Univerzitet u Sarajevu

Elektrotehnički fakultet

Multimedijalni Sistemi

Laboratorijska vježba 7. - Digitalizacija Signala

Izrada laboratorijske vježbe vrši se u ovom *Jupyter Notebook*-u. Isti je potrebno konvertovati u PDF dokument i predati na Zamger.

Ime i prezime studenta, broj indeksa:

Amila Laković

Datum izrade izvještaja:

19.05.2021.

Zadatak 1.

Potrebno je implementirati funkciju `sinusoida(pocetak, kraj, frekvencija)` koja vrši izračunavanje funkcije `sin(x)` u intervalu (pocetak, kraj) sa korakom `1/frekvencija`. Ova funkcija kao rezultat treba vratiti sve vrijednosti x i y respektivno.

Nakon toga potrebno je implementirati funkciju `uzorkovanje(x, y, fnovo, fstaro)` koja formira nove vrijednosti x i y na osnovu postojećih. Frekvencija uzorkovanja jednaka je `fnovo`, što znači da je potrebno dodati uzorke x i y iz nizova nakon svakih `fstaro/fnovo` uzoraka.

Naprimjer, ukoliko su definisane sljedeće vrijednosti:

```
x = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0]
y = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
fstaro = 10
fnovo = 5
```

Potrebno je da rezultat funkcije budu sljedeće vrijednosti:

```
xnew = [0, 0.2, 0.4, 0.6, 0.8, 0]
ynew = [0, 2, 4, 6, 8, 0]
```

Kako je frekvencija uzorkovanja dva puta manja od originalne, to se uzima svaki drugi uzorak iz nizova x i y.

Detaljne informacije o uzorkovanju moguće je pronaći u Poglavlju 4 materijala za rad na predmetu, s početkom na str. 106.

Rješenje:

In [25]:

```
import numpy as np

def sinusoida(pocetak, kraj, frekvencija):
    #postavljanje pocetnih vrijednosti
    x = []
    y = []
```

```
    step = 1/frekvencija

    #prolazak kroz interval po objasnjenom stepu
    #postavljanje vrijednosti x i y
    for i in np.arange(pocetak, kraj, step):
        x.append(i)
        y.append(np.sin(i))

    return x, y

def uzorkovanje(x, y, fnovo, fstaro):
    #postavljanje pocetnih vrijednosti
    xnew = []
    ynew = []
    step = fstaro/fnovo

    #prolazak kroz interval po objasnjenom stepu
    #postavljanje novih vrijednosti x i y
    for i in np.arange(0, len(x), step):
        xnew.append(i)
        ynew.append(np.sin(i))

    return xnew, ynew
```

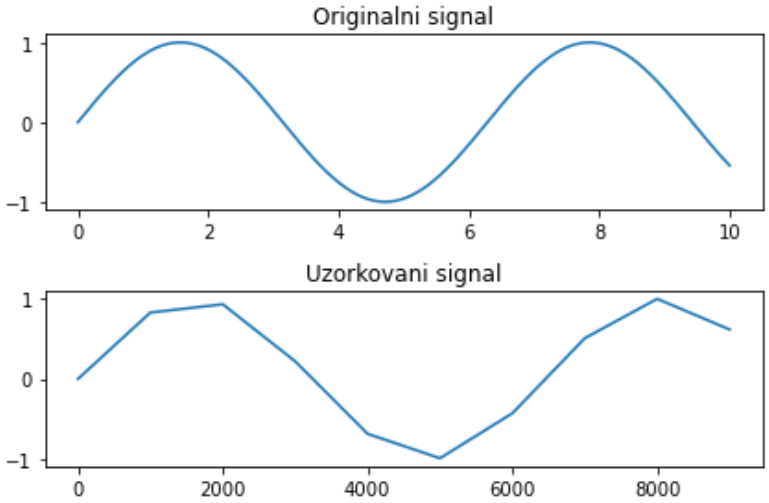
Nakon implementacije funkcije, potrebno je biti moguće izvršiti programski kod ispod tako da daje prikazani ispis. Osim toga, potrebno je dodati još primjera kako bi se implementacija adekvatno testirala.

In [26]:

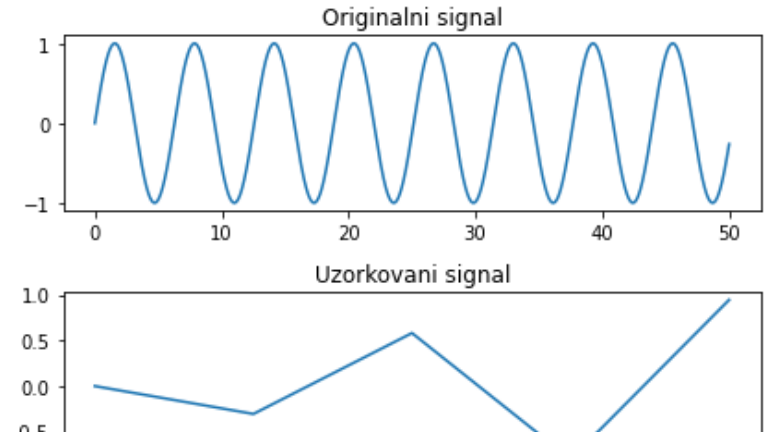
```
import matplotlib.pyplot as plt

[x, y] = sinusoida(0, 10, 1000)
[xnew, ynew] = uzorkovanje(x, y, 1, 1000)

print("Uzorkovanje - nije došlo do alijasinga:")

plt.figure(1)
plt.subplot(211)
plt.title('Originalni signal')
plt.plot(x, y)
plt.subplot(212)
plt.title('Uzorkovani signal')
plt.plot(xnew, ynew)
plt.tight_layout()
plt.show()

[x, y] = sinusoida(0, 50, 1000)
[xnew, ynew] = uzorkovanje(x, y, 0.1, 1000)

print("Uzorkovanje - došlo je do alijasinga:")

plt.figure(1)
plt.subplot(211)
plt.title('Originalni signal')
plt.plot(x, y)
plt.subplot(212)
plt.title('Uzorkovani signal')
plt.plot(xnew, ynew)
plt.tight_layout()
plt.show()

#DODATNI PRIMJERI

print("Dodatni primjeri: ")

#manji raspon, veci step
[x, y] = sinusoida(0, 20, 500)
#manji step, veci raspon
[xnew, ynew] = uzorkovanje(x, y, 0.5, 500)
```

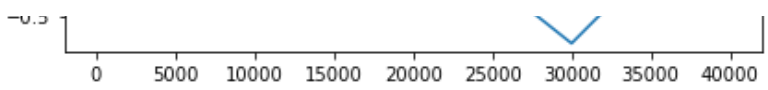
```
plt.figure(1)
plt.subplot(211)
plt.title('Originalni signal')
plt.plot(x, y)
plt.subplot(212)
plt.title('Uzorkovani signal')
plt.plot(xnew, ynew)
plt.tight_layout()
plt.show()

[x, y] = sinusoida(0, 100, 500)
[xnew, ynew] = uzorkovanje(x, y, 1, 500)

plt.figure(1)
plt.subplot(211)
plt.title('Originalni signal')
plt.plot(x, y)
plt.subplot(212)
plt.title('Uzorkovani signal')
plt.plot(xnew, ynew)
plt.tight_layout()
plt.show()

#manji step
[x, y] = sinusoida(0, 100, 50)
[xnew, ynew] = uzorkovanje(x, y, 1, 50)

plt.figure(1)
plt.subplot(211)
plt.title('Originalni signal')
plt.plot(x, y)
plt.subplot(212)
plt.title('Uzorkovani signal')
plt.plot(xnew, ynew)
plt.tight_layout()
plt.show()
```

Uzorkovanje - nije došlo do alijasinga:

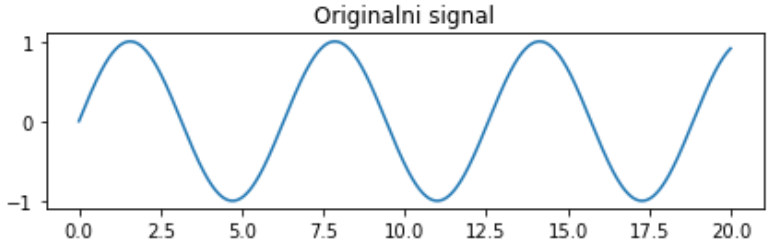


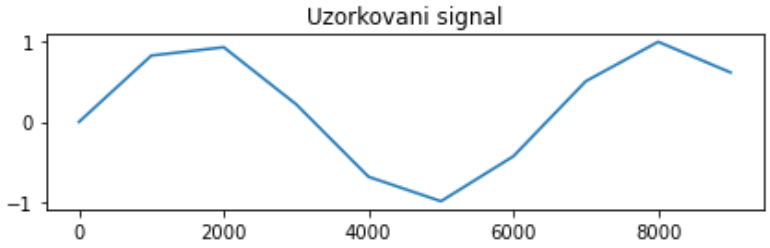
Uzorkovanje - došlo je do alijasinga:

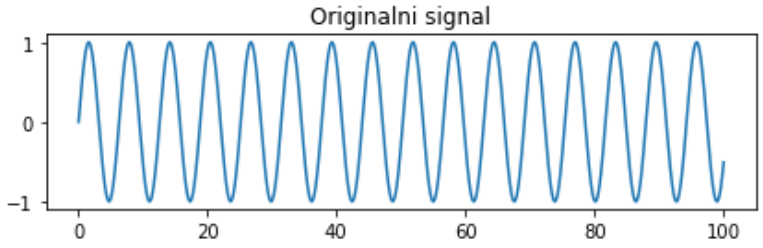


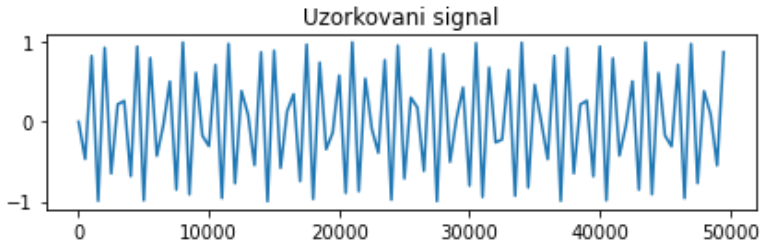


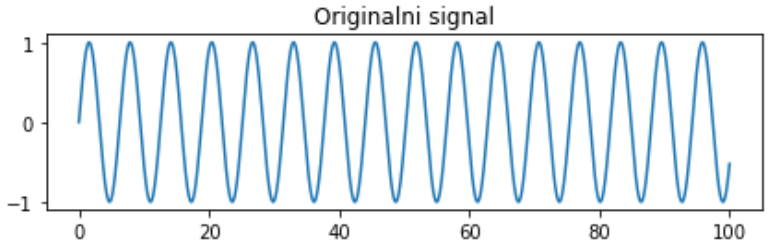
Dodatni primjeri:

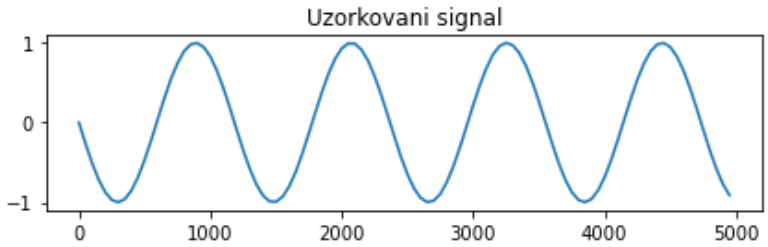












Zadatak 2.

Potrebno je implementirati funkciju `kvantizacija(y, broj_nivoa)` koja vrši kvantizaciju vrijednosti `y` proslijeđene kao njen parametar na broj kvantizacijskih nivoa proslijeđenih kao parametar `broj_nivoa`. Ova funkcija kao rezultat treba vratiti novu vrijednost y.

Izračunavanje vrijednosti svakog nivoa vrši se prema sljedećoj formuli:

$$r = \frac{\max(y) - \min(y)}{n - 1}$$

$$q_i = q_{i-1} + r, q_0 = \min(y)$$

pri čemu `r` predstavlja rezoluciju tj. korak kvantizacije, a `qi` predstavlja vrijednost kvantizacijskog nivoa na poziciji `i`, pri čemu je `n` broj kvantizacijskih nivoa.

Nakon određivanja svih kvantizacijskih nivoa, svakoj vrijednosti `y` potrebno je dodijeliti vrijednost onog kvantizacijskog nivoa najbližu stvarnoj vrijednosti y.

Naprimjer, ukoliko su definisane sljedeće vrijednosti:

```
y = [0, 0.2, 0.4, 0.6, 0.8, 1]
n = 3
```

Potrebno je da rezultat funkcije budu sljedeće vrijednosti:

```
y = [0, 0, 0.5, 0.5, 1, 1]
```

Jer je rezolucija jednaka 0.5 za 3 kvantizacijska nivoa između 0 i 1, a kvantizacijski nivoi su 0, 0.5 i 1.

Detaljne informacije o kvantizaciji moguće je pronaći u Poglavlju 4 materijala za rad na predmetu, s početkom na str. 106.

Rješenje:

In [27]:

```
import numpy as np

def sinusoida(pocetak, kraj, frekvencija):
    #postavljanje pocetnih vrijednosti
    x = []
    y = []
    step = 1/frekvencija

    #prolazak kroz interval po objasnjenom stepu
    #postavljanje vrijednosti x i y
    for i in np.arange(pocetak, kraj, step):
        x.append(i)
        y.append(np.sin(i))

    return x, y

#nadji min razliku u q od oredjenog y
def nadjimin(y, q, indeks):
    razlika = np.abs(y - q[0])
    for j in np.arange(1, len(q)):
        trenutna_razlika = np.abs(y - q[j])
        if(razlika > trenutna_razlika):
            razlika = trenutna_razlika
            indeks = j
    return indeks

def kvantizacija(y, broj_nivoa):
    #postavljanje pocetnih vrijednosti
    q = []
    q_indeks = 0
    ynew = []

    #postvaljanje q0 vrijednosti
    q0 = np.min(y)
    q.append(q0)

    #izracunavanje r po navedenoj formuli
    r = (max(y) - min(y))/(broj_nivoa - 1)
```

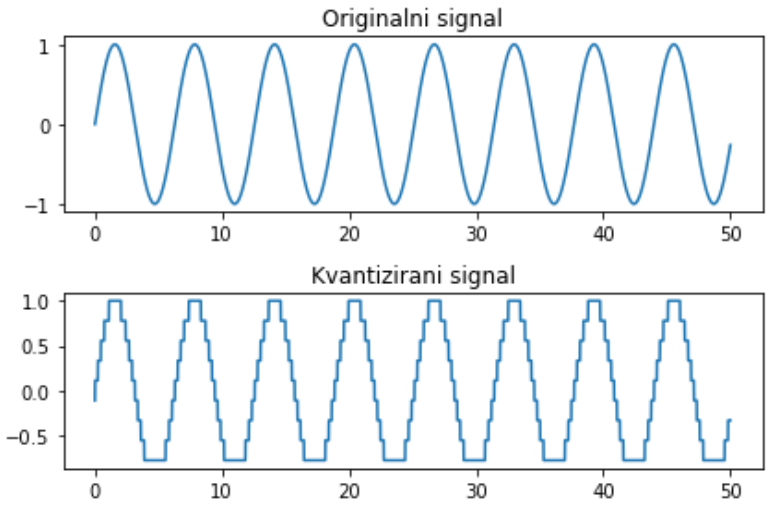
```
    #postavljenje ostalih q vrijednosti po navedenoj formuli
    for i in np.arange(1, broj_nivoa):
        vrijednost = q[i-1] + r
        q.append(vrijednost)

    #dodjeljivanje y najblize q
    for i in np.arange(0, len(y)):
        q_indeks = nadjimin(y[i], q, q_indeks)
        ynew.append(q[q_indeks])

    return ynew
```

Nakon implementacije funkcije, potrebno je biti moguće izvršiti programski kod ispod tako da daje prikazani ispis. Osim toga, potrebno je dodati još primjera kako bi se implementacija adekvatno testirala.

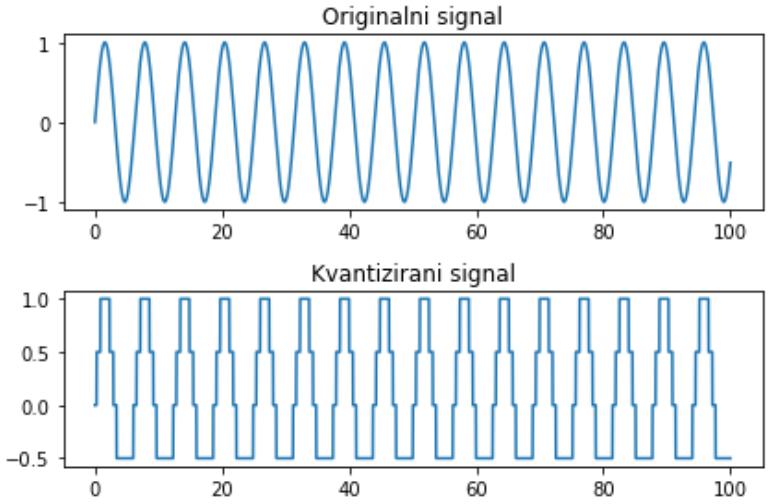
In [28]:

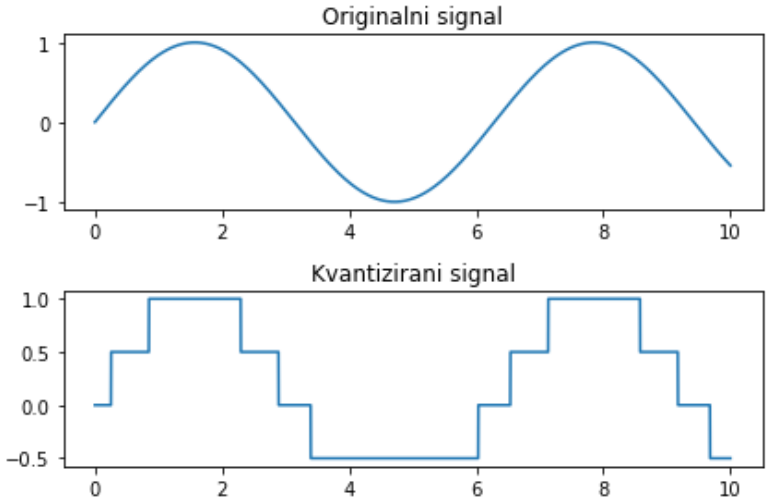
```
import matplotlib.pyplot as plt

[x, y] = sinusoida(0, 50, 1000)
ynew = kvantizacija(y, 10)

plt.figure(1)
plt.subplot(211)
plt.title('Originalni signal')
plt.plot(x, y)
plt.subplot(212)
plt.title('Kvantizirani signal')
plt.plot(x, ynew)
plt.tight_layout()
plt.show()

#DODATNI PRIMJERI

print("Dodatni primjeri: ")

#veci raspon, manji step
[x, y] = sinusoida(0, 100, 200)
#manji broj nivoa
ynew = kvantizacija(y, 5)

plt.figure(1)
plt.subplot(211)
plt.title('Originalni signal')
plt.plot(x, y)
plt.subplot(212)
plt.title('Kvantizirani signal')
plt.plot(x, ynew)
plt.tight_layout()
plt.show()

#manji raspon, veci step
[x, y] = sinusoida(0, 10, 2000)
ynew = kvantizacija(y, 5)

plt.figure(1)
plt.subplot(211)
plt.title('Originalni signal')
plt.plot(x, y)
plt.subplot(212)
plt.title('Kvantizirani signal')
plt.plot(x, ynew)
plt.tight_layout()
plt.show()

[x, y] = sinusoida(0, 20, 2000)
ynew = kvantizacija(y, 20)

plt.figure(1)
```

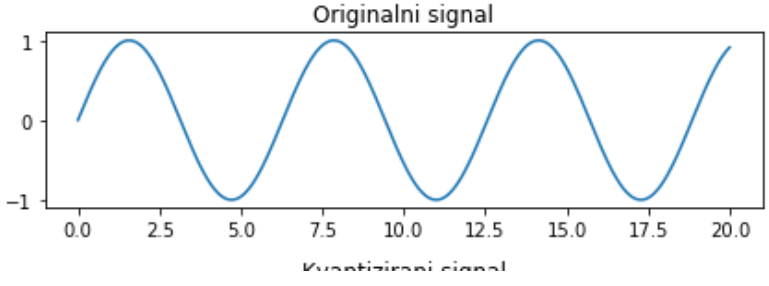
```
plt.subplot(211)
plt.title('Originalni signal')
plt.plot(x, y)
plt.subplot(212)
plt.title('Kvantizirani signal')
plt.plot(x, ynew)
plt.tight_layout()
plt.show()
```

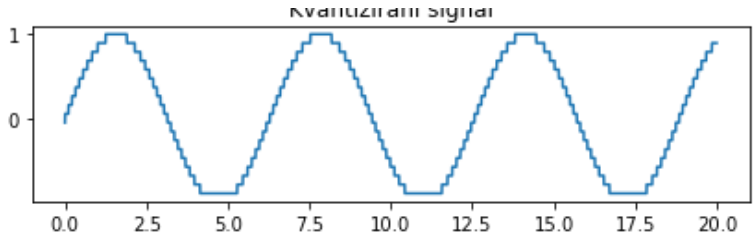


Dodatni primjeri:









In [ ]: