

Ek - z Transform

z Transform, Laplace Transformunun ayriksal dunyadaki karsiligidir, transform edilen surekli fonksiyon $f(t)$ degil, ayriksal, bir vektor olarak gorulebilecek $x(n)$ 'dir. z Transform

$$Z[x(n)] \rightsquigarrow X(z) = \sum_{-\infty}^{\infty} x(n)z^{-n}$$

ki z bir kompleks sayidir.

Gelisiguzel (casual) sistemlerden gelen verilerde sadece $n > 0$ veriye bakilabilir, o zaman alt sinir sifir olur

$$X(z) = \sum_0^{\infty} x(n)z^{-n}$$

z Transform ne ise yarar? Laplace Transform diferansiyel denklemlerin cozulmesine yardim ediyordu. z Transform benzer sekilde farklilik (difference) denklemlerin cozulmesine yardim eder. Farklilik denklemi mesela

$$y(n) = 0.85y(n-1) + x(n) \quad (3)$$

sekinde olabilir. Daha genel olarak farklilik denklemleri su sekilde belirtilebilir,

$$\sum_{k=0}^N a_k y(n-k) = \sum_{l=0}^M b_l x(n-l) \quad (1)$$

iki ustteki ornek, bu genel denklemin $N = 1, M = 0$ oldugu halidir, katsayilar $a_1 = 0.85, b_0 = 1$. Genel formdan $y(n)$ 'i disari cekebiliriz, o zaman k sifir yerine $k = 1$ 'den baslar

$$y(n) + \sum_{k=1}^N a_k y(n-k) = \sum_{l=0}^M b_l x(n-l)$$

Genel cozum icin farklilik denkleminin bu formuna z Transform uygulayabiliriz.

Ama ondan once kaydirma islemi, lineerlik gibi bazi temel ozellikleri, islemleri gorelim. Mesela

$$X(z) = x(0) + x(1)z^{-1} + x(2)z^{-2} + \dots \quad (2)$$

ise, bu dizin uzerinde zaman kaydirma islemi yapsak, yani -1 indeksi 0 haline gelse, onun gibi tum degerler bir ileri kaysa, $x(-1), x(0)$ olur, ve transform

$$x(-1) + x(0)z^{-1} + x(1)z^{-2} + \dots$$

Simdi z^{-1} 'i disari cekelim

$$= x(-1) + z^{-1} \left[x(0) + x(1)z^{-1} + \dots \right]$$

Koseli parantez icine bakarsak, oradaki degerler (2)'deki seriye benzemiyor mu? O zaman oraya direk $X(z)$ degerini koyabiliriz

$$= x(-1) + z^{-1}X(z)$$

Bir daha kaydirirsak,

$$z^{-2}X(z) + z^{-1}x(-1) + x(-2)$$

elde ederiz. Genel olarak m kadar kaydirirsak

$$z^{-m}X(z^{-1}) + z^{-m+1}x(-1) + z^{-m+2}x(-2) + \dots + x(-m)$$

Eger baslangic sartlari sifir ise, ustteki formilde $x(-1), x(-2), \dots$ tamamen sifir kabul edilebilir, ve daha basit su formulu elde ederiz.

$$Z[x(n-m)] \rightsquigarrow z^{-m}X(z^{-1})$$

Ayrica, z Transformun lineerlik ozelligi sayesinde

$$Z(ax(n)) = aZ(x(n)) \rightsquigarrow aX(z)$$

Simdi bu bilgiyle beraber (1)'in z Transformunu yapalim.

$$\begin{aligned} Y(z) + \sum_{k=1}^N a_k z^{-k} Y(z) &= \sum_{l=0}^M b_l z^{-l} X(z) \\ \rightarrow Y(z) \left[1 + \sum_{k=1}^N a_k z^{-k} \right] &= \sum_{l=0}^M b_l z^{-l} X(z) \\ \rightarrow Y(z) &= \frac{\sum_{l=0}^M b_l z^{-l} X(z)}{1 + \sum_{k=1}^N a_k z^{-k}} \end{aligned} \tag{5}$$

Formulun bolumdeki ust kismini acarsak

$$b_0 z^0 + b_1 z^{-1} + b_2 z^{-2} + \dots + b_m z^{-M}$$

$b_0 z^{-M}$ disari cekilirse

$$= b_0 z^{-M} (z^M + \frac{b_1}{b_0} z^{M-1} + \frac{b_2}{b_0} z^{M-2} + \dots + \frac{b_M}{b_0})$$

Bolumun alt kismini acarsak

$$1 + a_1 z^{-1} + a_2 z^{-2} + \dots + a_N z^{-N}$$

z^{-N} disari cekersek

$$= z^{-N} (z^N + z^{N-1} + \dots + a_N)$$

Bu yeni formlari bolumde $Y(z)$ icinde yerine koyelim

$$\begin{aligned} Y(z) &= \frac{b_0 z^{-M} (z^M + \dots + \frac{b_M}{b_0})}{z^{-N} (z^N + \dots + a_N)} \\ &= b_0 z^{N-M} \frac{z^M + \dots + \frac{b_M}{b_0}}{z^N + \dots + a_N} \end{aligned}$$

Bolum ve bolene bir daha bakalim simdi. Burada gosterilenler birer polinom, ve Cebirin Temel Teorisi'ne (Fundamental Theorem of Algebra) gore n derecesindeki bir $p(x)$ polinomunun mutlaka n tane reel ya da kompleks koku vardır.

Bir polinomun koku var ise, bu polinom $p(x)$ su sekilde de gosterilebilir

$$p(x) = (x - r_n)(x - r_{n-1}) \dots (x - r_1)$$

Bu mantigi iki ustteki formule uygularsak

$$= b_0 z^{N-M} \frac{\prod_{l=1}^M (z - z_l)}{\prod_{k=1}^N (z - p_k)}$$

Yani $b_0 z^{N-M}$ haricindeki ifadeler bir polinoma sebebiye verirler, ve bu polinomun kokleri bulunabilir, kokler bulununca cozum olan z degerleri bulunmus olacaktır. Bu z degerlerini alip z Transformunu olusturuz, ve ya tabloya bakarak, ya da baska sekilde ters transform yaparak farksal denklemin cozumune ulasmaya calisiriz.

Cozmek istedigimiz ornek (3)'e tekrar bakalim. Ustteki formule bu noktada gerek yok, (5)'e gore bile bu denklemin z Transformunu bulabiliriz.

$$Y(z) = \frac{1}{1 - 0.85z^{-1}}$$

Eger yaygın z Transformların tablosuna bakarsak, $|z| > 0.85$ için üstteki formülün ters z Transformunun

$$h(n) = Z^{-1}[Y(z)] = 0.85^n u(n)$$

olduğunu öğreniyoruz, ve $u(n)$ şöyle

$$u(n) = \begin{cases} 1, & \text{eger } n \geq 0 \\ 0, & \text{eger } n < 0 \end{cases}$$

Dikkat edilirse ters z Transform tablosunda y değil h bazlı sonuçlar gösteriliyor, h fonksiyonları vurucu cevabı (impulse response) fonksiyonlarıdır, ve hesaplamaları burum (convolution) üzerinden olur. Bir h , bir lineer sistemi tekil olarak temsil ettiği için h 'i alıp y bazlı sonuca gitmek çok kolaydır, hemen h 'in tarif ettiği burumu yaparız.

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = \sum_{k=-\infty}^{\infty} x(k)0.85^{n-k}u(n-k).$$

Kodlama

Düşünelim ki (3) ile tarif edilen farklılık denklemi bir şirketin patent portfolyunu temsil ediyor.

Bu şirket, her sene portfolyo $y(n)$ 'e, $x(n)$ kadar yeni patent ekliyor, ama her sene aynı zamanda elindeki patentlerin yüzde 15'i "eskiyor", yani zamanları dolarak portfolyondan çıkıyor. Bu eksiltme işlemini bir önceki $y(n)$ 'i 0.85 ile çarparak temsil ediyoruz.

Boyle bir problemde $x(n)$ bize veri olarak verilecektir, ve toplamsal / kumulatif (cumulative) $y(n)$ 'i hesaplamamız istenecektir. Bu hesap literatürde “değer kaybeden kumulatif toplam (cumulative sum with depreciation)” olarak biliniyor.

O zaman üstteki formüldeki sonucu kodlarsak, ve örnek veriyle

```
patents = np.array([ 4.,  3.,  2.,  8.,  4.,
                    4., 10.,  4., 10.,  7.] )
```

```
def u(n,k):
    if n-k < 0: return 0
    return 1.
```

```
def y(n,data):
```

```

    sum = 0
    for k in range(len(data)):
        sum += data[k]*(0.85**(n-k))*u(n,k)
    return sum

for n in range(len(patents)):
    print y(n,patents)

```

4.0
 6.4
 7.44
 14.324
 16.1754
 17.74909
 25.0867265
 25.323717525
 31.5251598962
 33.7963859118

Bu sistemi otomatik olarak cozen Python islemi lfilter cagrisidir.

```

from scipy.signal import *

a = np.array([ 4., 3., 2., 8., 4.,
               4., 10., 4., 10., 7.])
d = 0.15
res = lfilter((1,),(1,d-1),a)
k = [a[0]]
for inv in a[1:]: k.append((1-d)*k[-1] + inv)
print np.array(k)

```

[4. 6.4 7.44 14.324 16.1754 17.74909
 25.0867265 25.32371752 31.5251599 33.79638591]

Kaynaklar

- [1] Introduction to DSP and Filter Design, B. A. Shenoi, pg. 41
- [2] Digital Signal Processing, Ifeachor, pg. 105
- [3] Digital Signal Processing using Matlab, Slicer, pg. 119