ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

ELM 368 SAYISAL İŞARET İŞLEME LABORATUVARI

ÖN HAZIRLIK ÇALIŞMASI

VE

ÖDEV-2

1 AMAÇ

- Fark denklemi ile verilen doğrusal ve zamanda değişmez sistemlerin cevabını hesaplamak.
- Dürtü cevabı verilen doğrusal ve zamanda değişmez sistemlerin cevabını hesaplamak.

2 KODLAR

2.1 Fark denklemi verilen DZD sistemlerde çıkışı hesaplamak

Doğrusal zamanda değişmez sistemler aşağıda verildiği gibi sabit katsayılı fark denklemi ile ifade edilebilir.

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

Bilgisayar ortamında sabit katsayılı fark denklemi ile ifade edilebilen bir sistemi tanımlayabilmek için tek ihtiyacımız olan şey a ve b katsayılarıdır. Herhangi bir $x[n]$ giriş işaretine sistemin üreteceği $y[n]$ çıkış işaretini, scipy kütüphanesinin signal modülündeki *lfilter*() fonksiyonu ile hesaplayabileceğiniz gibi herhangi bir hazır fonksiyon kullanmadan kendiniz de döngüler kullanarak hesaplayabilirsiniz.

2.1.1 Hazır komut kullanmadan çıkış işaretini hesaplama

Aşağıda fark denklemi ile ifade edilen sistemin girişine $x[n] = \delta[n] - \delta[n-1]$ işareti uygulandığında verdiği tepkiyi hazır fonksiyon kullanmadan veren kod parçasını yazalım:

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

Fark denklemi verilen bir sistemin doğrusal olabilmesi için başlangıç koşulu sıfır olmak zorundadır. Diğer bir ifade ile, DZD sistemler için $n < 0$ için $y[n] = 0$'dır. $n = 0$'dan $n = 3$'e kadar $y[n]'$i hesaplamak istersek aşağıda gösterildiği gibi her n değeri için $y[n]$'i yinelemeli olarak hesaplayabiliriz;

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

$n = 0$ için: $\quad y[0] = -\frac{1}{2}y[-1] + \delta[0] + \delta[-1] - 2\delta[-2] = 1$

$n = 1$ için: $\quad y[1] = -\frac{1}{2}y[0] + \delta[1] + \delta[0] - 2\delta[-1] = \frac{1}{2}$

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

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

El ile yaptığımız bu işlemleri Python'da hazır kod kullanmadan yapmak için önce giriş işaretini tanımlamak gerekir. Diğer adımları inceleyiniz:

```
import numpy as np

x = np.zeros(20)
y = np.zeros(20)

y[-1] = 0

x[-1] = 0
x[0] = 1
x[1] = -1


for n in range(0,4):
    y[n] = -0.5*y[n-1]+x[n]+2*x[n-1]

print(y[0:4])

konsol çıktısı → [1.0, 0.5, -2.25, 1.125]
```

Yukarıda $y[n]$ dizisini $n = 0$'dan $n = 3$'e kadar hesaplayan kod verilmiştir. Burada x ve y dizileri numpy array olarak tanımlanmıştır. Kodu inceleyince göreceğiniz üzere, x ve y vektörleri sıfır başlangıç değerlerine sahip olacak şekilde tanımlanmıştır. Hem x hem de y için normalde sadece 5 uzunlukta iki vektör bu problemde yeterli olmasına rağmen negatif indisli değerleri de kullanabilmek için 20 uzunlukta iki vektör oluşturulmuştur (daha uzun bir dizi oluşturulmuştur).

```
x = np.zeros(20)
y = np.zeros(20)
```

Bu işlemin ardından giriş dizisi ve çıkış dizisi oluşturulmuştur. Bu örnekte $y[0]$'ı bulmak için $x[0]$ değerine ilave olarak $x[-1]$ ve $y[-1]$ değerlerine de ihtiyaç vardır. $y[-1] = 0$ değeri atanmıştır. Neden sıfır? Çünkü, tekrar vurgulamak gerekirse, DZD sistemler için $n < 0$ için $y[n] = 0$'dır. $y[0]$'ı bulmak istediğimizde $x[-1]$ değerine de ihtiyacımız olduğuna dikkat ediniz. Bu örnekte $x[n] = \delta[n] - \delta[n-1]$ olduğu için $x[-1] = 0$ olduğunu hesaplayarak görebilirsiniz. Kodda verilen "x[-1]=0" satırı, bu değer ataması, esasen bu örnekte gereksizdir zira negatif indisli olması sebebiyle "x[-1]=x[19]"dur ve x[19]'un değeri "x = np.zeros(20)" satırı sebebiyle zaten sıfıra eşittir. *Parantez açmak gerekirse, x[-1]=x[19], x[-2]=x[18]=0, x[-3]=x[17]=0, vb.'dir; dolayısıyla, $y[n]$ dizisini $n = 0$'dan $n = 3$'e kadar hesaplamak için bize x[-1], x[0],…,x[3] değerlerini depolamak için beş elemana sahip bir vektör gerekmektedir. Bunun 4 tanesi pozitif tarafta kalırken 1 tanesi negatif değere sahiptir. Beş elemandan daha uzun bir vektör tanımlayarak, ihtiyacımızdan daha fazla olan bölgeyi x[-1] gibi negatif indisli değerlere ayırma imkanı oluşturduk-örneğin x[0]…x[10] arası pozitif indisli elemanlar için kullanılırken x[19], x[18], …, x[11] arası, sırasıyla x[-1], x[-2],…,x[-9] için kullanılabilir.Benzer açıklama y[n] için de geçerlidir.*

Giriş işareti $x[n] = \delta[n] - \delta[n-1]$ olduğu için aşağıdaki iki satır giriş işaretini tanımlamak için kullanılmıştır.

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

Aşağıda verilen döngü n değişkenini 0'dan başlatıp birer birer artarak 3'e eşit olacak şekilde ilerletmektedir.

```
for n in range(0,4):
    y[n] = -0.5*y[n-1]+x[n]+2*x[n-1]
```

2.1.2 Hazır komut kullanarak çıkış işaretini hesaplama

Katsayılarını bildiğiniz bir doğrusal sabit katsayılı fark denklemi ile ifade edilen 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. DZD sistemleri tanımlamak için kullanılan sabit katsayılı fark denklemlerini hatırlamak gerekirse:

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

Yukarıda bahsedilen a ve b katsayıları bu fark denklemindeki parametrelere karşı gelmektedir. Bir önceki bölümdeki fark denklemini şu şekilde verilmişti:

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

Dolayısıyla,

$a_0 = 1,\ a_1 = 1/2, a_2 = 0, a_3 = 0, \ldots$

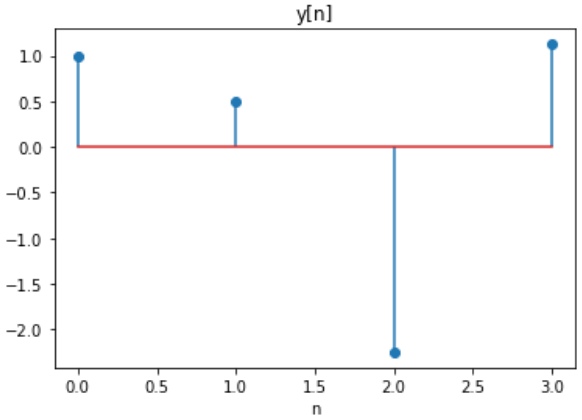
$b_0 = 1,\ b_1 = 2, b_2 = 0, b_3 = 0, \ldots$

olmaktadır. Buradaki a ve b parametreleri bir vektör olarak aşağıdaki gibi tanımlanmalıdır.

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

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

```
from matplotlib import pyplot as plt
n=np.arange(0,4)
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 biliniyorsa herhangi bir giriş işaretine sistemin vereceği cevabı, $y[n]$'i, giriş işareti ile dürtü cevabının konvolüsyonu alınarak bulunabilir. Bunun için aşağıda verilen konvolüsyon toplamı ifadesi kullanılır.

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

Konvolüsyon toplamı ifadelerinden birincisi şu şekilde ifade edilebilir:

$y[n] = \cdots + x[-2]h[n+2] + x[-1]h[n+1] + x[0]h[n] + x[1]h[n-1] + x[2]h[n-2] + \cdots$.

Konvolüsyon toplamı ifadelerinden ikincisi şu şekilde ifade edilebilir:

$y[n] = \cdots + h[-2]x[n+2] + h[-1]x[n+1] + x[0]h[n] + x[1]h[n-1] + x[2]h[n-2] + \cdots$.

x dizisinin sonlu olduğunu düşünelim: x dizisinin nx_{min} ve nx_{max} aralığında sıfırdan farklı değerler aldığını varsayalım. Bu durumda y dizisi şu şekilde olur:

$$y[n] = x[nx_{min}]h[n-nx_{min}] + x[nx_{min}+1]h[n-(nx_{min}+1)] + \cdots + x[nx_{max}]h[n-nx_{max}]$$

Benzer şekilde, h dizisinin de sonlu olduğunu ve nh_{min} ve nh_{max} aralığında sıfırdan farklı değerler aldığını varsayalım. Yukarıdaki toplamda x dizisinin aldığı indis değerleri soldan sağa doğru artarken, h dizisinin aldığı indis değerlerinin büyükten küçüğe doğru azaldığına dikkat ediniz. $n < nx_{min}$ için $x[n] = 0$'dır ve $n - nx_{min} < nh_{min}$ için $h[n] = 0$'dır. Dolayısıyla, bu toplamda y dizisinin sıfırdan farklı değer alacağı en küçük n değeri $n - nx_{min} = nh_{min}$ ile hesaplanabilir: $n_{min} = nx_{min}+nh_{min}$. Benzer analiz yapıldığında y dizisinin sıfırdan farklı değer alacağı en büyük n değerinin $n_{max} = nx_{max}+nh_{max}$ olduğu görülecektir. Bu sebeple, y dizisinin uzunluğu $n_{max} - n_{min} + 1 = (nx_{max}+nh_{max})-(nx_{min}+nh_{min})+1$ olur. Özel durum olarak $nx_{min}=0$, $nx_{max} = M-1$, $nh_{min} = 0$ ve $nh_{max} = N-1$ olsun. Diğer bir deyişle, x ve h dizileri sırasıyla $0, \ldots, M-1$ ve $0, \ldots, N-1$ aralığında değerler alan M ve N uzunluğunda diziler olsun. Bu durumda $n_{min} = 0$ ve $n_{max} = M + N - 1$ olur.

2.2.1 Hazır fonksiyon kullanmadan konvolüsyon hesaplama

Aşağıdaki giriş işareti ve dürtü cevabı için sistemin çıkışını hesaplayalım.

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

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

x ve h'nin konvolüsyonunu $y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$ formülünü kullanarak hesapladığınızda aşağıda verilen $y[n]$ işareti elde edilir.

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

x dizisinin sıfırdan farklı değer aldığı en küçük ve en büyük indisler sırasıyla 0 ve 2 indis değerleridir. h dizisinin ise sıfırdan farklı değer aldığı en küçük ve en büyük indisler sırasıyla 0 ve 1 indis değerleridir. Dolayısıyla, nx_{min}=0, nx_{max}=2, nh_{min}=0 ve $nh_{max} = 1$ olduğundan, y dizisinin sıfırdan farklı değer alacağı en küçük ve en büyük indis değerleri sırasıyla $n_{min} = nx_{min}$+nh_{min}=0 ve $n_{max} = nx_{max}$+nh_{max}=3 olup, dizinin uzunluğu ise $n_{max} - n_{min} + 1 = 4$ olacaktır. Python'da bu hesaplamayı yaparken bu sınırlara dikkat etmek gerekmektedir. Bu problemde $n_{min} \geq 0$ olduğu için y dizisine negatif indis değerlerinde bir değer atama durumu olmayacaktır. Ancak, aksi durumda, dizinin negatif indise sahip elemanlarına da değer ataması yapılması gerektiğinden kodlama aşamasında bu durumu dikkate almak gerekecektir. Her iki durumu birlikte ele alabilmek için x ve h dizilerinin başlangıç indisleri negatif değerli olsa bile başlangıç indis değerlerini sıfır varsayıp işlem yapılacak ve sonrasında ise yukarıda verilen formülleri kullanarak çıkış dizisi olan y'nin indislerini $n_{min} \ldots . n_{max}$ arasında olacak şekilde düzenlenecektir. Konvolüsyon toplamını hesaplayan Python kodu aşağıda verilmektedir.

```
import numpy as np

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

nXmin = 0
nHmin = 0

M = len(x)
N = len(h)
L = M + N -1
y=np.zeros(L)

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

nY = [i for i in range(nXmin+nHmin,nXmin+nHmin+L)]
print(nY)
Konsol çıktısı → [ 0   1   2   3]
```

Yukarıdaki kod x ve h dizilerinin konvolüsyonu ile y dizisini üretmektedir. Burada dikkat edilmesi gereken iki unsur var. Bunlardan birincisi, x işaretinin uzunluğu N ve h işaretinin uzunluğu M olursa, y işaretinin uzunluğu $N+M-1$ olur. Dolayısıyla y dizisini hesaplayan dıştaki döngü $N+M-1$ iterasyondan oluşur. İkincisi, $h[n-k]$ hesaplanırken $n-k$'nın sıfırdan küçük ve $n-k$'nın N'den büyük olduğu değerlerde programın hata üretmemesi için bu aralık toplama katılmaz, aşağıda koyu renkle gösterilen if satırı bu amaçla eklenmiştir.

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

Toplam sembolünü hesaplatmak için ise her bir n değeri için başta y[n]'i sıfıra eşitleyip (bknz, aşağıda koyu işaretlenmiş satır) içteki döngüde her bir k değeri için "x[k]*h[n-k]" çarpımı mevcut y[n] değerinin üzerine eklenmektedir.

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

Son olarak ise indis düzeltmesi yapılması gerekmektedir. Yukarıdaki örnekte nx_{min} ve nh_{min} zaten sıfıra eşittir.Dolayısıyla bu örnek için bir indis düzeltmesine gerek olmayıp çıkışta elde edilen dizinin indisi 0,,,,,L arasındadır. Bu iki dizinin herhangi birinin başlangıç indis değerlerinin sıfıra eşit olmadığı durumda indis değerleri 0,,,,,L arasında değer almayıp, yukarıda bahsedildiği gibi $n_{min} = nx_{min}+nh_{min}$ ile $n_{max} = nx_{max}+nh_{max}$ arasında değer alacaktır. Bu durumda indis değerlerini hesaplamak için y dizisinin ilk elemanının indisi sıfır yerine $0+nx_{min}+nh_{min}$ olacaktır. Dolayısıyla, indis değerlerine $nx_{min}+nh_{min}$ değerini eklemek yeterli olacaktır. Aşağıdaki verilen örnekte bu durum gösterilecektir.

Aşağıdaki giriş işareti ve dürtü cevabı için sistemin çıkışını hesaplayalım.

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

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

Bu problem için nx_{min}=-1, nx_{max}=2, nh_{min}=-2 ve $nh_{max} = 2$ olduğundan, y dizisi $n_{min} = nx_{min}+nh_{min}$=-3 ile $n_{max} = nx_{max}+nh_{max}$=4 alt ve üst indis değerleri arasında sıfırdan farklı değer alıp dizinin uzunluğu $n_{max} - n_{min} + 1 = 8$ olacaktır. Yukarıda verilen Python kodunda ilk 5 satır aşağıdaki gibi değiştirilecektir. Bu problemde x dizisinin ilk elemanın indisi normalde -1 olmasına rağmen aşağıda verilen kodda ilk elemanın indisi sanki sıfırmış gibi tanımlanmıştır. Benzer şekilde, h dizisinin ilk elemanın indisi normalde -2 olmasına rağmen aşağıda verilen kodda ilk elemanın indisi sanki sıfırmış gibi tanımlanmıştır. Bu iki durumu işaret etmek üzere x dizisinin başlangıç indisini tutan nXmin değişkenine -1, x dizisinin başlangıç indisini tutan nHmin değişkenine -2 değeri atanmıştır.

```
import numpy as np
x=np.array([1,0,2,-1],dtype=float)
h=np.array([1,0,0,0,3],dtype=float)
nXmin = -1
nHmin = -2
```

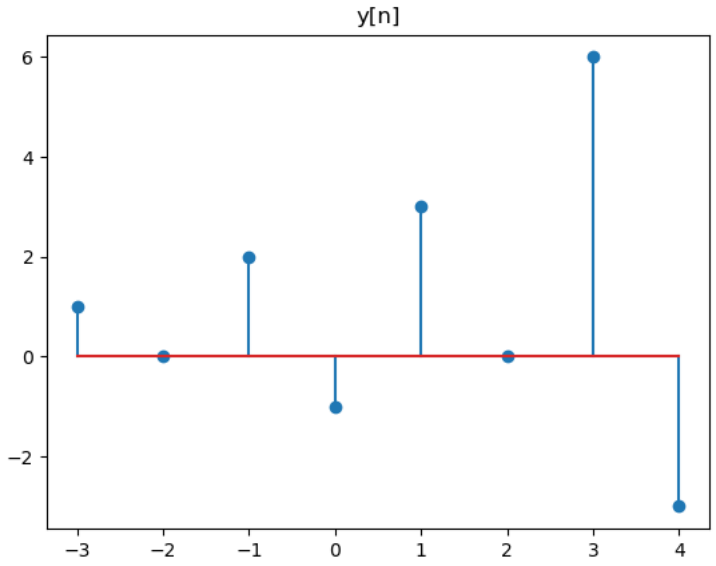
Kod çalıştırıldığında son üç satır şu sonucu verecektir:

```
print(y)
Konsol çıktısı → [ 1.  0.  2. -1.  3.  0.  6. -3.]

nY = [i for i in range(nXmin+nHmin,nXmin+nHmin+L)]
print(nY)

Konsol çıktısı → [-3, -2, -1, 0, 1, 2, 3, 4]
```

y vektörünün ilk elemanı 1 değerindedir ve nY vektörünün ilk değeri olan -3, y vektörünün ilk elemanının indisine karşı gelmektedir: y[-3]=1, y[-2]=0, y[-1]=2, …

Aşağıdaki kod çalıştırılarak elde edilen sonuç çizdirilebilir. Burada, yatay eksendeki değerlere dikkat ediniz.

```
import matplotlib.pyplot as plt
plt.stem(nY,y)
plt.title('y[n]')
```



2.2.2 Hazır komut kullanarak konvolüsyon hesaplama

2.2.1'de verilen *x* ve *h* dizilerinin konvolüsyonlarını Numpy kütüphanesinin veya Scipy kütüphanesinin signal modülündeki *convolve()* fonksiyonu ile hesaplayabilirsiniz.

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

3 ÖDEV-2:

Aşağıdaki sorularda istenenleri Python'da kodlayarak bulunuz. Kodlarınızı .py formatında değil "jupyter notebook" formatı olan .ipynb formatında tek bir dosya halinde teslim ediniz. Rastgele seçilen öğrencilere bu bölümden soru sorulabilir; sorulacak sorulara cevap vermeye hazırlıklı olunuz.

SORU 1: $y[n] - 1.2y[n-1] = x[n] - x[n-4]$ fark denklemiyle ifade edilen DZD bir sistem için aşağıda verilen soruları cevaplayınız.

a) Bu sistemin dürtü cevabını el yordamıyla bulunuz.

SORU 2: Aşağıdaki sisteme (a) ve (b) şıklarında verilen giriş işaretleri uygulandığında bu iki sistemin çıkışında elde edilecek işaretleri bulunuz. Hem giriş işaretinin hem de çıkış işaretinin grafiklerini çizdiriniz. Not: Toplam 2 çıkış işareti elde edeceksiniz.

SİSTEM-1: $h_1[n] = \frac{1}{3}(\delta[n+1] + \delta[n] + \delta[n-1])$

a) $x_1[n] = \cos\left(\frac{\pi}{5}n\right)(u[n-10] - u[n-20])$

b) $x_2[n] = (-0.3)^n(u[n-3] - u[n-5])$

4 ÖDEV-2 EK ÇALIŞMA:

Aşağıdaki sorularda istenenleri Python'da kodlayınız. Kodlarınızı .py formatında değil "jupyter notebook" formatı olan .ipynb formatında hazırlayınız. Ödev-1 Ek Çalışma, Ödev-1'in bir parçasıdır ve teslim edilmesi zorunludur; rastgele seçilen öğrencilere bu bölümden soru sorulabilir; sorulacak sorulara cevap vermeye hazırlıklı olunuz.

SORU 1 (40 PUAN): $y[n] - \mathbf{1.2}y[n-1] = x[n] - x[n-4]$ fark denklemiyle ifade edilen DZD bir sistem için aşağıda verilen soruları cevaplayınız. Not: bu sorudaki sistem yukarıdaki birinci sorudaki sistemdir.

a) Bu sistem kararlı mıdır? Neden?

b) Bu sistemin girişine $x[n] = u[n-3]$ işaretini uygulayın. Bunun için önce $x[n]'$i n=0,1,…10 aralığı için üretip Scipy.signal kütüphanesinin *convolve()* fonksiyonuna ve aynı kütüphanenin *lfilter()* fonksiyonuna uygulayıp çıkış işaretini elde edip çizdirin (indislere dikkat edilmelidir). *convolve()* fonksiyonunu kullanırken üçüncü parametre olarak 'same' girildiğinde giriş ve çıkış işaretlerinin boyları aynı olur.

SORU 1-EK (0 PUAN): (Bu soruyu yapmanız zorunlu değildir, puan değeri yoktur fakat öğrenmenizi sağlar. Yaparsanız cevabınızı **1EK** başlığı altında ipynb dosyasına ekleyiniz):

$y[n] - \mathbf{0.8}y[n-1] = x[n] - x[n-4]$ 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 $x[n] = u[n-3]$ işaretini uygulayın. Bunun için önce $x[n]'$i n=0,1,…10 aralığı için üretip Scipy.signal kütüphanesinin *convolve()* fonksiyonuna ve aynı kütüphanenin *lfilter()* fonksiyonuna uygulayıp çıkış işaretini elde edip çizdirin (indislere dikkat edilmelidir). *convolve()* fonksiyonunu kullanırken üçüncü parametre olarak 'same' girildiğinde giriş ve çıkış işaretlerinin boyları aynı olur. Bu iki fonksiyon farklı sonuç üretti ise sebebini açıklayınız.

SORU-2 (45 PUAN): Aşağıdaki sisteme (a), (b) ve (c) şıklarında verilen giriş işaretleri uygulandığında bu sistemin çıkışında elde edilecek işaretleri bulunuz. Hem giriş işaretinin hem de çıkış işaretinin grafiklerini çizdiriniz. Not: Toplam 3 çıkış işareti elde edeceksiniz.

SİSTEM-2: $h_2[n] = \delta[n-3] - \delta[n-4])$

a) $x_3[n] = \cos\left(\frac{\pi}{5}n\right)(u[n] - u[n-20])$

b) $x_4[n] = (-0.3)^n(u[n-1] - u[n-5])$

c) $x_5[n] = n * u[n]$

SORU-3 (15 PUAN): Bu ödevin size kazandırdıklarını özetleyiniz (en az 5 cümle, en az 256 karakter).

5 TESLİM ŞEKLİ ve ZAMANI

Bu dokümanda verilen örnek kodları kendiniz bir Jupyter Notebook'ta yazarak verilen sonuçlarla karşılaştırınız. Aynı dokümanın devamında olacak şekilde, ÖDEV-2 başlığı altında verilen sorularda istenenleri Python'da (Jupyter Notebook kullanarak) kodlayınız. Yaptığınız çalışmayı aşağıdaki formata uygun isimle kaydediniz:

ÖDEV-2: **ÖDEV2_OgrenciKOD.ipynb**

Ayrı bir doküman olarak ÖDEV-2 EK ÇALIŞMA başlığı altında verilen sorularda istenenleri Python'da kodlayınız. Yaptığınız çalışmayı aşağıdaki formata uygun isimle kaydediniz:

ÖDEV-2 EK ÇALIŞMA: **ÖDEV2_EK_OgrenciKOD.ipynb**

Bu iki .ipynb uzantılı dosyayı GTUZEM'e yükleyiniz. Sisteme geç yüklenen dosyalar kabul edilmeyecektir. Jupyter Notebook'ta yapacağınız çözümler birinci ödevde verilen **ÖDEV1_OgrenciKOD.ipynb** isimli **şablona göre hazırlanmalıdır. Ödev-1 yazan yerleri Ödev-2 olarak düzenlemeyi unutmayınız.**