Signals in Time Domain

March 3, 2024

```
Index Number -  220067G / 220071M
Group        -  A6
Date         -  01/03/2024
```

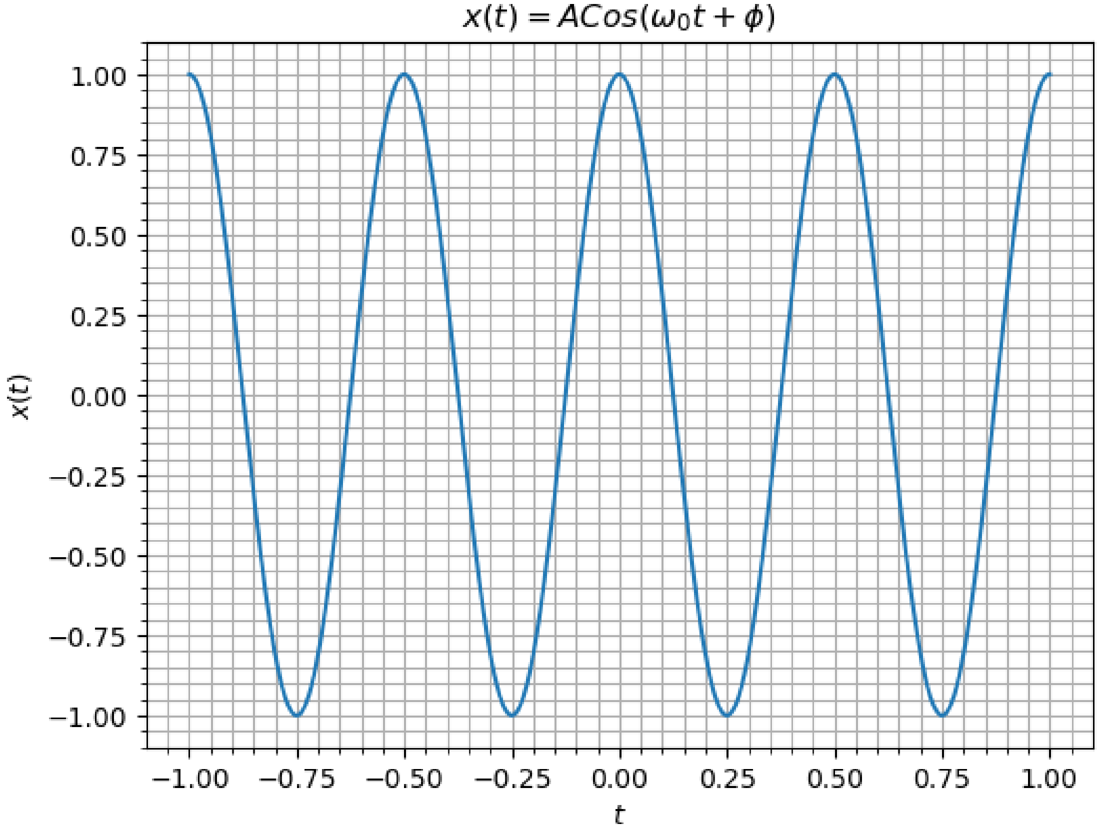
1 Workshop 1: Signals in Time Domain

[1]:
```
import numpy as np      #importing numpy library
import matplotlib.pyplot as plt     #importing matplotlib library
%matplotlib inline
```

1.1 Real CT Sinusoidal Signal

$x(t) = ACos(\omega_0 t + \phi)$

[2]:
```
fs=44100 #sampling frequency
ts=1/fs #sampling time
t=np.arange(-1.,1.,ts) #time for 2 seconds
A=1.0 #Amplitude
f=2.0 #Frequency

x=A*np.cos(2*np.pi*f*t) #Signal

fig,ax=plt.subplots()
ax.plot(t,x)

ax.set(xlabel='$t$',ylabel='$x(t)$')  #setting x and y labels
ax.minorticks_on()
ax.grid(visible=True, which="both")   #enabling minor grid
ax.title.set_text(r'$x(t)= ACos(\omega_{0}t+\phi)$')   #setting title of the␣
 ↪plot
plt.show()
```

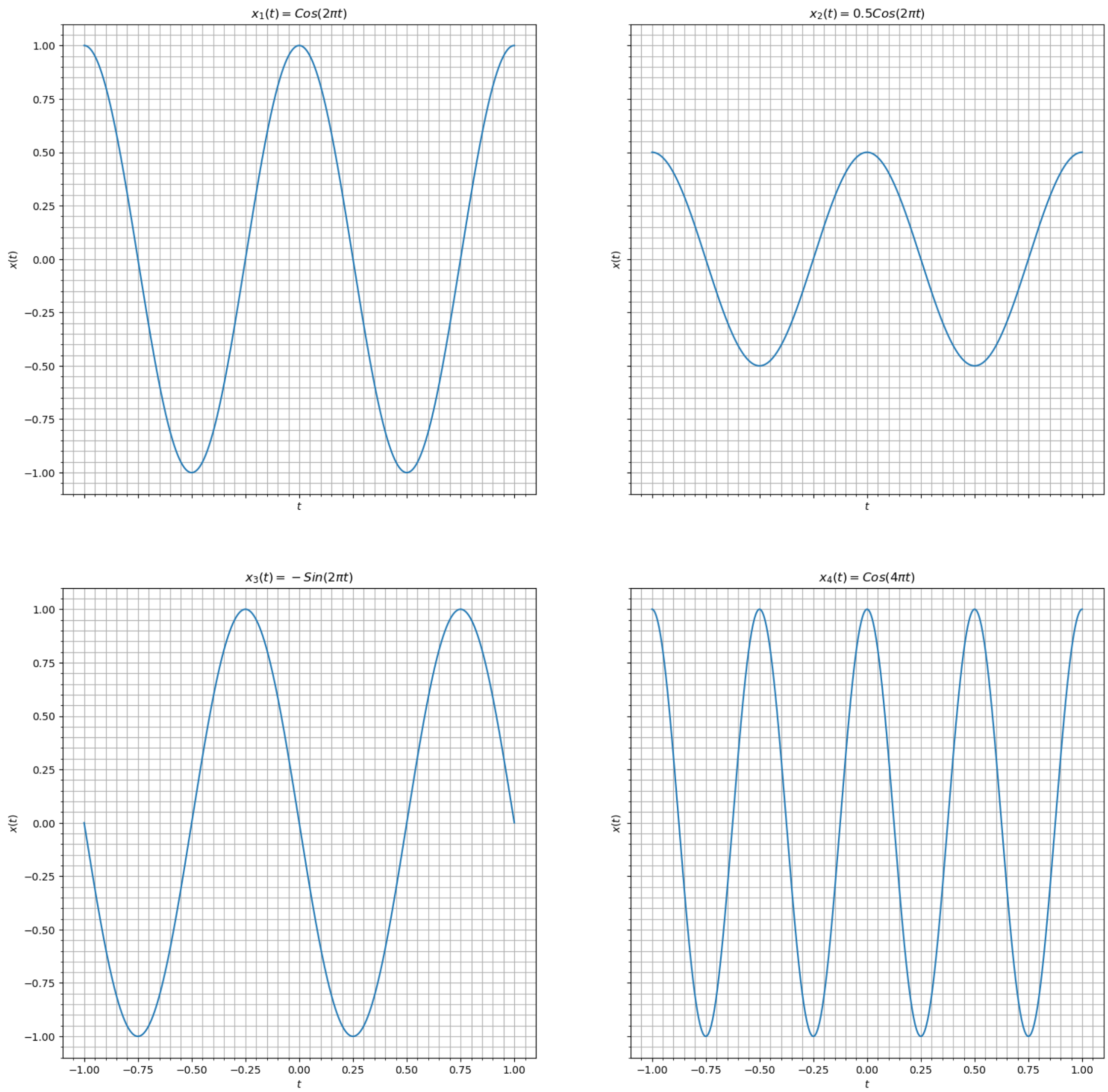


1.1.1 Task 1 - Plot following Signals

1. $x_1(t) = cos(2\pi t)$
2. $x_2(t) = 0.5cos(2\pi t)$
3. $x_3(t) = -sin(2\pi t)$
4. $x_4(t) = cos(4\pi t)$

[4]:
```
fs = 44100 #44,100-Hz sampling frequency
ts = 1/fs
t = np.arange(-1., 1., ts) #A linearly-spaced array with step ts
f = 2 # 2 Hz
fig, axes = plt.subplots(2,2, sharex='all', sharey='all', figsize=(18,18))

#####################  Signals  #####################

x1=np.cos(2*np.pi*t)
x2=0.5*np.cos(2*np.pi*t)
x3=-np.sin(2*np.pi*t)
x4=np.cos(4*np.pi*t)
```

```python
######################################################

#plotting x1
axes[0,0].plot(t, x1)
axes[0,0].set_title(r'$x_{1}(t)=Cos(2\pi t)$')    #setting title of the plot
axes[0,0].set(xlabel='$t$',ylabel='$x(t)$')       #setting x and y labels
axes[0,0].minorticks_on()
axes[0,0].grid(visible=True, which='both')        #enabling grid

#plotting x2
axes[0,1].plot(t, x2)
axes[0,1].set_title(r'$x_{2}(t)=0.5Cos(2\pi t)$')  #setting title of the plot
axes[0,1].set(xlabel='$t$',ylabel='$x(t)$')        #setting x and y labels
axes[0,1].minorticks_on()
axes[0,1].grid(visible=True, which='both')         #enabling grid

#plotting x3
axes[1,0].plot(t, x3)
axes[1,0].set_title(r'$x_{3}(t)=-Sin(2\pi t)$')    #setting title of the plot
axes[1,0].set(xlabel='$t$',ylabel='$x(t)$')        #setting x and y labels
axes[1,0].minorticks_on()
axes[1,0].grid(visible=True, which='both')         #enabling grid

#plotting x4
axes[1,1].plot(t, x4)
axes[1,1].set_title(r'$x_{4}(t)=Cos(4\pi t)$')     #setting title of the plot
axes[1,1].set(xlabel='$t$',ylabel='$x(t)$')        #setting x and y labels
axes[1,1].minorticks_on()
axes[1,1].grid(visible=True, which='both')         #enabling grid


plt.show()
```

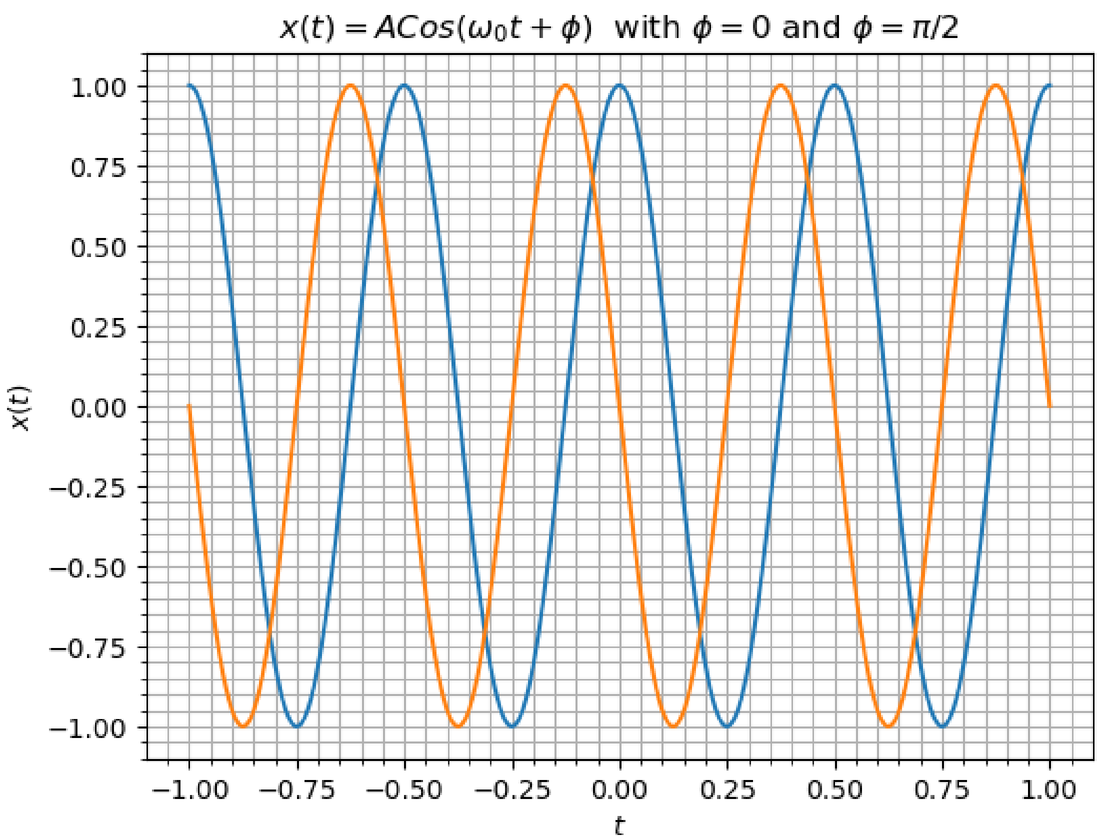


$x_1(t)$ is an even signal.
$x_2(t)$ is an even signal.
$x_3(t)$ is an odd signal.
$x_4(t)$ is an even signal.

1.1.2 Plotting 2 waves in a same plot

[2]:
```
fs = 4.4e4                    #44KHz sampling frequency
ts = 1/fs
t = np.arange(-1., 1., ts)   #A linearly-spaced array with step ts
f = 2                         #2 Hz
w0=2*np.pi*f                  #angular frequency
```

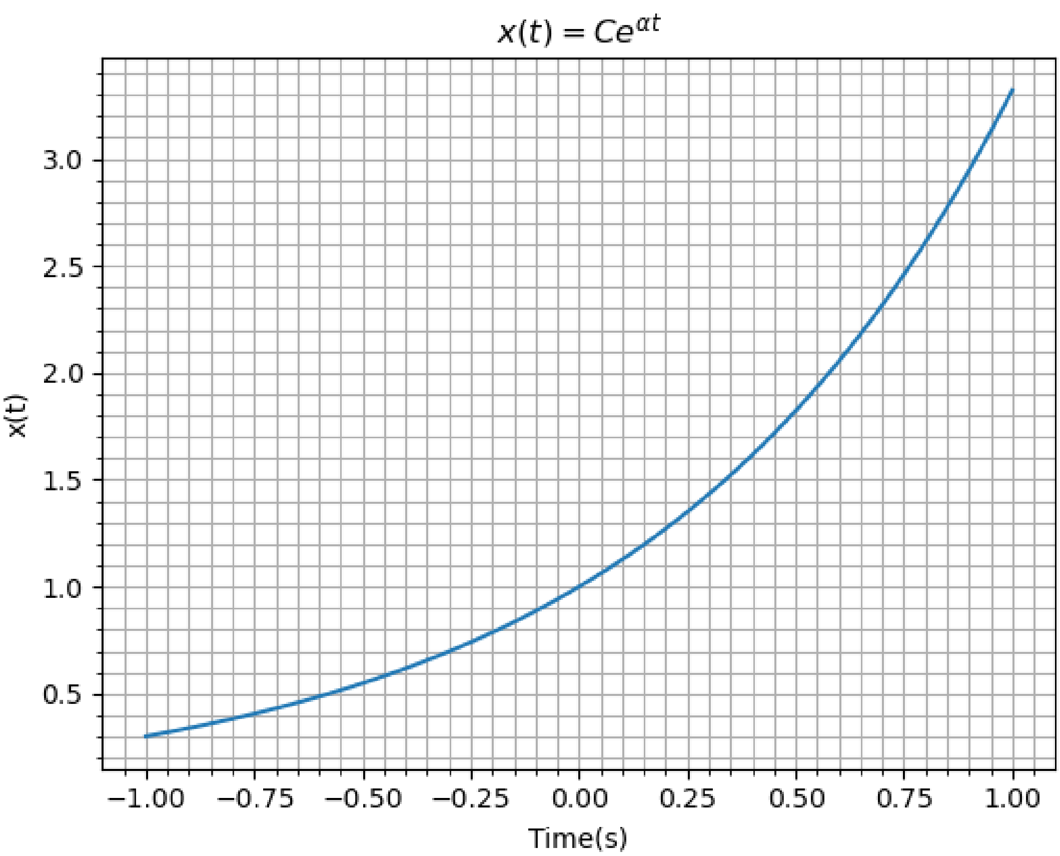
```
phi=0                        #phase
x1=np.cos(w0*t+phi)          #Signal-1

phi=np.pi/2
x2=np.cos(w0*t+phi)          #Signal-2


fig,ax=plt.subplots()
ax.plot(t,x1,label=r'$x(t)=Cos(\omega_{0}t)$')         #plotting x1
ax.plot(t,x2,label=r'$x(t)=Cos(\omega_{0}t+\pi/2)$')   #plotting x2
ax.set(xlabel='$t$',ylabel='$x(t)$')                   #setting x and y labels
ax.minorticks_on()
ax.grid(visible=True, which='both')                    #enabling grid

ax.set_title(r'$x(t)=ACos(\omega_{0}t+\phi)$  with $\phi=0$ and $\phi=\pi/2$')

plt.show()
```



1.2 Real CT Exponential

$x(t) = Ce^{\alpha t}$

[4]:
```
fs=4.4e4 #44KHz sampling frequency
ts=1/fs #sampling time
t=np.arange(-1.,1.,ts)

C=1
a=1.2

xt=C*np.exp(a*t) #Signal

fig,ax=plt.subplots()

ax.plot(t,xt)
ax.set(xlabel='t',ylabel='x(t)')      #setting x and y labels
ax.minorticks_on()
ax.grid(visible=True, which='both')         #enabling grid
ax.title.set_text(r'$x(t)=Ce^{\alpha t}$') #setting title of the plot

plt.show()
```

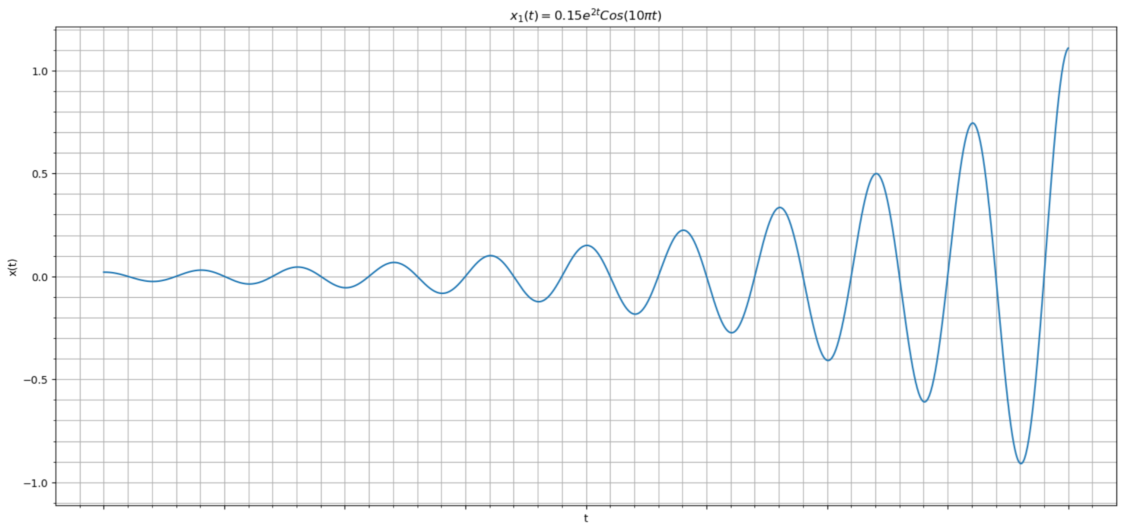


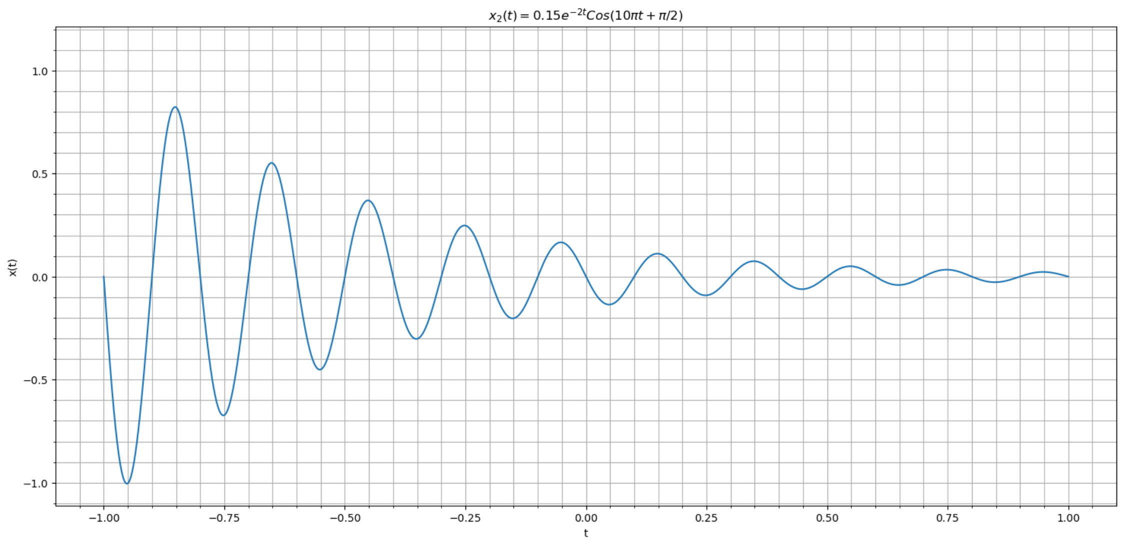
when α is positive signal grows with t
when α is negative signal decays with t

1.3 Growing Sinusoidal Signal $\quad x(t) = Ce^{rt}cos(\omega_0 t + \theta)$

1. $x_1(t) = 0.15e^{2t}Cos(10\pi t)$
2. $x_2(t) = 0.15e^{-2t}Cos(10\pi t + \pi/2)$

[3]:
```
fs=4.4e4 #44KHz sampling frequency
ts=1/fs  #sampling time
t=np.arange(-1.,1.,ts)
f=5 #5Hz

xt1=0.15*np.exp(2*t)*np.cos(2*np.pi*f*t)            #Signal_1
xt2=0.15*np.exp(-2*t)*np.cos(2*np.pi*f*t+np.pi/2)   #Signal_2

fig,ax=plt.subplots(2,1,sharex="all",sharey="all",figsize=(18,18)) #creating␣
 ↪subplots

ax[0].plot(t,xt1)                                 #plotting xt1
ax[0].set(xlabel='t',ylabel='x(t)')         #setting x and y labels
ax[0].minorticks_on()
ax[0].grid(visible=True, which='both')                      #enabling grid
ax[0].title.set_text(r'$x_{1}(t)=0.15e^{2t}Cos(10\pi t)$') #setting title of␣
 ↪the plot

ax[1].plot(t,xt2)                            #plotting xt2
ax[1].set(xlabel='t',ylabel='x(t)')    #setting x and y labels
ax[1].minorticks_on()
ax[1].grid(visible=True, which='both')                            #enabling grid
ax[1].title.set_text(r'$x_{2}(t)=0.15e^{-2t}Cos(10\pi t+\pi/2)$')#setting title␣
 ↪of the plot
```



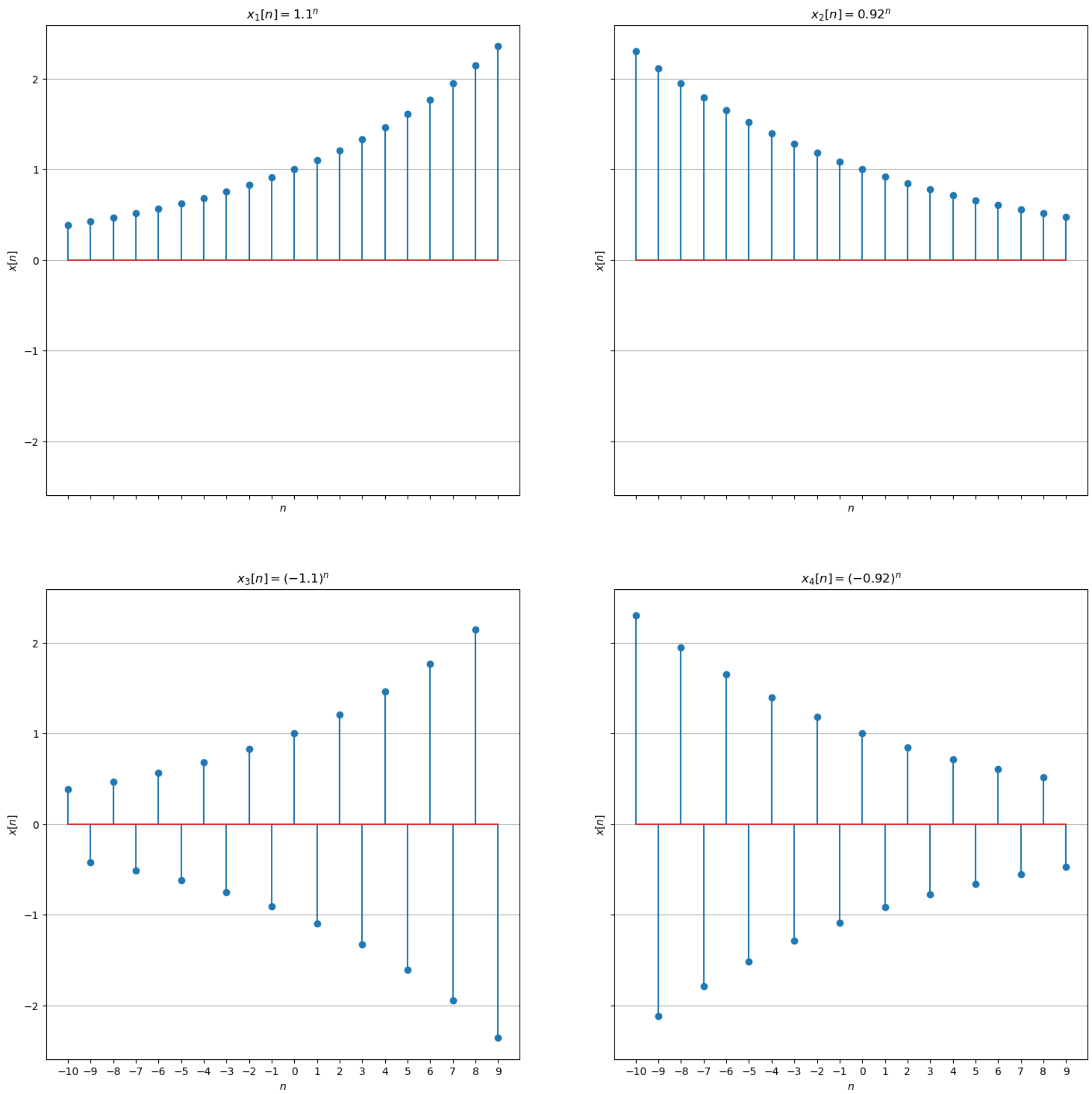


1.4 Real DT Exponential Signal $\quad x[n] = C\alpha^n$

1. $x_1[n] = 1.1^n$
2. $x_2[n] = 0.92^n$
3. $x_3[n] = -1.1^n$
4. $x_4[n] = -0.92^n$

[8]:
```
n=np.arange(-10,10,1) #generating n
fig,axes=plt.subplots(2,2,sharex="all",sharey="all",figsize=(18,18))
C=1.0
alpha=1.1

xn1=C*alpha**n       #Signal_1
```

```
xn3=C*(-alpha)**n   #Signal_3

alpha=0.92

xn2=C*alpha**n      #Signal_2
xn4=C*(-alpha)**n   #Signal_4

axes[0,0].set_xticks(n)

axes[0,0].stem(n,xn1)                              #plotting xn1
axes[0,0].set(xlabel='$n$',ylabel='$x[n]$')        #setting x and y labels
axes[0,0].yaxis.grid(True)                         #enabling grid
axes[0,0].title.set_text(r'$x_{1}[n]=1.1^{n}$')        #setting title of the plot

axes[0,1].stem(n,xn2)                              #plotting xn2
axes[0,1].set(xlabel='$n$',ylabel='$x[n]$')        #setting x and y labels
axes[0,1].yaxis.grid(True)                         #enabling grid
axes[0,1].title.set_text(r'$x_{2}[n]=0.92^{n}$')       #setting title of the plot

axes[1,0].stem(n,xn3)                              #plotting xn3
axes[1,0].set(xlabel='$n$',ylabel='$x[n]$')        #setting x and y labels
axes[1,0].yaxis.grid(True)                         #enabling grid
axes[1,0].title.set_text(r'$x_{3}[n]=(-1.1)^{n}$')   #setting title of the plot

axes[1,1].stem(n,xn4)                              #plotting xn4
axes[1,1].set(xlabel='$n$',ylabel='$x[n]$')        #setting x and y labels
axes[1,1].yaxis.grid(True)                         #enabling grid
axes[1,1].title.set_text(r'$x_{4}[n]=(-0.92)^{n}$') #setting title of the plot

plt.show()
```

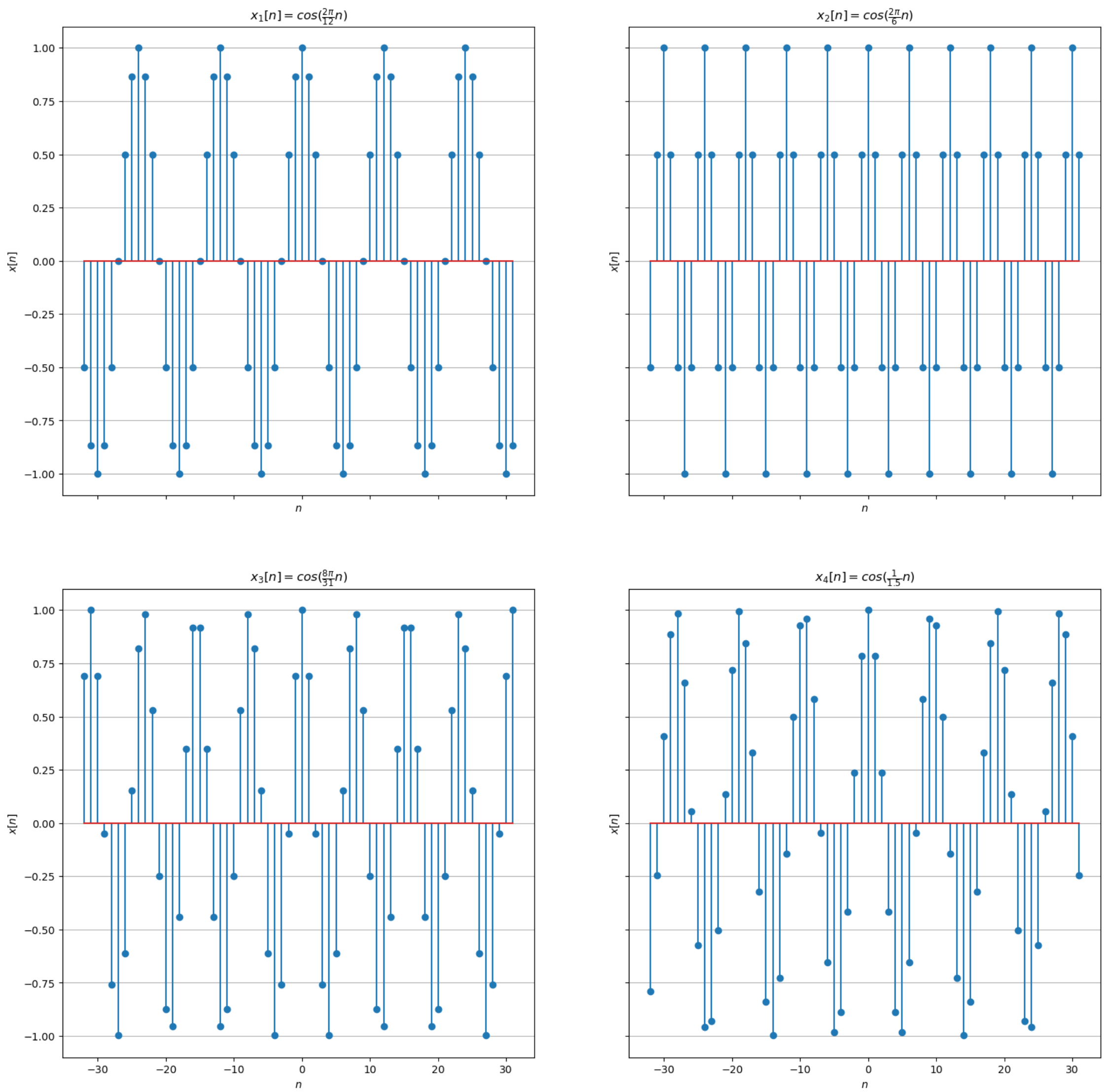


1.5 DT Sinusoids $\quad x[n] = Acos(\omega_0 n)$

1. $x_1[n] = cos(\frac{2\pi}{12}n)$
2. $x_2[n] = cos(\frac{2\pi}{6}n)$
3. $x_3[n] = cos(\frac{8\pi}{31}n)$
4. $x_4[n] = cos(\frac{1}{1.5}n)$

[9]:
```
n=np.arange(-32,32,1) #generating n

fig,axes=plt.subplots(2,2,sharex="all",sharey="all",figsize=(18,18))
```

```
xn1=np.cos(2*np.pi*n/12)    #Signal_1
xn2=np.cos(2*np.pi*n/6)     #Signal_2
xn3=np.cos(8*np.pi*n/31)    #Signal_3
xn4=np.cos(n/1.5)           #Signal_4

axes[0,0].stem(n,xn1)                                             #plotting xn1
axes[0,0].set(xlabel='$n$',ylabel='$x[n]$')                       #setting x and␣
 ↪y labels
axes[0,0].yaxis.grid(True)                                        #enabling grid
axes[0,0].title.set_text(r'$x_{1}[n]=cos(\frac{2\pi}{12} n)$')  #setting title␣
 ↪of the plot

axes[0,1].stem(n,xn2)                                             #plotting xn2
axes[0,1].set(xlabel='$n$',ylabel='$x[n]$')                       #setting x and␣
 ↪y labels
axes[0,1].yaxis.grid(True)                                        #enabling grid
axes[0,1].title.set_text(r'$x_{2}[n]=cos(\frac{2\pi}{6} n)$')   #setting title␣
 ↪of the plot

axes[1,0].stem(n,xn3)                                             #plotting xn3
axes[1,0].set(xlabel='$n$',ylabel='$x[n]$')                       #setting x and␣
 ↪y labels
axes[1,0].yaxis.grid(True)                                        #enabling grid
axes[1,0].title.set_text(r'$x_{3}[n]=cos(\frac{8\pi}{31} n)$')  #setting title␣
 ↪of the plot

axes[1,1].stem(n,xn4)                                             #plotting xn4
axes[1,1].set(xlabel='$n$',ylabel='$x[n]$')                       #setting x and␣
 ↪y labels
axes[1,1].yaxis.grid(True)                                        #enabling grid
axes[1,1].title.set_text(r'$x_{4}[n]=cos(\frac{1}{1.5} n)$')    #setting title␣
 ↪of the plot

plt.show()
```



1.5.1 Periodicity of DT signals.

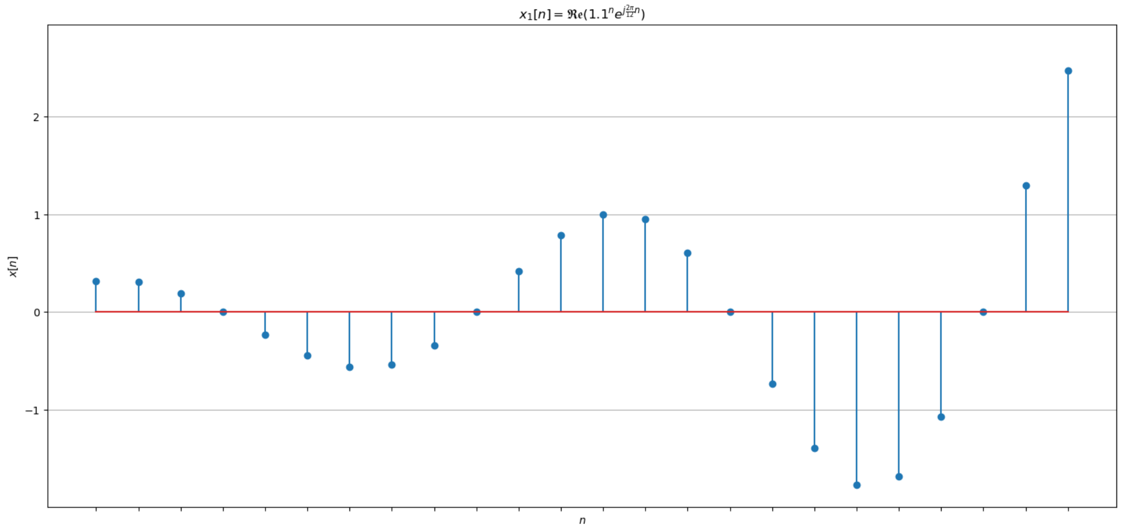
1. $x_1[n]$ - Periodic with a fundamental period of $N_0 = 12$
2. $x_2[n]$ - Periodic with a fundamental period of $N_0 = 6$
3. $x_3[n]$ - Periodic with a fundamental period of $N_0 = 31$
4. $x_4[n]$ - Aperiodic

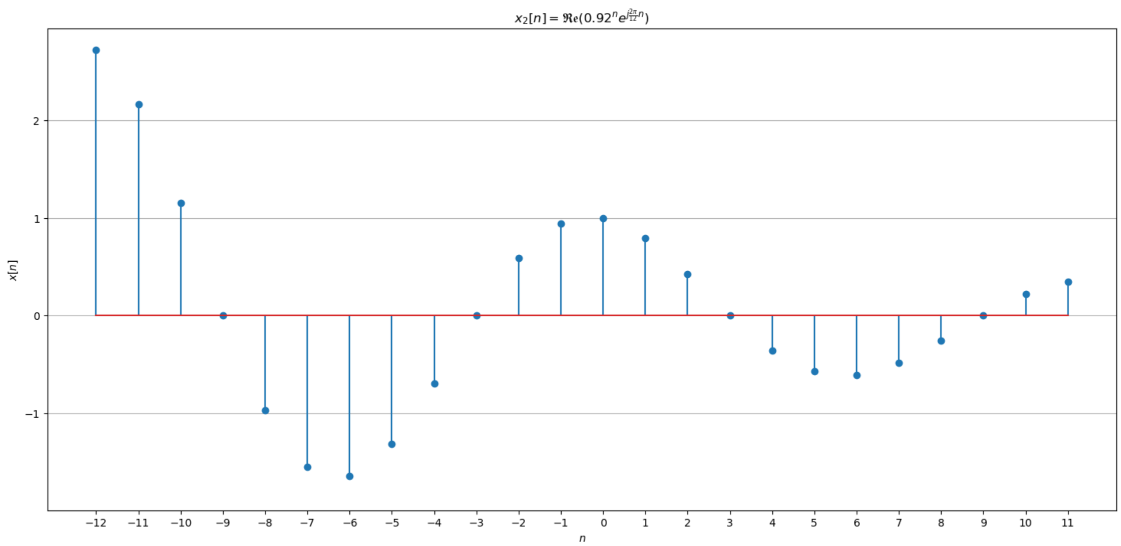
1.6 General Complex Exponential Signal (DT) $\quad x[n] = |C||\alpha|^n e^{j(\omega_0 n+\theta)}$

1. $x_1[n] = 1.1^n e^{j\frac{2\pi}{12}n}$

2. $x_2[n] = 0.92^n e^{j\frac{2\pi}{12}n}$

[13]:
```
n=np.arange(-12,12,1) #generating n
fig,axes=plt.subplots(2,1,sharex="all",sharey="all",figsize=(18,18))
theta=0

xn1=(1.1**n)*np.cos(2*np.pi*n/12+theta) #Signal_1
xn2=(0.92**n)*np.cos(2*np.pi*n/12+theta) #Signal_2



axes[0].set_xticks(n)

axes[0].stem(n,xn1) #plotting xn1
axes[0].set(xlabel='$n$',ylabel='$x[n]$') #setting x and y labels
axes[0].yaxis.grid(True) #enabling grid
axes[0].title.set_text(r'$x_{1}[n]=\mathfrak{Re}(1.
↪1^{n}e^{j\frac{2\pi}{12}n})$') #setting title of the plot

axes[1].stem(n,xn2) #plotting xn2
axes[1].set(xlabel='$n$',ylabel='$x[n]$') #setting x and y labels
axes[1].yaxis.grid(True) #enabling grid
axes[1].title.set_text(r'$x_{2}[n]=\mathfrak{Re}(0.
↪92^{n}e^{j\frac{2\pi}{12}n})$') #setting title of the plot




plt.show()
```



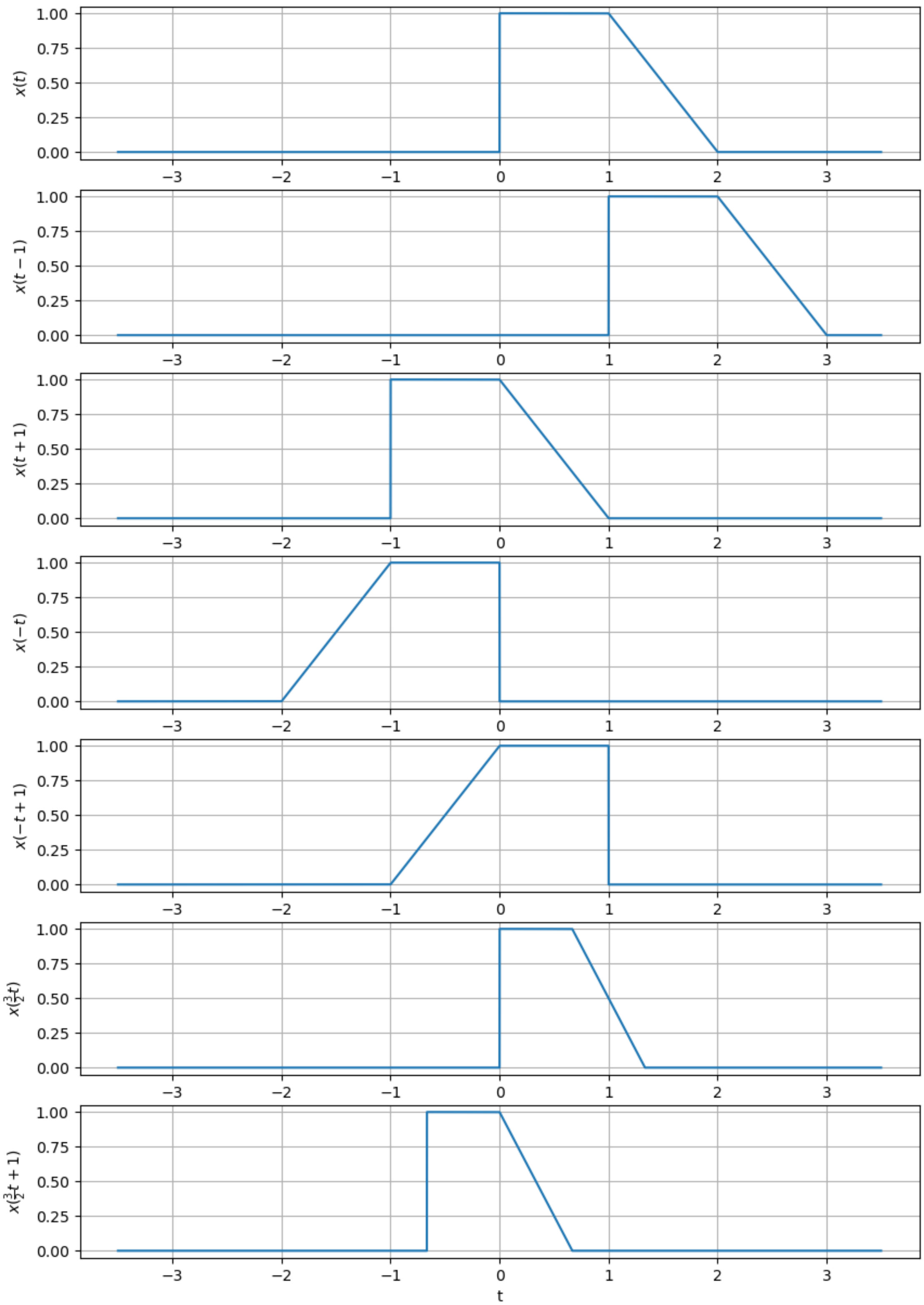


There won't be any change even if α is negative as we consider it's magnitude only.

1.7 Transformation of the Independent Variable

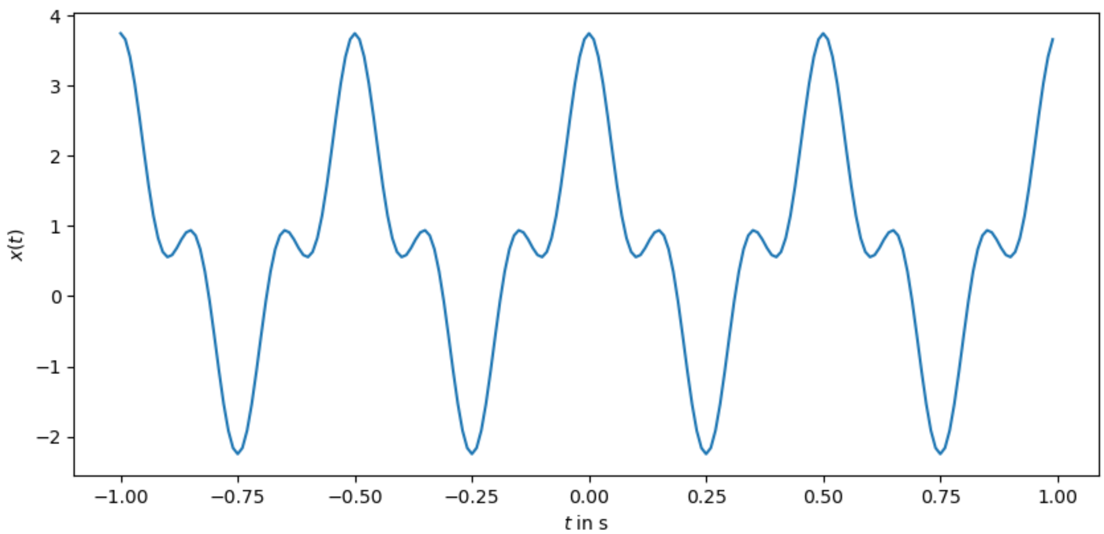
[5]:
```
def x(t):
    if(t<0):
        return 0.
    elif(t<1):
        return 1.
    elif(t<2):
        return 2.-t
    else:
        return 0.
```

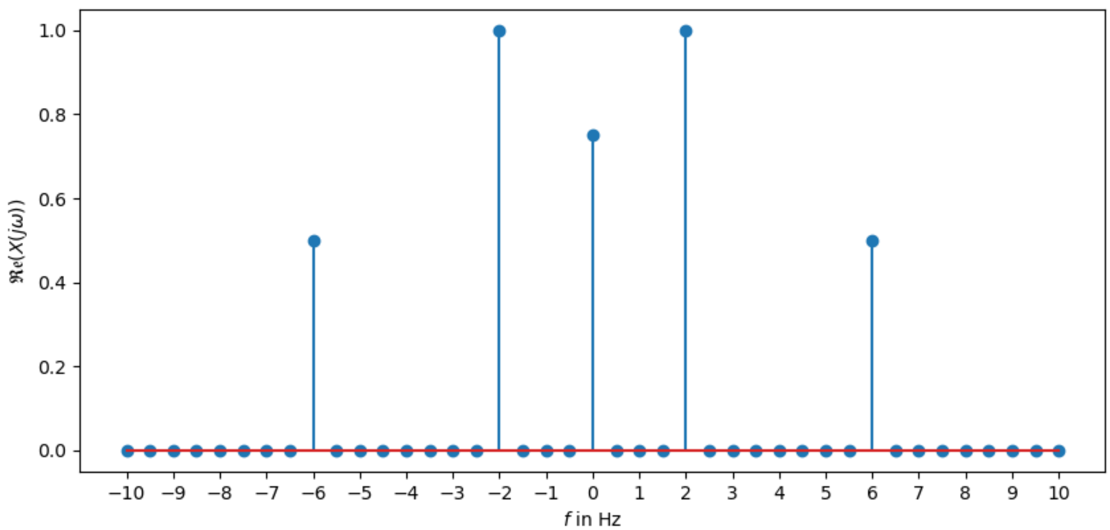
```
fs=4.4e4 #44KHz sampling frequency
ts=1/fs #sampling time
t=np.arange(-3.5,3.5,ts) #A linearly spaced array with step ts
fig,axes=plt.subplots(7,1,sharey="all",figsize=(10,15))

axes[0].plot(t,[x(t_) for t_ in t]) #initial Signal
axes[0].set(xlabel='t',ylabel='$x(t)$') #setting x and y labels
axes[0].grid(visible=True, which='both') #enabling grid

axes[1].plot(t,[x(t_-1) for t_ in t]) #Signal with a delay of 1
axes[1].set(xlabel='t',ylabel='$x(t-1)$') #setting x and y labels
axes[1].grid(visible=True, which='both') #enabling grid

axes[2].plot(t,[x(t_+1) for t_ in t]) #Signal with a delay of -1
axes[2].set(xlabel='t',ylabel='$x(t+1)$') #setting x and y labels
axes[2].grid(visible=True, which='both') #enabling grid

axes[3].plot(t,[x(-t_) for t_ in t]) #Signal with Time reversal
axes[3].set(xlabel='t',ylabel='$x(-t)$') #setting x and y labels
axes[3].grid(visible=True, which='both') #enabling grid

axes[4].plot(t,[x(-t_+1) for t_ in t]) #Signal with Time reversal and delay of -1
axes[4].set(xlabel='t',ylabel='$x(-t+1)$') #setting x and y labels
axes[4].grid(visible=True, which='both') #enabling grid

axes[5].plot(t,[x(3*t_/2) for t_ in t]) #Signal with Time scaling by 3/2
axes[5].set(xlabel='t',ylabel=r'$x(\frac{3}{2}t)$') #setting x and y labels
axes[5].grid(visible=True, which='both') #enabling grid

axes[6].plot(t,[x(3*t_/2+1) for t_ in t]) #Signal with Time scaling by 3/2 and delay of 1
axes[6].set(xlabel='t',ylabel=r'$x(\frac{3}{2}t+1)$') #setting x and y labels
axes[6].grid(visible=True, which='both') #enabling grid

plt.show()
```


$x(t)$
$x(t-1)$
$x(t+1)$
$x(-t)$
$x(-t+1)$
$x(\frac{3}{2}t)$
$x(\frac{3}{2}t+1)$
t

1.8 Observing a Signal in Frequency Domain

```
[ ]: fs=100              #100Hz sampling frequency
     ts=1/fs
     t=np.arange(-1.,1.,ts)                      #A linearly-spaced array with step␣
      ↪ts
     fig,axes=plt.subplots(2,1,figsize=(10,10))

     f=2                  #2Hz
     omega0=2*np.pi*f     #angular frequency

     xt=0.75+2*np.cos(omega0*t)+1*np.cos(3*omega0*t)

     Xf=np.fft.fft(xt)                           #computing the DFT of xt
     freq=np.fft.fftfreq(t.shape[-1],d=ts)       #computing the frequency axis

     axes[0].plot(t,xt)
     axes[0].set(xlabel='$t$ in s',ylabel='$x(t)$')                          ␣
      ↪#setting x and y labels
     valsubrange=np.concatenate((np.arange(0,21,1),np.arange(-1,-21,-1)))
     freqsubrange=np.concatenate((np.arange(0,21,1),np.arange(-1,-21,-1)))
     axes[1].stem(freq[freqsubrange],Xf.real[valsubrange]/len(t))            ␣
      ↪#plotting the real part of the DFT
     axes[1].set(xlabel='$f$ in Hz',ylabel=r'$\mathfrak{Re}(X(j\omega))$')   ␣
      ↪#setting x and y labels

     plt.xticks(np.arange(-10,11))
     plt.show()
```





Our signal has 2 sinusoids of frequencies 2Hz and 6Hz with amplitudes 1 and 0.5 respectively. Due to the 0.75 DC offset we have spike at 0Hz with amplitude 0.75.

1.9 Simple Audio Effects

Install the required packages first

[ ]:
```
!apt install libasound2-dev portaudio19-dev libportaudio2 libportaudiocpp0␣
↪ffmpeg
!pip install pyaudio
```

Before run the code make sure you have required packages installed and there is a wav file named "power_of_love.wav" in the same directory.

[ ]:
```python
import numpy as np
import pyaudio
import wave
from IPython.display import Audio

#import utility

CHUNK=8820 #44100 = 1 sec
wf = wave.open("power_of_love.wav", 'rb')

p=pyaudio.PyAudio()
nchannels=wf.getnchannels()

stream=np.array(np.zeros(nchannels),dtype=np.int16) #init stream
data = wf.readframes(CHUNK)

dtype = '<i2' #little-endian two-byte (int16) signed integers

sig = np.frombuffer(data, dtype=dtype).reshape(-1, nchannels)
signal_chunk = np.asarray(sig)

delayed = np.zeros(signal_chunk.shape, dtype=dtype)

i=0
alpha = 3
while data != '' and signal_chunk.shape[0] == CHUNK and i<120:
    i+=1
    modified_signal_chunk = alpha*signal_chunk + (1. - alpha)*delayed
    modified_signal_chunk_int16 = modified_signal_chunk.astype(np.int16)
    stream = np.vstack((stream, modified_signal_chunk_int16)) #append modified␣
 ↪to stream
    #byte_chunk = modifed_signal_chunk_int16.tobytes()
    #stream.write(byte_chunk)
    delayed = signal_chunk
    data = wf.readframes(CHUNK)
    sig = np.frombuffer(data, dtype=dtype).reshape(-1, nchannels)
    signal_chunk = np.asarray(sig)


stream = stream[1:] #pop stream init
byte_stream = stream.tobytes() #np array to bytes
p.terminate()
wf.close()

wfo = wave.open("power_of_love_eco.wav", 'wb') #writing the bot stream to a␣
 ↪output wav file
wfo.setnchannels(nchannels)
```

```
wfo.setsampwidth(wf.getsampwidth())
wfo.setframerate(wf.getframerate())
wfo.writeframes(byte_stream)
wfo.close()

Audio('power_of_love_eco.wav')
```

When we change value of alpha the strength of echo in the newly created audio file changes.