Name : V.Pavan sri sai

Roll No:AM.EN.U4ECE17063

ECE –A.

**Python Assignment**

**Functions:**

def comb\_step\_ramp(t,x):

for i in t:

if i <=-6:

k=-2

step(-6,i,k,x)

elif (i>-6) & (i<=-2) :

ramp(-2,i,4,1,x)

elif (i>-2) & (i<-1) :

step(-1,i,-1,x)

elif (i>=-1) & (i<=1) :

step(+1,i,2,x)

elif (i>1) & (i<=2) :

step(2,i,-1,x)

elif (i>2) & (i<=6) :

ramp(6,i,-3,1,x)

else:

x.append(3)

def step(j,i,k,x):

if j >=i:

x.append(k)

else:

x.append(0)

def ramp(j,i,k,l,x):

if j >=i:

x.append(l\*i+k)

else:

x.append(0)

**Codes:**

**Codes: Question No 1st,2nd,3rd,4th**

import matplotlib.pyplot as plt

import numpy as np

from comb\_step\_ramp import comb\_step\_ramp

'original function'

t = np.arange(-10, 10,0.01)

a=[]

comb\_step\_ramp(t,a)

print(a)

plt.subplot(2,2,1)

plt.step(t,a)

plt.axhline(0, color='black')

plt.axvline(0, color='black')

plt.xlabel('time')

plt.ylabel('function value')

plt.title('original function')

'TIME INVERSION x(-t)'

b=[]

comb\_step\_ramp(-t,b)

plt.subplot(2,2,2)

plt.step(t,b)

plt.axhline(0, color='black')

plt.axvline(0, color='black')

plt.xlabel('time')

plt.ylabel('function value')

plt.title('time inversion x(-t) function')

plt.show()

'AMPLITUDE INVERSION -x(t)'

c=[]

comb\_step\_ramp(t,c)

plt.subplot(2,2,3)

c[:] = [a\*-1 for a in c]

plt.step(t,c)

plt.axhline(0, color='black')

plt.axvline(0, color='black')

plt.xlabel('time')

plt.ylabel('function value')

plt.title('amplitude inversion -x(t) function')

plt.show()

'TIME SCALING BY 2t'

d=[]

comb\_step\_ramp(2\*t,d)

plt.subplot(2,2,4)

plt.step(t,d)

plt.axhline(0, color='black')

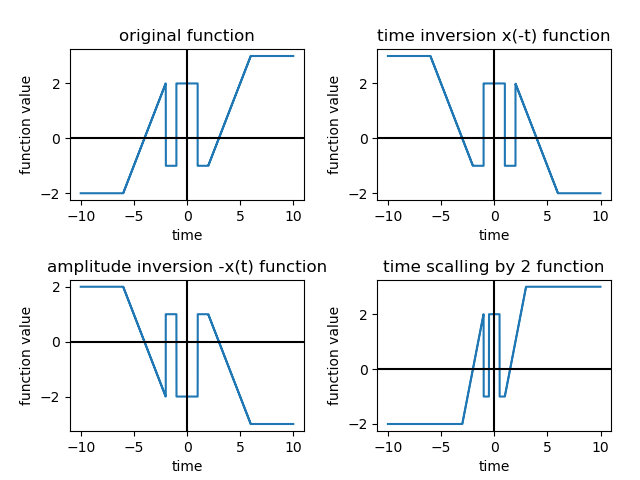
plt.axvline(0, color='black')

plt.xlabel('time')

plt.ylabel('function value')

plt.title('time scalling by 2 function')

plt.show()



Codes : Question No 5th ,6th ,7th

import matplotlib.pyplot as plt

import numpy as np

from comb\_step\_ramp import comb\_step\_ramp

t = np.arange(-10, 10,0.01)

'TIME SCALING BY t/2'

a=[]

comb\_step\_ramp(t/2,a)

plt.subplot(2,2,1)

plt.step(t,a)

plt.axhline(0, color='black')

plt.axvline(0, color='black')

plt.xlabel('time')

plt.ylabel('function value')

plt.title('time scaling by 1/2 function')

plt.show()

'AMPLITUDE SCALING BY 4'

b=[]

comb\_step\_ramp(t,b)

b[:] = [a\*4 for a in b]

plt.subplot(2,2,2)

plt.step(t,a)

plt.axhline(0, color='black')

plt.axvline(0, color='black')

plt.xlabel('time')

plt.ylabel('function value')

plt.title('amplitude scaling by 4 function')

plt.show()

'AMPLITUDE SCALING BY -4'

c=[]

comb\_step\_ramp(t,c)

c[:] = [a\*-4 for a in c]

plt.subplot(2,2,3)

plt.step(t,c)

plt.axhline(0, color='black')

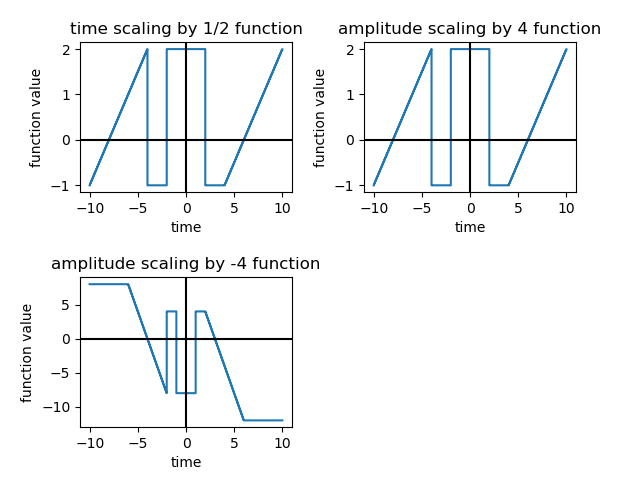
plt.axvline(0, color='black')

plt.xlabel('time')

plt.ylabel('function value')

plt.title('amplitude scaling by -4 function')

plt.show()



Codes : Question No 8,9,10,12,13,14,15

import matplotlib.pyplot as plt

import numpy as np

from comb\_step\_ramp import comb\_step\_ramp

from scipy.integrate import quad

t = np.arange(-10, 10,0.01)

'TIME SCALING t-2'

x=[]

t[:] = [x-2 for x in t]

comb\_step\_ramp(t,x)

plt.subplot(2,2,1)

t[:] = [x+2 for x in t]

plt.step(t,x)

plt.axhline(0, color='black')

plt.axvline(0, color='black')

plt.xlabel('time')

plt.ylabel('function value')

plt.title('time scaling by t-2 function')

plt.show()

'TIME SCALING t+2'

y=[]

t[:] = [x+2 for x in t]

comb\_step\_ramp(t,y)

plt.subplot(2,2,2)

t[:] = [x-2 for x in t]

plt.step(t,y)

plt.axhline(0, color='black')

plt.axvline(0, color='black')

plt.xlabel('time')

plt.ylabel('function value')

plt.title('time scaling by t+2 function')

plt.show()

'COMBINATION OF TIME SCALING t-2 AND TIME SCALING t+2'

z=[]

plt.subplot(2,2,3)

[z.append(x[i]+y[i]) for i in range(len(x))]

plt.step(t,z)

plt.axhline(0, color='black')

plt.axvline(0, color='black')

plt.xlabel('time')

plt.ylabel('function value')

plt.title('combination of t-2 and t+2 function')

plt.show()

'power or energy'

v=0

t = np.arange(-100000001, -100000000,1)

x=[]

comb\_step\_ramp(t,x)

print("value of function at near negative infinite is ",x[0])

for i in x:

if i>0:

v=1

for k in range(1,11,1):

l=0

for i in range((len(x)//100)-k):

if (x[i\*100]==x[i\*100+k\*100]):

l=l+1

else:

l=0

break

if (l== (20-k)):

m=1

print("it is power signal")

break

else:

m=0

if (m==0 & (v!=0)):

print("it is one sided power signal")

elif(m==0 & v==0):

print("it is energy signal")

else:

print("it is neither power nor energy")

'for energy'

def integrand(y):

return -2\*\*2

ans, err = quad(integrand, -1000000000, -3)

x1=ans

def integrand(y):

return (y+3)\*\*2

ans, err = quad(integrand, -3,-2)

x2=ans

def integrand(y):

return -1\*\*2

ans, err = quad(integrand, -2,0)

x3=ans

def integrand(y):

return 2\*\*2

ans, err = quad(integrand, 0,1)

x4=ans

def integrand(y):

return -1\*\*2

ans, err = quad(integrand, 1,2)

x5=ans

def integrand(y):

return (y-3)\*\*2

ans, err = quad(integrand, 2,3)

x6=ans

def integrand(y):

return 3\*\*2

ans, err = quad(integrand, 3,1000000000)

x7=ans

i=0

i=x1+x2+x3+x4+x5+x6+x7

'for average power'

print("energy of the signal when taken from -1000000000 to 1000000000 is ",i)

def integrand(y):

return -2\*\*2

ans, err = quad(integrand, -1000, -3)

x1=ans

def integrand(y):

return (y+3)\*\*2

ans, err = quad(integrand, -3,-2)

x2=ans

def integrand(y):

return -1\*\*2

ans, err = quad(integrand, -2,0)

x3=ans

def integrand(y):

return 2\*\*2

ans, err = quad(integrand, 0,1)

x4=ans

def integrand(y):

return -1\*\*2

ans, err = quad(integrand, 1,2)

x5=ans

def integrand(y):

return (y-3)\*\*2

ans, err = quad(integrand, 2,3)

x6=ans

def integrand(y):

return 3\*\*2

ans, err = quad(integrand, 3,1000)

x7=ans

i=0

i=x1+x2+x3+x4+x5+x6+x7

u=np.arange(-1000,1000,1)

print("average power when taken from -1000 to 1000 is ",(i/len(u)))

'for even or odd signal'

t = np.arange(-10, 10,0.01)

x=[]

comb\_step\_ramp(t,x)

y=[]

comb\_step\_ramp(-t,y)

m=0

for i in range(len(x)):

if (x[i]==y[i]):

m=1

else:

m=0

print("the given signal is odd signal")

break

if (m==1):

print("the given signal is even signal")

'for periodic or aperiodic'

m=0

for k in range(1,11,1):

if (m==1):

break

l=0

for i in range((len(x)//100)-k):

if (x[i\*100]==x[i\*100+k\*100]):

l=l+1

else:

l=0

break

if (l== (20-k)):

m=1

print("it is periodic signal")

break

else:

m=0

if (m==0):

print("the given signal is aperiodic signal")

'for bounded or unbounded'

m=0

t = np.arange(-10, 10,0.01)

x=[]

comb\_step\_ramp(t,x)

for i in range(len(x)):

if ( x[i] < 1000000000 ):

m=1

else:

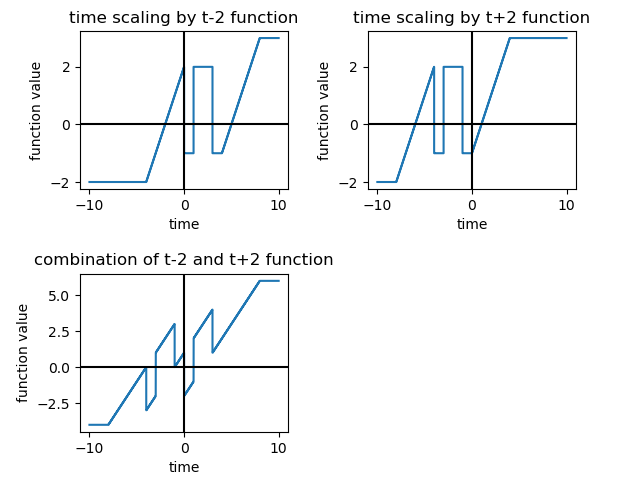
print("it is an unbounded function")

break

if(m==1):

print("the given function is bounded function")

Output:



value of function at near negative infinite is -2

it is one sided power signal

energy of the signal when taken from -1000000000 to 1000000000 is 4999999986.666667

average power when taken from -1000 to 1000 is 2.4933333333333336

the given signal is odd signal

the given signal is aperiodic signal

the given function is bounded function

Codes: Question No 11

import matplotlib.pyplot as plt

import numpy as np

from comb\_step\_ramp import comb\_step\_ramp

'EVEN SIGNAL'

t = np.arange(-10, 10,0.01)

a=[]

comb\_step\_ramp(t,a)

b=[]

comb\_step\_ramp(-t,b)

z=[]

plt.subplot(2,1,1)

[z.append((a[i]+b[i])/2) for i in range(len(a))]

plt.step(t,z)

plt.axhline(0, color='black')

plt.axvline(0, color='black')

plt.xlabel('time')

plt.ylabel('function value')

plt.title('the even signal')

plt.show()

'ODD SIGNAL'

d=[]

plt.subplot(2,1,2)

[d.append((a[i]-b[i])/2) for i in range(len(a))]

plt.step(t,d)

plt.axhline(0, color='black')

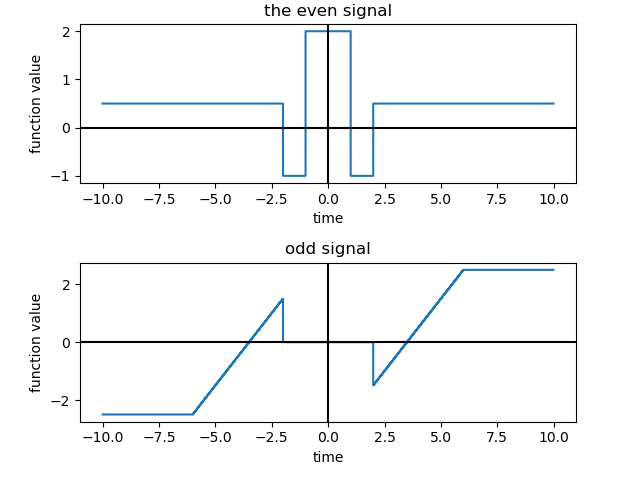
plt.axvline(0, color='black')

plt.xlabel('time')

plt.ylabel('function value')

plt.title('odd signal')

plt.show()



**Codes: Question No 16**

**16.1:**

**Static or Dynamic**

import matplotlib.pyplot as plt

import numpy as np

from comb\_step\_ramp3 import comb\_step\_ramp3

from stat\_dyn\_func import stat\_dyn\_fun

t = np.arange(-10, 10,0.01)

't'

a=[]

comb\_step\_ramp3(t,a)

stat\_dyn\_fun(a,a)

'-t'

b=[]

comb\_step\_ramp3(-t,b)

stat\_dyn\_fun(a,b)

'-x(t)'

c=[]

comb\_step\_ramp3(t,c)

c[:] = [x\*-1 for x in c]

stat\_dyn\_fun(a,c)

'x(2t)'

d=[]

comb\_step\_ramp3(2\*t,d)

stat\_dyn\_fun(a,d)

'x(t/2)'

e=[]

comb\_step\_ramp3(t/2,e)

stat\_dyn\_fun(a,e)

'4x(t)'

f=[]

comb\_step\_ramp3(t,f)

f[:] = [x\*4 for x in f]

stat\_dyn\_fun(a,f)

'-4x(t)'

g=[]

comb\_step\_ramp3(t,g)

g[:] = [x\*-4 for x in g]

stat\_dyn\_fun(a,g)

't-2'

h=[]

t[:] = [x-2 for x in t]

comb\_step\_ramp3(t,h)

stat\_dyn\_fun(a,h)

't+2'

t = np.arange(-10, 10,0.01)

i=[]

t[:] = [x+2 for x in t]

comb\_step\_ramp3(t,i)

stat\_dyn\_fun(a,i)

't-2 + t+2'

j=[]

for k in range(len(h)):

j.append(h[k]+i[k])

stat\_dyn\_fun(a,j)

**Function:**

def stat\_dyn\_fun(a,x):

m=0

for i in range(len(x)):

if (x[i] == a[i]):

m=1

else:

m=0

print("the function is dynamic")

break

if(m==1):

print("the given function is static")

Output:

the given function is static

the function is dynamic

the function is dynamic

the function is dynamic

the function is dynamic

the function is dynamic

the function is dynamic

the function is dynamic

the function is dynamic

the function is dynamic

**Question No** : **16.2**

import matplotlib.pyplot as plt

import numpy as np

from cau\_noncau import cau\_noncau

t = np.arange(-10, 10,0.01)

'general form is a(b\*t-c+d) c is shift which is t0'

cau\_noncau(1,1,0,0,1)

cau\_noncau(1,-1,0,0,2)

cau\_noncau(-1,1,0,0,3)

cau\_noncau(1,2,0,0,4)

cau\_noncau(1,0.5,0,0,5)

cau\_noncau(4,1,0,0,6)

cau\_noncau(-4,1,0,0,7)

cau\_noncau(1,1,-2,0,8)

cau\_noncau(1,1,2,0,9)

cau\_noncau(1,1,-2,2,10)

**Function:**

def cau\_noncau(a,b,c,d,e):

if(b==0.5):

b=3

c=9

if ((b==1) & (c==0) & (d==0)):

print(e," is causal")

elif((b==-1) & (c==0) & (d==0)):

print(e," is non-causal")

elif((b>1) & (c==0) & (d==0)):

print(e," is non-causal")

elif((b<0) & (c==0) & (d==0)):

print(e," is non-causal")

elif((b==3) & (c==9) & (d==0)):

print(e," is anti-causal")

elif((b==1) & (c>0) & (d==0)):

print(e," is anti-causal")

elif((b==1) & (c<0) & (d==0)):

print(e," is causal")

elif((c\*d)<0):

print(e," is non-causal")

**Output:**

1 is causal

2 is non-causal

3 is causal

4 is non-causal

5 is anti-causal

6 is causal

7 is causal

8 is causal

9 is anti-causal

10 is non-causal

**Question No : 16.3**

import matplotlib.pyplot as plt

import numpy as np

from comb\_step\_ramp3 import comb\_step\_ramp3

from linear\_nonlinear import lin\_nonlin

't'

t = np.arange(-10, 10,0.01)

x=[]

comb\_step\_ramp3(t,x)

lin\_nonlin(x)

'-t'

y=[]

comb\_step\_ramp3(-t,y)

lin\_nonlin(y)

'-x(t)'

z=[]

comb\_step\_ramp3(t,z)

z[:] = [x\*-1 for x in z]

lin\_nonlin(z)

'x(2t)'

u=[]

comb\_step\_ramp3(2\*t,u)

lin\_nonlin(u)

'x(t/2)'

x=[]

comb\_step\_ramp3(t/2,x)

lin\_nonlin(x)

'4x(t)'

y=[]

comb\_step\_ramp3(t,y)

y[:] = [x\*4 for x in y]

lin\_nonlin(y)

'-4x(t)'

z=[]

comb\_step\_ramp3(t,z)

z[:] = [x\*-4 for x in z]

lin\_nonlin(z)

't-2'

x=[]

t[:] = [x-2 for x in t]

comb\_step\_ramp3(t,x)

lin\_nonlin(x)

't+2'

t = np.arange(-10, 10,0.01)

y=[]

t[:] = [x+2 for x in t]

comb\_step\_ramp3(t,y)

lin\_nonlin(y)

z=[]

for i in range(len(x)):

z.append(x[i]+y[i])

lin\_nonlin(z)

**Function:**

def lin\_nonlin(x):

k=0

l=0

m=1

for i in range(len(x)):

k=0

if(m!=0):

for j in range(len(x)):

if (round((x[i]+x[j]),3)==round(x[i+j],3)):

k=k+1

else:

print("not linear")

m=0

break

if(k==len(x)):

l=l+1

if (l==len(x)):

print("linear function")

Output:

not linear

not linear

not linear

not linear

not linear

not linear

not linear

not linear

not linear

not linear

**Question No: 16.4**

import matplotlib.pyplot as plt

import numpy as np

from comb\_step\_ramp3 import comb\_step\_ramp3

from inv\_noninv import inv\_noninv

't'

t = np.arange(-10, 10,0.01)

x=[]

comb\_step\_ramp3(t,x)

inv\_noninv(x)

'-t'

y=[]

comb\_step\_ramp3(-t,y)

inv\_noninv(y)

'-x(t)'

z=[]

comb\_step\_ramp3(t,z)

z[:] = [x\*-1 for x in z]

inv\_noninv(z)

'x(2t)'

u=[]

comb\_step\_ramp3(2\*t,u)

inv\_noninv(u)

'x(t/2)'

x=[]

comb\_step\_ramp3(t/2,x)

inv\_noninv(x)

'4x(t)'

y=[]

comb\_step\_ramp3(t,y)

y[:] = [x\*4 for x in y]

inv\_noninv(y)

'-4x(t)'

z=[]

comb\_step\_ramp3(t,z)

z[:] = [x\*-4 for x in z]

inv\_noninv(z)

't-2'

x=[]

t[:] = [x-2 for x in t]

comb\_step\_ramp3(t,x)

inv\_noninv(x)

't+2'

t = np.arange(-10, 10,0.01)

y=[]

t[:] = [x+2 for x in t]

comb\_step\_ramp3(t,y)

inv\_noninv(y)

z=[]

for i in range(len(x)):

z.append(x[i]+y[i])

inv\_noninv(z)

**Function:**

def inv\_noninv(x):

k=0

l=0

m=1

for i in range(len(x)):

k=0

if(m!=0):

for j in range(len(x)):

if(i!=j):

if (round(x[i],3)==round(x[j],3)):

print("not one to one")

print("so non-invertible\n")

m=0

break

else:

k=k+1

if(k==len(x)-1):

l=l+1

if (l==len(x)):

print("invertible")

**Output:**

not one to one

so non-invertible

not one to one

so non-invertible

not one to one

so non-invertible

not one to one

so non-invertible

not one to one

so non-invertible

not one to one

so non-invertible

not one to one

so non-invertible

not one to one

so non-invertible

not one to one

so non-invertible

not one to one

so non-invertible

**Question No : 16.5 Stable and Unstable.**

**Code:**

import matplotlib.pyplot as plt

import numpy as np

from comb\_step\_ramp3 import comb\_step\_ramp3

from sta\_unsta\_func import sta\_unsta\_func

t = np.arange(-10, 10,0.01)

't'

a=[]

comb\_step\_ramp3(t,a)

sta\_unsta\_func(a)

'-t'

b=[]

comb\_step\_ramp3(-t,b)

sta\_unsta\_func(b)

'-x(t)'

c=[]

comb\_step\_ramp3(t,c)

c[:] = [x\*-1 for x in c]

sta\_unsta\_func(c)

'x(2t)'

d=[]

comb\_step\_ramp3(2\*t,d)

sta\_unsta\_func(d)

'x(t/2)'

e=[]

comb\_step\_ramp3(t/2,e)

sta\_unsta\_func(e)

'4x(t)'

f=[]

comb\_step\_ramp3(t,f)

f[:] = [x\*4 for x in f]

sta\_unsta\_func(f)

'-4x(t)'

g=[]

comb\_step\_ramp3(t,g)

g[:] = [x\*-4 for x in g]

sta\_unsta\_func(g)

't-2'

h=[]

t[:] = [x-2 for x in t]

comb\_step\_ramp3(t,h)

sta\_unsta\_func(h)

't+2'

t = np.arange(-10, 10,0.01)

i=[]

t[:] = [x+2 for x in t]

comb\_step\_ramp3(t,i)

sta\_unsta\_func(i)

't-2 + t+2'

j=[]

for k in range(len(h)):

j.append(h[k]+i[k])

sta\_unsta\_func(j)

**Function:**

def sta\_unsta\_func(x):

m=0

for i in range(len(x)):

if ((x[i]<100000000) & (x[i]>-1000000)):

m=1

else:

m=0

print("the given funnction is not stable")

break

if(m==1):

print("the given function is stable")

**Output:**

the given function is stable

the given function is stable

the given function is stable

the given function is stable

the given function is stable

the given function is stable

the given function is stable

the given function is stable

the given function is stable

the given function is stable

**Question : Time Variant or Time Invariant**

import matplotlib.pyplot as plt

import numpy as np

from timevariant\_invariant import time\_var\_invar

from comb\_step\_ramp1 import comb\_step\_ramp1

t = np.arange(-10, 10,0.01)

'general form is a(b\*t-c+d) c is shift which is t0'

time\_var\_invar(1,1,2,0,t)

time\_var\_invar(-1,1,2,0,t)

time\_var\_invar(1,-1,2,0,t)

time\_var\_invar(1,2,2,0,t)

time\_var\_invar(1,0.5,2,0,t)

time\_var\_invar(4,1,2,0,t)

time\_var\_invar(-4,1,2,0,t)

a=time\_var\_invar(1,1,2,-2,t)

b=time\_var\_invar(1,1,2,2,t)

if((a==b) & (a==1)):

print("x(t-2)+x(t+2) is also time invariant")

**Function:**

from comb\_step\_ramp1 import comb\_step\_ramp1

import numpy as np

import matplotlib.pyplot as plt

def time\_var\_invar(a,b,c,d,t):

t = np.arange(-10, 10,0.01)

x=[]

comb\_step\_ramp1(b\*t,x)

t[:] = [x+c+d for x in t]

x[:] = [x\*a for x in x]

t= np.arange(-10, 10,0.01)

y=[]

t[:] = [x-c+d for x in t]

comb\_step\_ramp1(b\*t,y)

t[:] = [x+c+d for x in t]

y[:] = [x\*a for x in y]

m=0

for i in range(len(x)):

if(i<(2000-((c-d)\*200))):

a=round(x[i],3)

b=round(y[i+(c-d)\*100],3)

if(a==b):

m=1

else:

m=0

break

if(m==1):

print("time invariant\n")

else:

print("time variant")

return m

**output:**

time invariant

time invariant

time invariant

time invariant

time invariant

time invariant

time invariant

time invariant

time invariant

x(t-2)+x(t+2) is also time invariant

**Question No:17**

**Codes:**

import matplotlib.pyplot as plt

import numpy as np

from impulse import impulse

from step import step

from comb\_step\_ramp4 import comb\_step\_ramp4

t = np.arange(-10, 10,0.01)

x=[]

comb\_step\_ramp4(t,x)

t = np.arange(-10, 10,0.01)

i=[]

impulse(t,i)

z=[]

for j in range(len(t)):

z.append(i[j]\*x[j])

plt.step(t,z)

plt.xlabel('time')

plt.ylabel('function value')

plt.title('x(t)\*d(t)')

plt.show()

t = np.arange(-10, 10,0.01)

i=[]

impulse(-t,i)

t = np.arange(-10, 10,0.01)

x=[]

comb\_step\_ramp4(t,x)

z=[]

for j in range(len(t)):

z.append(i[j]\*x[j])

plt.step(t,z)

plt.xlabel('time')

plt.ylabel('function value')

plt.title('x(t)\*d(-t)')

plt.show()

t = np.arange(-10, 10,0.01)

t[:]=[x+2 for x in t]

i=[]

impulse(t,i)

t = np.arange(-10, 10,0.01)

x=[]

comb\_step\_ramp4(t,x)

z=[]

for j in range(len(t)):

z.append(round(i[j]\*x[j],3))

plt.step(t,z)

plt.xlabel('time')

plt.ylabel('function value')

plt.title('x(t)\*d(t+2)')

plt.show()

t = np.arange(-10, 10,0.01)

t[:]=[x-2 for x in t]

i=[]

impulse(t,i)

t = np.arange(-10, 10,0.01)

x=[]

comb\_step\_ramp4(t,x)

z=[]

for j in range(len(t)):

z.append(round(i[j]\*x[j],3))

plt.step(t,z)

plt.axhline(0, color='black')

plt.axvline(0, color='black')

plt.xlabel('time')

plt.ylabel('function value')

plt.title('x(t)\*d(t-2)')

plt.show()

t = np.arange(-10, 10,0.01)

i=[]

impulse(t,i)

t = np.arange(-10, 10,0.01)

x=[]

comb\_step\_ramp4(t,x)

z=[]

for j in range(len(t)):

z.append(round(i[j]\*x[j],3))

z[:]=[x\*4 for x in z]

plt.step(t,z)

plt.xlabel('time')

plt.ylabel('function value')

plt.title('x(t)\*4\*d(t)')

plt.show()

t = np.arange(-10, 10,0.01)

x=[]

comb\_step\_ramp4(t,x)

t = np.arange(-10, 10,0.01)

u=[]

step(t,u)

z=[]

for j in range(len(t)):

z.append(u[j]\*x[j])

plt.step(t,z)

plt.axhline(0, color='black')

plt.axvline(0, color='black')

plt.xlabel('time')

plt.ylabel('function value')

plt.title('x(t)\*u(t)')

plt.show()

t = np.arange(-10, 10,0.01)

u=[]

step(-t,u)

t = np.arange(-10, 10,0.01)

x=[]

comb\_step\_ramp4(t,x)

z=[]

for j in range(len(t)):

z.append(u[j]\*x[j])

plt.step(t,z)

plt.axhline(0, color='black')

plt.axvline(0, color='black')

plt.xlabel('time')

plt.ylabel('function value')

plt.title('x(t)\*u(-t)')

plt.show()

t = np.arange(-10, 10,0.01)

t[:]=[x+2 for x in t]

u=[]

step(t,u)

t = np.arange(-10, 10,0.01)

x=[]

comb\_step\_ramp4(t,x)

z=[]

for j in range(len(t)):

z.append(round(u[j]\*x[j],3))

plt.step(t,z)

plt.axhline(0, color='black')

plt.axvline(0, color='black')

plt.xlabel('time')

plt.ylabel('function value')

plt.title('x(t)\*u(t+2)')

plt.show()

t = np.arange(-10, 10,0.01)

t[:]=[x-2 for x in t]

u=[]

step(t,u)

print(i)

t = np.arange(-10, 10,0.01)

x=[]

comb\_step\_ramp4(t,x)

z=[]

for j in range(len(t)):

z.append(round(u[j]\*x[j],3))

plt.step(t,z)

plt.axhline(0, color='black')

plt.axvline(0, color='black')

plt.xlabel('time')

plt.ylabel('function value')

plt.title('x(t)\*u(t-2)')

plt.show()

t = np.arange(-10, 10,0.01)

u=[]

step(t,u)

t = np.arange(-10, 10,0.01)

x=[]

comb\_step\_ramp4(t,x)

z=[]

for j in range(len(t)):

z.append(round(u[j]\*x[j],3))

z[:]=[x\*4 for x in z]

plt.step(t,z)

plt.axhline(0, color='black')

plt.axvline(0, color='black')

plt.xlabel('time')

plt.ylabel('function value')

plt.title('x(t)\*4\*u(t)')

plt.show()

**Functions:**

**impulse:**

def impulse(t,i):

for j in t:

if (round(j,3)==0):

i.append(1)

else:

i.append(0)

**step:**

def step(t,u):

for j in t:

if (round(j,3)>=0):

u.append(1)

else:

u.append(0)

Pictures:

