Sayısal Analiz Projesi

Habil Çoban

May 7, 2024

İçindekiler

1 Giriş

Program 10 farklı işlemi yerine getirmek amaçlı yapılmıştır. Programda yapılabilecek işlemler 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

$$\text{Yapılanlar 1 ile gösterilmiştir : } \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Bu yöntemlerde kullanılan fonksiyonları hesaplamak için ise bir parser yazılmıştır. Kullanılan parse yönteminden sonraki bölümlerde bahsedilecektir.

2 Parse İşlemi

2.1 Kullanılan yöntem

String Parse işlemi bir den fazla yöntemle yapılabiliyor. Yöntemler arasında bir kaçını araştırdım ve **Recursive Descent Parsing(RDP)** kullanmayı tercih ettim. RDP öz yinelemeli olarak çalışan **Top-down** bir parsing yöntemidir. Son tokene ulaşana kadar fonksiyonlar ilerler ve sona ulaşıldığında hesaplanan değerler döndürülerek sonuca ulaşılır.

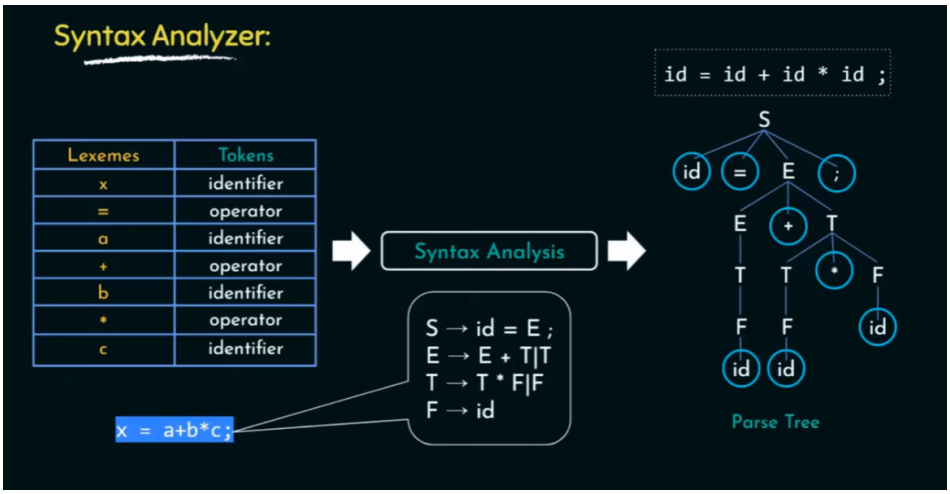

Figure 1: Neso Academy videosundan alıntı.

2.2 Tokenlere ayırma

Öncelikle typdef enum ile token türleri belirlenmiştir. Token için struct tanımlanmıştır

```
typedef enum {
    TOKEN_NUM, //e ,pi gibi sabitler bu kategoride
    TOKEN_PLUS,
    TOKEN_MINUS,
    TOKEN_MUL,
    TOKEN_DIV,
    TOKEN_LPAREN,
    TOKEN_RPAREN,
    TOKEN_EXPONENT,

    TOKEN_SIN,
    TOKEN_COS,
    TOKEN_LOG,
    TOKEN_TAN,
    TOKEN_COT,
    TOKEN_ASIN,
    TOKEN_ACOS,
    TOKEN_ATAN,
    TOKEN_ACOT,

    TOKEN_VARIABLE,
    TOKEN_EOF,  // End of the expression
    TOKEN_INVALID
}

TokenType;

typedef struct TokenST {
    TokenType type;
    double value;
} Token;

Token getNextToken(char **input,float*varX);
```

getNextToken() fonksiyonu ise aldıgı parametrelere göre sonraki tokeni döndürüyor. Parametre olarak fonksiyonu işaret eden bir pointer ve degiskenimizin degerini veriyoruz.

2.3 Fonksiyonu hesaplama

```
double parseDegree3(char **input,Token* cr,float* varX);//Ucuncu derece oncelik
double parseDegree2(char **input,Token* cr,float* varX);//Ikinci derece oncelik
double parseDegree1(char **input,Token* cr,float* varX); //En yuksek oncelik

double funcEvaluate(char *input,float varX);
```

Fonksiyonumuzun değerini hesaplarken kullandığımız ana fonksiyonlar bunlar. İlk olarak **funcEvaluate()** fonksiyonunu çağırıyoruz ve parse işlemini başlatıyoruz. Sonrasında sırayla **parseDegree3()**, **parseDegree2()** ve **parseDegree1()** rekürsif olarak çağırılıyor. Fonksiyonların içinde kendi öncelik derecelerine göre çağırabilecekleri fonksiyonlar var. Örneğin 3 derecede toplama ve çıkarma yani öncelik olarak en düşük operatörler varken 1 derecede üstel , sin ve cos gibi fonksiyonlar yer alıyor.

Her fonksiyon önce kendinden daha yüksek öncelikli olanı çağırır. Fonksiyonlar kendi içinde barındırdığı if else kontrolleri ile şuanki tokenin hangi işleme ait olduğunu buluyor. Bulunduktan

sonra o işlemden yeniden parse işlemi dallanmaya başlıyor. Son token e kadar gidildikten sonra geriye doğru hesaplamalar yapılarak sonuç döndürülüyor.

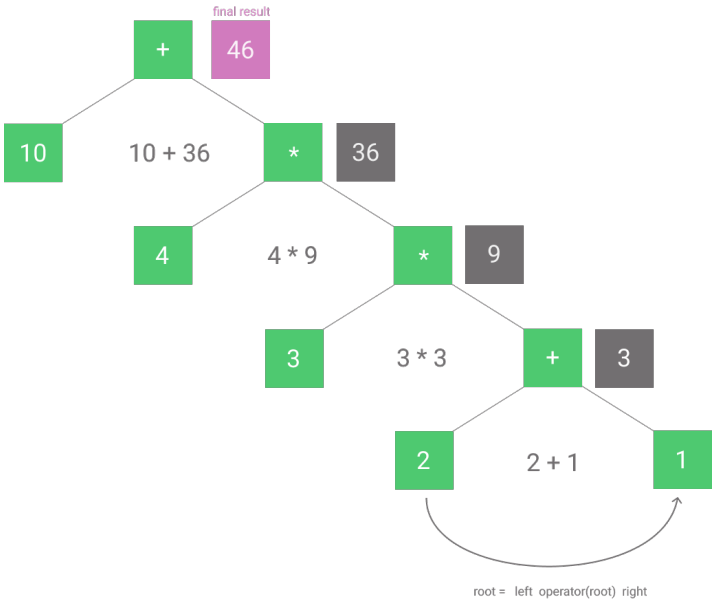

Figure 2: Örnek bir parse işlemi

3 Program Menüsü

Kodu çalıştırdığımızda karşımıza ilk olarak ana menü geliyor. Buradan istediğimiz işlemin numarasını girmemiz gerekiyor.

```
------------------
Program Secenekleri
------------------
1  + Bisection Yontemi
2  + Regula-Falsi Yontemi
3  + Newton Raphson Yontemi
4  + NxN'lik bir matrisin ters
5  + Gauus Eleminasyon
6  + Gauss Seidal Yontemi
7  + Sayisal Turev
8  + Simpson Yontemi
9  + Trapez Yontemi
10 + Gregory newton Enterpolasyonu
0  - Programdan Cik
------------------
Yapilacak islem secin :
```

Figure 3: Menü görseli

3.1 Desteklenen fonksiyonlar

Desteklene fonksiyonlar ve yazımları aşagıdaki gibidir. Fonksiyonları kombine ederek birlikte kullanabilirsiniz. Dikkkat edilmesi gereken bir husus ise üs alma , taban yazma gibi işlemlerde birden fazla ifade varsa parantez kullanmayı unutmayın.(x^5 için x^2+3 yerine x^(2+3) şeklinde olmalıdır)

3.1.1 Polinom

Polinom ifadeleri $x^2, x^3, x^{(2+3)}x^x \ldots$ gibi ifadeler $a_p \times x\hat{}x_{exp}$ şeklinde girilebilmektedir .

a_p: x'in katsayısı
x: değişken
x_{exp}: polinomun derecesi

3.1.2 Üstel

Üstel ifadeler $2^3\,, (7.2)^4\,, 5^x$, $e^{(x+2)}$ $\ldots$ gibi ifadeler $n\hat{}n_{exp}$ şeklinde girilebilmektedir .

n: üstel ifadenin tabanı
n_{exp}: ifadenin üssü

3.1.3 Logaritmik

Logaritmik ifadeler $\log_2 3\,, \log_{(7.2)} 4, \log_5 x\,, \log_e (x+2)$ $\ldots$ gibi ifadeler $\log\_a_l(b_l)$ şeklinde girilebilmektedir . Doğal logaritma yoktur onun yerine $\log\_e(b_l)$ şeklinde giriş yapılmaktadır.

a_l: logaritmanın tabanı
b_l: logaritmanın içi
e: euler sayısı

3.1.4 Trigonometrik

Trigonometrik ifadeler $\sin(pi)\,,\ \cos(x),\ \tan(17)\,,\ \cot(x+2), \ldots$ gibi ifadeler $f_{trig}(a_t)$ şeklinde girilebilmektedir.

f_{trig}: trigonometrik fonksiyonun ismi (sin,cos,tan,cot)
a_t : trigonometik ifadenin içindeki değer
pi : π 'nin değeridir.

3.1.5 Ters Trigonometrik

Ters Trigonometrik ifadeler $\arcsin(pi)\,,\ \arccos(x),\ \arctan(17)\,, \operatorname{arccot}(x+2), \ldots$ gibi ifadeler $f_{atrig}(a_t)$ şeklinde girilebilmektedir.

f_{atrig}: ters trigonometrik fonksiyonun ismi (asin,acos,atan,acot)
a_t : ters trigonometik ifadenin içindeki değer
pi : π 'nin değeridir.

3.2 Matris İşlemleri

3.2.1 Matris Girişi

Matris işlemlerinde ilk adım matrisi girmektir. Bunun için önce boyut(NxN) ya da denklem sayısı istenir. Daha sonra sırayla matrisin tüm elemanları kullanıcıdan istenir.

Ör: $\begin{bmatrix} 3 & 2 & 2 \\ 4 & 7 & 28 \\ 1.2 & 6 & 15 \end{bmatrix}$

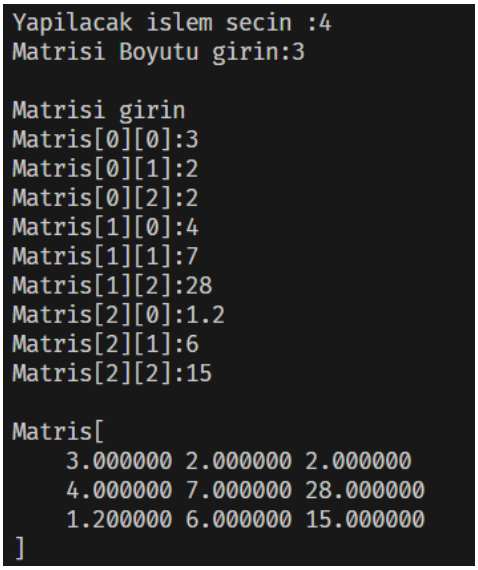

Figure 4: Matris girme

3.2.2 Denklem Girişi

Denklem sistemi girerken önce denklem sayisi girmemiz gerekiyor. Sırayla ifadelerin katsayıları ve sonucları giriliyor.

$$DenklemSistemi: \begin{bmatrix} -2 & 1 \\ 3 & -1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 8 \\ -11 \end{bmatrix}$$

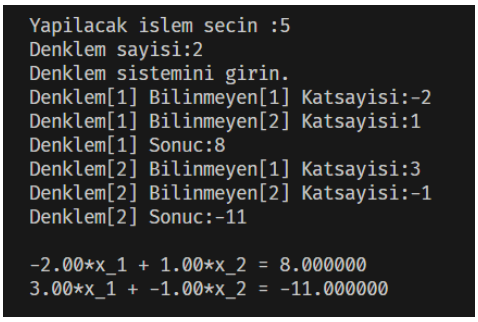

Figure 5: Denklem sistemi girme

4 Yöntemler ve Örnekler

4.1 Bisection

4.1.1 Parametreler

- *Fonksiyon* : Kullanıcıdan alınan fonksiyon
- *Sol* sınır: Arama işlemi icin sol sınır
- *Sag* sınır: Arama işlemi icin sag sınır
- *Epsilon* : Hata payı

- *D*urma Kosulu = $\left\{ \begin{array}{l} f(x) < Epsilon \\ İterasyon\ \ sayısı < 1000 \end{array} \right\}$

4.1.2 Örnek

Fonksiyon olarak $f(x) = x^3 - x - 2$ ve aralık olarak [1,2] verilmiştir.

```
----------
Fonksiyonu giriniz:x^3-x-2
Kok aranacak ilk araligi girin.
Sol sinir:1
Sag sinir:2
Epsilon degeri girin:0.01
x^3-x-2

Kok Bulundu: 1.5205 Degeri:-0.0052
Iterasyon Sayisi : 10
Press any key to continue . . .
```

Figure 6: Bisection örnek

4.2 Regula Falsi

4.2.1 Parametreler

- *Fonksiyon* : Kullanıcıdan alınan fonksiyon
- *Sol* sınır: Arama işlemi icin sol sınır
- *Sag* sınır: Arama işlemi icin sag sınır
- *Epsilon* : Hata payı
- *D*urma Kosulu = $\left\{ \begin{array}{l} f(x) < Epsilon \\ İterasyon\ \ sayısı < 1000 \end{array} \right\}$

4.2.2 Örnek

Fonksiyon olarak $f(x) = x^3 - 2x^2 + 3x - 5$ ve aralık olarak [0,2] verilmiştir.

```
----------
Fonksiyonu giriniz:x^3-2*x^2+3*x-5
Kok aranacak ilk araligi girin.
Sol sinir:0
Sag sinir:2
Epsilon degeri girin:0.01
Kok Bulundu: 1.8437 Degeri:0.0000
Iterasyon Sayisi : 10
Press any key to continue . . .
```

Figure 7: Regula-Falsi örnek

4.3 Newton Raphson

4.3.1 Parametreler

- *Fonksiyon* : Kullanıcıdan alınan fonksiyon
- *Baslangıc* sınır: Arama işlemi icin baslangıc noktası
- *Epsilon* : Hata payı

- $Durma\ \ Kosulu = \left\{ \begin{array}{l} f(x) < Epsilon \\ İterasyon\ \ sayısı < 1000 \end{array} \right\}$

4.3.2 Örnek

Fonksiyon olarak $f(x) = x * log_{10}x - 1.2$ ve baslangıç noktası 5 verilmiştir.

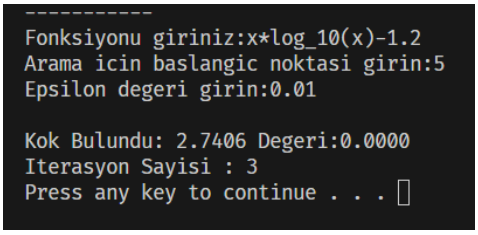

Figure 8: Newton-Raphson örnek

4.4 Matris Tersi

4.4.1 Parametreler

- N : Kare matrisin boyutu
- *Matris[i][j]* : Matristeki sayı

4.4.2 Örnek

Örnek matris olarak $\begin{bmatrix} -1.2 & 3 \\ 4 & 5 \end{bmatrix}$ verilmiştir.

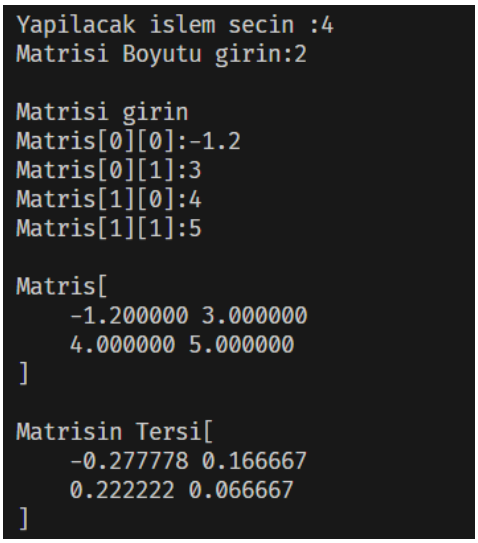

Figure 9: Matris Tersi Örnek

4.5 Gauss Eliminasyon

4.5.1 Parametreler

- N : Denklem sistemi sayısı
- *Denklem[i][j]* : Denklem katsayısı

4.5.2 Örnek

$$DenklemSistemi: \begin{bmatrix} 4 & -3 & 1 \\ -2 & 1 & -3 \\ 1 & -1 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ -4 \\ 2 \end{bmatrix}$$

```
Yapilacak islem secin :5
Denklem sayisi:3
Denklem sistemini girin.
Denklem[1] Bilinmeyen[1] Katsayisi:4
Denklem[1] Bilinmeyen[2] Katsayisi:-3
Denklem[1] Bilinmeyen[3] Katsayisi:1
Denklem[1] Sonuc:-8
Denklem[2] Bilinmeyen[1] Katsayisi:-2
Denklem[2] Bilinmeyen[2] Katsayisi:1
Denklem[2] Bilinmeyen[3] Katsayisi:-3
Denklem[2] Sonuc:-4
Denklem[3] Bilinmeyen[1] Katsayisi:1
Denklem[3] Bilinmeyen[2] Katsayisi:-1
Denklem[3] Bilinmeyen[3] Katsayisi:2
Denklem[3] Sonuc:2

Denklem Sistemi:
4.00*x_1 + -3.00*x_2 + 1.00*x_3 = -8.000000
-2.00*x_1 + 1.00*x_2 + -3.00*x_3 = -4.000000
1.00*x_1 + -1.00*x_2 + 2.00*x_3 = 2.000000

Cozumler:
x_1 : -0.6667
x_2 : 2.6667
x_3 : 2.6667
```

Figure 10: Gauss Eliminasyon Örnek

4.6 Gauss Seidal

4.6.1 Parametreler

- N : Denklem sistemi sayısı
- *Denklem[i][j]* : Denklem katsayısı
- *Epsilon* : Hata payı
- *Bilinmeyen[i]* : Bilinmeyen için başlangıç değeri

4.6.2 Örnek

$$DenklemSistemi : \begin{bmatrix} 1 & 2 \\ 4 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 3 \end{bmatrix}$$

```
Yapilacak islem secin :6
Denklem sayisi:2
Denklem sistemini girin
Denklem[1] Bilinmeyen[1] Katsayisi:1
Denklem[1] Bilinmeyen[2] Katsayisi:2
Denklem[1] Sonucu:5
Denklem[2] Bilinmeyen[1] Katsayisi:4
Denklem[2] Bilinmeyen[2] Katsayisi:2
Denklem[2] Sonucu:3
Epsilon degeri girin
0.01
Baslangic degelerini girin
Bilinmeyen[1]:4
Bilinmeyen[2]:3

Genisletilmis Matris:
    4.000000 2.000000 3.000000
    1.000000 2.000000 5.000000

8 iterasyonda bulundu.
Sonuclar:
Bilinmeyen[1] : -0.6680
Bilinmeyen[2] : 2.8340
```

Figure 11: Gauss Seidal Örnek

4.7 Sayısal Türev

4.7.1 Parametreler

- *Türev Türü* : Alınacak Türevin türü (İleri,Geri,Merkezi)
- *Fonksyion*
- *Adım boyutu* : Hata payı (h)
- *Türev noktası*

4.7.2 Örnek

Örnek fonksyion $f(x) = x^e + 5 - x^2$

```
-----------
1=Merkezi Fark 2=Ileri Fark 3=Geri fark
Turevin turunu secin: 1
Adim boyutunu girin:0.01
Fonksiyonu giriniz:x^e+5-x^2
Turev alinacak noktayi girin:5

Sonuc:33.184676
Press any key to continue . . .
```

Figure 12: Türev Örnek

4.8 Simpson Yöntemi

4.8.1 Parametreler

- *Fonksyion*
- *Sol ve sağ sınır* : Simpson uygulanacak aralık
- *Adım sayısı* : Simpson metodu için adım sayısı çift olmalıdır

4.8.2 Örnek

Örnek fonksyion $f(x) = ln(x^2 + sin(x))$
Aralık [5,10]

```
-----------
Fonksiyonu giriniz:log_e(x^2+sin(x))
Integral alinacak araligi girin.
Sol sinir:5
Sag sinir:10
Adim sayisi:20
Sonuc: 19.9640
Press any key to continue . . .
```

Figure 13: Simpson Örnek

4.9 Trapez Yöntemi

4.9.1 Parametreler

- *Fonksyion*
- *Sol ve sağ sınır* : Trapez uygulanacak aralık
- *Adım sayısı*

4.9.2 Örnek

Örnek fonksyion $f(x) = 2^{cos(x)} + x^{log_{pi}e^{5x}}$
Aralık [1,3]

```
Fonksiyonu giriniz:2^(cos(x))+x^(log_pi(e^(5*x)))
Integral alinacak araligi girin.
Sol sinir:1
Sag sinir:3
Adim sayisi:40
Sonuc: 202017.5796
Press any key to continue . . .
```

Figure 14: Trapez Örnek

4.10 Gregory Newton Enterpolasyonu

4.10.1 Parametreler

- *Değer sayısı* : Verilecek noktaların sayısı
- *Noktaların sırasıyla değerleri*

4.10.2 Örnek

Verilen değerler $\begin{bmatrix} 0 & 1 \\ 1 & 3 \\ 2 & 8 \\ 3 & 13 \\ 4 & 21 \end{bmatrix}$ - Tahmin edilen deger : 8

```
Kac adet deger girilecek :5
1. degeri girin:1
2. degeri girin:3
3. degeri girin:8
4. degeri girin:13
5. degeri girin:21
Fonksiyonda istenen noktayi girin:8

x = 8.0000 icin y = 101.0000
```

Figure 15: Gregory-Newton Örnek

References

https://www.nesoacademy.org/
https://github.com/codeplea/tinyexpr
https://datatrained.com/post/recursive-descent-parser/