Laboratorijska vježba 6

Fourierova transformacija - Uvod

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

Danas su Fourierove teorije dalje razvijene; medu 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 multimedijskih primjena (MP3, JPEG).

Diskretna Fourierova transformacija

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

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

Inverzna diskretna Fourierova transformacija (DFT) se definiše kao:

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

Zadatak 6.1

```
# -*- coding: utf-8 -*-
"""
Odrediti DFT signala x(t)=5+2cos(2pit-90)+3cos(4pit)
"""

import numpy as np
import matplotlib.pyplot as plt

t=np.linspace(0,10,200)

x=5+2*np.cos(2*np.pi*t-90)+3*np.cos(4*np.pi*t)

plt.plot(t,x)
plt.grid()
```

```
#Kontinualni signal u vremenskom domenu

#Frekvencija uzoraka Fs=4Hz -> t=kTs=k/T=k/4
#Pa imamo novi izgled signala

import scipy as sp
import matplotlib.pyplot as plt

k=sp.arange(0,4,1)
Fs=4 #frekvencija uzorkovanja
x=5+2*sp.cos(0.5*sp.pi*k-0.5*sp.pi)+3*sp.cos(sp.pi*k)

plt.figure(2)
plt.stem(k,x)
plt.grid()

#dalje ra unamo DFT

X=sp.fft(x)
print(X)
N=len(X)
F=sp.linspace(0,Fs,N+1)
F=F[:-1] #na  vektor -1
plt.figure(3)
plt.stem(F[0:N],sp.absolute(X[0:N]))
plt.title('DFT na e sekvence za 4 uzorka')
plt.grid()
```

Zadatak 6.2

```
# -*- coding: utf-8 -*-
"""
Created on Thu Dec 10 18:47:48 2020

@author: Minja
"""

import numpy as np
import matplotlib.pyplot as plt

t=np.linspace(0,10,200)

x=5+2*np.cos(2*np.pi*t-90)+3*np.cos(4*np.pi*t)

plt.plot(t,x)
plt.grid()
#Kontinualni signal u vremenskom domenu

#Frekvencija uzoraka Fs=4Hz -> t=kTs=k/T=k/4
#Pa imamo novi izgled signala

import scipy as sp
import matplotlib.pyplot as plt

k=sp.arange(0,4,1)
Fs=4
x=5+2*sp.cos(0.5*sp.pi*k-0.5*sp.pi)+3*sp.cos(sp.pi*k)

plt.figure(2)
plt.stem(k,x)
plt.grid()

#dalje ra unamo DFT

X=sp.fft(x)
print(X)
```

```
F=sp.linspace(0,Fs,len(X)+1)
F=F[:-1]

N=len(X)

plt.figure(3)
plt.stem(F[0:N],sp.absolute(X[0:N]))
plt.title('DFT na e sekvence za 4 uzorka')
plt.grid()

#Inverzna FT
k=sp.arange(0,4,1)
x=sp.ifft(X)

plt.figure(4)
plt.stem(k,x)
plt.grid()
```

Zadatak 6.3

```
# -*- coding: utf-8 -*-
"""
Izra unati dft i nacrtati amplitudni spektar signala  x[n]=n(u[n]-u[n-7])
"""

import numpy as np
import matplotlib.pyplot as plt
import scipy as sp

t=np.arange(0,16,1) #defini emo vektor vremenskih trenutaka
x=t*(np.heaviside(t,1)-np.heaviside(t-7,1)) #na   signal

plt.figure(1)
plt.stem(t,x)
plt.grid()

N=len(x) #broj uzoraka
X=sp.fft(x) #Izra unavamo DFT na eg signala
w=sp.linspace(0,sp.pi*2,N+1) #defini emo vektor na ih stvarnih vrijednosti
                                #Fourierova transformacija se ra una od 0-2pi
w=w[:-1]
plt.figure(2)
plt.stem(w,X) #crtanje spektra

plt.figure(3)
plt.stem(w,sp.absolute(X)) # crtanje amplitudnog spektra, uzimaju se pozitivne vrijednosti
                           #imamo kompleksan broj a amplitudni spektar je
                           #korijen sume kvadrata realne i kvadrata imaginarne vrijednosti
```

Zadatak 6.4

```
# -*- coding: utf-8 -*-
"""
Nacrtati spektar signala x[t]=sin(2*pi*0.125*t)
"""

import numpy as np
import matplotlib.pyplot as plt
import scipy as sp


t=np.arange(16)
```

```
x=np.sin(2*np.pi*t*0.125)
plt.figure(1)
plt.stem(t,x,use_line_collection=True)

X=sp.fft.fft(x)

w=np.linspace(0,np.pi*2,len(x)+1)
w=w[:-1]
plt.figure(2)
plt.stem(w,np.absolute(X),use_line_collection=True)
```

Zadatak 6.5

```
# -*- coding: utf-8 -*-
"""
x(t) = sin(2 500t ) uz frekvenciju
fs = 8 kHz u N = 8000 uzoraka.
"""
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp

t=np.arange(0,16,1) #defini emo vektor vremenskih trenutaka
x = np.sin(2*np.pi*0.0625*t)
plt.figure(1)
plt.stem(t,x,use_line_collection=True)

X=sp.fft.fft(x)
A=np.absolute(X)
N=len(X)

w=np.linspace(0,np.pi*2,N+1)
w=w[:-1]


plt.figure(2)
plt.stem(w,A,use_line_collection=True)
```

Zadatak 6.6

```
# -*- coding: utf-8 -*-
"""
Odrediti DFT x[n]=2delta[n]-3delta[n-2]+delta[n-4]-4delta[n-6]
"""

import numpy as np
import matplotlib.pyplot as plt
import scipy as sp

t=np.arange(8)
x1=2*(np.heaviside(t,1)-np.heaviside(t-0.01,1))
x2=3*(np.heaviside(t-2,1)-np.heaviside(t-2.01,1))
x3=(np.heaviside(t-4,1)-np.heaviside(t-4.01,1))
x4=4*(np.heaviside(t-6,1)-np.heaviside(t-6.01,1))

x=x1-x2+x3-x4

plt.figure(1)
plt.stem(t,x)
plt.grid()

X=sp.fft.fft(x)
```

```
N=len(X)
F=sp.linspace(0,sp.pi*2,N+1)
F=F[:-1]

plt.figure(2)
plt.stem(F,X)

plt.figure(3)
plt.stem(F,sp.absolute(X))
plt.title('DFT na e sekvence')
plt.grid()

#Inverzna FT
k=sp.arange(8)
x=sp.ifft(X)

plt.figure(4)
plt.stem(k,x)
plt.grid()
```