Homework Aula 01 - 01_08_05_2020_Bruno_Lima_Q_S_ELE_22

May 8, 2020

0.1 EET-01-Sinais e sistemas de tempo discreto

0.1.1 Aluno: Bruno Lima Queiroz Santos

Exercício Aula 01 - 01

a) Representar graficamente as sequências básicas apresentadas na Aula 01 com a função "stem" do matlab (ou GNU Octave).

A)Sequência pulso unitário, impulso de tempo discreto, ou impulso.

i)

$$\delta[n] = \begin{cases} 0, & \text{if } n \neq 0 \\ 1, & \text{if } n = 0 \end{cases}$$

[1]:
```
import numpy as np
```

[2]:
```
" Perfil do tempo (domínio) do sinal"
N=16
From=-7
To=From+(N-1)

" Formato do dado na forma de vetor "
x=np.linspace(start=From,
              stop=To,
              num=N
             )
""" Returns
num evenly spaced samples,
calculated over the interval [start, stop]. """



" Formato do sinal na forma de vetor "

y=np.zeros(shape=(N))
```

```
X=np.array([0])

" y e x estão relacionados por uma mesma indexação "

for i in X :
    y[
        np.where(
            x==i
        )[0]
    ]=1.0
```

[3]:
```
import matplotlib.pyplot as plt
```

[4]:
```
markerline, stemlines, baseline=plt.stem(x,
                                         y,
                                         markerfmt='or',
                                         linefmt='-r',
                                         basefmt="k")
#basefmt=" " oculta a baseline
#markerline.set_markerfacecolor('none')
plt.title('Sequência impulso')
plt.ylabel('y')
plt.xlabel('t')
plt.show()

print("Figura 1 representa a função do item i)")
```

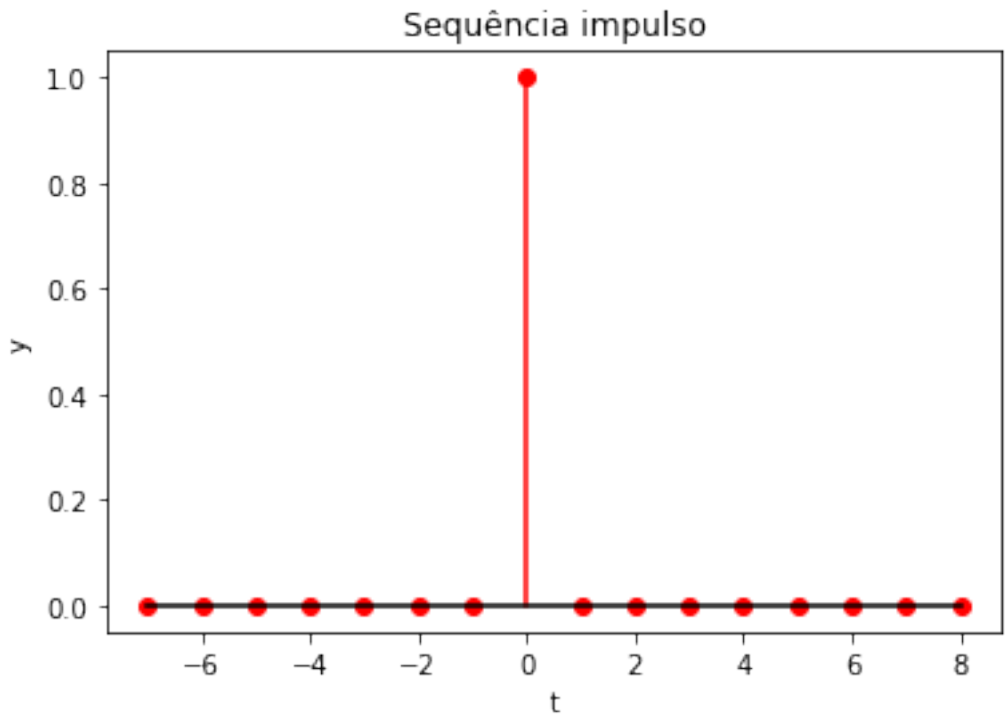


Figura 1 representa a função do item i)

ii) Segunda sequência pulso unitário, impulso de tempo discreto, ou impulso.

$$\rho[n] = \begin{cases} 0, & \text{if } n \neq (-3,1,2,7) \\ 1, & \text{if } n = (-3,1,2,7) \end{cases}$$

ou,

$$\rho[n] = a_{-3}\delta[n+3] + a_1\delta[n-1] + a_2\delta[n-2] + a_7\delta[n-7]$$

Cuidado, o pacote abaixo não possui suporte para Windows. Apenas até a versão 2.9.4 há o bom funcionamento nesse sistema operacional. Ela encontra-se atualmente disponível pelo anaconda. (Basta incluir adequadamente a pasta rpy2 referente a essa versão no diretório do Python).

A seguir, utiliza-se R para a resolução do problema. (Obviamente é um requisito ter essa linguagem também).

[5]:
```
import rpy2.rinterface as ri
ri.initr()
#rr=ri.parse

rb=ri.baseenv
rg=ri.globalenv
rs=ri.StrSexpVector
rp=rb.get("parse")
```

```
re=rb["eval"]

"""~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"""

filename=ri.StrSexpVector([""])
n=ri.IntSexpVector(["-1"])
"""~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"""

""" Estruturação da função """

""" ########################################      """

                # Código R #
# ================================================== #

text=ri.StrSexpVector["""
#The function
stem <- function(x,y,pch=16,linecol=1,clinecol=1,...){
    if (missing(y)){
        y = x
        x = 1:length(x)
    }
    plot(x,y,pch=pch,...)
    for (i in 1:length(x)){
       lines(c(x[i],x[i]), c(0,y[i]),col=linecol)
    }
    lines(c(x[1]-2,x[length(x)]+2), c(0,0),col=clinecol)
}
"""])

# ================================================== #

""" ########################################      """


k=rp(filename,n,text)
k=re(k)


""" ---------------------------------------- """


""" Execução  """

""" ########################################      """

                # Código R #
```

```
# ================================================== #
text=ri.StrSexpVector(["""
#An example
x <- seq(-4, 8, by = 1)
#y <- seq(0,0,length.out=13)
y <- rep(0,13)
Y<-c(2,4,2,1.5)
#Y<- as.numeric(c(2.0,4.0,-2.0,-1.5))
X<-which(x %in% c(-3,1,2,7)) #match(c(-3,1,2,7),x)
for (i in seq(1,length(Y),by=1))
{
    y[X[i]]<-Y[i]
}
png(file="fileName1.png")
stem(x,y,col=4,
    linecol=4,clinecol=2,
    main="Impulso de tempo discreto",
    xlab="t"
)
dev.off()
"""])


# ================================================== #

""" ########################################      """

k=re(
    rp(filename,n,text)
)

""" ---------------------------------------------- """


from IPython.display import Image,display

i1=Image(filename='fileName1.png',width=400,height =400)
display(i1)

v=2
print("figura %d" %(v), "Sequência ii)")
v+=1

""" ########################################      """

                # Código R #
# ================================================== #
```

```
text=rs(["""
unlink("fileName1.png")
"""])

# ================================================== #

""" ########################################      """

k=re(
    rp(filename,n,text)
)
```

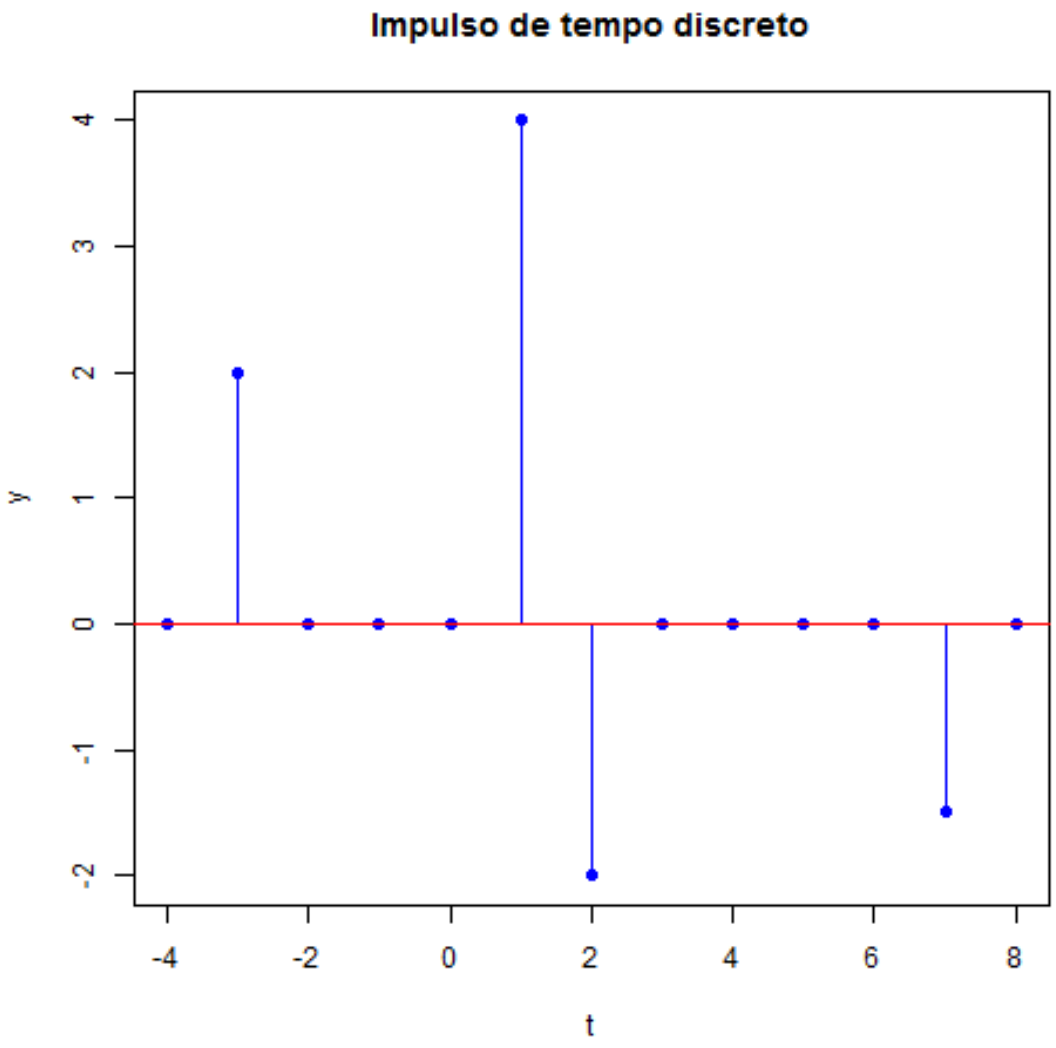


figura 2 Sequência ii)

O código em R para a função stem é disponibilizado por meio dos links 1 e 2.

B) Sequência degrau unitário

$$u[n] = \begin{cases} 1, & \text{if } n \geq 0 \\ 0, & \text{if } n < 0 \end{cases}$$

[6]:
```
" Perfil do tempo (domínio) do sinal"
N=14
From=-6
To=From+(N-1)

" Formato do dado na forma de vetor "
x=np.linspace(start=From,
              stop=To,
              num=N
             )




" Formato do sinal na forma de vetor "

y=np.zeros(shape=(N))

X=np.arange(0,8,1)

" y e x estão relacionados por uma mesma indexação "

# no Python, não há um recurso igual ao match
for i in X :
    y[
        np.where(
            x==i
        )[0]
    ]=1.0

# corrigiram-se os valores de y nos índices em que os valores
# do tempo requerem valor de y não nulo.

#import matplotlib.pyplot as plt

markerline, stemlines, baseline=plt.stem(x,
                                         y,
                                         markerfmt='or',
```

```
                                        linefmt='-r',
                                        basefmt="k")
#basefmt=" " oculta a baseline
#markerline.set_markerfacecolor('none')
plt.title('Sequência degrau unitário')
plt.ylabel('y')
plt.xlabel('t')

plt.show()

print("Figura 3 representa a função do item B)")
```

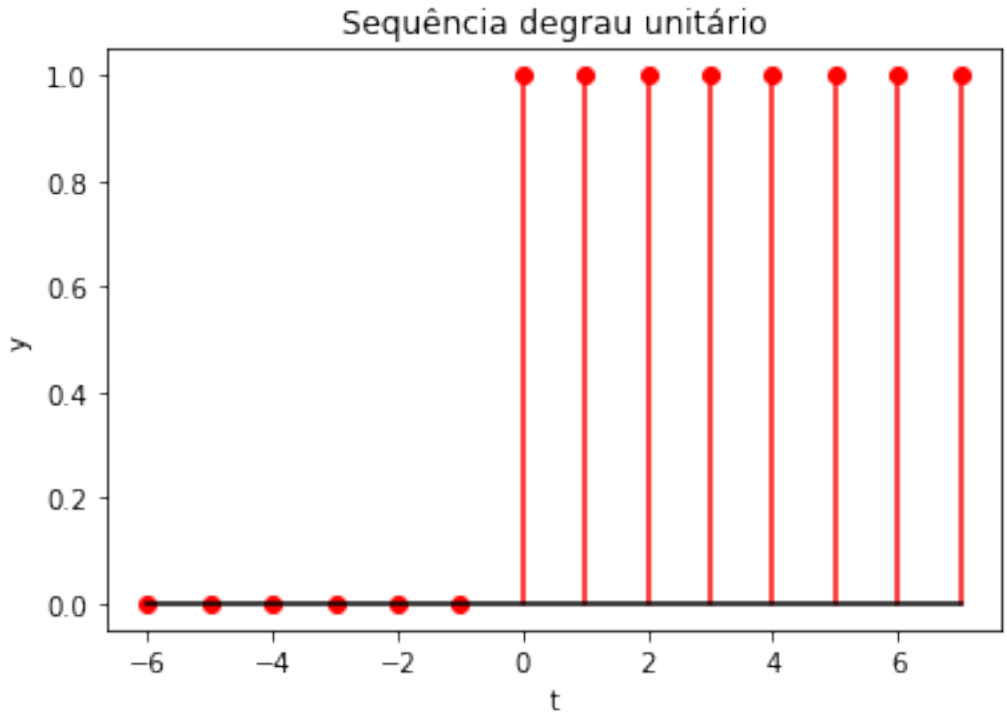


Figura 3 representa a função do item B)

C) Sequência exponencial

$$y[n] = A\alpha^n u[n]$$

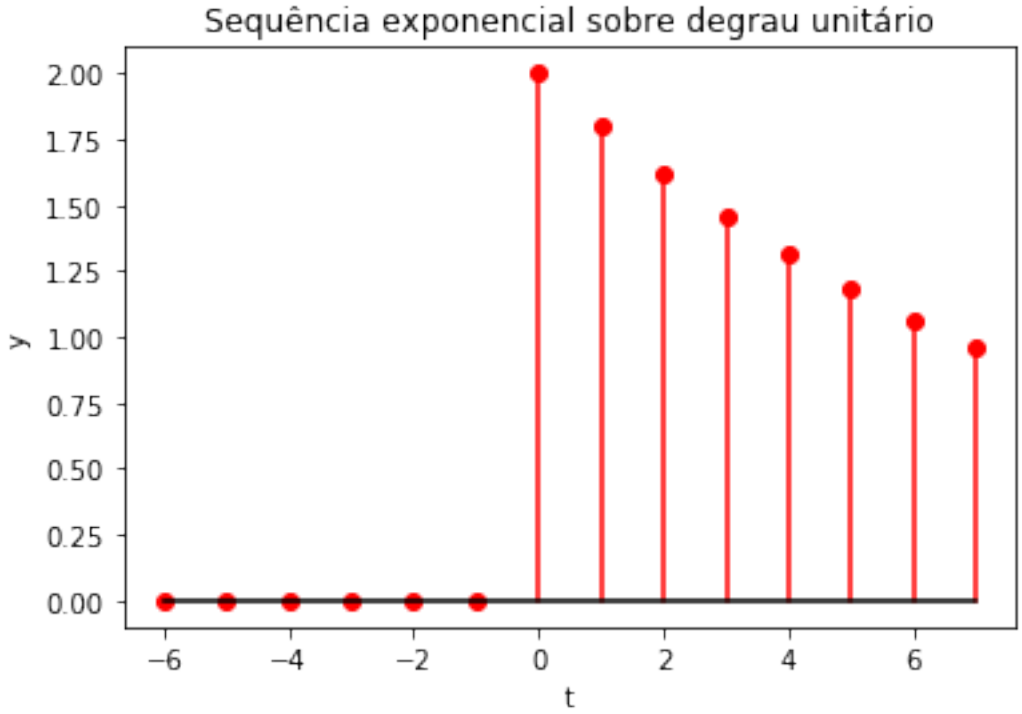
C.1) $0 < \alpha < 1$ e $A > 0$

Nesse caso, é esperado que os valores da sequência sejam positivos e decresçam com o aumento de n.

Digamos $\alpha = 0.9, A = 2.0$

[7]:
```
" Perfil do tempo (domínio) do sinal"
N=14
From=-6
To=From+(N-1)

" Formato do dado na forma de vetor "
x=np.linspace(start=From,
              stop=To,
              num=N
             )




" Formato do sinal na forma de vetor "

u=np.zeros(shape=(N))

X=np.arange(0,N,1)

" y e x estão relacionados por uma mesma indexação "

# no Python, não há um recurso igual ao match
for i in X :
    u[
        np.where(
            x==i
        )[0]
    ]=1.0

for i in np.arange(0,N,1):
    u[i]=2.0*u[i]*(0.9**x[i])
y=u


markerline, stemlines, baseline=plt.stem(x,
                                         y,
                                         markerfmt='or',
                                         linefmt='-r',
                                         basefmt="k")
#basefmt=" " oculta a baseline
#markerline.set_markerfacecolor('none')
plt.title('Sequência exponencial sobre degrau unitário')
plt.ylabel('y')
plt.xlabel('t')

plt.show()
```

```
print("Figura 4 representa a função do item C.1)")
```




```
Figura 4 representa a função do item C.1)
```

C.2) $-1 < \alpha < 0$ e $A \neq 0$

Nesse caso, é esperado que os valores da sequência sejam alternantes entre números positivos e negativos, e que decresçam sua magnitude com o aumento de n.

Digamos $\alpha = -0.9$, $A = 2.0$

[8]:
```
" Perfil do tempo (domínio) do sinal"
N=14
From=-6
To=From+(N-1)

" Formato do dado na forma de vetor "
x=np.linspace(start=From,
              stop=To,
              num=N
             )
```

```
" Formato do sinal na forma de vetor "

u=np.zeros(shape=(N))

X=np.arange(0,N,1)

" y e x estão relacionados por uma mesma indexação "

# no Python, não há um recurso igual ao match
for i in X :
    u[
        np.where(
            x==i
        )[0]
    ]=1.0

for i in np.arange(0,N,1):
    u[i]=2.0*u[i]*((-0.9)**x[i])
y=u


markerline, stemlines, baseline=plt.stem(x,
                                         y,
                                         markerfmt='or',
                                         linefmt='-r',
                                         basefmt="k")
#basefmt=" " oculta a baseline
#markerline.set_markerfacecolor('none')
plt.title('Sequência exponencial sobre degrau unitário')
plt.ylabel('y')
plt.xlabel('t')

plt.show()

print("Figura 5 representa a função do item C.2)")
```

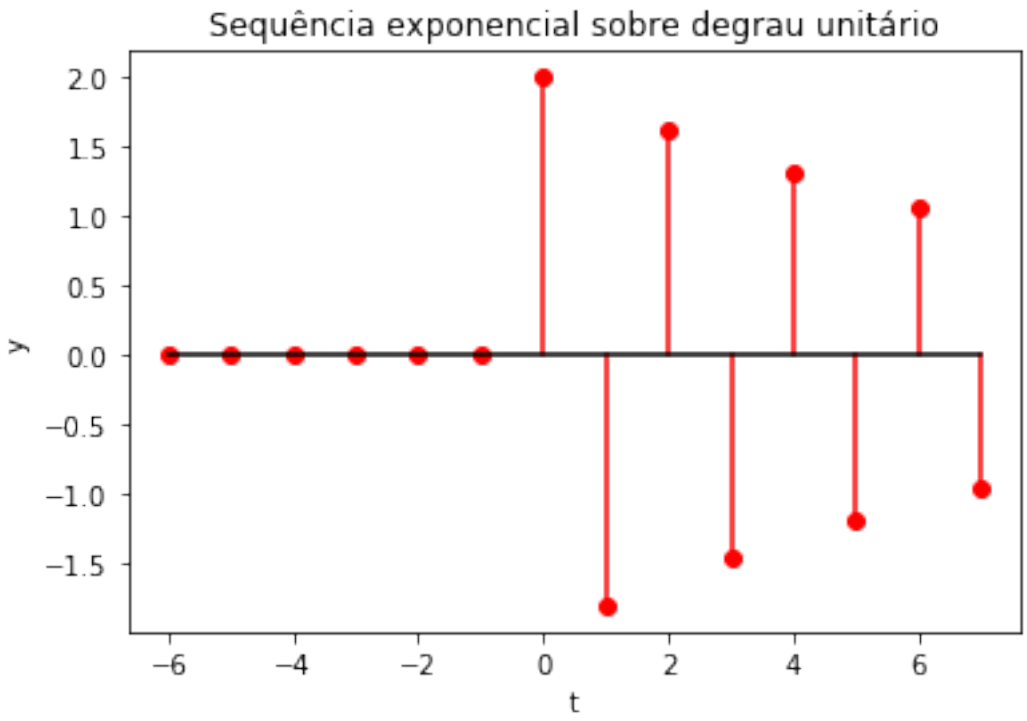


Figura 5 representa a função do item C.2)

C.3) $\alpha > 1$ e $A \neq 0$
Nesse caso, é esperado que a magnitude dos valores da sequência cresça com o aumento de n.
Digamos $\alpha = 1.5, A = 2.0$

[9]:
```
" Perfil do tempo (domínio) do sinal"
N=14
From=-6
To=From+(N-1)

" Formato do dado na forma de vetor "
x=np.linspace(start=From,
              stop=To,
              num=N
             )




" Formato do sinal na forma de vetor "

u=np.zeros(shape=(N))
```

```
X=np.arange(0,N,1)

" y e x estão relacionados por uma mesma indexação "

# no Python, não há um recurso igual ao match
for i in X :
    u[
        np.where(
            x==i
        )[0]
    ]=1.0

for i in np.arange(0,N,1):
    u[i]=2.0*u[i]*((1.5)**x[i])
y=u


markerline, stemlines, baseline=plt.stem(x,
                                         y,
                                         markerfmt='or',
                                         linefmt='-r',
                                         basefmt="k")
#basefmt=" " oculta a baseline
#markerline.set_markerfacecolor('none')
plt.title('Sequência exponencial sobre degrau unitário')
plt.ylabel('y')
plt.xlabel('t')

plt.show()

print("Figura 6 representa a função do item C.3)")
```

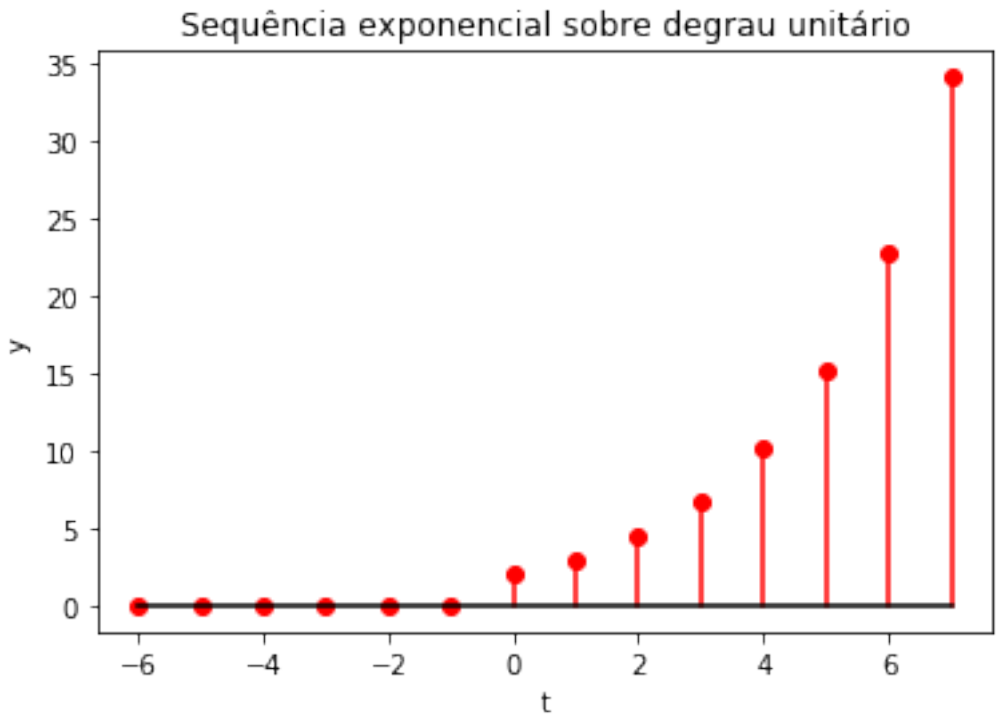


Figura 6 representa a função do item C.3)

C.4) $\alpha < -1$ e $A \neq 0$
Nesse caso, é esperado que a magnitude dos valores da sequência cresça com o aumento de n.
Digamos $\alpha = -1.5, A = 2.0$

[10]:
```
" Perfil do tempo (domínio) do sinal"
N=14
From=-6
To=From+(N-1)

" Formato do dado na forma de vetor "
x=np.linspace(start=From,
              stop=To,
              num=N
             )




" Formato do sinal na forma de vetor "

u=np.zeros(shape=(N))
```

```
X=np.arange(0,N,1)

" y e x estão relacionados por uma mesma indexação "

# no Python, não há um recurso igual ao match
for i in X :
    u[
        np.where(
            x==i
        )[0]
    ]=1.0

for i in np.arange(0,N,1):
    u[i]=2.0*u[i]*((1.5)**x[i])
y=u


markerline, stemlines, baseline=plt.stem(x,
                                         y,
                                         markerfmt='or',
                                         linefmt='-r',
                                         basefmt="k")
#basefmt=" " oculta a baseline
#markerline.set_markerfacecolor('none')
plt.title('Sequência exponencial sobre degrau unitário')
plt.ylabel('y')
plt.xlabel('t')

plt.show()

print("Figura 7 representa a função do item C.4)")
```

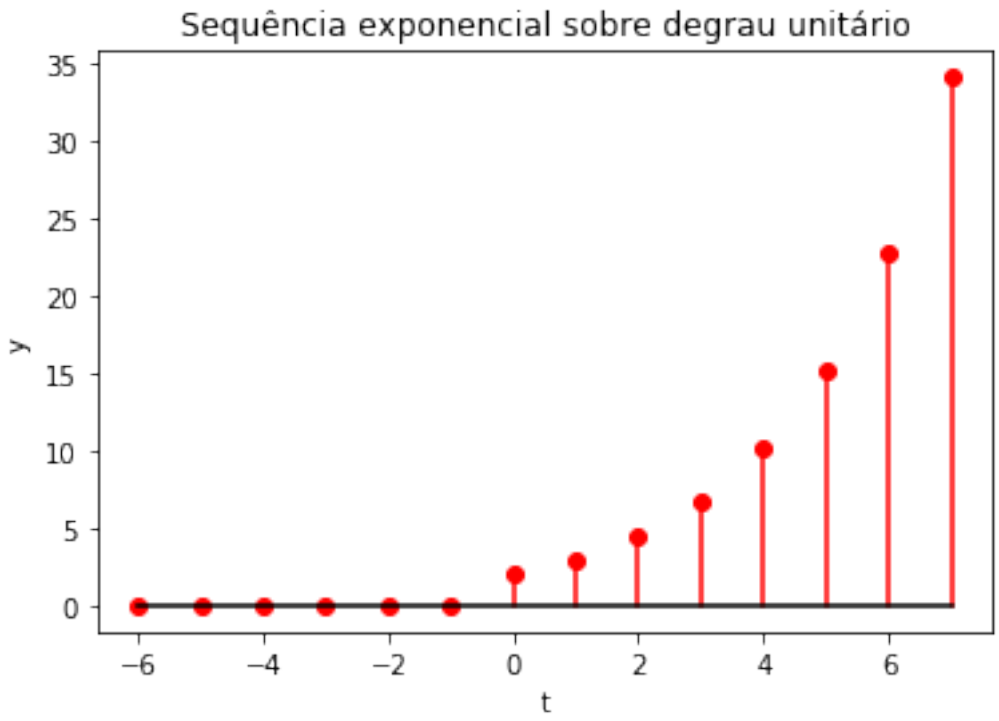


Figura 7 representa a função do item C.4)