

# Laboratorijska vježba 6

## Fourierova transformacija - Uvod

Fourierova analiza proizlazi iz glavne ideje da svaku periodičnu funkciju možemo zapisati kao sumu (ne nužno konačnu) sinusa različitih amplituda, faza i frekvencija. Takva suma naziva se Fourierov red.

Danas su Fourierove teorije dalje razvijene; među njima je najvažnija diskretna Fourierova transformacija (DFT) i algoritam za brzo i efikasno izračunavanje DFT-a u vremenu - brza Fourierova transformacija (FFT), koja je temelj velikog dijela modernih multimedijских primjena (MP3, JPEG).

## Diskretna Fourierova transformacija

Diskretna Fourier-ova transformacija (DFT) konačne sekvence  $x[n]$  dužine  $N$  definirana je sljedećim izrazom:

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-\frac{j2\pi kn}{N}}, \text{ za } 0 \leq k \leq N$$

Inverzna diskretna Fourierova transformacija (DFT) se definiše kao:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X_N[k] e^{\frac{j2\pi kn}{N}}, \text{ za } 0 \leq n \leq N$$

---

### Zadatak 6.1

---

```
1 # -*- coding: utf-8 -*-
2 """
3 Odrediti DFT signala x(t)=5+2cos(2pit-90)+3cos(4pit)
4 """
5
6 import numpy as np
7 import matplotlib.pyplot as plt
8
9 t=np.linspace(0,10,200)
10
11 x=5+2*np.cos(2*np.pi*t-90)+3*np.cos(4*np.pi*t)
12
13 plt.plot(t,x)
14 plt.grid()
```

```

15 #Kontinualni signal u vremenskom domenu
16
17 #Frekvencija uzoraka Fs=4Hz -> t=kTs=k/T=k/4
18 #Pa imamo novi izgled signala
19
20 import scipy as sp
21 import matplotlib.pyplot as plt
22
23 k=sp.arange(0,4,1)
24 Fs=4 #frekvencija uzorkovanja
25 x=5+2*sp.cos(0.5*sp.pi*k-0.5*sp.pi)+3*sp.cos(sp.pi*k)
26
27 plt.figure(2)
28 plt.stem(k,x)
29 plt.grid()
30
31 #dalje ra unamo DFT
32
33 X=sp.fft(x)
34 print(X)
35 N=len(X)
36 F=sp.linspace(0,Fs,N+1)
37 F=F[:-1] #na vektor -1
38 plt.figure(3)
39 plt.stem(F[0:N],sp.absolute(X[0:N]))
40 plt.title('DFT na e sekvence za 4 uzorka')
41 plt.grid()

```

## Zadatak 6.2

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Dec 10 18:47:48 2020
4
5 @author: Minja
6 """
7
8 import numpy as np
9 import matplotlib.pyplot as plt
10
11 t=np.linspace(0,10,200)
12
13 x=5+2*np.cos(2*np.pi*t-90)+3*np.cos(4*np.pi*t)
14
15 plt.plot(t,x)
16 plt.grid()
17 #Kontinualni signal u vremenskom domenu
18
19 #Frekvencija uzoraka Fs=4Hz -> t=kTs=k/T=k/4
20 #Pa imamo novi izgled signala
21
22 import scipy as sp
23 import matplotlib.pyplot as plt
24
25 k=sp.arange(0,4,1)
26 Fs=4
27 x=5+2*sp.cos(0.5*sp.pi*k-0.5*sp.pi)+3*sp.cos(sp.pi*k)
28
29 plt.figure(2)
30 plt.stem(k,x)
31 plt.grid()
32
33 #dalje ra unamo DFT
34
35 X=sp.fft(x)
36 print(X)

```

```

37 F=sp.linspace(0,Fs,len(X)+1)
38 F=F[:-1]
39
40 N=len(X)
41
42 plt.figure(3)
43 plt.stem(F[0:N],sp.absolute(X[0:N]))
44 plt.title('DFT na e sekvence za 4 uzorka')
45 plt.grid()
46
47 #Inverzna FT
48 k=sp.arange(0,4,1)
49 x=sp.ifft(X)
50
51 plt.figure(4)
52 plt.stem(k,x)
53 plt.grid()

```

### Zadatak 6.3

```

1  # -*- coding: utf-8 -*-
2  """
3  Izra unati dft i nacrtati amplitudni spektar signala x[n]=n(u[n]-u[n-7])
4  """
5
6  import numpy as np
7  import matplotlib.pyplot as plt
8  import scipy as sp
9
10 t=np.arange(0,16,1) #defini emo vektor vremenskih trenutaka
11 x=t*(np.heaviside(t,1)-np.heaviside(t-7,1)) #na signal
12
13 plt.figure(1)
14 plt.stem(t,x)
15 plt.grid()
16
17 N=len(x) #broj uzoraka
18 X=sp.fft(x) #Izra unavamo DFT na eg signala
19 w=sp.linspace(0,sp.pi*2,N+1) #defini emo vektor na ih stvarnih vrijednosti
20                                     #Fourierova transformacija se ra una od 0-2pi
21 w=w[:-1]
22 plt.figure(2)
23 plt.stem(w,X) #crtanje spektra
24
25 plt.figure(3)
26 plt.stem(w,sp.absolute(X)) # crtanje amplitudnog spektra, uzimaju se pozitivne
    vrijednosti
27                                     #imamo kompleksan broj a amplitudni spektar je
28                                     #korijen sume kvadrata realne i kvadrata imaginarne
    vrijednosti

```

### Zadatak 6.4

```

1  # -*- coding: utf-8 -*-
2  """
3  Nacrtati spektar signala x[t]=sin(2*pi*0.125*t)
4  """
5
6  import numpy as np
7  import matplotlib.pyplot as plt
8  import scipy as sp
9
10
11 t=np.arange(16)

```

```

12 x=np.sin(2*np.pi*t*0.125)
13 plt.figure(1)
14 plt.stem(t,x,use_line_collection=True)
15
16 X=sp.fft.fft(x)
17
18 w=np.linspace(0,np.pi*2,len(x)+1)
19 w=w[:-1]
20 plt.figure(2)
21 plt.stem(w,np.absolute(X),use_line_collection=True)

```

### Zadatak 6.5

```

1 # -*- coding: utf-8 -*-
2 """
3 x(t) = sin(2 500t ) uz frekvenciju
4 fs = 8 kHz u N = 8000 uzoraka.
5 """
6 import numpy as np
7 import matplotlib.pyplot as plt
8 import scipy as sp
9
10 t=np.arange(0,16,1) #defini emo vektor vremenskih trenutaka
11 x = np.sin(2*np.pi*0.0625*t)
12 plt.figure(1)
13 plt.stem(t,x,use_line_collection=True)
14
15 X=sp.fft.fft(x)
16 A=np.absolute(X)
17 N=len(X)
18
19 w=np.linspace(0,np.pi*2,N+1)
20 w=w[:-1]
21
22
23 plt.figure(2)
24 plt.stem(w,A,use_line_collection=True)

```

### Zadatak 6.6

```

1 # -*- coding: utf-8 -*-
2 """
3 Odrediti DFT x[n]=2delta[n]-3delta[n-2]+delta[n-4]-4delta[n-6]
4 """
5
6 import numpy as np
7 import matplotlib.pyplot as plt
8 import scipy as sp
9
10 t=np.arange(8)
11 x1=2*(np.heaviside(t,1)-np.heaviside(t-0.01,1))
12 x2=3*(np.heaviside(t-2,1)-np.heaviside(t-2.01,1))
13 x3=(np.heaviside(t-4,1)-np.heaviside(t-4.01,1))
14 x4=4*(np.heaviside(t-6,1)-np.heaviside(t-6.01,1))
15
16 x=x1-x2+x3-x4
17
18 plt.figure(1)
19 plt.stem(t,x)
20 plt.grid()
21
22 X=sp.fft.fft(x)
23
24

```

```
25 N=len(X)
26 F=sp.linspace(0,sp.pi*2,N+1)
27 F=F[:-1]
28
29 plt.figure(2)
30 plt.stem(F,X)
31
32 plt.figure(3)
33 plt.stem(F,sp.absolute(X))
34 plt.title('DFT na e sekvence')
35 plt.grid()
36
37 #Inverzna FT
38 k=sp.arange(8)
39 x=sp.ifft(X)
40
41 plt.figure(4)
42 plt.stem(k,x)
43 plt.grid()
```