÜLTESİ

Bilgisayar Mühendisliği Bölümü

SAYISAL ANALİZ DÖNEM PROJESİ

BLM1022 – Sayısal Analiz, Grup: 2

Ders Sorumlusu:

Öğr. Gör. Dr. Ahmet ELBİR

Hazırlayan:

İsim: ARINÇ AYDEMİR

No: 21011013

E-posta: arinc.aydemir@std.yildiz.edu.tr

İçindekiler

1 Ön Bilgi

Program, 10 tane belirli işlemi yerine getirebilmek için tasarlanmıştır. Bu işlemler sırasıyla şöyledir:

1. Bisection yöntemi
2. Regula-Falsi yöntemi
3. Newton-Raphson 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

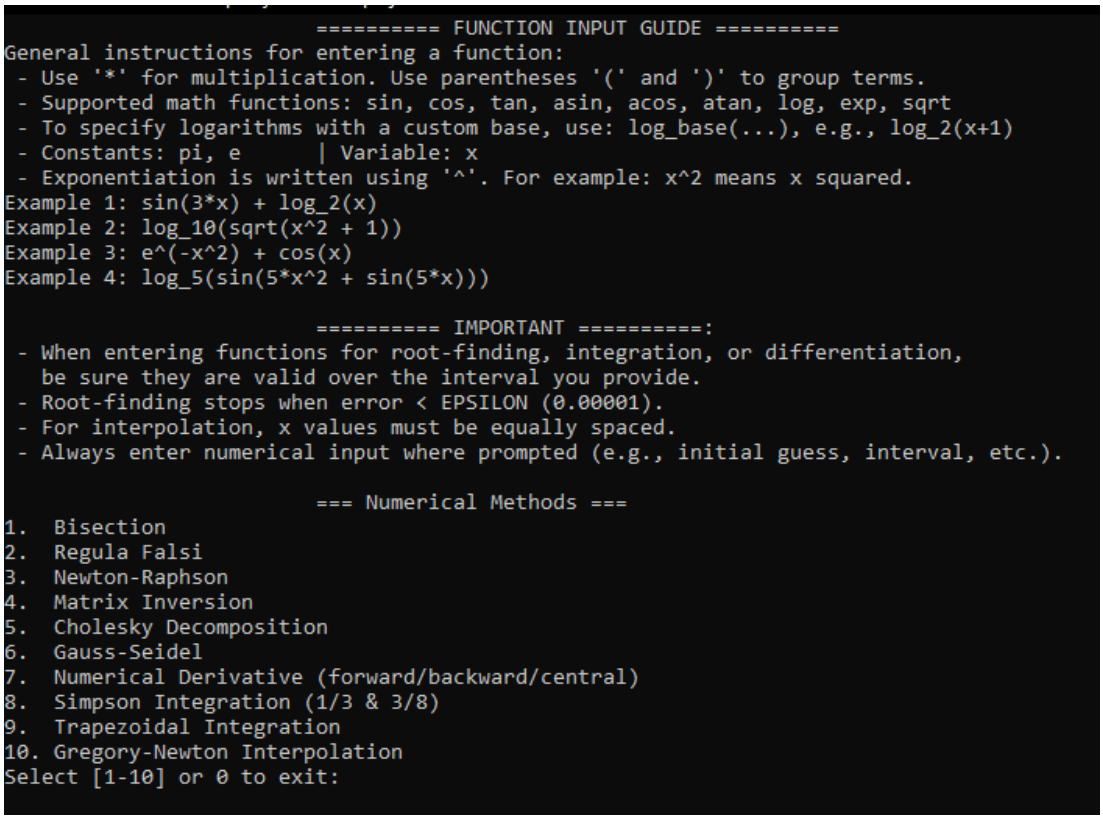
1.1 Değerlendirme Tablosu

YÖNTEMLERİN YAPILIP YAPILMADIĞINI AŞAĞIDAKİ TABLODA GÖSTERİLDİĞİ GİBİ 1/0 OLARAK GÖSTERİNİZ									
1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1

2 Ana Menü

Çalıştırılmak istenilen işlem, program çalıştırıldıktan sonra numarası girilip gereken parametrelerin verilmesiyle çalışır. Ana menüde ‘0’ girdisi verilene kadar program çalışmaya devam eder.

2.1 Ekran Çıktısı



3 Fonksiyon Formatı Açıklaması

Kullanıcıdan alınan matematiksel ifadeler, `char[]` türünde dizelere yazılır ve daha sonra bir parser fonksiyonu tarafından değerlendirilir. Kod, kullanıcıdan alınan bu fonksiyonları çözümlemek için ön işlem, token ayrıştırma ve değerlendirme işlemleri uygular.

3.1 Desteklenen Semboller ve Kurallar

- Çarpma işlemi açıkça belirtilmelidir. Örn: `3*x`, `2*sin(x)`
- Parantezler mutlaka doğru kullanılmalıdır. Örn: `(x+1)*(x-2)`
- Üs alma işlemi için `^` operatörü kullanılır. Örn: `x^2`
- Logaritma işlemleri için:
  - Doğal logaritma: `log(x)` $\equiv \ln(x)$
  - Üs logaritmalar: `log_2(x+1)` $\equiv \frac{\log(x+1)}{\log(2)}$
- Desteklenen sabitler:
  - `pi` ≈ 3.14159
  - `e` ≈ 2.71828
- Desteklenen fonksiyonlar:
  - `sin(x)`
  - `cos(x)`
  - `tan(x)`
  - `asin(x)`
  - `acos(x)`
  - `atan(x)`
  - `log(x)`
  - `exp(x)`
  - `sqrt(x)`

3.2 Geçerli Örnekler

Aşağıdaki örnekler kullanıcı tarafından doğrudan programa giriş olarak verilebilir:

- `sin(3*x) + log_2(x)` $\rightarrow \sin(3x) + \log_2(x)$
- `log_10(sqrt(x^2 + 1))` $\rightarrow \log_{10}(\sqrt{x^2 + 1})$
- `e^(-x^2) + cos(x)` $\rightarrow e^{-x^2} + \cos(x)$
- `log_5(sin(5*x^2 + sin(5*x)))` $\rightarrow \log_5(\sin(5x^2 + \sin(5x)))$
- `x^(sin(log_5(x^3)))` $\rightarrow x^{\sin(\log_5(x^3))}$

4 Bisection Yöntemi

4.1 Parametreler

Fonksiyon

start: Başlangıç değeri

end: Bitiş değeri

epsilon: Hata miktarı

Durma koşulu:

- $|f(c)| < \epsilon$
- veya $(b - a) < \epsilon$

Maksimum iterasyon: Kodda `max_iter = 20` ile sınırlandırılmıştır.

4.2 Örnek: $f(x) = x^3 - 7x^2 + 14x - 6$ çözümü

start: 0 **end:** 1 **epsilon:** 0.00001

```
Enter function f(x):
x^3-7*x^2+14*x-6
Enter interval [a b]:
 a = 0
 b = 1

Iter    a          b          c          f(a)       f(b)       |b-a|
0       0.000000   1.000000   0.500000   -6.000000  2.000000   1.000000
1       0.500000   1.000000   0.750000   -0.625000  2.000000   0.500000
2       0.500000   0.750000   0.625000   -0.625000  0.984375   0.250000
3       0.500000   0.625000   0.562500   -0.625000  0.259766   0.125000
4       0.562500   0.625000   0.593750   -0.161865  0.259766   0.062500
5       0.562500   0.593750   0.578125   -0.161865  0.054047   0.031250
6       0.578125   0.593750   0.585938   -0.052624  0.054047   0.015625
7       0.578125   0.585938   0.582031   -0.052624  0.001031   0.007813
8       0.582031   0.585938   0.583984   -0.025716  0.001031   0.003906
9       0.583984   0.585938   0.584961   -0.012322  0.001031   0.001953
10      0.584961   0.585938   0.585449   -0.005640  0.001031   0.000977
11      0.585449   0.585938   0.585693   -0.002303  0.001031   0.000488
12      0.585693   0.585938   0.585815   -0.000636  0.001031   0.000244
13      0.585693   0.585815   0.585754   -0.000636  0.000198   0.000122
14      0.585754   0.585815   0.585785   -0.000219  0.000198   0.000061
15      0.585785   0.585815   0.585800   -0.000010  0.000198   0.000031
16      0.585785   0.585800   0.585793   -0.000010  0.000094   0.000015

Root approximation: 0.585793
```

5 Regula-Falsi Yöntemi

5.1 Parametreler

Fonksiyon

start: Başlangıç değeri

end: Bitiş değeri

epsilon: Hata miktarı

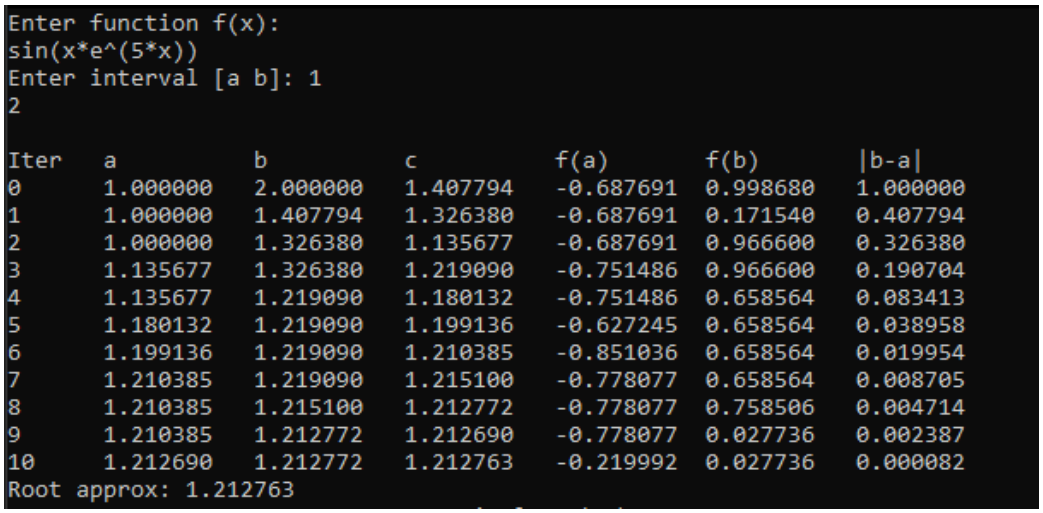
Durma koşulu:

- $|f(c)| < \epsilon$
- veya $(b - a) < \epsilon$

Maksimum iterasyon: Kodda `max_iter = 20` ile sınırlandırılmıştır.

5.2 Örnek: $f(x) = \sin(x \cdot e^{5x})$ çözümü

start: 1 **end:** 2 **epsilon:** 0.00001



```
Enter function f(x):
sin(x*e^(5*x))
Enter interval [a b]: 1
2

Iter    a          b          c          f(a)       f(b)       |b-a|
0       1.000000   2.000000   1.407794   -0.687691  0.998680   1.000000
1       1.000000   1.407794   1.326380   -0.687691  0.171540   0.407794
2       1.000000   1.326380   1.135677   -0.687691  0.966600   0.326380
3       1.135677   1.326380   1.219090   -0.751486  0.966600   0.190704
4       1.135677   1.219090   1.180132   -0.751486  0.658564   0.083413
5       1.180132   1.219090   1.199136   -0.627245  0.658564   0.038958
6       1.199136   1.219090   1.210385   -0.851036  0.658564   0.019954
7       1.210385   1.219090   1.215100   -0.778077  0.658564   0.008705
8       1.210385   1.215100   1.212772   -0.778077  0.758506   0.004714
9       1.210385   1.212772   1.212690   -0.778077  0.027736   0.002387
10      1.212690   1.212772   1.212763   -0.219992  0.027736   0.000082
Root approx: 1.212763
```

6 Newton-Raphson Yöntemi

6.1 Parametreler

Fonksiyon

initial guess: Başlangıç tahmini (x_0)

epsilon: Hata miktarı

Durma koşulu:

- $|x_{n+1} - x_n| < \epsilon$

Türev hesaplama: Sayısal türev kullanılır:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}, \quad h = 10^{-4}$$

Maksimum iterasyon: Kodda `max_iter = 20` olarak sınırlandırılmıştır.

6.2 Örnek: $f(x) = x^{\sin(\log_5(x^3))} - 0.5$ çözümü

initial guess: 0.5 **epsilon:** 0.00001

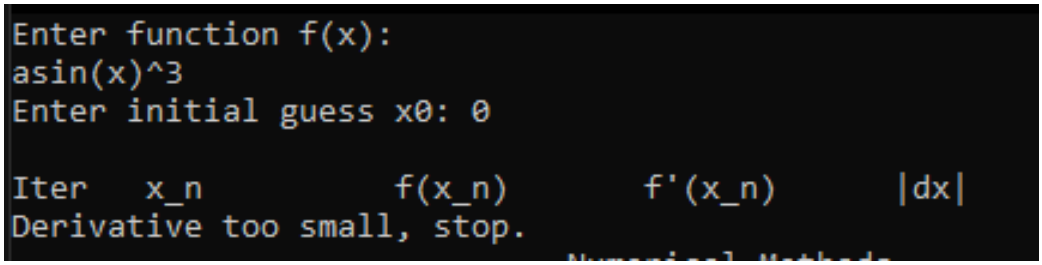
```
Enter function f(x):
x^(sin(log_5(x^3))) - 0.5
Enter initial guess x0: 0.5

Iter   x_n          f(x_n)       f'(x_n)      |dx|
0      0.500000     1.447193     -5.128607    0.282180
1      0.782180     0.614726     -1.215428    0.505769
2      1.287949     0.621856     0.761788     0.816311
3      0.471639     1.597442     -5.436579    0.293832
4      0.765471     0.636219     -1.358850    0.468204
5      1.233675     0.583414     0.652807     0.893702
6      0.339974     2.153841     -0.371515    5.797449
7      6.137422     0.149137     -0.372607    0.400253
8      6.537675     0.017726     -0.287326    0.061692
9      6.599367     0.000355     -0.275897    0.001286
10     6.600653     0.000000     -0.275663    0.000001
Root approx: 6.600654
```

6.3 Örnek: $f(x) = \arcsin(x)^3$ – Türev çok küçük hatası

initial guess: 0 **epsilon:** 0.00001

Fonksiyonun $f(x) = \arcsin(x)^3$ türevi, $x = 0$ civarında oldukça küçük değerlere sahiptir. Bu nedenle sayısal türev $|f'(x)| < \epsilon$ şartını sağlayamaz ve algoritma "`Derivative too small, stop.`" mesajı ile durur.



6.4 Örnek: $f(x) = x^3 - 2x + 2$ – Maksimum iterasyon limiti

initial guess: 0 **epsilon:** 0.00001

max iteration: 20

$$f(x) = x^3 - 2x + 2$$

Bu fonksiyonun yaklaşık kökü -1.769 civarında yer alıyor. Newton–Raphson yöntemi bazen belirli bir başlangıç noktasından hareket edip, gerçek çözümün etrafındaki "çekim bölgesi"ne giremeden, kendini iki farklı değerin arasında gidip gelir ama gerçek köke yakınsamak yerine kısır bir döngüye hapsolur

```
Enter function f(x):
x^3 - 2*x + 2
Enter initial guess x0: 0

Iter    x_n          f(x_n)       f'(x_n)      |dx|
0       0.000000     2.000000     -2.000000    1.000000
1       1.000000     1.000000     1.000000     1.000000
2       0.000000     2.000000     -2.000000    1.000000
3       1.000000     1.000000     1.000000     1.000000
4       0.000000     2.000000     -2.000000    1.000000
5       1.000000     1.000000     1.000000     1.000000
6       0.000000     2.000000     -2.000000    1.000000
7       1.000000     1.000000     1.000000     1.000000
8       0.000000     2.000000     -2.000000    1.000000
9       1.000000     1.000000     1.000000     1.000000
10      0.000000     2.000000     -2.000000    1.000000
11      1.000000     1.000000     1.000000     1.000000
12      0.000000     2.000000     -2.000000    1.000000
13      1.000000     1.000000     1.000000     1.000000
14      0.000000     2.000000     -2.000000    1.000000
15      1.000000     1.000000     1.000000     1.000000
16      0.000000     2.000000     -2.000000    1.000000
17      1.000000     1.000000     1.000000     1.000000
18      0.000000     2.000000     -2.000000    1.000000
19      1.000000     1.000000     1.000000     1.000000
Max iteration limit reached without convergence.
                              Numerical Methods
```

7 Kare Matrisin Tersi

7.1 Parametreler

Matris boyutu: n

Katsayı matrisi: $A[n][n]$

Durma koşulu: Eğer herhangi bir satırdaki pivot sıfır veya çok küçükse işlem durdurulur ve kullanıcıya hata mesajı basılır.

Maksimum iterasyon: Yoktur.

7.2 Örnek: Tersi alınabilen 4×4 matris

n: 4

$$A = \begin{bmatrix} 4 & 2 & 3 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \\ 5 & 1 & 0 & 2 \end{bmatrix}$$

```
Enter matrix size n: 4
Enter matrix A elements:
Enter element A[0][0]: 4
Enter element A[0][1]: 2
Enter element A[0][2]: 3
Enter element A[0][3]: 1
Enter element A[1][0]: 1
Enter element A[1][1]: 1
Enter element A[1][2]: 1
Enter element A[1][3]: 1
Enter element A[2][0]: 0
Enter element A[2][1]: 1
Enter element A[2][2]: 2
Enter element A[2][3]: 3
Enter element A[3][0]: 5
Enter element A[3][1]: 1
Enter element A[3][2]: 0
Enter element A[3][3]: 2
Inverse matrix:
  0.166667   -0.500000    0.000000    0.166667
 -0.500000    2.833333   -0.666667   -0.166667
  0.500000   -1.166667    0.333333   -0.166667
 -0.166667   -0.166667    0.333333    0.166667
```

Bu örnekte verilen 4×4 matris terslenebilirdir.

7.3 Örnek: Tersi alınamayan 3×3 matris

n: 3

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
Enter matrix size n: 3
Enter matrix A elements:
Enter element A[0][0]: 1
Enter element A[0][1]: 2
Enter element A[0][2]: 3
Enter element A[1][0]: 4
Enter element A[1][1]: 5
Enter element A[1][2]: 6
Enter element A[2][0]: 7
Enter element A[2][1]: 8
Enter element A[2][2]: 9
[ERROR] Matrix is singular: pivot at row 2 is zero. Cannot compute inverse.
```

Bu örnekte determinant sıfır olduğundan matrisin tersi alınamaz. Kod kullanıcıyı bilgilendirir ve işlem durdurulur.

8 Cholesky (ALU) Yöntemi

8.1 Parametreler

Matris boyutu: n

Katsayı matrisi: $A[n][n]$

Sabitler vektörü: $c[n]$

Durma koşulu: Belirtilmemiştir, çözüm LU ayrıştırması ile iki aşamalı olarak hesaplanır.

8.2 Örnek: 3×3 sistem çözümü

n: 3

$$A = \begin{bmatrix} 3.6 & 2.4 & -1.8 \\ 4.2 & -5.8 & 2.1 \\ 0.8 & 3.5 & 6.5 \end{bmatrix}, \quad c = \begin{bmatrix} 6.3 \\ 7.5 \\ 3.7 \end{bmatrix}$$

```
Enter matrix size n: 3
Enter matrix A elements:
Enter element A[0][0]: 3.6
Enter element A[0][1]: 2.4
Enter element A[0][2]: -1.8
Enter element A[1][0]: 4.2
Enter element A[1][1]: -5.8
Enter element A[1][2]: 2.1
Enter element A[2][0]: 0.8
Enter element A[2][1]: 3.5
Enter element A[2][2]: 6.5

Now enter the constants vector c:
 You're entering c[0]: 6.3
 You're entering c[1]: 7.5
 You're entering c[2]: 3.7

L matrix:
    3.6000     0.0000     0.0000
    4.2000    -8.6000     0.0000
    0.8000     2.9667     8.3488

U matrix:
    1.0000     0.6667    -0.5000
    0.0000     1.0000    -0.4884
    0.0000     0.0000     1.0000

Forward (y values):
y[1] = 1.7500
y[2] = -0.0174
y[3] = 0.2817

Backward (solution x):
x[1] = 1.8108
x[2] = 0.1201
x[3] = 0.2817
```

Katsayılar matrisi biri alt üst üçgen diğeri üst üçgen olan iki ayrı matrise ayrıştırılır ve çözüme ulaşılır

9 Gauss-Seidel Yöntemi

9.1 Parametreler

Matris boyutu: n

Katsayı matrisi: $A[n][n]$

Sabitler vektörü: $c[n]$

İlk tahmin: $x_0[n]$

Tolerans: ϵ

Maksimum iterasyon: Belirtilen sınır

Durma koşulu: $\max(|dx_i|) < \epsilon$ sağlandığında işlem sonlandırılır.

9.2 Örnek: Gauss-Seidel ile 3×3 sistem çözümü

n: 3

$$A = \begin{bmatrix} 3 & 1 & -2 \\ -1 & 4 & -3 \\ 1 & -1 & 4 \end{bmatrix}, \quad c = \begin{bmatrix} 9 \\ -8 \\ 1 \end{bmatrix}, \quad x_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Tolerans: 0.001

Maksimum iterasyon: 15

```
Enter the number of variables (system size n): 3

Now enter the coefficient matrix A (3x3):
  Enter A[0][0]: 3
  Enter A[0][1]: 1
  Enter A[0][2]: -2
  Enter A[1][0]: -1
  Enter A[1][1]: 4
  Enter A[1][2]: -3
  Enter A[2][0]: 1
  Enter A[2][1]: -1
  Enter A[2][2]: 4

Now enter the constants vector c:
 You're entering c[0]: 9
 You're entering c[1]: -8
 You're entering c[2]: 1

Now enter initial guesses for the unknowns (x1 to x3):
for x1: 1
for x2: 1
for x3: 1

Enter the desired tolerance (e.g., 0.001): 0.001
Note: Iterations will stop early if the maximum change in variables is below the tolerance (0.001)

Enter the maximum number of iterations: 15

Iter       x1         |dx1 |          x2         |dx2 |          x3         |dx3 |
-------------------------------------------------------------------------------------
    1     1.000000     -            1.000000     -            1.000000     -
    2     3.333333      2.333333    -0.416667     1.416667    -0.687500     1.687500
    3     2.680556      0.652778    -1.845486     1.428819    -0.881510     0.194010
    4     3.027488      0.346933    -1.904261     0.058775    -0.982937     0.101427
    5     2.979462      0.048026    -1.992337     0.088077    -0.992950     0.010013
    6     3.002146      0.022684    -1.994176     0.001838    -0.999080     0.006131
    7     2.998672      0.003474    -1.999642     0.005466    -0.999579     0.000498
    8     3.000162      0.001490    -1.999643     0.000001    -0.999951     0.000373
    9     2.999914      0.000248    -1.999985     0.000342    -0.999975     0.000023

Final Solution (x vector):
  x1 =   2.999914
  x2 =  -1.999985
  x3 =  -0.999975
```

Her iterasyon adımında her değişken için bulunan en son değer kullanılır Her adımda toleransa göre yakınsama kontrolü yapılır. Tolerans sağlandığında işlem durur.

9.3 Ek Örnek: Köşegen Dominant Hale Getirme ile Gauss-Seidel

n: 3

$$A = \begin{bmatrix} -1 & 4 & -3 \\ 3 & 1 & -2 \\ 1 & -1 & 4 \end{bmatrix}, \quad c = \begin{bmatrix} -8 \\ 9 \\ 1 \end{bmatrix}, \quad x_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Tolerans: 0.001

Maksimum iterasyon: 15

```
Enter the number of variables (system size n): 3

Now enter the coefficient matrix A (3x3):
  Enter A[0][0]: -1
  Enter A[0][1]: 4
  Enter A[0][2]: -3
  Enter A[1][0]: 3
  Enter A[1][1]: 1
  Enter A[1][2]: -2
  Enter A[2][0]: 1
  Enter A[2][1]: -1
  Enter A[2][2]: 4

Now enter the constants vector c:
 You're entering c[0]: -8
 You're entering c[1]: 9
 You're entering c[2]: 1

Now enter initial guesses for the unknowns (x1 to x3):
for x1: 1
for x2: 1
for x3: 1

Enter the desired tolerance (e.g., 0.001): 0.001
Note: Iterations will stop early if the maximum change in variables is below the tolerance (0.0

Enter the maximum number of iterations: 15

Iter       x1         |dx1 |          x2         |dx2 |          x3         |dx3 |
-------------------------------------------------------------------------------------
    1     1.000000     -             1.000000     -             1.000000     -
    2     3.333333      2.333333    -0.416667      1.416667    -0.687500      1.687500
    3     2.680556      0.652778    -1.845486      1.428819    -0.881510      0.194010
    4     3.027488      0.346933    -1.904261      0.058775    -0.982937      0.101427
    5     2.979462      0.048026    -1.992337      0.088077    -0.992950      0.010013
    6     3.002146      0.022684    -1.994176      0.001838    -0.999080      0.006131
    7     2.998672      0.003474    -1.999642      0.005466    -0.999579      0.000498
    8     3.000162      0.001490    -1.999643      0.000001    -0.999951      0.000373
    9     2.999914      0.000248    -1.999985      0.000342    -0.999975      0.000023

Final Solution (x vector):
  x1 =   2.999914
  x2 =  -1.999985
  x3 =  -0.999975
```

Bu örnekte sistem başlangıçta diyagonal baskın değildir. Kod satır değişimi yaparak köşegenlerdeki değerleri baskın hale getirir ve çözümün yakınsamasını kolaylaştırır.

10 Sayısal Türev

10.1 Parametreler

Fonksiyon: $f(x)$

Nokta: x değerinde türev alınır

Adım boyutu: h (küçük bir reel sayı, örneğin 0.0001)

Durma koşulu: $f(x+h)$ veya $f(x-h)$ değeri tanımsız veya karmaşık hale gelirse işlem durdurulur ve kullanıcıya hata mesajı gösterilir.

10.2 Örnek: $f(x) = e^{-x} \cdot \cos(x)$ için türev

Nokta: $x = 0.75$ **Adım boyutu:** $h = 0.25$

```
Enter function f(x):
e^(-x)*cos(x)
Point x: 0.75
Step size h (0.00001): 0.25

Numerical derivative at x = 0.750000
  Forward  diff : -0.587437
  Backward diff : -0.746622
  Central  diff : -0.667029
```

10.3 Örnek: $f(x) = \sqrt{x}$ – Tanımsız türev durumu

Nokta: $x = 0$ **Adım boyutu:** $h = 0.1$

```
Enter function f(x):
sqrt(x)
Point x: 0
Step size h (0.00001): 0.1
Complex or undefined value encountered in numerical_derivative, aborting.
```

$f(x) = \sqrt{x}$ ve $x = 0$ noktasında türev alınmak istenmiş. Sayısal türev hesaplanırken kullanılan formül, $f(x+h)$ ve $f(x-h)$ gibi iki değeri kullanıyor. Burada $h = 0.1$ verilmiş, yani $f(x - h) = f(-0.1)$ değeri hesaplanmaya çalışılıyor. Ama $f(-0.1) = \sqrt{-0.1}$ olduğundan karmaşık sayı oluşur ve kod durdurulup ve hata bastırılır

11 Simpson Yöntemleri (1/3 ve 3/8)

11.1 Parametreler

Fonksiyon: $f(x)$

Alt sınır: a **Üst sınır:** b **Parça sayısı:** n

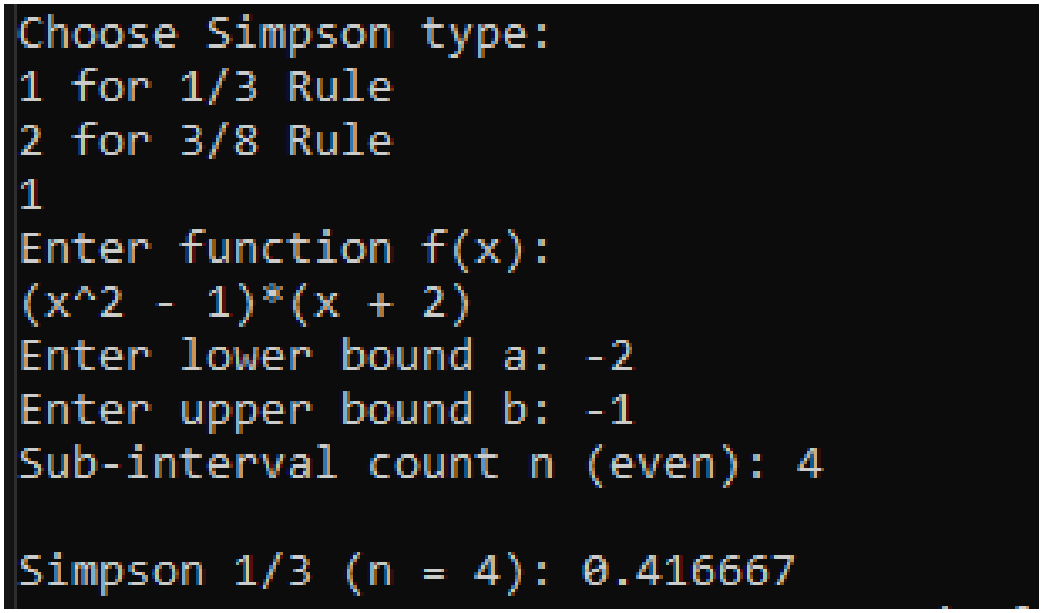
Simpson 1/3 yöntemi için n çift sayı olmalıdır.

Simpson 3/8 yöntemi için n değeri 3'ün katı olmalıdır.

Durma koşulu: Geçersiz n değeri girildiğinde işlem iptal edilir.

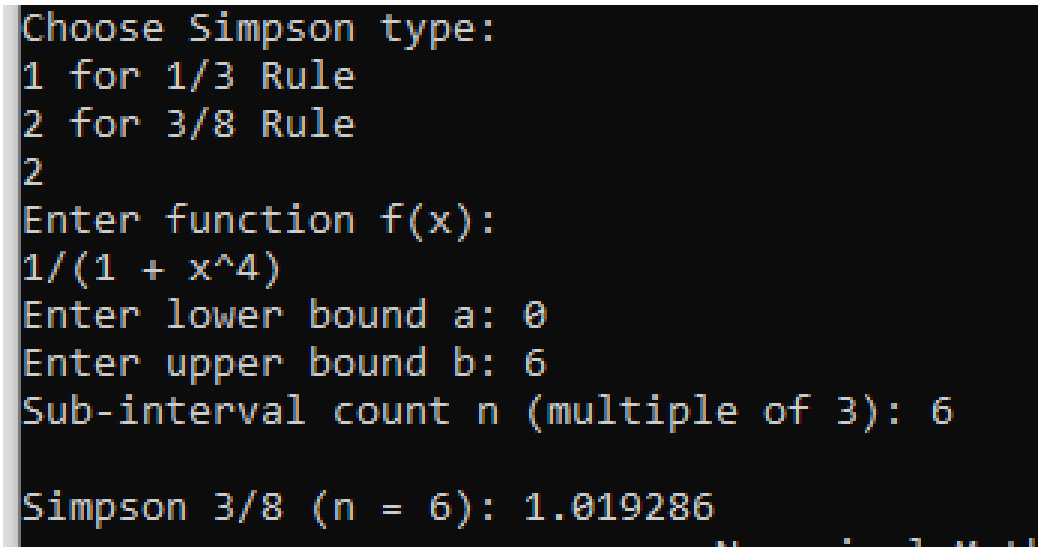
11.2 Örnek: $(x^2 - 1)(x + 2)$ fonksiyonu için Simpson 1/3

Alt sınır: $a = -2$ **Üst sınır:** $b = -1$ **Parça sayısı:** $n = 4$ (çift)



11.3 Örnek: $\frac{1}{1+x^4}$ fonksiyonu için Simpson 3/8

Alt sınır: $a = 0$ **Üst sınır:** $b = 6$ **Parça sayısı:** $n = 6$ (3'ün katı)



12 Trapezoid Yöntemi

12.1 Parametreler

Fonksiyon: $f(x)$

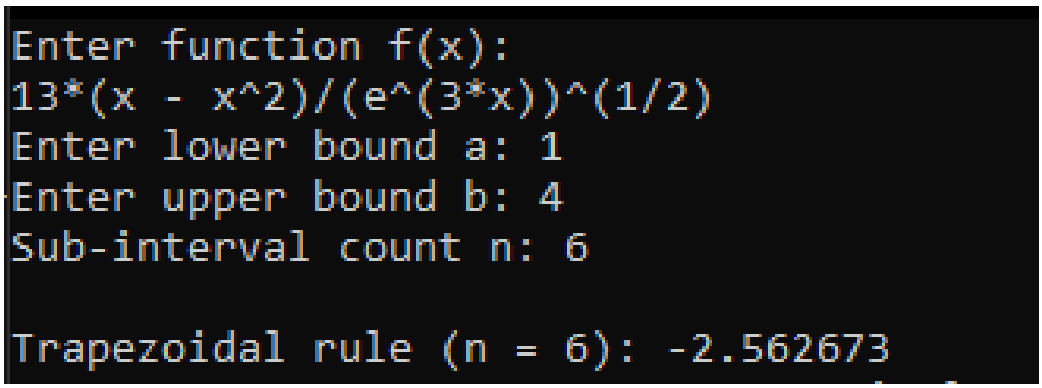
Alt sınır: a **Üst sınır:** b

Parça sayısı: n (pozitif tamsayı)

Durma koşulu: $n \leq 0$ girilirse işlem iptal edilir.

12.2 Örnek: $\frac{13(x-x^2)}{\sqrt{e^{3x}}}$ fonksiyonu için Trapezoid Yöntemi

Alt sınır: $a = 1$ **Üst sınır:** $b = 4$ **Parça sayısı:** $n = 6$



13 Değişken Dönüşümsüz Gregory-Newton Enterpolasyonu

13.1 Parametreler

Veri noktası sayısı: n

Başlangıç noktası: x_0

Adım aralığı: h

Verilen y değerleri: $f(x_0), f(x_1), ..., f(x_{n-1})$

Enterpolasyon noktası: x_p

13.2 Örnek 1:

n: 5 x_0**:** 2 **h:** 2 x_p**:** 3

Y değerleri: $[10, 50, 122, 226, 362]$

```
Enter number of data points: 5
Enter starting x (x0): 2
Enter step size h: 2
Enter value to interpolate x_p: 3
Enter 5 y values:
  f(2) = 10
  f(4) = 50
  f(6) = 122
  f(8) = 226
  f(10) = 362

x         f(x)        delta^1f(x)     delta^2f(x)     delta^3f(x)     delta^4f(x)
2         10          40          32          0           0
4         50          72          32          0
6         122         104         32
8         226         136
10        362

Interpolated value at x = 3 : 26

Interpolated polynomial in terms of x:
f(x) roughly = 10 + 40*((x - 2)/2) + 16*((x - 2)/2)*((x - 2)/2 - 1)
```

Kodun oluşturduğu fonksiyon:

$$f(x) \approx 10 + 40 \cdot \left(\frac{x-2}{2}\right) + 16 \cdot \left(\frac{x-2}{2}\right) \cdot \left(\frac{x-2}{2} - 1\right)$$

Sadeleştirilmiş hali:

$$f(x) = 4x^2 - 4x + 2$$

Kod bu veri noktalarına karşılık ikinci dereceden polinom üretmiştir. Enterpolasyon değeri $x = 3$ için 26 olarak bulunur.

13.3 Örnek 2:

n: 7 x_0**:** 0 **h:** 1 x_p**:** 2.5

Y değerleri: $[-4, -2, 2, 62, 160, 326, 578]$

```
Enter number of data points: 7
Enter starting x (x0): 0
Enter step size h: 1
Enter value to interpolate x_p: 2.5
Enter 7 y values:
  f(0) = -4
  f(1) = -2
  f(2) = 14
  f(3) = 62
  f(4) = 160
  f(5) = 326
  f(6) = 578

x         f(x)        delta^1f(x)     delta^2f(x)     delta^3f(x)     delta^4f(x)     delta^5f(x)     delta^6f(x)
0         -4          2          14          18          0          0          0
1         -2          16          32          18          0          0
2         14          48          50          18          0
3         62          98          68          18
4         160         166         86
5         326         252
6         578

Interpolated value at x = 2.5 : 32.875

Interpolated polynomial in terms of x:
f(x) roughly = -4 + 2*((x - 0)/1) + 7*((x - 0)/1)*((x - 0)/1 - 1) + 3*((x - 0)/1)*((x - 0)/1 - 1)*((x - 0)/1 - 2)
```

Kodun oluşturduğu fonksiyon:

$$f(x) \approx -4 + 2x + 7x(x-1) + 3x(x-1)(x-2)$$

Sadeleştirilmiş hali:

$$f(x) = 3x^3 - 2x^2 + x - 4$$

Kod bu veri kümesine karşılık üçüncü dereceden bir polinom oluşturmuştur. $x = 2.5$ için enterpolasyon sonucu 32.875 olarak hesaplanır.