

Signali i sistemi

Domaci 1

Anja Vujačić
Br. indeksa: 0307/2021

Parametri

N = 0, P = 3, Q = 0, R = 1

Sadžaj

1 Zadatak 1.	2
1.1 Prvi deo	2
1.2 Govorna sekvenca	9
2 Zadatak 2.	12
2.1 Prvi deo	12
2.2 Drugi deo	15
2.3 Drugi deo	27
3 Zadatak 3.	32
3.1 Prvi deo	32
4 Zadatak 3.	32
4.1 Prvi deo	32

1 Zadatak 1.

Osnovne osobine i vremenske transformacije signala

1.1 Izabrati signal i prikazati grafik signala $x[n]$.

$$X[n] = (n+1)(u[n+2] - u[n-3]) \quad (1)$$

Na narednim slikama se može videti postupak transformacije signala gde su primjene transformacije pomeranje, inverzija i skaliranje, u navedenom redosledu. Za slučaj pod a, prvo se izvrsava pomeranje za 3 mesta u levo, odnosno ceo grafik se translira 3 mesta u levo. Nakon toga sledi inverzija - odnosno grafik se preslikava u odnosu na y osu. Poslednja faza je faza skaliranja vremenske promjenljive - signal je kompirovan, određen broj odbiraka je izgubljen tj svaki drugi odbirak je izgubljen postoje u pitanju $2n$. Taj fenomen - da neki odbirci budu izgubljeni naziva se decimacija.

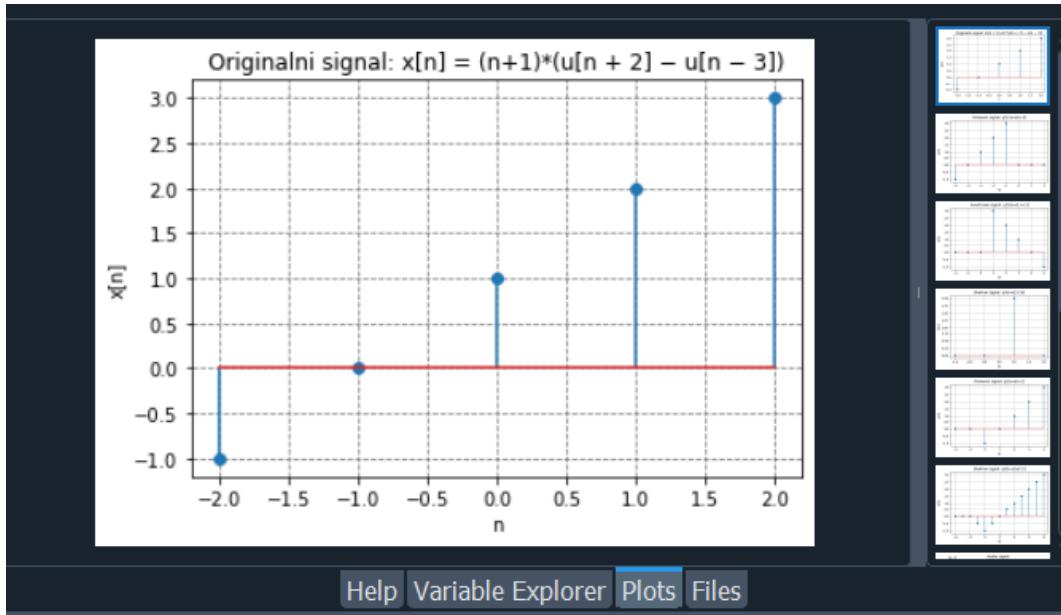
U slučaju pod b, prvo se vrši pomeranje za dva mesta u desno. U ovom primeru nema inverzije, tako da je sledeća faza skaliranje signala za $1/2$. U ovom slučaju, suprotno pod a, signal će se razvuci. U rezultujućem signalu neke odbirke ćemo imati, a neke ne tj nastace praznine gde ne znamo vrednosti. Primjenjuje se postupak interpolacije tj sprovescemu linearu interpolaciju kao najjednostavniju. Takodje su moguće primene i drugih vršata interpolacije. Vrednosti koje fale dobijemo kao linearu kombinaciju poznatih vrednosti. U prilogu su date slike, a kasnije je dat analitički oblik i formula.

Pod a treba napraviti signal:

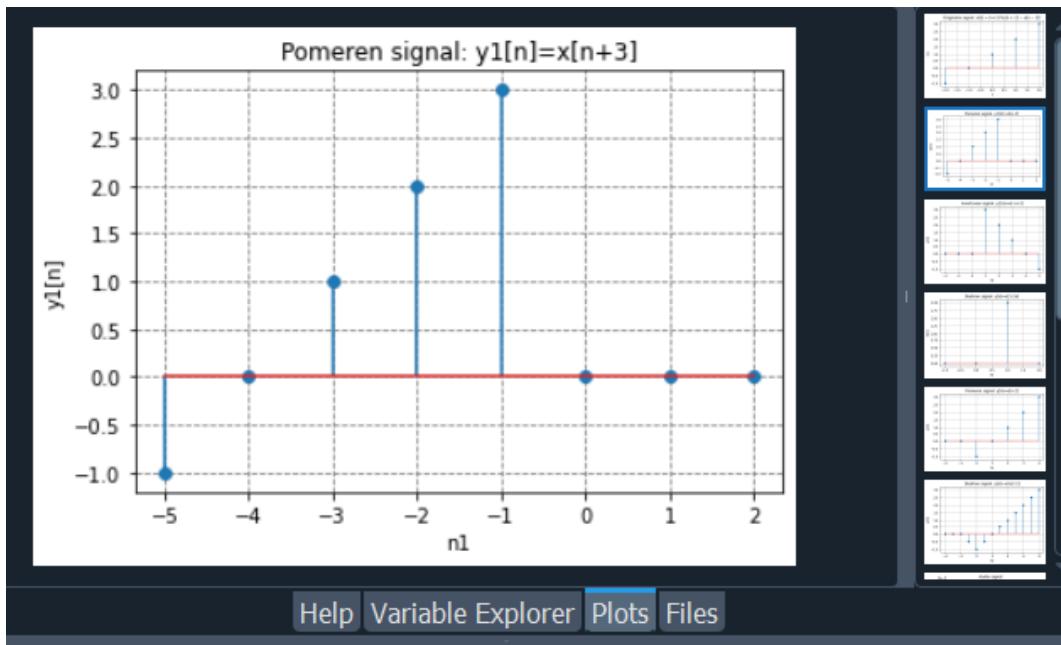
$$y1[n] = x[3-2n] \quad (2)$$

Pod b:

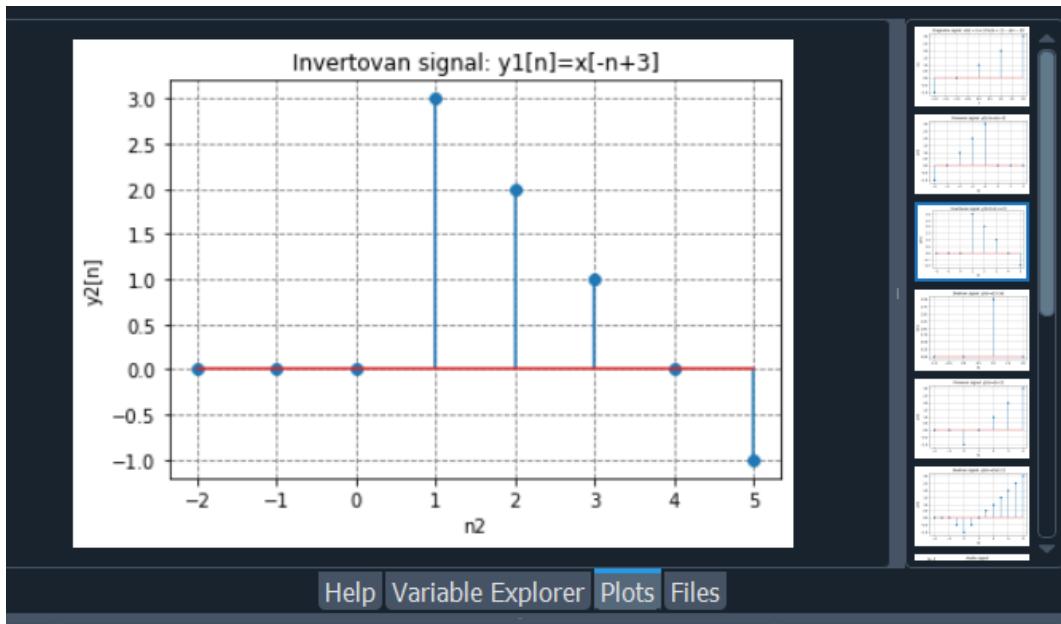
$$y2[n] = x[n/2-2] \quad (3)$$



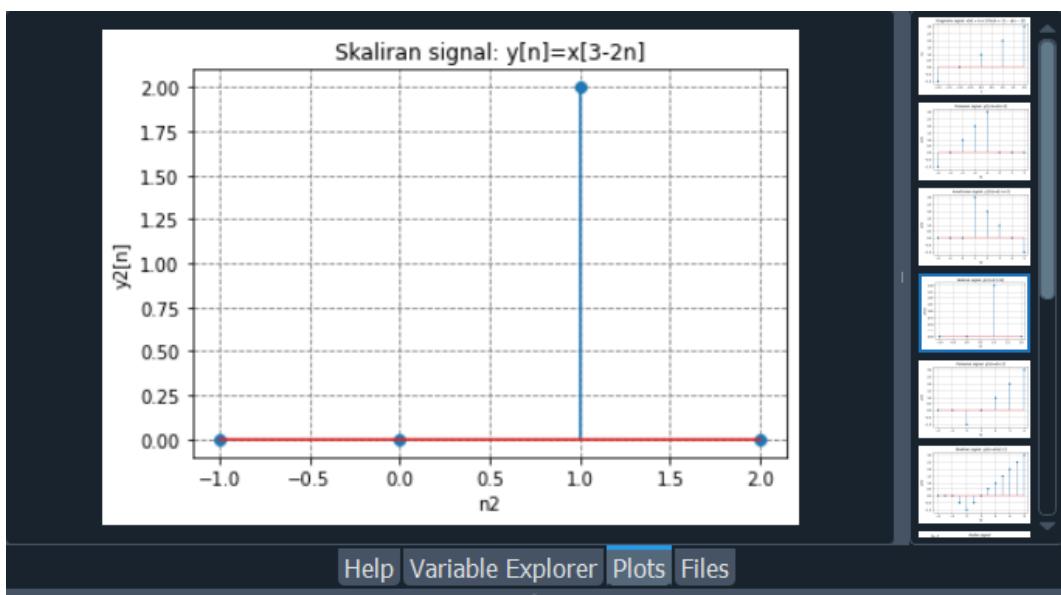
Slika 1: Inicijalni signal



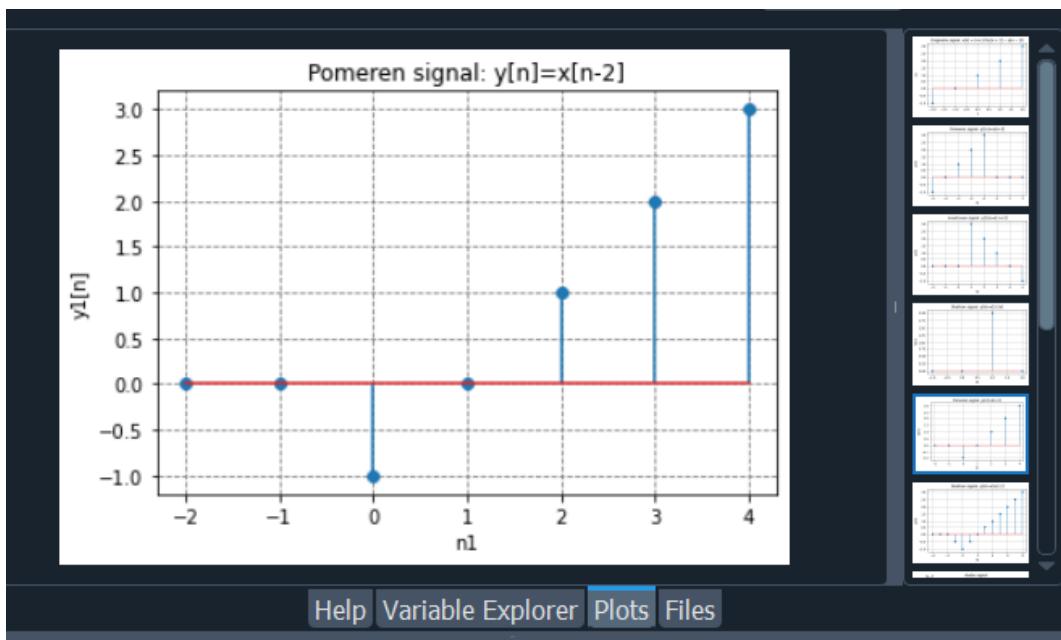
Slika 2: a) Pomeren signal za 3



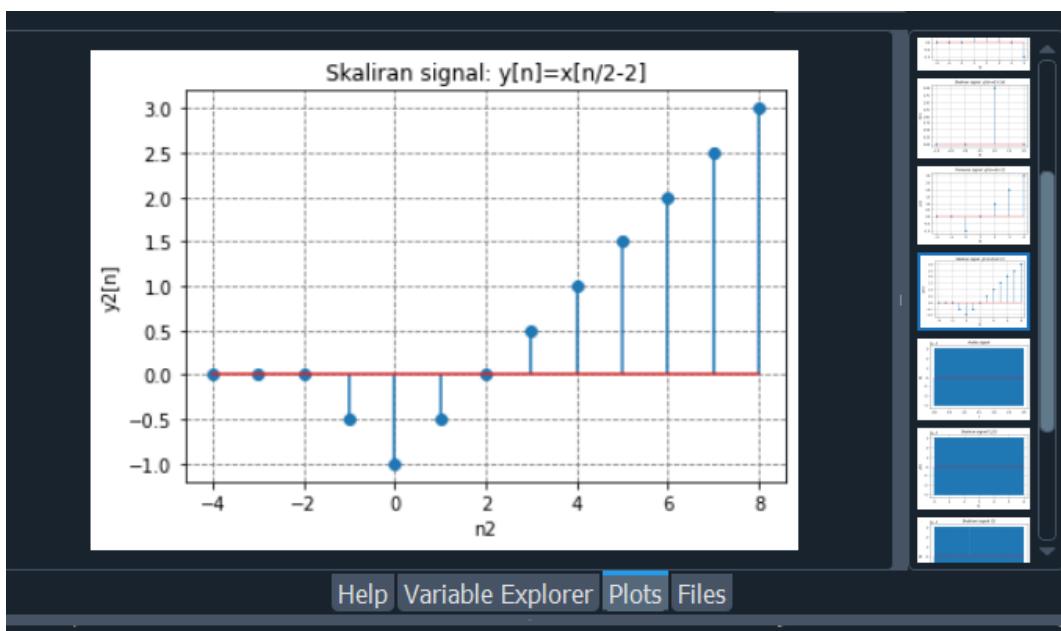
Slika 3: a) Invertovan signal



Slika 4: a) Skaliran signal, $2n$



Slika 5: b) Pomeren inicijalani signal za -2



Slika 6: b) Skaliran signal za $1/2n$

Za slučaj pod b) predlaze se metoda linearne interpolacije.

Nazovimo pomereni signal $x[n-2]$ sa $r[n]$ tj $r[n] = x[n-2]$ Nakon skaliranja, uz linearnu interpolaciju, rezultujući oblik signala $y2[n]$ je:

$$y2[n] = \begin{cases} r[k] & , \quad n = 2k \\ \frac{r[k]+r[k+1]}{2} & , \quad n = 2k+1 \end{cases} \quad (4)$$

Za skiciranje konacanog grafika potrebno je na osnovu izraza (4) izracunati vrednosti. Vidimo da će vrednost signala biti nula za $n \leq -2$ i $n \geq 9$, dok ostale vrednosti dobijamo direktnom zamenom.

Za skiciranje grafika za primer a koriscen je sledeći kod u programskom jeziku Python:

```
#Zadatak 1
start_n=-2
d_n=1
end_n=3
n=np.arange(start_n, end_n, d_n)
x =(n+1)*((n>=-2)*(n<3))

plt.figure()
plt.stem(n,x)
plt.xlabel('n')
plt.ylabel('x[n]')
plt.title('Originalni signal: x[n] = (n+1)*(u[n + 2] - u[n - 3])')
plt.grid(b=True,which="both",color='grey',linestyle='--')

plt.show()
```

Slika 7:

```
#Pomeranje za 3:
n0=3

y1 = np.concatenate((x,np.zeros(n0)))
n1 = np.concatenate((np.arange(start_n-n0, start_n, d_n),n))
plt.figure()
plt.stem(n1,y1)
plt.xlabel('n1')
plt.ylabel('y1[n]')
plt.title('Pomeren signal: y1[n]=x[n+3]')
plt.grid(b=True,which="both",color='grey',linestyle='--')

plt.show()
```

Slika 8:

```
#Inverzij:
y2 = y1[::-1]
n2=-n1[::-1]
plt.figure()
plt.stem(n2,y2)
plt.xlabel('n2')
plt.ylabel('y2[n]')
plt.grid(b=True,which="both",color='grey',linestyle='--')

plt.title('Invertovan signal: y1[n]=x[-n+3]')
plt.show()
```

Slika 9:

```

#Skaliranje

t=np.arange(0,len(n2),1)
skaliranje=2

y3 = y2[::-skaliranje] # ubrzavanje
n3 = n2[::-skaliranje]/skaliranje
plt.figure()
plt.figure()
plt.stem(n3,y3)
plt.xlabel('n2')
plt.grid(b=True,which="both",color='grey',linestyle='--')

plt.ylabel('y2[n]')
plt.title('Skaliran signal: y[n]=x[3-2n]')
plt.show()

```

Slika 10:

1.2 Rad sa govornim sekvencama

Potrebito je snimiti govornu sekvencu u trajanju od 3s sa frekvencijom 8000Hz pa potom formirati odgovarajuće signale i reprodukovati ih sa razlicitim frekven-cijama. Formirani su signali $y1[n] = S[n/2]$ sa frekvencijom 8kHz i $y2 = S[2n]$ sa frekvencijom 8kHz. Takodje originalni signal je reprodukovana sa frekvencijama 4kHz i 16kHz.

Python kod za snimanje sekvence u trajanju od 3s i sa frekvencijom 8kHz, koji takodje pravi grafik i skicira govornu sekvencu:

```
%%%

# snimanje i analiza zvucnog signala

duration = 3 # trajanje snimka
samplerate=8000 # ucestanost odabiranja

myrecording=sd.rec(int(duration*samplerate), samplerate=samplerate, channels=1)
sd.wait()
wavfile.write('./sekvenca.wav', samplerate, myrecording)

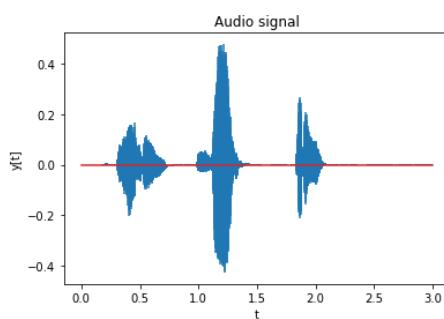
samplerate, data=wavfile.read('./sekvenca.wav')

dt=1/samplerate
t=np.arange(0,dt*len(data),dt)
chanel1=data
plt.figure()
plt.stem(t,chanel1, markerfmt=" ")
plt.xlabel('t')
plt.ylabel('y[t]')
plt.title('Audio signal')
plt.show()

sd.play(data,samplerate)
```

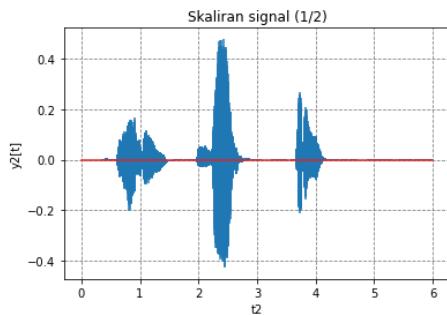
Slika 11:

Na slici ispod dat je grafik generisan za govornu sekvencu "Jedan dva tri":

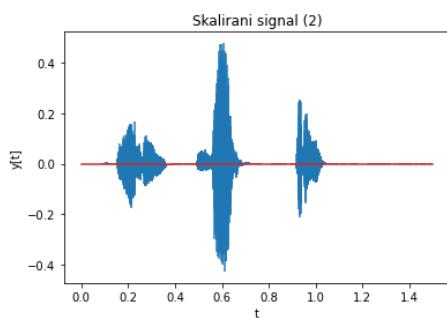


Slika 12:

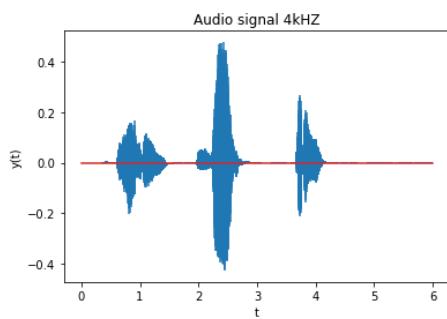
Na narednim slikama prikazani su grafici traženih funkcija:



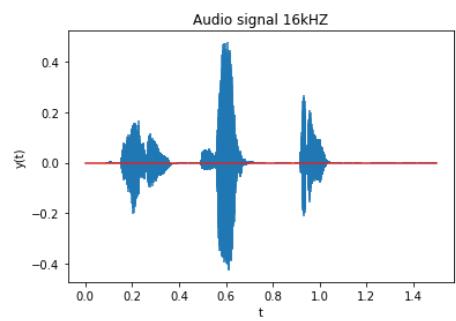
Slika 13:



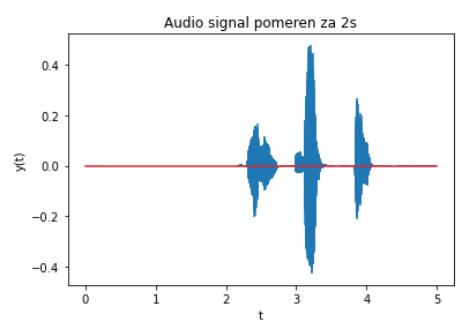
Slika 14:



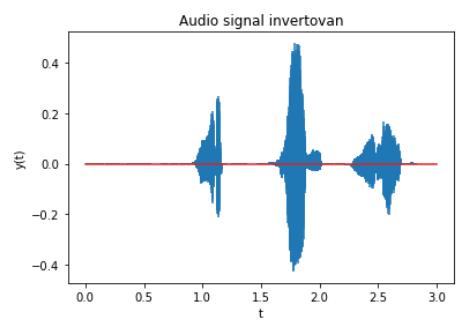
Slika 15:



Slika 16:



Slika 17:



Slika 18:

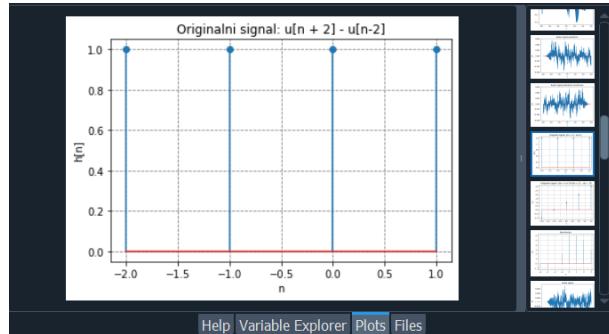
U okviru teksta zadatka, trazeno je da se posmatra i palindrom. U ovom resenju, ta stacka je realizovana tako sto se generise nova sekvenca, koju korisnik izgovara i koja je palindrom. Ta sekvenca se invertuje i prikazuje na grafiku.

Kodovi za posmatrane funkcije prilozeni su u fajlu.

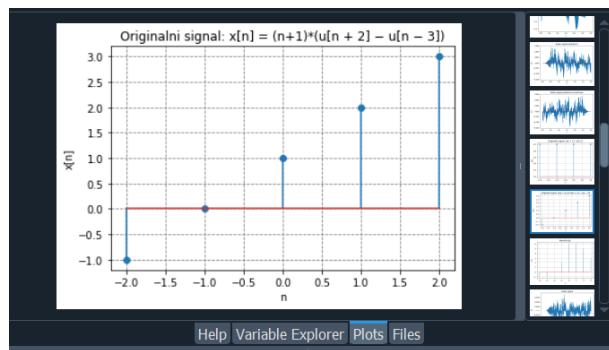
2 Zadatak 2.

Konvolucija

2.1 Analitički odrediti i skicirati konvoluciju signala $x[n] = (n + 1)(u[n + 2] - u[n - 3])$ i $h[n] = u[n + 2] - u[n - 2]$



Slika 19: Signal $x[n]$



Slika 20: Signal $h[n]$

```

start_h=-2
d_n=1
end_n=2
n=np.arange(start_n, end_n, d_n)
h =((n>=-2)*(n<2))

plt.figure()
plt.stem(n,h)
plt.xlabel('n')
plt.ylabel('h[n]')
plt.title('Originalni signal: u[n + 2] - u[n-2]')
plt.grid(b=True,which="both",color='grey',linestyle='--')

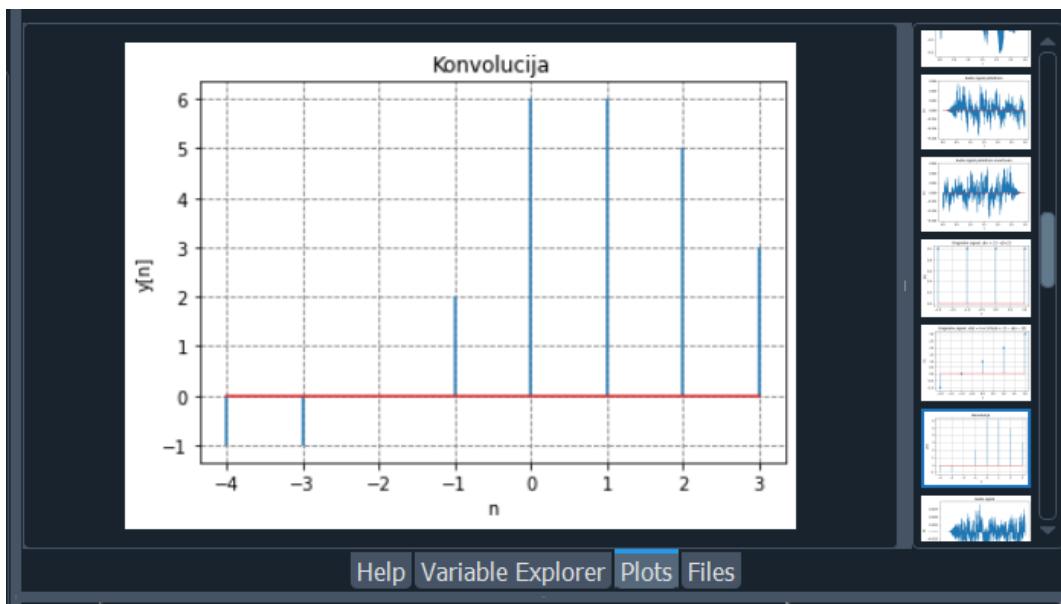
#%%
start_n=-2
d_n=1
end_n=3
n=np.arange(start_n, end_n, d_n)
x =((n+1)*((n>=-2)*(n<3)))

plt.figure()
plt.stem(n,x)
plt.xlabel('n')
plt.ylabel('x[n]')
plt.title('Originalni signal: x[n] = (n+1)*(u[n + 2] - u[n - 3])')
plt.grid(b=True,which="both",color='grey',linestyle='--')

plt.show()

```

Slika 21:



Slika 22: Grafik konvolucije

```

#%%
odziv=np.convolve(h,x)
#odziv=odziv/max(np.absolute(odziv))
no=np.arange(-4,4,1)

plt.figure()
plt.stem(no,odziv,markerfmt=" ")
plt.xlabel('n')
plt.ylabel('y[n]')
plt.title('Konvolucija')
plt.grid(b=True,which="both",color='grey',linestyle='--')
plt.show()

```

Slika 23: Python kod za konvoluciju

Postupak analitickog racunanja dat je na sledecoj slici:

2.2 Primena konvolucije u obradi zvucnih signala

Po zahtevu zadatka zadatka, snimljena je govorna sekvencia u trajanju od 3s i ucestanoscu $fs=48\text{kHz}$. odziv nekog sistema na bilo koju pobudu može dobiti kao konvolucija te pobude sa impulsnim odzivom sistema, postupak ce biti objasnjen na dalje.

Kod koji generise proizvoljnu sekvencu u trajanju od tri sekunde, i prikazuje njen grafik u programskom jeziku Python glasi:

```
%%%
#konvolucija zvucni signal Snimiti govornu sekvencu trajanja 3 sekunde sa učestanošću
#odabiranja fs = 48kHz.

duration = 3 # trajanje snimka
samplerate=48000 # ucestanost odabiranja

myrecording=sd.rec(int(duration*samplerate), samplerate=samplerate,channels=1)
sd.wait()
wavfile.write('./sekvenca2.wav', samplerate,myrecording)

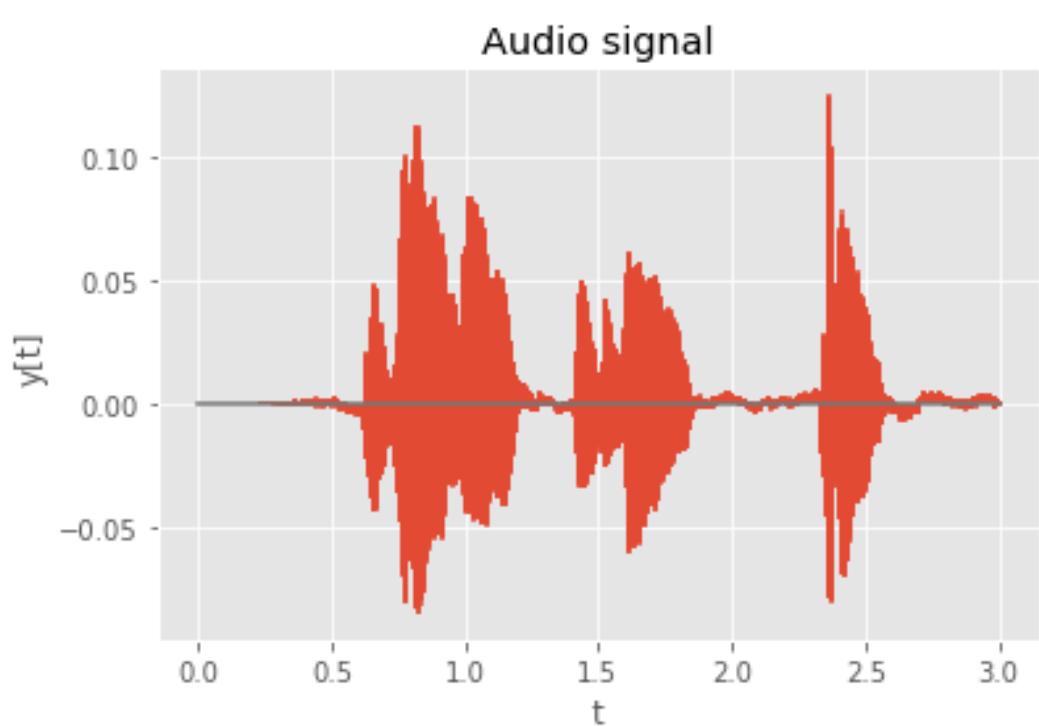
samplerate, data=wavfile.read('./sekvenca2.wav')

dt=1/samplerate
t=np.arange(0,dt*len(data),dt)
channel1=data
plt.figure()
plt.stem(t,channel1, markerfmt=" ")
plt.xlabel('t')
plt.ylabel('y[t]')
plt.title('Audio signal')
plt.show()
```

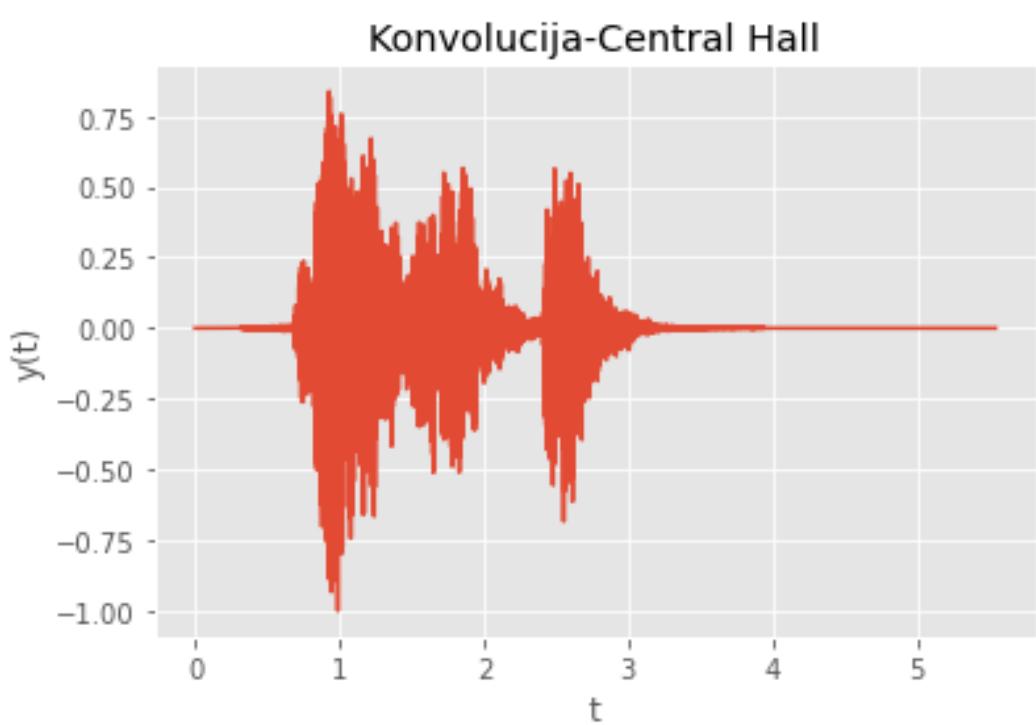
Slika 24: Python kod za konvoluciju

Dalje kako se odziv nekog sistema na bilo koju pobudu može dobiti kao konvolucija te pobude sa impulsnim odzivom sistema, u prilogu se nalaze grafici nastali kao konvolucija nase gorovne sekvence i odzva sistema.

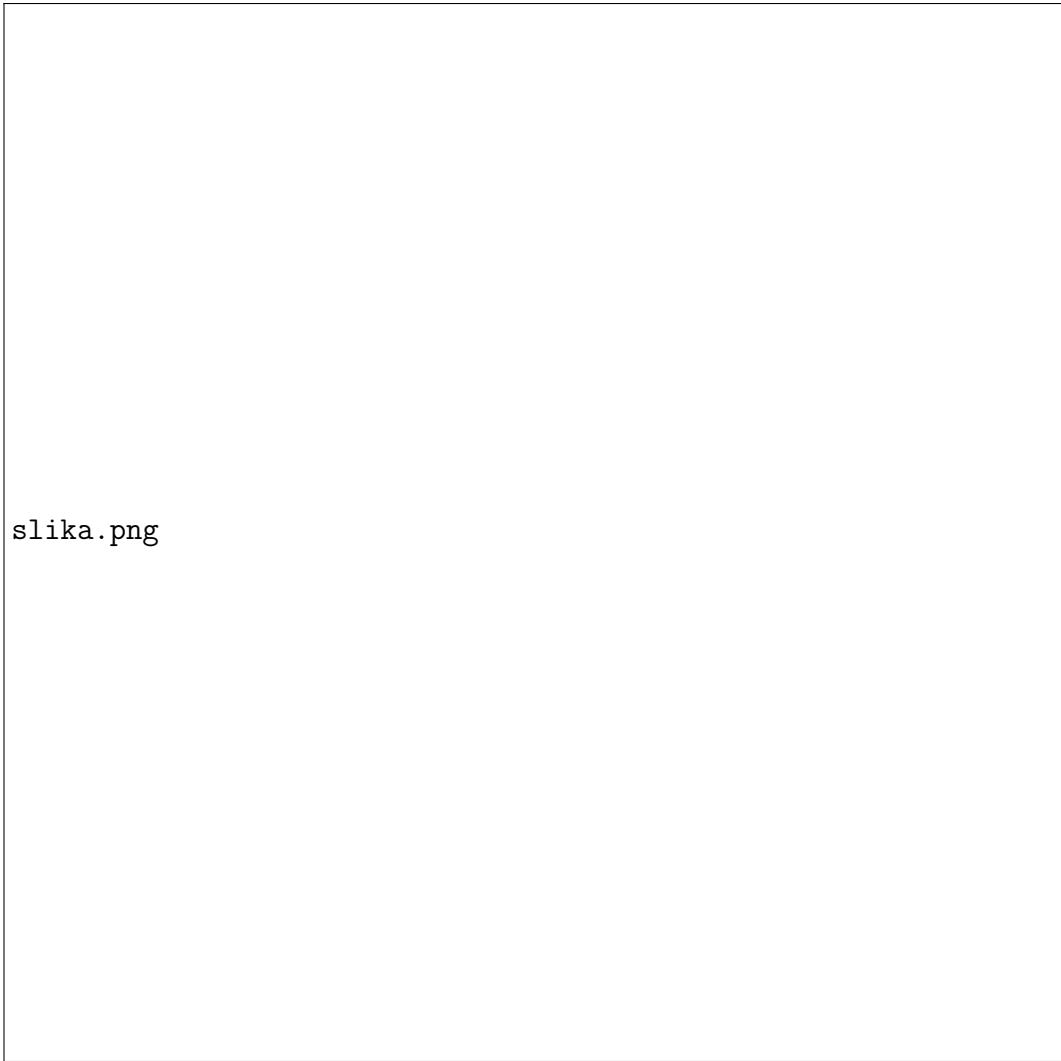
Pre toga, dacemo prikaz Python koda za generisanje konvolucije. Generisanje ostalih se postize na isti nacin;



Slika 25:



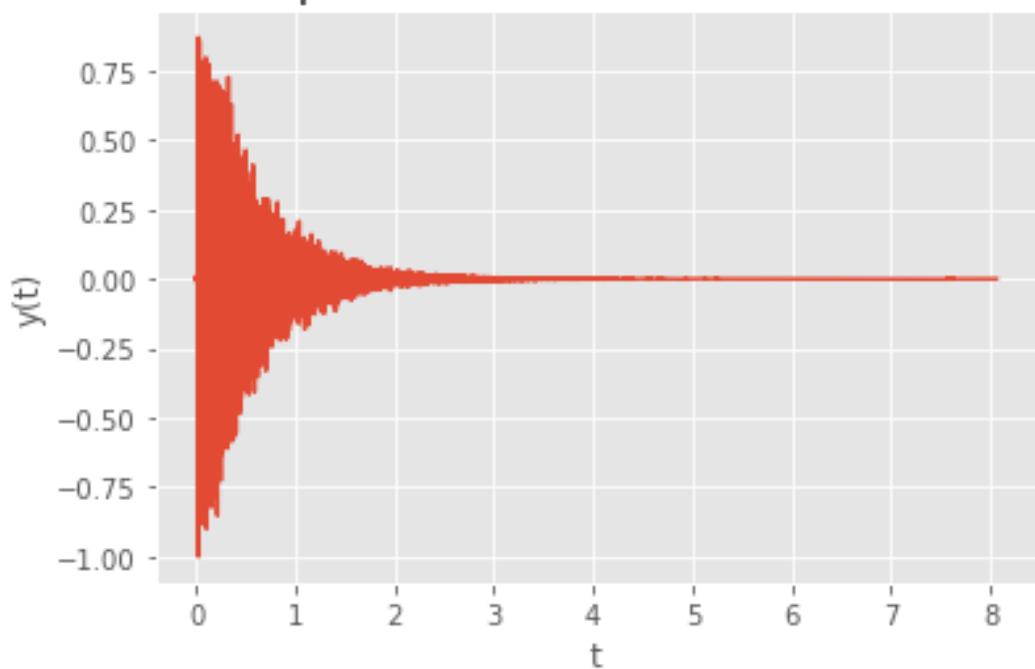
Slika 26:



slika.png

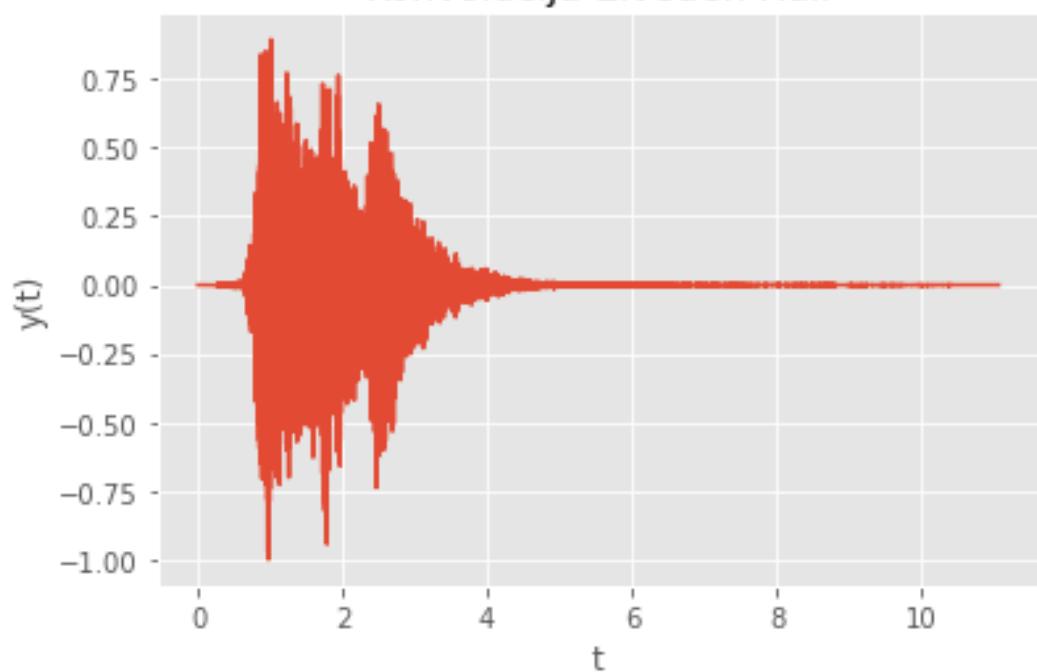
Slika 27:

Impulsni odziv-Elveden Hall - Suffolk



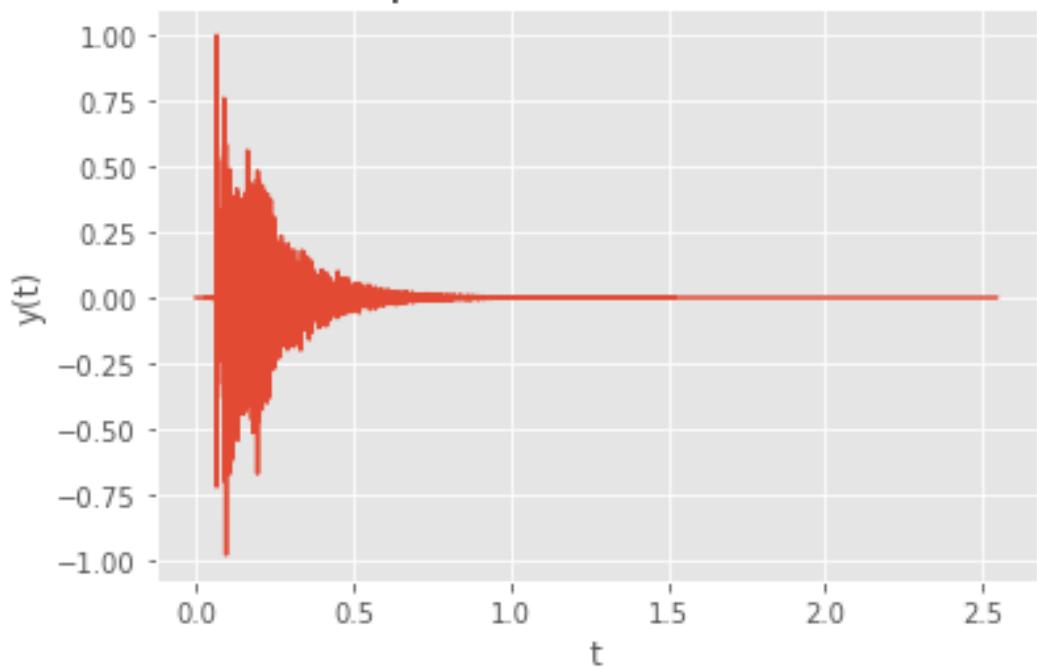
Slika 28:

Konvolucija-Elveden Hall

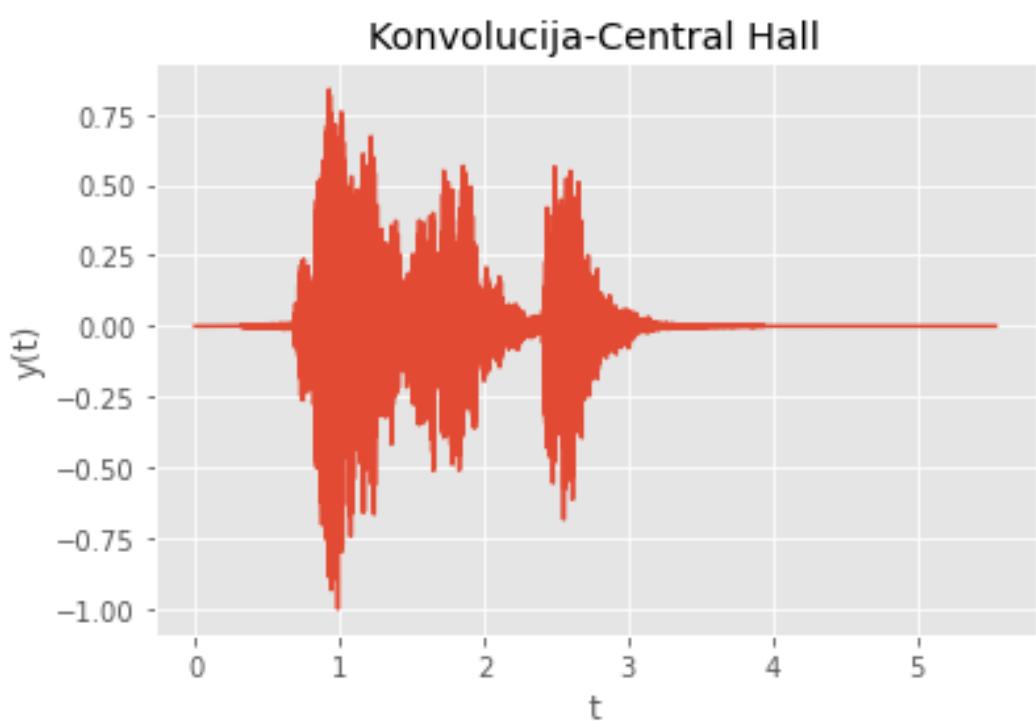


Slika 29:

Impulsni odziv-Central Hall

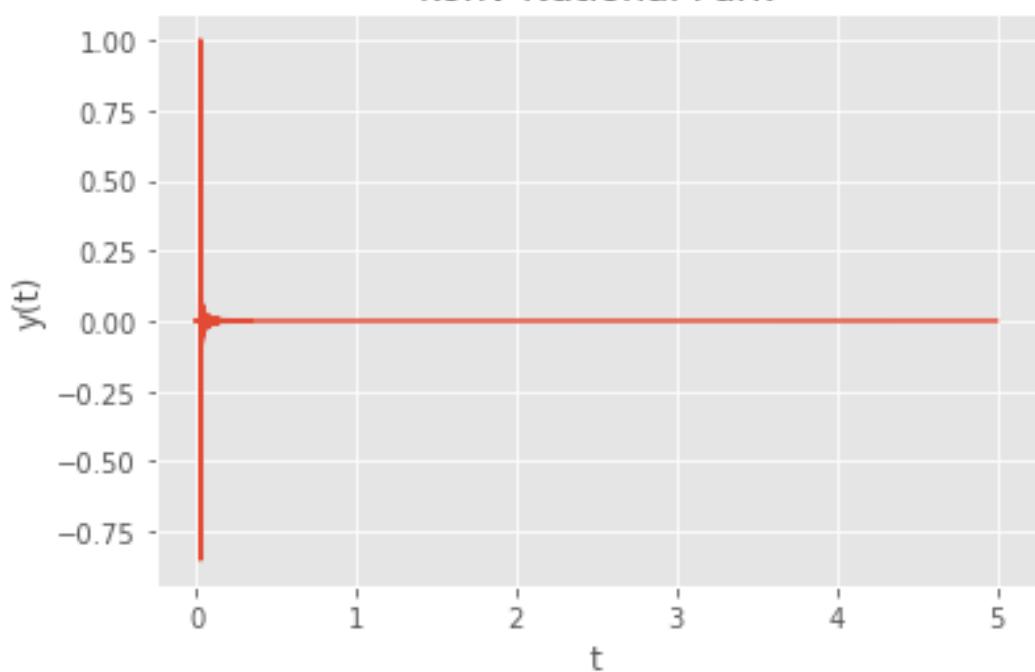


Slika 30:

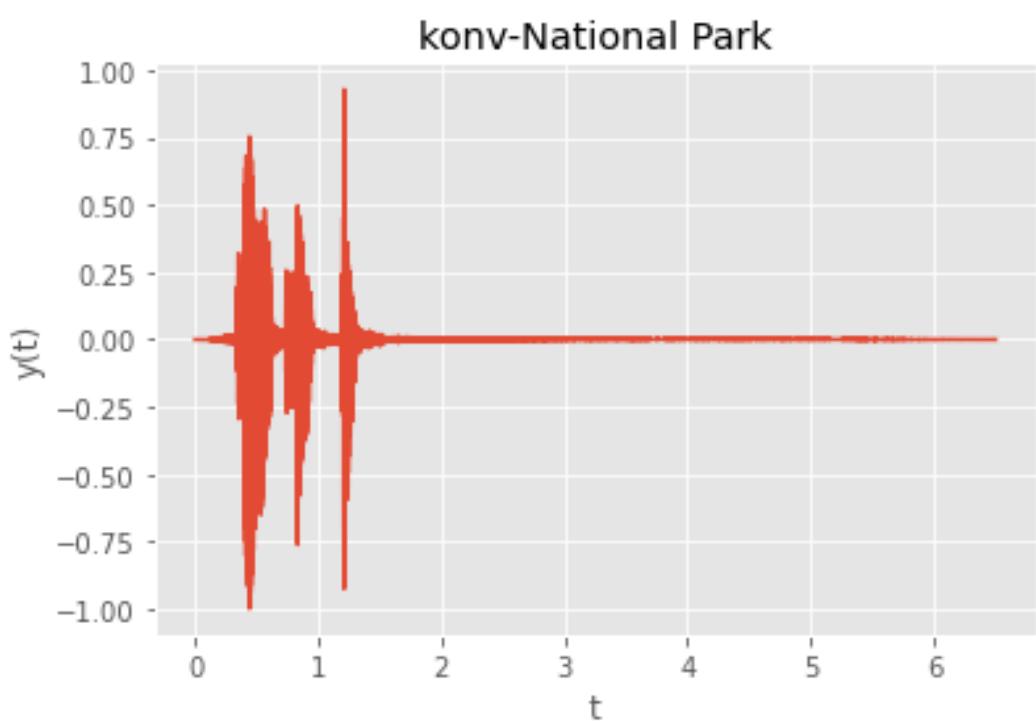


Slika 31:

konv-National Park

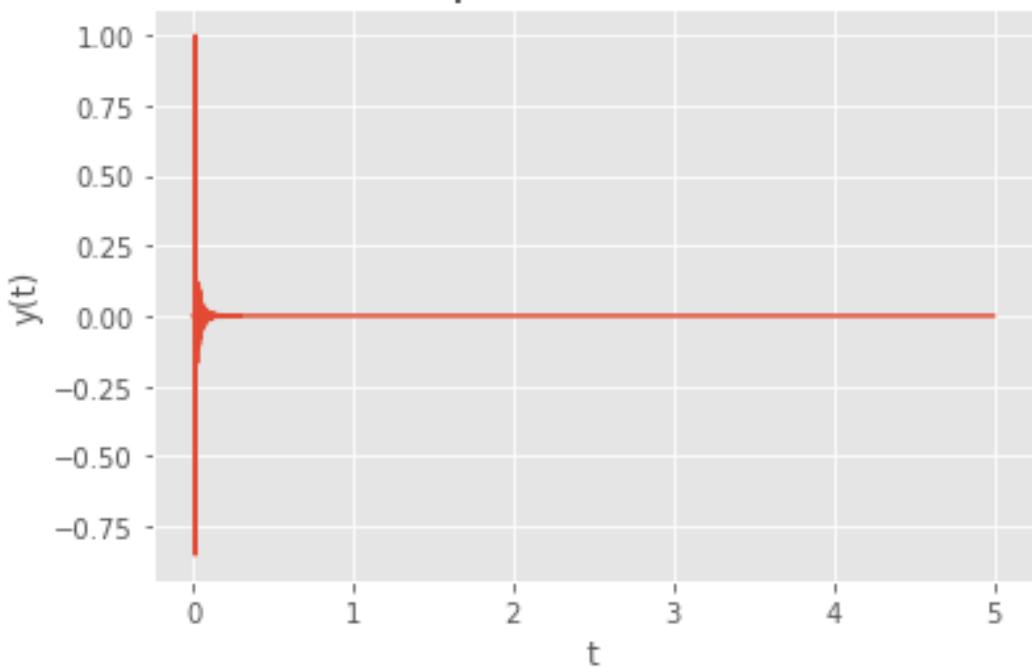


Slika 32:

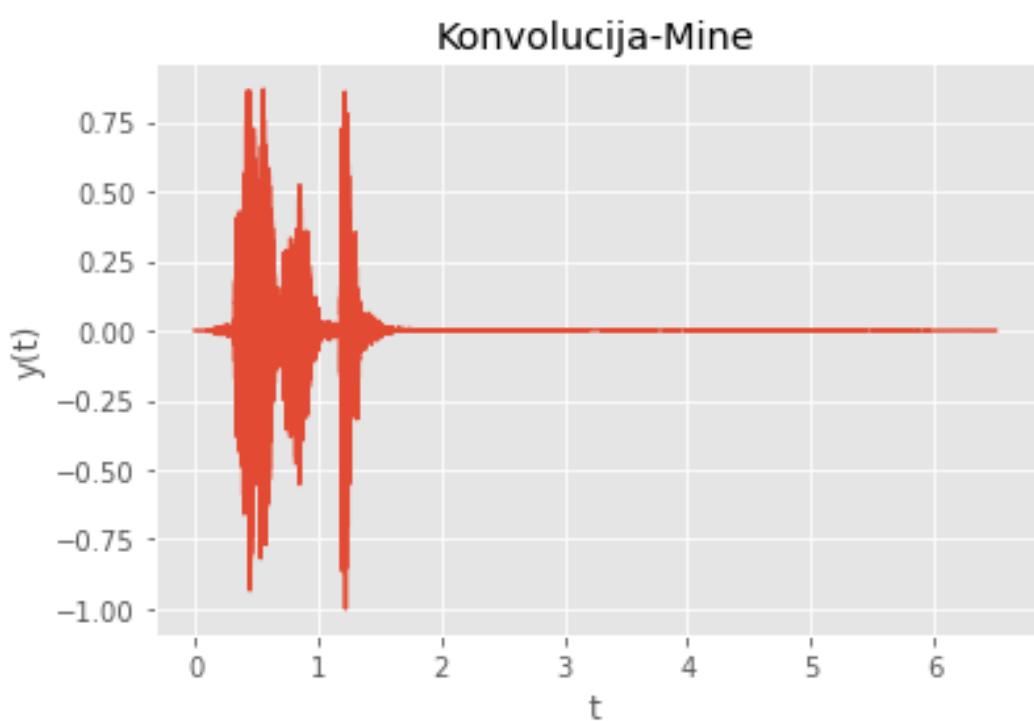


Slika 33:

Impulsni odziv-Mine



Slika 34:



Slika 35:

2.3 Primena konvolucije u obradi slike

Kako je vrednost parametra $P = 3$ na trimenice se maske M1, M5, M6 i M9. Dobijeni su sledeci rezultati:



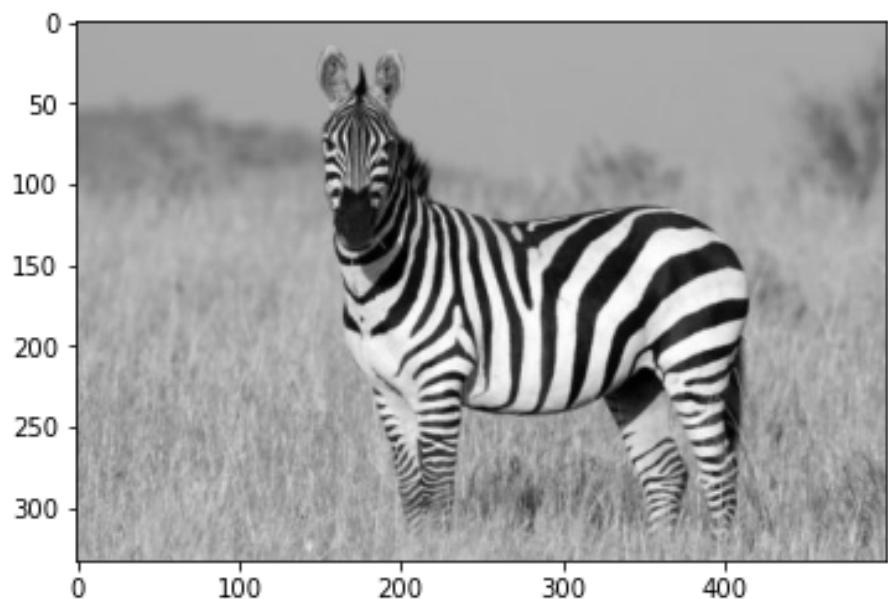
Slika 36: Originalna slika

Slika je prvo prebacena u crno-belu sliku, pomocu sledeceg Python koda

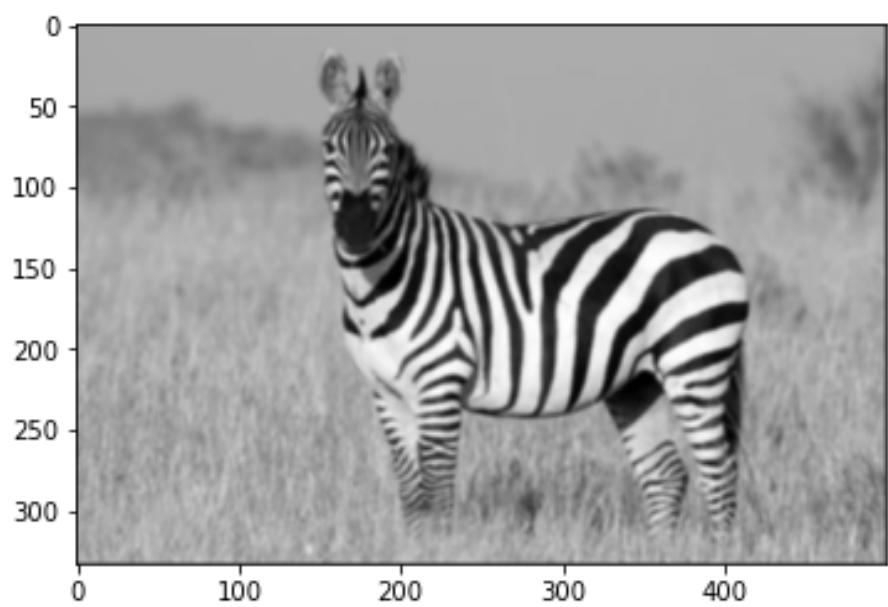
```
def rgb2gray(rgb):
    return np.dot(rgb[...,:3], [0.2989, 0.5870, 0.1140])

img = mpimg.imread('C:\_Faks\Semestar III\SIS\Prvi domaci zadatak 22_23\Test_slika.png')
img = rgb2gray(img)
plt.figure()
imgplot = plt.imshow(img, cmap=plt.get_cmap('gray'))
plt.show()
```

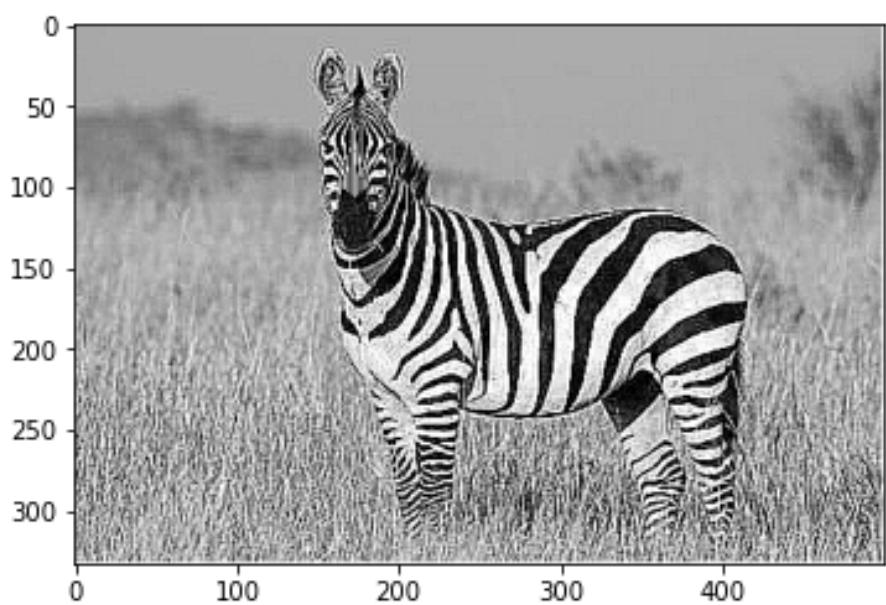
Slika 37: Kod



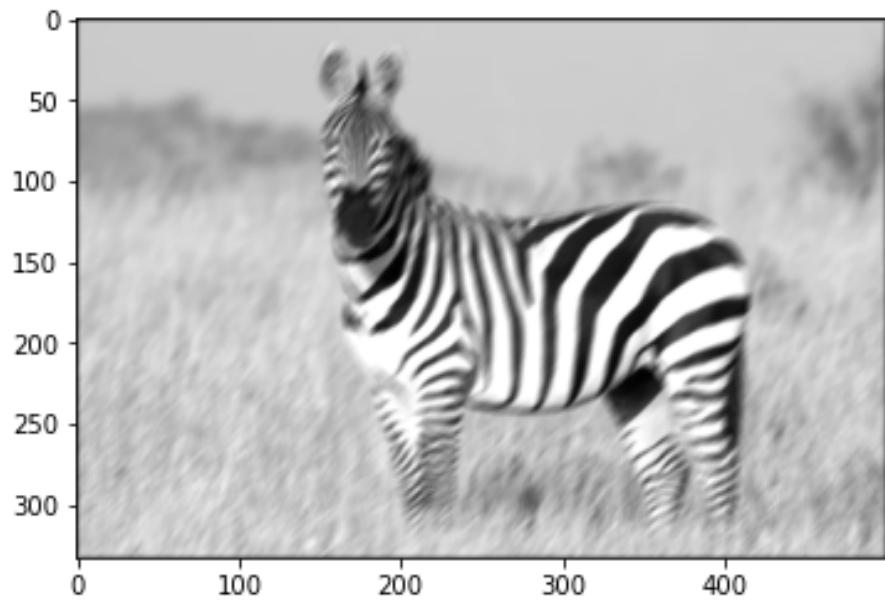
Slika 38: Originalna slika u crno-beloj boji



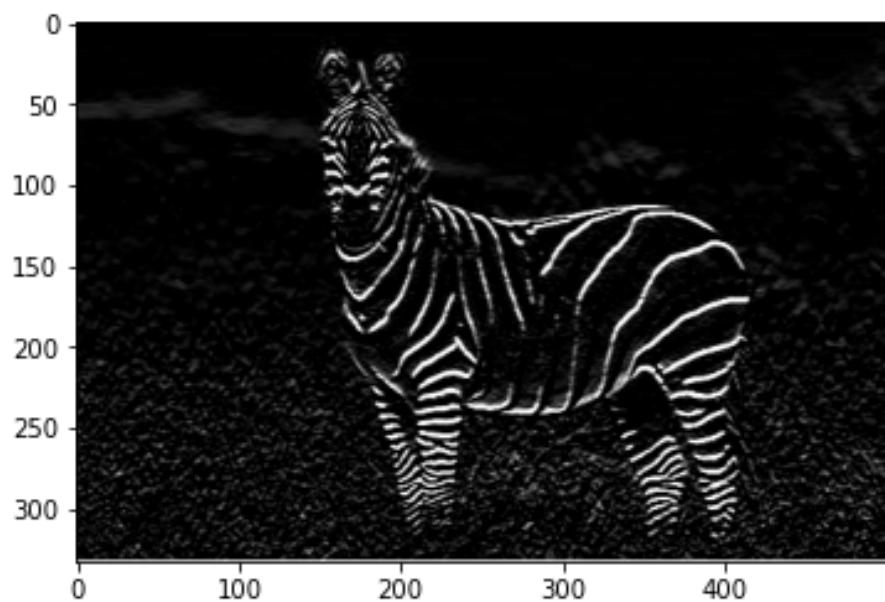
Slika 39: Posle primene maske M1



Slika 40: Posle primene maske M5



Slika 41: Posle primene maske M6



Slika 42: Posle primene maske M9

```

M1=np.array([[1/9, 1/9, 1/9],[1/9, 1/9, 1/9],[1/9, 1/9, 1/9]])
img_m0=signal.convolve2d(img, M1, mode='same').clip(0,1)
plt.figure()
imgplot = plt.imshow(img_m0, cmap=plt.get_cmap('gray'))
plt.show()

M5=np.array([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])
img_sharp=signal.convolve2d(img,M5, mode='same').clip(0,1)
plt.figure()
imgplot = plt.imshow(img_sharp, cmap=plt.get_cmap('gray'))
plt.show()
%%

M6=np.array([[0,0,0,0,0,1/5],[0,0,0,0,1/5,0],[0,0,0,1/5,0,0], [0,0,1/5,0,0,0], [0,1/5,0,0,0,0], [1/5,0,0,0,0,0]])
img_sharp=signal.convolve2d(img, M6, mode='same').clip(0,1)
plt.figure()
imgplot = plt.imshow(img_sharp, cmap=plt.get_cmap('gray'))
plt.show()

%%

M9=np.array([[-1, -2, -1],[-2, 0, 0],[1, 2, 1]])
img_sharp=signal.convolve2d(img,M9, mode='same').clip(0,1)
imgplot = plt.imshow(img_sharp, cmap=plt.get_cmap('gray'))
plt.show()

```

Slika 43: Python kod za primenu maski nad slikom

3 Zadatak 3.

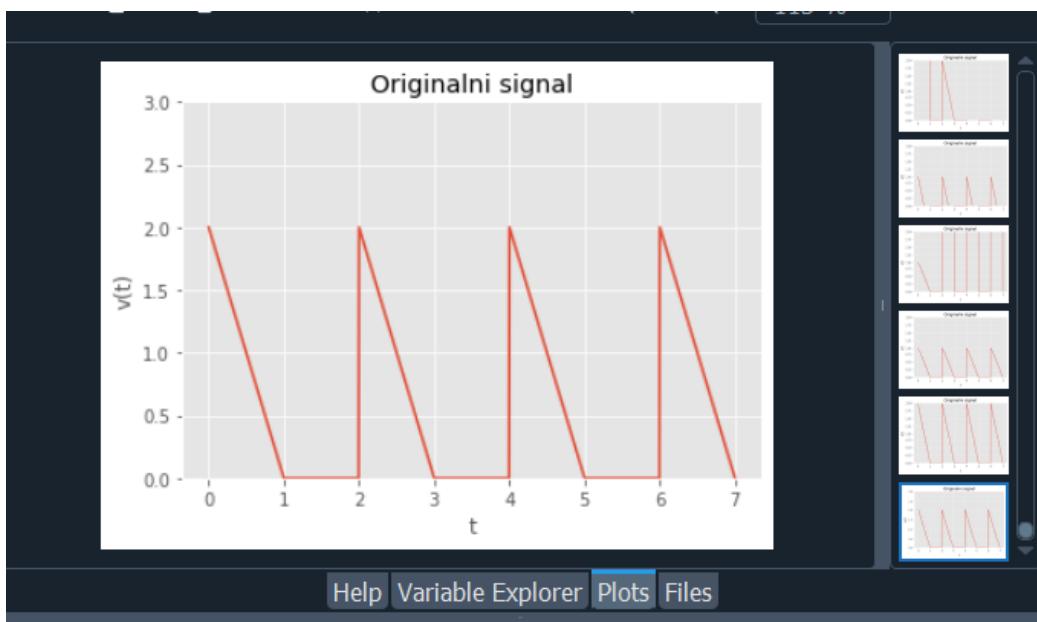
Osnovne osobine signala

3.1 Analitičkim postupkom odrediti da li je sistem S:

4 Zadatak 3.

Furijeovi redovi

4.1 Napisati analitički oblik signala $v(t)$, odrediti njegovu osnovnu periodu T i učestanost ω_0 , i izvesti izraz za koeficijente Furijeovog reda a_k



Slika 44:

Sa slike se vidi da je period signala $T = 2s$. Odavde sledi da je kružna učestanost $\omega_0 = \frac{2\pi}{T} = \pi rad/s$. Sa slike se može očitati i analitički oblik signala $v(t)$:

$$v(t) = \sum_{k=-\infty}^{\infty} 2 \cdot (2k - t + 1) (u(t - 2k) - u(t - 2k - 1)) \quad (5)$$

Za skiciranje grafika korišćen je sledeći kod u programskom jeziku Python:

```
#Furijeov red

def u(t):
    return 0 if t<0 else 1

def v(L):
    def vk(t):
        s=0
        for k in range(-L,L+1):
            s += 2*( 2*k-t+1) * ( u ( t - 2 * k ) - u ( t - 2 * k -1))
        return s
    return vk

dt = 0.005

data = np . arange ( 0 , 7 + dt , dt )
values = np . vectorize ( v ( 9 ) , otypes =[ float ] ) ( data )
start_t=-4
d_t=0.01
end_t=4
t=np.arange(start_t, end_t, d_t)
plt.figure()
plt.plot(data,values)
plt.xlabel('t')
plt.ylabel('v(t)')
plt.ylim([0,3])
plt.title('Originalni signal')
plt.show()
```

Slika 45:

Izvodenje koeficijenata Furijeovog reda

Furijeov red funkcije $v(t)$ dat je sumom:

$$v(t) = \sum_{k=-\infty}^{+\infty} a_k \cdot e^{j\omega_0 kt} \quad (6)$$

gde su koeficijenti a_k definisani sa:

$$a_k = \frac{1}{T} \int_T v(t) e^{-j\omega_0 kt} dt \quad (7)$$

Odredimo sada koeficijente Furijeovog reda:

$$a_k = \frac{1}{T} \int_T v(t) e^{-j\omega_0 kt} dt = - \int_0^1 (t-1) e^{-j\omega_0 kt} dt = \left\{ \begin{array}{l} u = t-1 \\ du = dt \end{array}, \begin{array}{l} dv = e^{-j\omega_0 kt} \\ v = -\frac{1}{k\omega_0} e^{-j\omega_0 kt} \end{array} \right\} \quad (8)$$

Vidimo da koeficijent a_k nije definisan prethodnom reklacijom za $k = 0$, tako da koeficijent a_0 moramo računati posebno, koristeći se sledecom relacijom uzimajući $k = 0$.

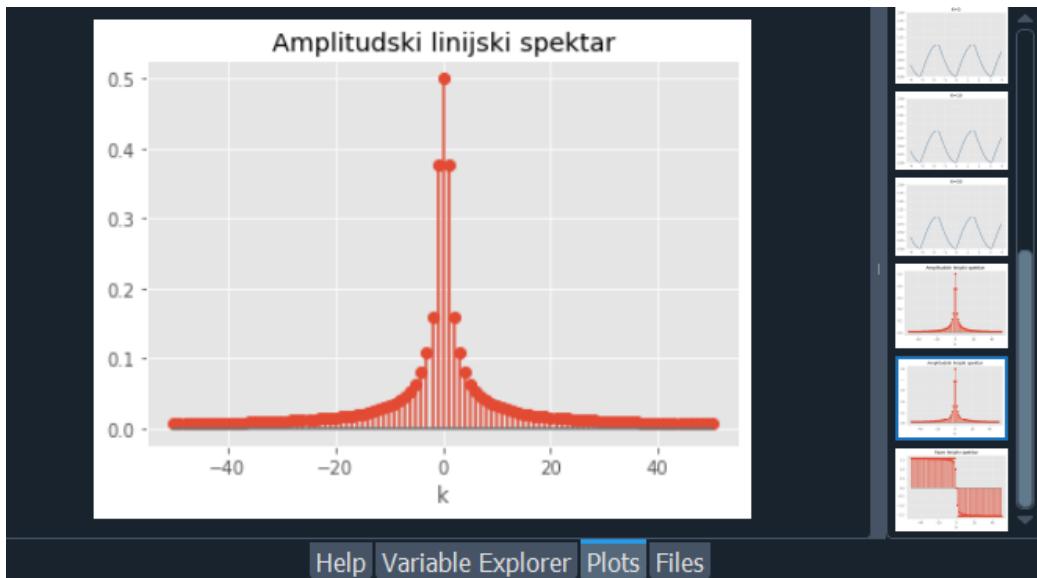
$$a_0 = \frac{1}{T} \int_T v(t) dt = \frac{1}{2} \int_0^1 v(t) dt = \frac{1}{2} \int_0^1 (2-2t) dt = \frac{1}{2} \cdot (2t - t \cdot t) \Big|_0^1 = \frac{1}{2} \quad (9)$$

Konačno, koeficijenti a_k Furijeovog reda funkcije $v(t)$ definisani su sa

$$a_k = \begin{cases} \frac{1}{2}, & k = 0 \\ \frac{1-jk\pi-(-1)^k}{\pi^2 k^2}, & k \neq 0 \end{cases} \quad (10)$$

$$\begin{aligned}
 d\Phi &= \frac{1}{T} \int_0^T u(t) e^{-j\omega_0 t} dt = - \int_0^1 (t-1) e^{-j\omega_0 t} dt \\
 - \int_0^1 (t-1) e^{-j\omega_0 t} dt &\quad u = t-1 \quad du = e^{-j\omega_0 t} \\
 du = dt &\quad u = -e^{-j\omega_0 t} \\
 &= - \frac{(t-1) e^{-j\omega_0 t}}{j\omega_0} - \int - \frac{e^{-j\omega_0 t}}{j\omega_0} dt = \quad u = -j\omega_0 t \\
 &= - \frac{(t-1) e^{j\omega_0 t}}{j\omega_0} - \frac{1}{j^2 \omega_0^2} \int e^u du = \\
 &= - \frac{(t-1) e^{j\omega_0 t}}{j\omega_0} - \frac{1}{j^2 \omega_0^2} \cdot e^u = \\
 &= - \frac{(t-1) \cdot e^{j\omega_0 t}}{j\omega_0} - \frac{e^{-j\omega_0 t}}{j^2 \omega_0^2} \\
 - \int_0^1 (t-1) \cdot e^{-j\omega_0 t} dt &= \frac{(t-1) \cdot e^{j\omega_0 t}}{j\omega_0} + \frac{e^{-j\omega_0 t}}{j^2 \omega_0^2} + C \\
 \int_0^1 -(t-1) \cdot e^{-j\omega_0 t} dt &= \frac{e^{-j\omega_0}}{j^2 \omega_0^2} - \frac{(-1)}{j\omega_0} - \frac{1}{j^2 \omega_0^2} + \frac{e^{-j\omega_0}}{j^2 \omega_0^2} \\
 &= \frac{e^{-j\omega_0} + j\omega_0 - 1}{j^2 \omega_0^2} \quad \omega_0 = \pi \\
 &= \frac{e^{-j\pi} + j\pi - 1}{j^2 \pi^2} = \frac{e^{(-j)\pi^k}}{j^2 \pi^2} + \frac{j\pi - 1}{j^2 \pi^2} = \\
 \alpha_k &= \frac{e^{-j\pi} + j\pi - 1}{j^2 \pi^2} = \frac{(-1)^k}{\pi^2} - \frac{j\pi - 1}{\pi^2} = \boxed{\frac{1 - j\pi - (-1)^k}{\pi^2}}
 \end{aligned}$$

Slika 46:



Slika 47:

```
#Furijeov red

def u(t):
    return 0 if t<0 else 1

def v(L):
    def vk(t):
        s=0
        for k in range(-L,L+1):
            s += 2*( 2*k-t+1) * ( u ( t - 2 * k ) - u ( t - 2 * k -1))
        return s
    return vk

dt = 0.005

data = np . arange ( 0 , 7 + dt , dt )
values = np . vectorize ( v (9) , otypes =[ float ]) ( data )
start_t=-4
d_t=0.01
end_t=4
t=np.arange(start_t, end_t, d_t)
plt.figure()
plt.plot(data,values)
plt.xlabel('t')
plt.ylabel('v(t)')
plt.ylim([0,3])
plt.title('Originalni signal')
plt.show()
```

Slika 48:

Lista Kodova