EEE 3005 Signals and Systems

EXPERIMENT/HOMEWORK #1

CONTINUOUS AND DISCRETE TIME SIGNAL PLOTS

Student Name

Ahmad Zameer Nazarı

220702706

Name of Lecturers

Dr. Öğr. Üyesi Kutlu Karayahşi

Arş. Gör. Ali Can Erüst

Date:

29/10/2024 – 04/11/2024

[EEE-3005]. HW1. 220702706 - Ahmad Zameer Nazarı

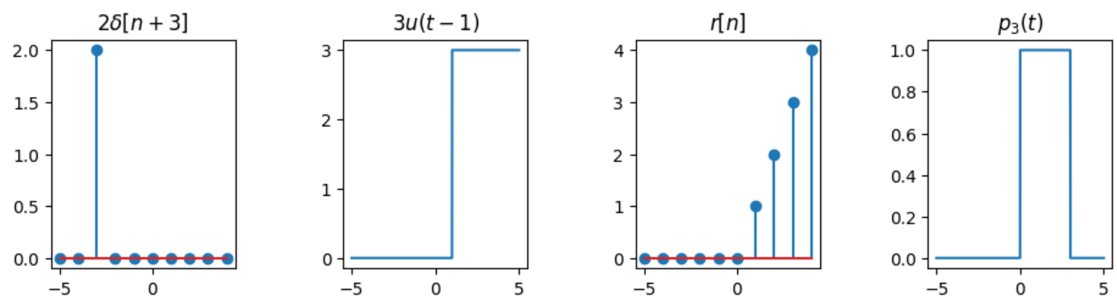
November 4, 2024

1 1ST ASSIGNMENT

[1]:
```python
# importing some essential libraries
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
```

[2]:
```python
# defining some elementary signals for repeated use

# unit step
def step(t,pos):
    return np.heaviside(t+pos,1)

# unit impulse
def impulse(t, pos):
    return np.where(t == -pos, 1, 0)

# unit ramp
def ramp(t):
    sig = []
    for i in range(len(t)):
        value = (t[i] if t[i]>=0 else 0)
        sig.append(value)
    return sig

# rectangular pulse
def rect_pulse(t, tau):
    return np.heaviside(t,1) - np.heaviside(t-tau,1)



# testing with some random values
t = np.linspace(-5,5,1000)
n = np.arange(-5,5).astype(int)

impulse_test = 2*impulse(n,3)
step_test = 3*step(t,-1)
```

```
ramp_test = ramp(n)
pulse_test = rect_pulse(t, 3)

fig, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
fig.tight_layout(pad=3)
plt.rc('axes.spines', **{'bottom':True, 'left':True, 'right':False, 'top':
  ↪False})
plt.rcParams.update({"text.usetex": True})

ax1 = plt.subplot(141)
ax1.stem(n,impulse_test)
plt.title('$2\delta[n+3]$')

ax2 = plt.subplot(142)
ax2.plot(t,step_test)
plt.title('$3u(t-1)$')

ax3 = plt.subplot(143)
ax3.stem(n,ramp_test)
plt.title('$r[n]$')

ax4 = plt.subplot(144)
ax4.plot(t, pulse_test)
plt.title('$p_3(t)$')

plt.show()
```



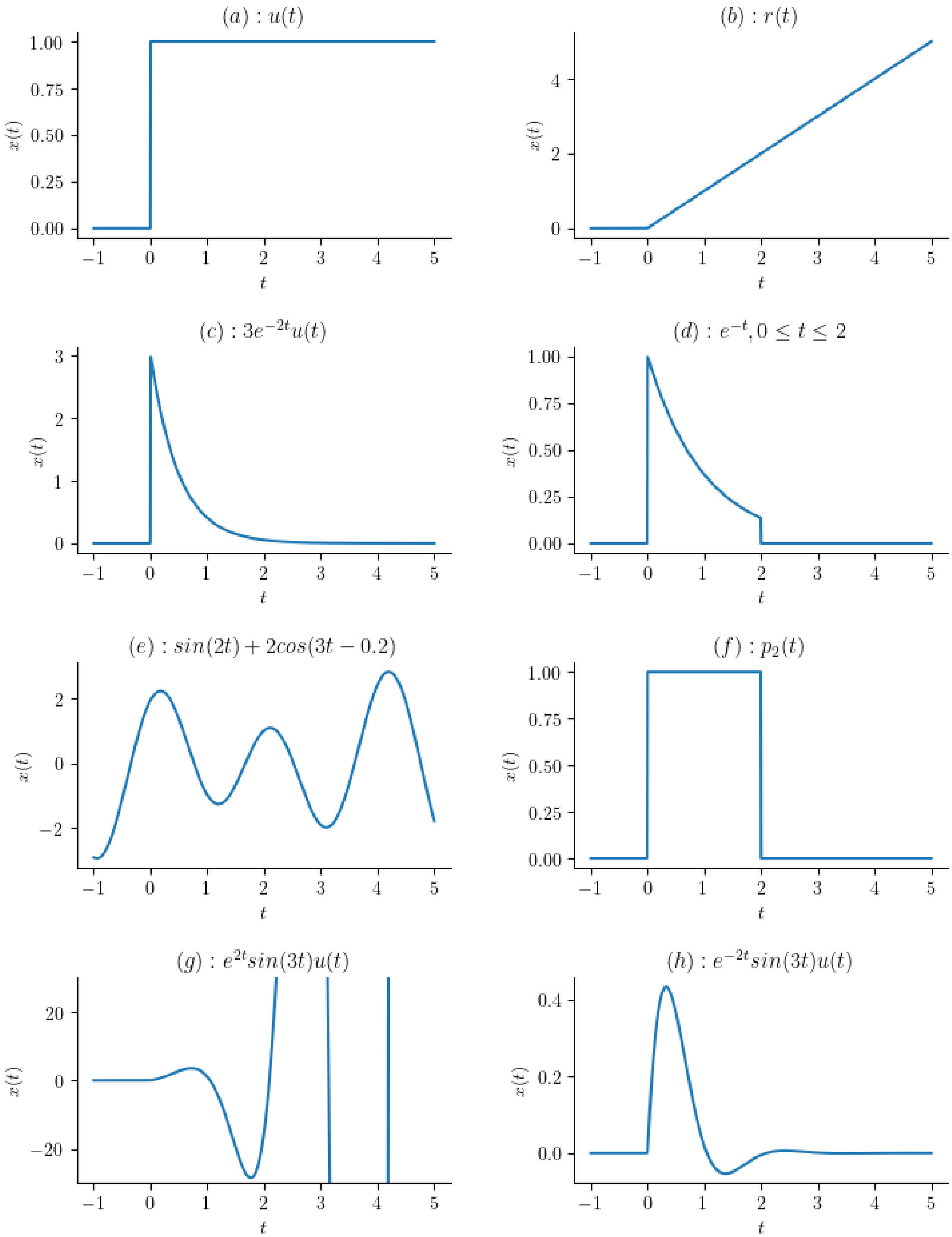
1.1 Question #1

plotting continuous time signals

1.1.1 *parts (a) to (h)*

[3]:
```
# defined time interval for 1st question
t = np.linspace(-1,5,1000)



# INITIALIZING SIGNALS a to h.
u_t = step(t,0)
r_t = ramp(t)
x_1c = 3*np.exp(-2*t)*u_t

def x_1d(t):
    signal = []
    for i in range(len(t)):
        if t[i] >= 0 and t[i]<=2:
            value = np.exp(-t[i])
        else:
            value = 0
        #value = (np.exp(-t[i]) if t[i] >= 0 && t[i]<=2 else 0)
        signal.append(value)
    return signal

x_1e = np.sin(2*t) + 2*np.cos(3*t-0.2)
x_1f = rect_pulse(t, 2)
x_1g = np.exp(2*t)*np.sin(3*t)*u_t
x_1h = np.exp(-2*t)*np.sin(3*t)*u_t



# PLOTTING
fig, axes = plt.subplots(nrows=4, ncols=2, figsize=(8, 10))
fig.tight_layout(pad=4)
plt.rc('axes.spines', **{'bottom':True, 'left':True, 'right':False, 'top':
 ↪False})
for ax in axes.flat:
    ax.set(xlabel='$t$', ylabel='$x(t)$')
    ax.xaxis.set_major_locator(ticker.MultipleLocator(1))

# Part a
ax1 = plt.subplot(421)
ax1.plot(t,u_t)
plt.title('$(a): u(t)$')

# Part b
ax2 = plt.subplot(422)
ax2.plot(t,r_t)
plt.title('$(b): r(t)$')
```

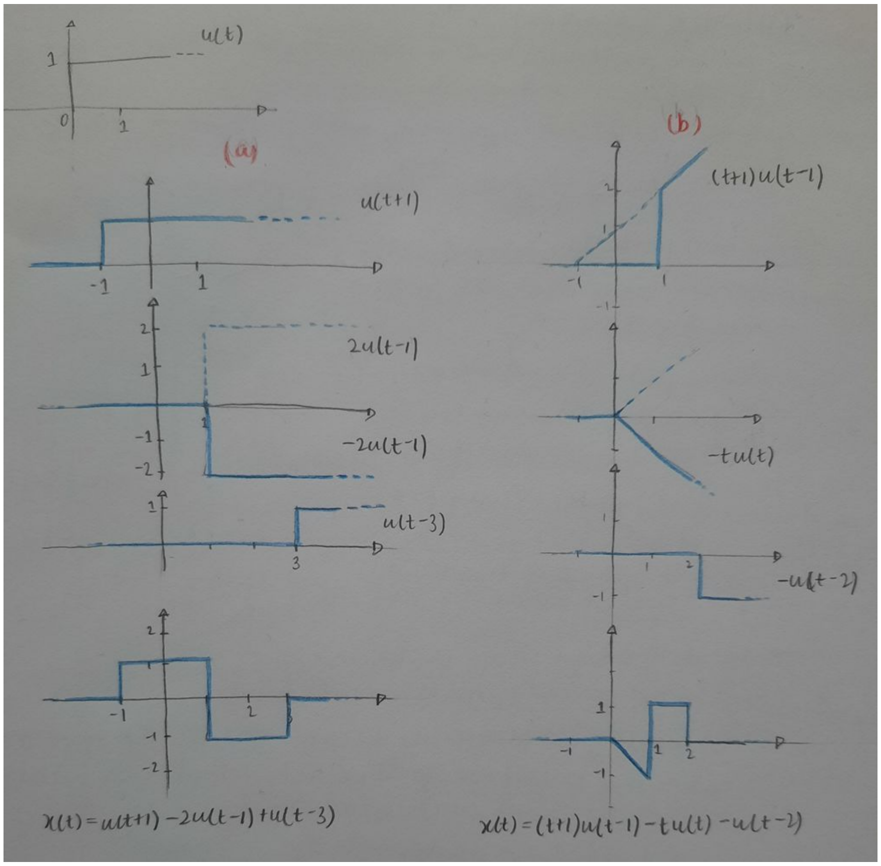
```python
# Part c
ax3 = plt.subplot(423)
ax3.plot(t,x_1c)
plt.title('$(c): 3e^{-2t}u(t)$')

# Part d
ax4 = plt.subplot(424)
ax4.plot(t, x_1d(t))
plt.title('$(d): e^{-t} , 0 \leq t \leq 2$')

# Part e
ax5 = plt.subplot(425)
ax5.plot(t,x_1e)
plt.title('$(e): sin(2t)+2cos(3t-0.2)$')

# Part f
ax6 = plt.subplot(426)
ax6.plot(t, x_1f)
plt.title('$(f):p_2(t)$')

# Part g
ax7 = plt.subplot(427)
ax7.plot(t,x_1g)
plt.ylim(-30,30) # the function has very abrupt fluctuations so i scaled y-axis␣
↪down, so it could be better viewed
plt.title('$(g): e^{2t}sin(3t)u(t)$')

# Part h
ax8 = plt.subplot(428)
ax8.plot(t,x_1h)
plt.title('$(h): e^{-2t}sin(3t)u(t)$')

plt.show()
```



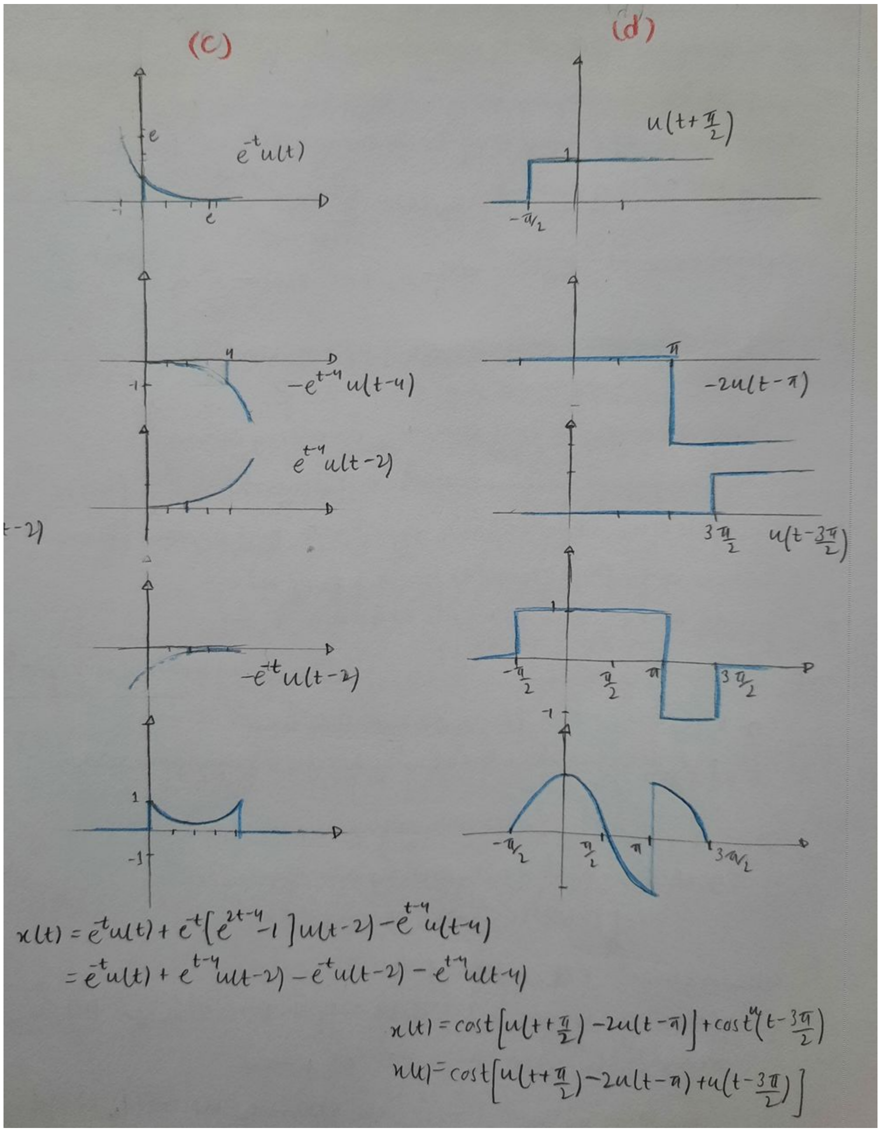
1.2 Question #2

sketching continuous time signals

1.2.1 *parts (a) & (b)*


u(t)
(a)
u(t+1)
2u(t-1)
-2u(t-1)
u(t-3)
x(t) = u(t+1) - 2u(t-1) + u(t-3)
(b)
(t+1)u(t-1)
-tu(t)
-u(t-2)
x(t) = (t+1)u(t-1) - tu(t) - u(t-2)

1.2.2 *parts (c) & (d)*

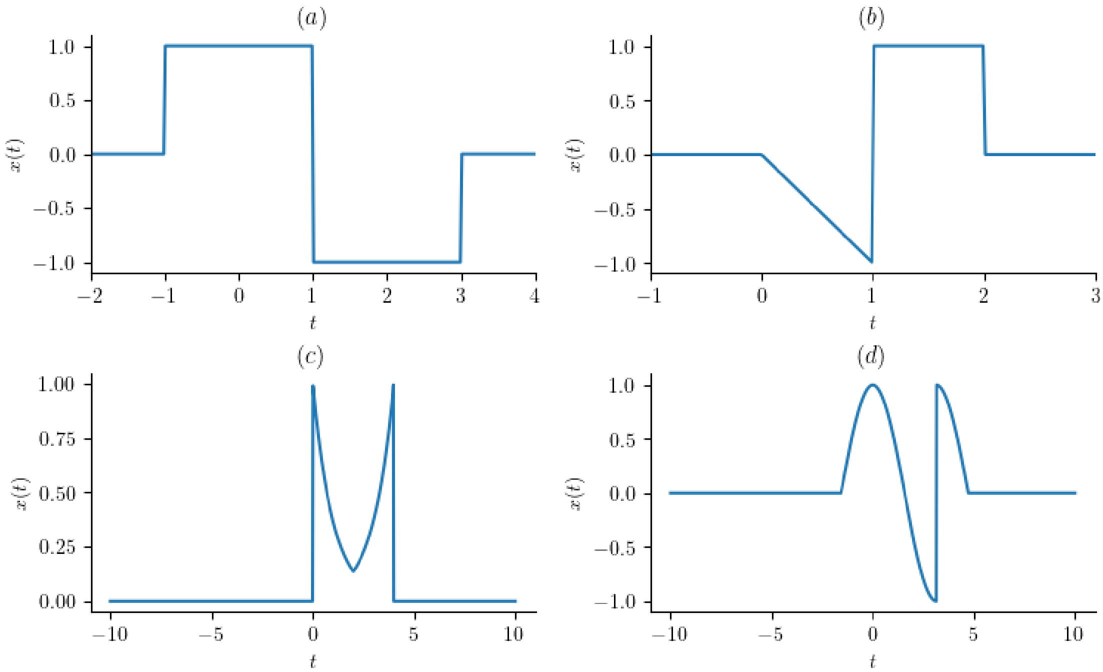


1.2.3 *part (e)*

plotting continous time signals (a) to (d)

[4]:
```
# defining time interval
t = np.linspace(-10,10,1000)



# defining signals
x_2a = step(t,1) - 2*step(t,-1) + step(t,-3)
x_2b = (t+1)*step(t,-1) - t*step(t,0) - step(t,-2)
x_2c = np.exp(-t)*step(t,0) + np.exp(-t)*(np.exp((2*t)-4)-1)*step(t,-2) - np.
 ↪exp(t-4)*step(t,-4)
```

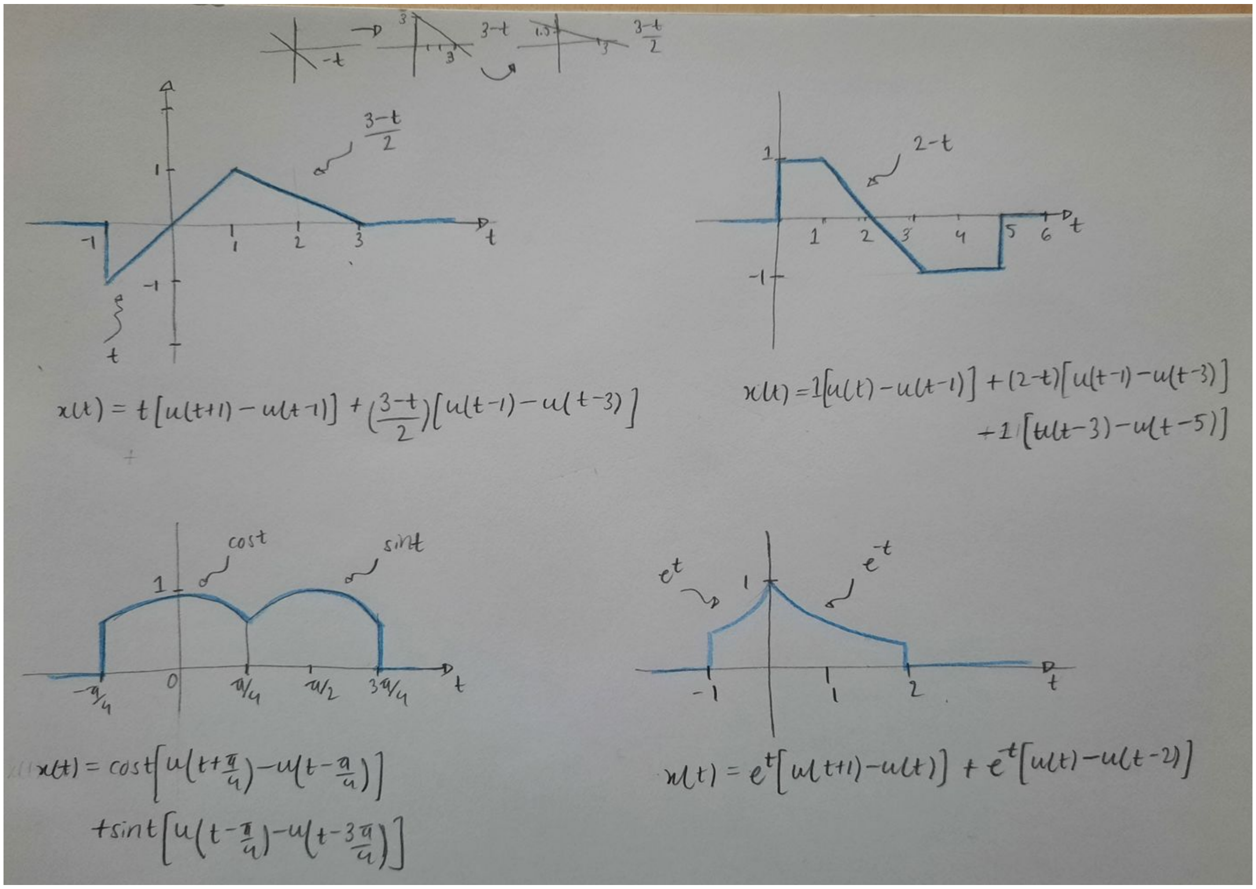
```python
x_2d = np.cos(t)*(step(t,(np.pi/2)) - 2*step(t,-np.pi)) + np.
 ↪cos(t)*step(t,-(3*np.pi)/2)


# PLOTTING
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(8,5))
fig.tight_layout(pad=3.0)
plt.rc('axes.spines', **{'bottom':True, 'left':True, 'right':False, 'top':
 ↪False})
for ax in axes.flat:
    ax.set(xlabel='$t$', ylabel='$x(t)$')

# Part a
ax1 = plt.subplot(221)
ax1.plot(t,x_2a)
plt.title('$(a)$')
plt.xlim(-2,4)

# Part b
ax2 = plt.subplot(222)
ax2.plot(t,x_2b)
plt.title('$(b)$')
plt.xlim(-1,3)

# Part c
ax3 = plt.subplot(223)
ax3.plot(t,x_2c)
plt.title('$(c)$')

# Part d
ax4 = plt.subplot(224)
ax4.plot(t,x_2d)
plt.title('$(d)$')

plt.show()
```



1.3 Question #3

determining the signals.

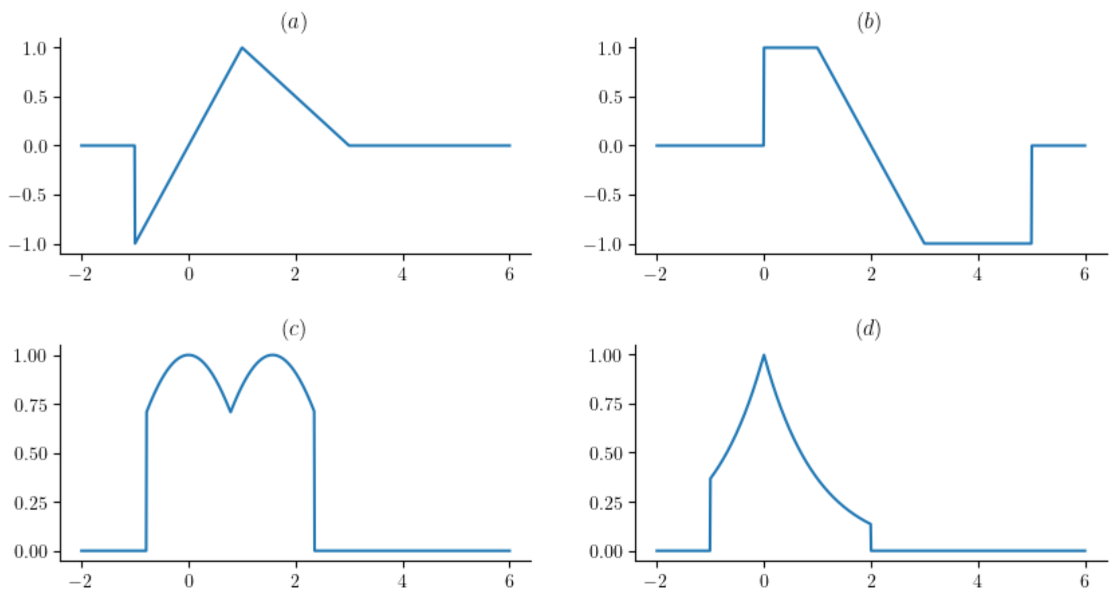
1.3.1 *parts (a) to (d)*



verifying the determined signals for parts (a) to (d)

[5]:
```
# defining interval
t = np.linspace(-2,6,1000)

# defining signals
x_3a = t*(step(t,1)-step(t,-1)) + ((3-t)/2)*(step(t,-1)-step(t,-3))
x_3b = step(t,0) - step(t,-1) + (2-t)*(step(t,-1)-step(t,-3)) - step(t,-3) +␣
↪step(t,-5)
x_3c = np.cos(t)*(step(t, np.pi/4)-step(t,-np.pi/4)) + np.sin(t)*(step(t,-np.pi/
↪4)-step(t,-0.75*np.pi))
x_3d = np.exp(t)*(step(t,1)-step(t,0)) + np.exp(-t)*(step(t,0)-step(t,-2))



# plotting
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(9,5))
fig.tight_layout(pad=3.0)
plt.rc('axes.spines', **{'bottom':True, 'left':True, 'right':False, 'top':
↪False})
```

```python
# Part a
ax1 = plt.subplot(221)
ax1.plot(t,x_3a)
plt.title('$(a)$')

# Part b
ax2 = plt.subplot(222)
ax2.plot(t,x_3b)
plt.title('$(b)$')

# Part c
ax3 = plt.subplot(223)
ax3.plot(t,x_3c)
plt.title('$(c)$')

# Part d
ax4 = plt.subplot(224)
ax4.plot(t,x_3d)
plt.title('$(d)$')

plt.show()
```



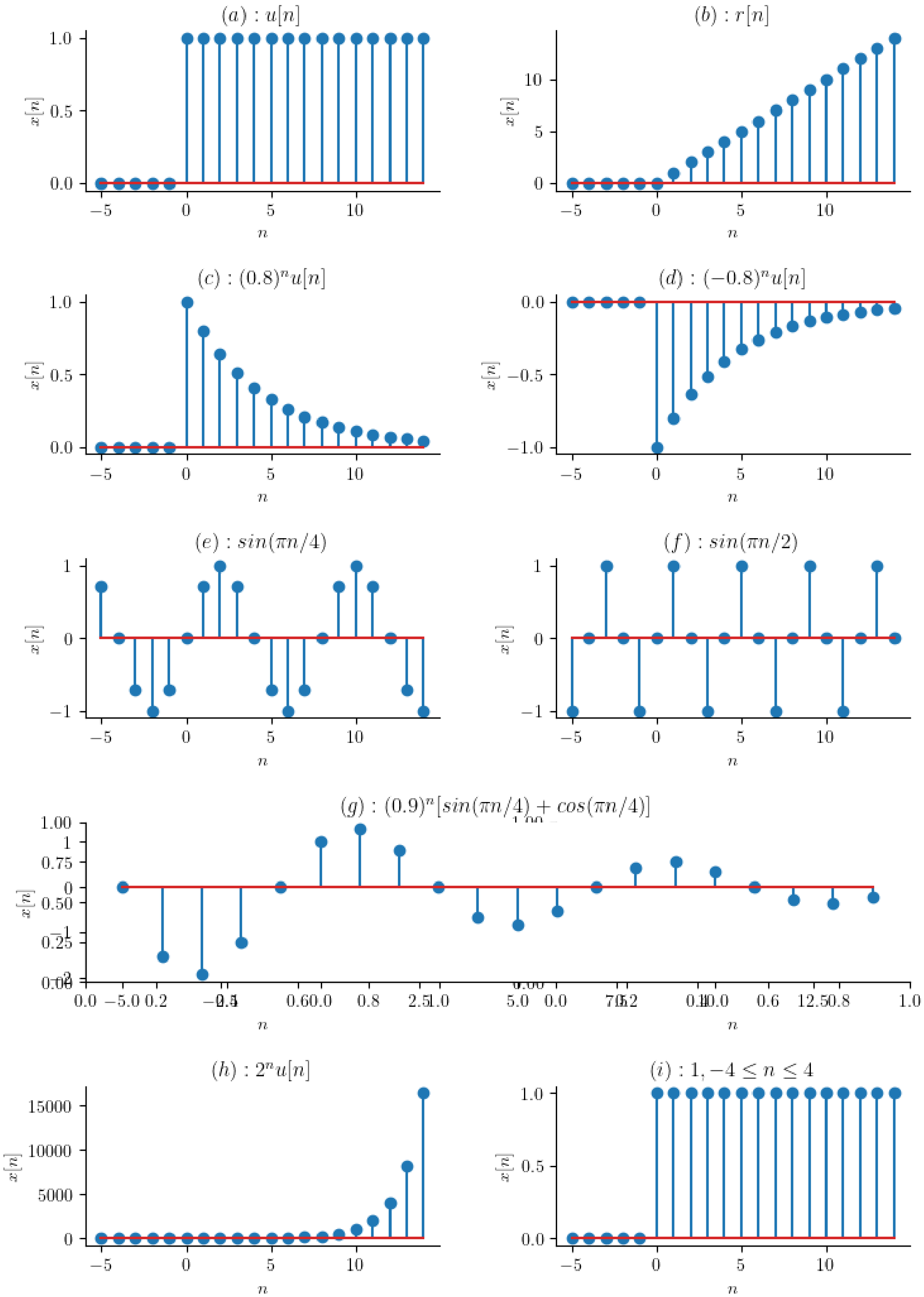
1.4 Question #4

plotting discrete time signals

1.4.1 *parts (a) to (i)*

[6]:
```
# defined time interval for 1st question
n = np.arange(-5,15).astype(float)


# initializing discrete time signals
u_n = step(n,0)
r_n = ramp(n)
x_4c = (0.8**n)*u_n
x_4d = (-0.8**n)*u_n
x_4e = np.sin((np.pi*n)/4)
x_4f = np.sin((np.pi*n)/2)
x_4g = (0.9**n)*( np.sin((np.pi*n)/4) + np.cos((np.pi*n)/4) )
x_4h = (2**n)*u_n
x_4i = u_n


# PLOTTING
fig, axes = plt.subplots(nrows=5, ncols=2, figsize=(8, 11))
fig.tight_layout(pad=4)
plt.rc('axes.spines', **{'bottom':True, 'left':True, 'right':False, 'top':
 ↪False})
for ax in axes.flat:
    ax.set(xlabel='$n$', ylabel='$x[n]$')

# Part a
ax1 = plt.subplot(521)
ax1.stem(n,u_n)
plt.title('$(a): u[n]$')

# Part b
ax2 = plt.subplot(522)
ax2.stem(n,r_n)
plt.title('$(b): r[n]$')

# Part c
ax3 = plt.subplot(523)
ax3.stem(n,x_4c)
plt.title('$(c): (0.8)^{n}u[n]$')

# Part d
ax4 = plt.subplot(524)
ax4.stem(n,x_4d)
plt.title('$(d): (-0.8)^{n}u[n]$')

# Part e
```

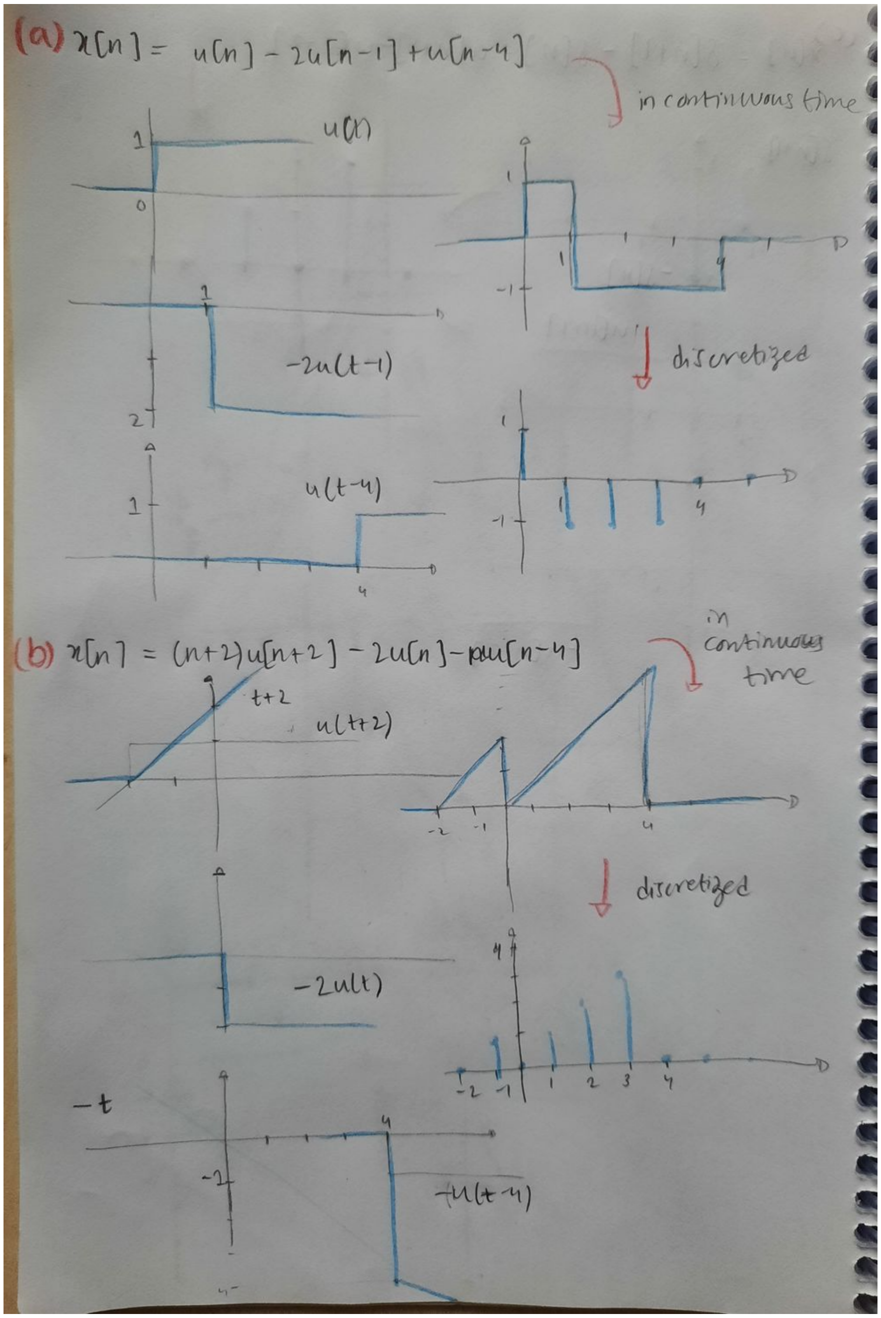
```
ax5 = plt.subplot(525)
ax5.stem(n,x_4e)
plt.title('$ (e): sin( \pi n /4 )$')

# Part f
ax6 = plt.subplot(526)
ax6.stem(n,x_4f)
plt.title('$(f): sin(\pi n /2 )$')

# comparing parts (e) and (f), their plots show the significance ot sampling i␣
↪believe

# Part g
ax7 = plt.subplot(5,2, (7,8))
ax7.stem(n,x_4g)
plt.title('$(g): (0.9)^{n}[sin(\pi n / 4 ) + cos( \pi n /4 )]$')

# Part h
ax8 = plt.subplot(529)
ax8.stem(n,x_4h)
plt.title('$(h): 2^{n}u[n] $')

# Part i
ax8 = plt.subplot(5,2,10)
ax8.stem(n,x_4i)
plt.title('$(i): 1 , -4 \leq n \leq 4$')

plt.show()
```


(a) : u[n]
(b) : r[n]
(c) : (0.8)^n u[n]
(d) : (−0.8)^n u[n]
(e) : sin(πn/4)
(f) : sin(πn/2)
(g) : (0.9)^n [sin(πn/4) + cos(πn/4)]
(h) : 2^n u[n]
(i) : 1, −4 ≤ n ≤ 4
x[n]
n

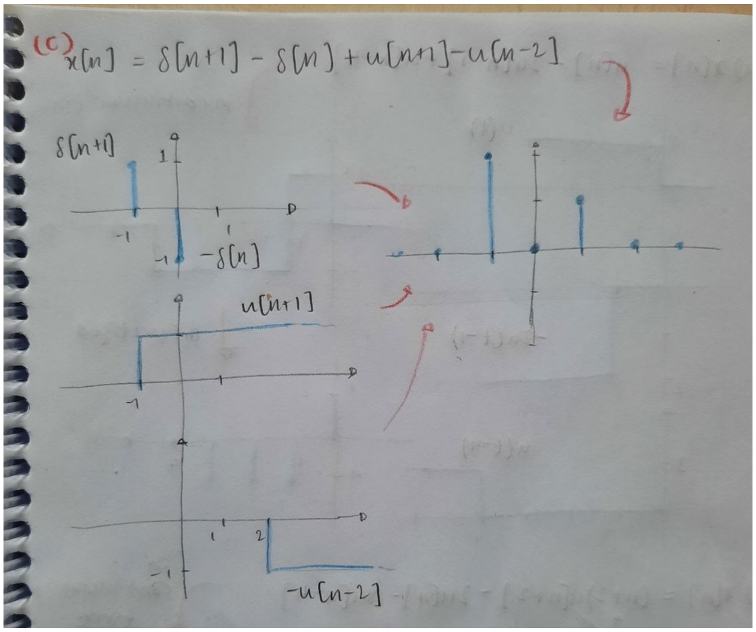
1.5 Question #5

sketching discrete time signals (a) to (d)

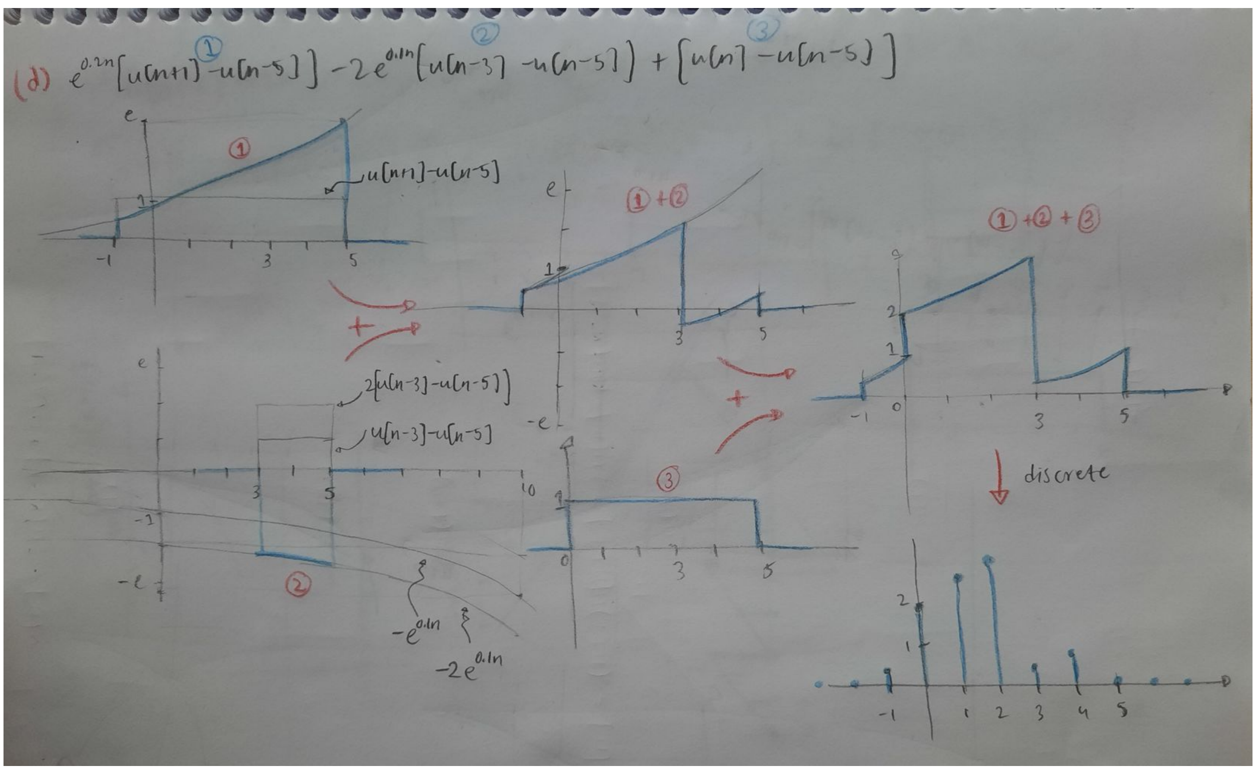
1.5.1 *parts (a) & (b)*



1.5.2 *part (c)*



1.5.3 *part (d)*

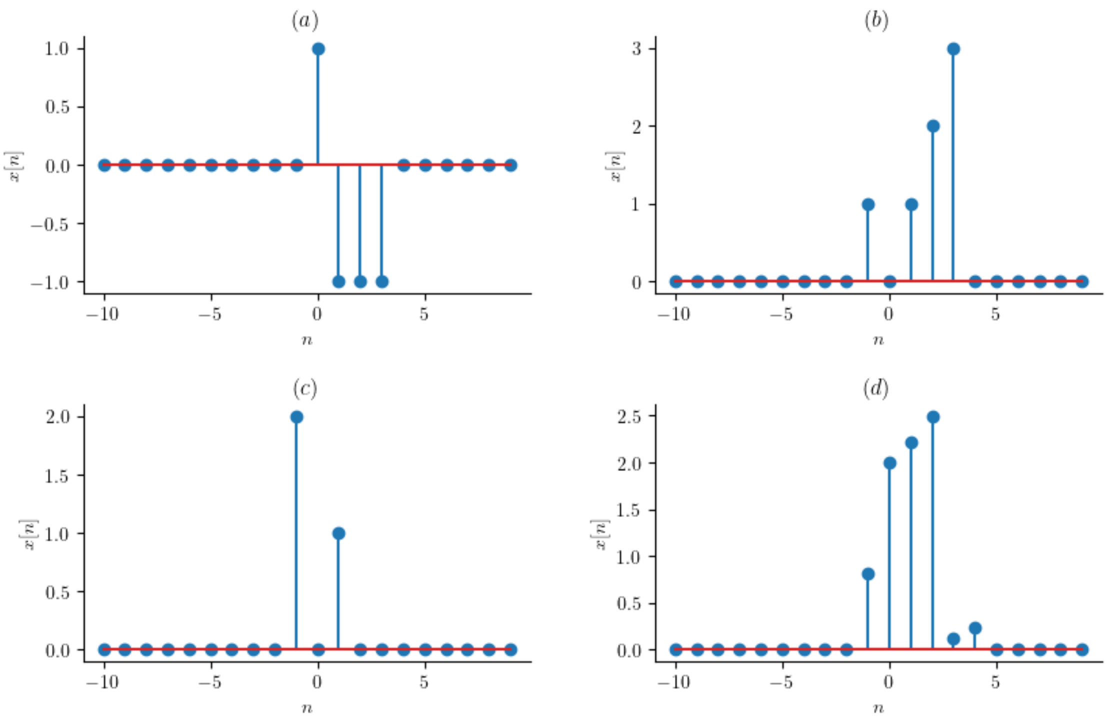


1.5.4 *part (e)*

plotting discrete time signals of parts (a) to (d)

[7]:

```python
# defined time interval for 1st question
n = np.arange(-10,10).astype(int)


# initializing discrete time signals
x_5a = step(n,0) - 2*step(n,-1) + step(n,-4)
x_5b = (n+2)*step(n,2) - 2*step(n,0) - n*step(n,-4)
x_5c = impulse(n,1) - impulse(n,0) + step(n,1) - step(n,-2)
x_5d = np.exp(0.2*n)*step(n,1) + step(n,0) - 2*np.exp(0.1*n)*step(n,-3) -␣
 ↪((1-np.exp(0.1*n))**2)*step(n,-5)


# PLOTTING
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(9, 6))
fig.tight_layout(pad=4.0)
plt.rc('axes.spines', **{'bottom':True, 'left':True, 'right':False, 'top':
 ↪False})
for ax in axes.flat:
    ax.set(xlabel='$n$', ylabel='$x[n]$')

# Part a
ax1 = plt.subplot(221)
ax1.stem(n,x_5a)
plt.title('$(a)$')

# Part b
ax2 = plt.subplot(222)
ax2.stem(n,x_5b)
plt.title('$(b)$')

# Part c
ax3 = plt.subplot(223)
ax3.stem(n,x_5c)
plt.title('$(c)$')

# Part d
ax3 = plt.subplot(224)
ax3.stem(n,x_5d)
plt.title('$(d)$')

plt.show()
```



all files available at: https://github.com/az-yugen/EEE-3005.-Signals-Systems-LAB