ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

ELM 368 SAYISAL İŞARET İŞLEME LABORATUVARI

ÖN HAZIRLIK ÇALIŞMASI

Laboratuvar 2

1 AMAÇ

- Fark denklemi verilen doğrusal ve zamanda değişmez sistemlerde çıkış hesaplamak.
- Dürtü cevabı verilen bir sistemin çıkışını hesaplamak.

2 KODLAR

2.1 Fark denklemi verilen DZD sistemlerde çıkışı hesaplamak

Eğer bir sistemin giriş-çıkış arasındaki ilişkisi aşağıdaki denklem ile ifade edilebiliyor ise, biz bu tür sistemlere sabit katsayılı fark denklemi ile ifade edilebilen sistemler deriz.

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{l=0}^{M} b_l x[n-l]$$

Bilgisayar ortamında sabit katsayılı fark denklemi ile ifade edilebilen bir sistemi tanımlayabilmek için tek ihtiyacımız olan şey a ve b katsayılarıdır. Herhangi bir giriş işaretine sistemin üreteceği çıkışı, a ve b katsayılarını giriş olarak alan scipy kütüphanesinin signal modülündeki "lfilter" fonksiyonu ile hızlı bir şekilde hesaplayabileceğiniz gibi herhangi bir hazır fonksiyon kullanmadan kendiniz de döngüler kullanarak hesaplayabilirsiniz. Aklınızda kalıcı olması için aşağıdaki fark denklemi ile ifade edilen sistemin $x[n] = \delta[n] - \delta[n-1]$ girişine verdiği tepkiyi hazır fonksiyon kullanmadan veren kod parçasını vereceğim.

$$y[n] + \frac{1}{2} y[n-1] = x[n] + 2x[n-1]$$

Fark denklemi verilen bir sistemin doğrusal olabilmesi için başlangıç koşulu sıfır olmak zorundadır. Bu da giriş işareti ($x[n] = \delta[n] - \delta[n-1]$) $n = 0$'da ilk olarak sıfırdan farklı bir değer aldığından dolayı $n < 0$ olduğu tüm durumlarda $y[n] = 0$ olmak zorunda olduğu anlamına gelir. $n = 0$'dan $n = 3$'e çıkış işareti

$y[n]'$i hesaplamak istersek aşağıda gösterildiği gibi yinelemeli olarak her bir n değeri için $y[n]$'i hesaplayabiliriz;

$$y[n] = -\frac{1}{2}y[n-1] + x[n] + 2x[n-1]$$
$$= -\frac{1}{2}y[n-1] + \delta[n] - \delta[n-1] + 2(\delta[n-1] - \delta[n-2])$$
$$= -\frac{1}{2}y[n-1] + \delta[n] + \delta[n-1] - 2\delta[n-2]$$

$y[0]$ için;

$$y[0] = -\frac{1}{2}y[-1] + \delta[0] + \delta[-1] - 2\delta[-2] = 1$$

$y[1]$ için;

$$y[1] = -\frac{1}{2}y[0] + \delta[1] + \delta[0] - 2\delta[-1] = \frac{1}{2}$$

$y[2]$ için;

$$y[2] = -\frac{1}{2}y[1] + \delta[2] + \delta[1] - 2\delta[0] = -\frac{9}{4}$$

$y[3]$ için;

$$y[3] = -\frac{1}{2}y[2] + \delta[3] + \delta[2] - 2\delta[1] = \frac{9}{8}$$

2.1.1 Hazır komut kullanmadan çıkış işaretini hesaplama

```
x=[0,1,-1,0,0]
y=[0]
for i in range(1,len(x)):
    y.append(-0.5*y[i-1]+x[i]+2*x[i-1])
print(y)
konsol çıktısı → [0, 1.0, 0.5, -2.25, 1.125]
```

Yukarıda $y[n]$ dizisini $n = -1$'den $n = 3$'e kadar bulan kodu paylaştım. Burada x ve y dizileri numpy array yerine liste veri tipi ile tanımladım. Bu örneği liste veri tipi ile göstermemin nedeni Python'da liste veri tipi en sık kullanılan veri tiplerinden birisi olduğu içindir. Kodu açıklamak gerekirse, öncelikle giriş ve çıkış dizilerini oluşturduk. Burada dikkat etmeniz gereken nokta indis vektöründen bağımsız olacak şekilde giriş işaretini tanımlamanız gerek. Sistemin giriş-çıkış ilişkisi aşağıda tekrardan veriyorum.

$$y[n] = -\frac{1}{2}y[n-1] + x[n] + 2x[n-1]$$

$y[0]$' ı bulmak istediğimizde $y[-1]$ ve $x[-1]$ değerlerine ihtiyacımız olduğu için ($y[-1] = x[-1] = 0$ olduğunu biliyoruz.) x ve y listelerinin ilk elemanları sıfır olacak şekilde ayarladım. Bunun yanında $y[n]$'i hesaplarken $x[n]$'e de ihtiyacımız olduğundan ve $n = 0$'dan $n = 3$'e kadar $y[n]$'i hesaplamak istediğimiz için x işaretini $n = -1,0,1,2,3$ indisleri üzerinde tanımladım.

```
x=[0,1,-1,0,0]
y=[0]
```

Aşağıda ise i değişkenini 1'den başlayıp birer birer artarak uç noktası x dizisinin boyuna eşit olacak şekilde bir döngü kurdum. Döngünün 1'den başlamasının nedeni x'in 1. indis değeri aslında indis vektöründe $n = 0$'a denk gelmesidir.

```
for i in range(1,len(x)):
    y.append(-0.5*y[i-1]+x[i]+2*x[i-1])
```

Daha sonra y listesinin append() metodu ile y listesinin sonuna $-\frac{1}{2}y[i-1] + x[i] + 2x[i-1]$ değerini ekledik. Her iterasyonda y listesinin boyu bir eleman artmaktadır. Liste veri tipini hiç kullanmadan doğrudan numpy kütüphanesinin array tipinde değişkenler kullanarak da aynı çıktıyı elde edebilirsiniz. Listeden farklı olarak eğer array kullanacaksanız y dizisinin uzunluğu $n = -1,0,1,2,3$

değerleri için en başta oluşturup daha sonra her bir indis için döngüde içerdeki elemanı değiştirmeniz gerekmektedir. Aşağıda liste yerine numpy array tipi değişkenlerin kullanıldığı ve aynı çıktıyı veren kod parçası verilmektedir.

```
import numpy as np
x=np.array([0,1,-1,0,0],dtype=float)

y=np.array([0,0,0,0,0],dtype=float)
for i in range(1,len(x)):
    y[i]=-0.5*y[i-1]+x[i]+2*x[i-1]
print(y)
konsol çıktısı→ [ 0.     1.     0.5   -2.25   1.125]
```

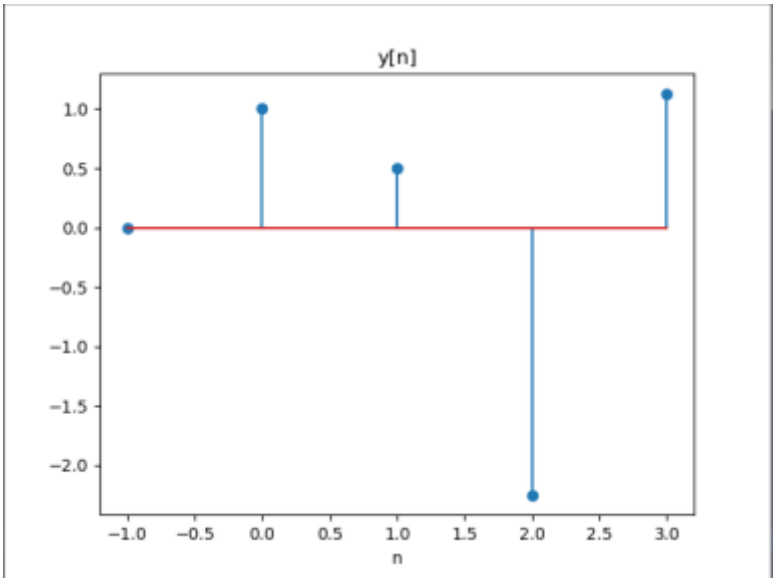
2.1.2 Hazır komut kullanarak çıkış işaretini hesaplama

Katsayılarını bildiğiniz bir doğrusal sabit katsayılı fark denklemi ile ifade edilen bir sistemin herhangi bir giriş işaretine verdiği cevabı Scipy kütüphanesinin Signal modülündeki lfilter(b,a,x) komutu ile hesaplayabilirsiniz. Burada "b" vektörü giriş işaretinin katsayılarına, "a" ise çıkış işaretinin katsayılarına karşılık gelmektedir. "x" ise girişe uygulamak istediğiniz tek boyutlu dizidir.

```
from scipy import signal
x=np.array([0,1,-1,0,0],dtype=float)
b=[1,2]
a=[1,0.5]
y=signal.lfilter(b,a,x)
print(y)
konsol çıktısı→ [ 0.     1.     0.5   -2.25   1.125]
```

Çıkış işaretinin grafiğini çizdirmek istersek eğer, Matplotlib pyplot'u import ettikten sonra $n = -1,0,1,2,3$ değerlerini alacak şekilde indis vektörünü oluşturup stem() ile çizdirebiliriz;

```
from matplotlib import pyplot as plt
n=np.arange(-1,4)
plt.stem(n,y)
plt.title('y[n]')
plt.xlabel('n')
```



2.2 Dürtü cevabı bilinen bir sistemin giriş işaretine verdiği cevabı bulmak

Dürtü cevabı $h[n]$ sistemin girişine dürtü işareti $\delta[n]$ uygulandığında sistemin çıkışında elde edilen işarettir. Eğer DZD bir sistemin dürtü cevabını biliyorsanız herhangi bir giriş işareti $x[n]$'e sistemin vereceği cevap $y[n]$'i giriş işareti ile dürtü cevabının konvolusyonunu alarak bulabilirsiniz;

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = \sum_{k=-\infty}^{\infty} h[k]x[n-k]$$

2.2.1 Hazır komut kullanmadan konvolusyon hesaplama

$$x[n] = \delta[n] + 2\delta[n-1] - \delta[n-2]$$

$$h[n] = \delta[n] + 3\delta[n-1]$$

İşaretlerinin konvolusyonunu $y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$ formülünü kullanarak hesapladığınızda aşağıda verilen $y[n]$ işaretini elde edersiniz;

$$y[n] = \delta[n] + 5\delta[n-1] + 5\delta[n-2] - 3\delta[n-3]$$

```
import numpy as np
h=np.array([1,2,-1],dtype=float)
x=np.array([1,3],dtype=float)

M=len(h)
N=len(x)
L=M+N-1
y=np.zeros([L])
for n in range(L):
    y[n]=0
    for k in range(N):
        if n-k<0 or n-k>=M:
            h_n_k=0
        else:
            h_n_k=h[n-k]
        y[n]=y[n]+x[k]*h_n_k
print(y)
Konsol çıktısı → [ 1.  5.  5. -3.]
```

Yukarıdaki kod x ve h dizilerinin konvolusyonu ile y dizisini üretmektedir. Bu kodta dikkat etmeniz gereken iki unsur var. Bunlardan birincisi x işaretinin uzunluğuna N, ve h işaretinin uzunluğuna M dersek, y işaretinin uzunluğu $N + M - 1$ olur. Dolayısıyla y dizisini hesaplayan dıştaki döngü $N + M - 1$ iterasyondan oluşur. Diğer önemli nokta ise $h[n-k]$ hesaplanırken $n-k$'nın negatif ve $n-k$'nın M'den büyük olduğu değerlerde program hata üretmemesi

için önceden bu aralıklardaki değerler için $h[n-k]$ sıfıra eşitlenmeli. Bunu yapabilmek için h_n_k adında bir değişken tanımlayıp ilgili aralıklar için sıfıra veya $h[n-k]$ değerine eşitledik.

```
if n-k<0 or n-k>=M:
            h_n_k=0
        else:
            h_n_k=h[n-k]
```

Toplam sembolünü hesaplatmak için ise her bir n değeri için başta y[n]'i sıfıra eşitleyip içteki döngüde her bir k değerini mevcut y[n] değeri üzerine ekledik.

```
y[n]=0
    for k in range(N):
        if n-k<0 or n-k>=M:
            h_n_k=0
        else:
            h_n_k=h[n-k]
        y[n]=y[n]+x[k]*h_n_k
```

2.2.2 Hazır komut kullanarak konvolusyon hesaplama

2.2.1'de verilen x ve h dizilerinin konvolusyonlarını numpy kütüphanesinin veya scipy kütüphanesinin signal modülündeki convolve() fonksiyonu ile hesaplayabilirsiniz.

```
import numpy as np
from scipy import signal
h=np.array([1,2,-1],dtype=float)
x=np.array([1,3],dtype=float)
print(np.convolve(x,h))
print(signal.convolve(x,h))
Konsol çıktısı →
                [ 1.  5.  5. -3.]
```

```
[ 1.  5.  5. -3.]
```

2.2.3 İki işaretin konvolusyonunun grafiğini çizdirmek

Konvolusyon işleminde dikkat ettiyseniz indis bilgisini işin içine hiç dahil etmedik. Ancak ayrık zamanlı $y[n]$ işaretini n'e göre çizdirmek istersek eğer, konvolusyon sonucu elde ettiğimiz dizinin hangi n değerlerine denk geldiğini bilmemiz gerek.

Çıkış işaretinin indis vektörünü bulmanın en kolay yolu giriş ve dürtü cevabı işaretlerinin başlangıç ve bitiş indislerindeki dürtülerin ayrı ayrı konvolusyonlarını almak. Şöyle ki; hem x hemde h işaretleri $n = 0$ indisinde sıfırdan farklı değere sahip. $\delta[n] * \delta[n] = \delta[n]$ olacağından dolayı $y[n]$ işaretinin sıfırdan farklı olduğu ilk indis $n = 0$ olacaktır. Bitiş indisine gelirsek, $x[n]$ işaretinin bitiş noktası $n = 2'$de $\delta[n-2]$ bileşeni vardır. $h[n]$ işareti ise $n = 1$ indisinde $3\delta[n-1]$ bileşeni ile bitmektedir. $\delta[n-2] * 3\delta[n-1] = 3\delta[n-3]$ olacağı için $y[n]$ işaretinin bitiş indisi $n = 3$ olur. Sonuç olarak $y[n]$ dizisini çizdirmek istersek $n = 0,1,2,3$ şeklinde bir indis vektörü ile çizdirmemiz gerekir. Mesela başka bir örnek yapalım;

$$x[n] = \delta[n+1] + 2\delta[n-1] - \delta[n-2]$$

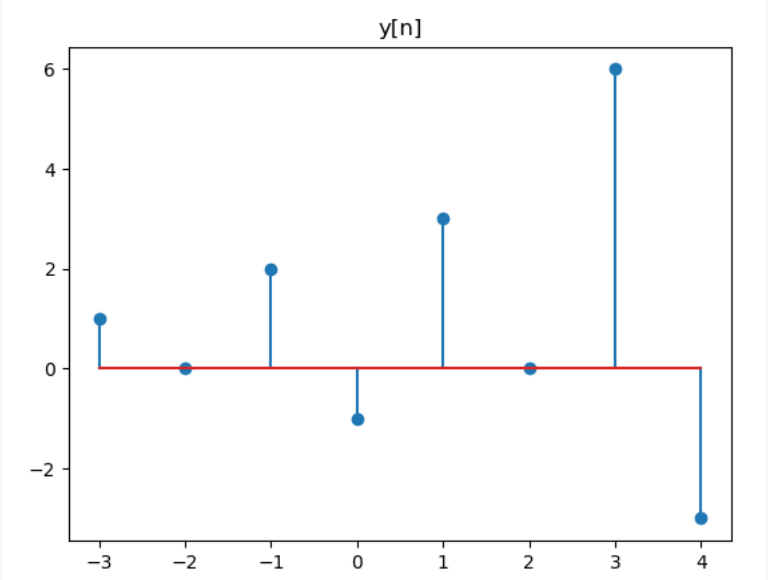
$$h[n] = \delta[n+2] + 3\delta[n-2]$$

$x[n]$ ve $h[n]$'in konvolusyonunu alırsak başlangıç ve bitiş indisleri yukarıda bahsettiğim yol ile $n = -3$'den $n = 4$'e kadar olduğunu buluruz. $y[n]$'i çizdirmek isersek -3'den 4'e kadar olan bir indis vektörü ile çizdirmemiz gerekir.

```
x=np.array([1,0,2,-1])
h=np.array([1,0,0,0,3])
y=np.convolve(x,h)
print(y)
```

```
Konsol çıktısı → [ 1  0  2 -1  3  0  6 -3]

n=np.arange(-3,5)
plt.stem(n,y)
plt.title('y[n]')
```



3 Çalışma Soruları

1) $y[n] = x[n] - x[n-1]$ fark denklemiyle ifade edilen DZD bir sistem için aşağıda verilen soruları cevaplayınız.

a) Bu sistemin dürtü cevabını el yordamıyla bulunuz.

b) Bu sistem kararlı mıdır? Neden?

c) Bu sistemin girişine $x[n] = \cos\left(\frac{\pi}{5}n\right)$ işaretini uygulayın. Bunun için önce $x[n]'$i 2 periyot olacak şekilde üretip scipy.signal kütüphanesinin convolve() fonksiyonu daha sonra yine aynı kütüphenin lfilter() fonksiyonuna uygulayıp çıkış işaretini elde edip çizdirin. (Convolve() fonksiyonunu kullanırken üçüncü parametre olarak 'same' girerseniz giriş ve çıkış işaretlerinin boyları aynı olur.)

2) Sistem-1: $y[n] - \frac{1}{2}y[n-1] = x[n]$ ile Sistem-2: $y[n] - 2y[n-1] = x[n]$ DZD sistemleri verilmektedir. Bu iki sistem için aşağıdaki soruları cevaplayın.

a) Python'da 100 noktalı olacak şekilde dürtü işareti oluşturun ve her iki sistemin bu oluşturduğunuz işarete verdiği cevabın grafiklerini çizdirin.

b) a-şıkkında elde ettiğiniz grafiklere bakarak bu sistemlerin kararlılıkları hakkında yorumda bulunabilir misiniz?

c) a-şıkkında elde ettiğiniz grafiklere bakarak sistemlerin nedensellikleri hakkında yorumda bulunabilir misiniz?

4 TESLİM ŞEKLİ ve ZAMANI

Kodlar bölümünde yazılan kodları kendiniz bir Jupyter Notebook'ta yazarak sonuçları gözlemleyin. Jupyter Notebook'ta yaptığınız çalışmayı **OgrenciNo_Ad_Soyad_LABno.ipynb** formatına bir isimle kaydedip Ders Kutusu'na yükleyiniz. Laboratuvar ön çalışmaları (ev ödevi), laboratuvarın yapılacağı gün sabah 05:00'e kadar sisteme yüklenmelidir. Sisteme geç yüklenen dosyalar kabul edilmeyecektir. Ön hazırlık çalışmasını yapmamış (sisteme ön çalışmasını yüklememiş) öğrenciler aynı yapılacak laboratuvar çalışmasına giremez. Ekte, örnek bir ödev çözümü şablonu verilmektedir (bknz: **101024099_AYSE_SEN_LAB1.ipynb**). Jupyter Notebook'ta yapacağınız çözümler **bu şablona göre hazırlanmalıdır.**