Domaći zadatak iz predmeta 13E053DOS Digitalna obrada signala

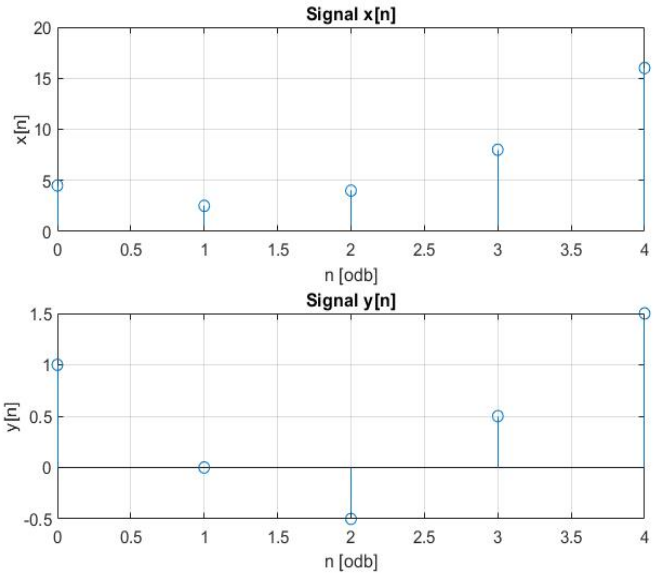
Aleksa Janjić 2019/0021

Decembar 2021.

$$P = 5$$
$$Q = 3$$
$$R = 0$$
$$S = 0$$

1 Zadatak 1

a) Na početku zadatka, kroz jednu *for*-petlju su definisane vrednosti vektora **x** i **y**, a potom su pomoću funkcije ***stem()*** grafički prikazani signali **x** i **y**:



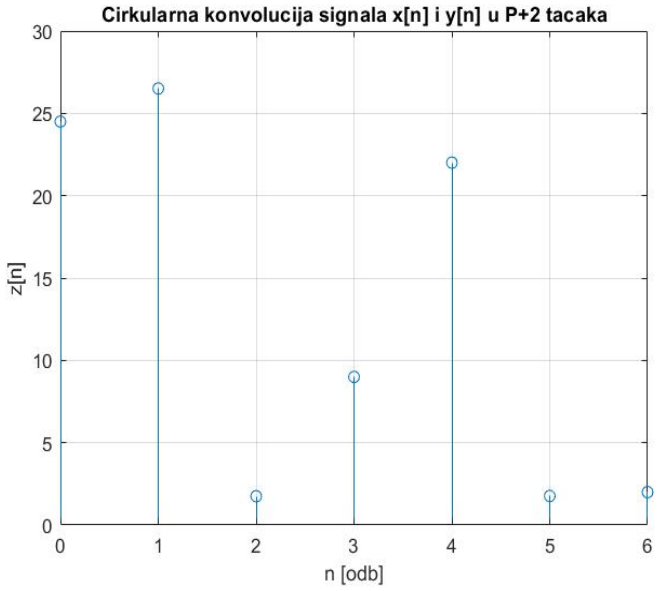
Slika 1: Vremenski oblici signala $x[n]$ i $y[n]$

Pošto se traži ciklična konvolucija u $P+2$ tačaka (u mom slučaju, P ima vrednost 5), neophodno je proširiti početne vektore **x** i **y**, odnosno, dopuniti ih nulama, što je u kodu i učinjeno, samo što je sačuvano u drugim vektorima **x1** i **y1**.

Ciklična (ili cirkularna) konvolucija signala x_1 i x_2 se računa po definiciji:

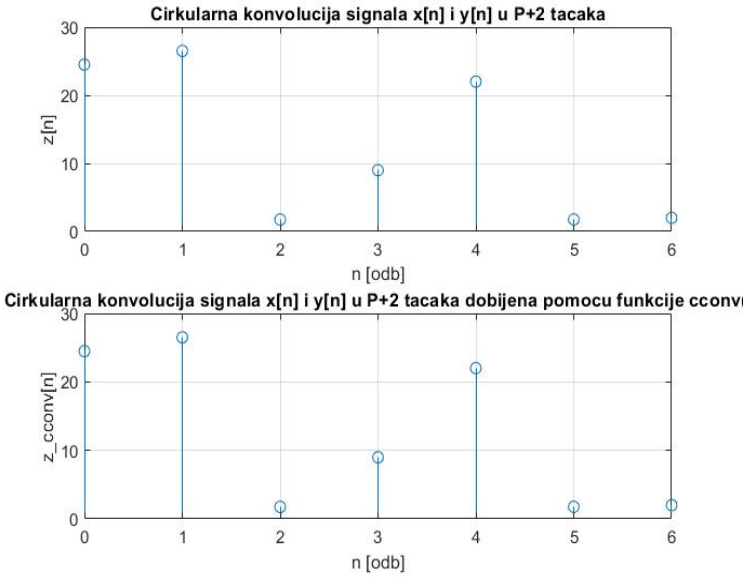
$$z[k] = \sum_{m=0}^{N-1} x[m]y\left[\langle k-m\rangle_N\right]$$

Zatim na način koji je urađen na vežbama, odredili smo kroz jednu *for*-petlju vrednost ciklične konvolucije u $N = P + 2$ (=7) tačaka i vrednosti sačuvali u vektoru **z**. Takođe, provere radi, korišćenjem ugrađene funkcije ***cconv()*** smo odredili vrednosti ciklične konvolucije u $P + 2$ tačaka koje su saglasne sa našim metodom i taj rezultat smestili u vektor **z_cconv**.



Slika 2: Cirkularna (ciklična) konvolucija signala $x[n]$ i $y[n]$ u $P + 2 = 7$ tačaka

Kao dodatak ovome, sledi grafik koji ilustruje validnost dobijenih rezultata na implementirani način u odnosu na dobijene vrednosti korišćenjem Matlab-ove funkcije ***cconv()***.



Slika 3: Cirkularna (ciklična) konvolucija signala $x[n]$ i $y[n]$ u $P + 2 = 7$ tačaka i pomoću funkcije ***cconv()***

b) Sada je na redu linearna konvolucija. Linearna konvolucija se računa po definiciji:

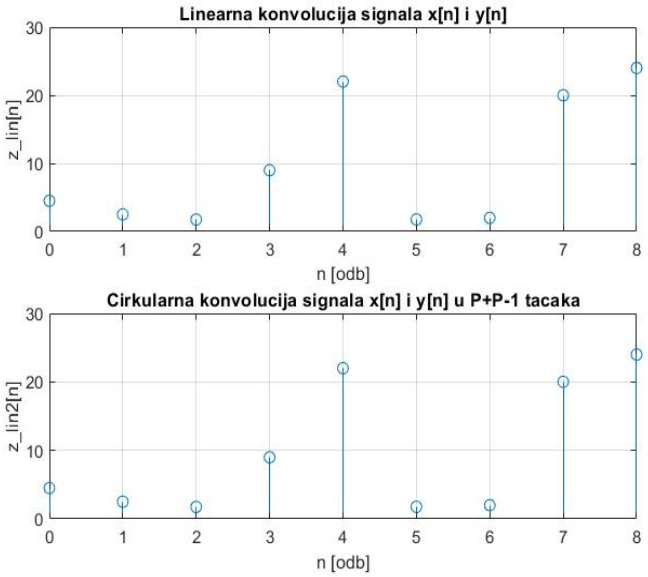
$$z[n] = x[n] * y[n] = y[n] * x[n] = \sum_{k=-\infty}^{\infty} x[k]y[n-k].$$

U opštem slučaju, ovu beskonačnu sumu je nemoguće računarski izračunati, međutim, znajući da su naši signali ograničenih dužina N_1 i N_2 respektivno, kao i poznavajući činjenicu da je linearna konvolucija dva ograničena signala dužina N_1 i N_2 takođe ograničenog trajanja $N_1 + N_2 - 1$, tada možemo ovu beskonačnu sumu predstaviti pomoću dve ugnježdene sume:

$$z[n] = \sum_{k=-\infty}^{\infty} x[k]y[n-k] = \sum_{n=0}^{(N_1+N_2-1)-1} \sum_{i=0}^{N_1-1} x[i]y[n-i]$$

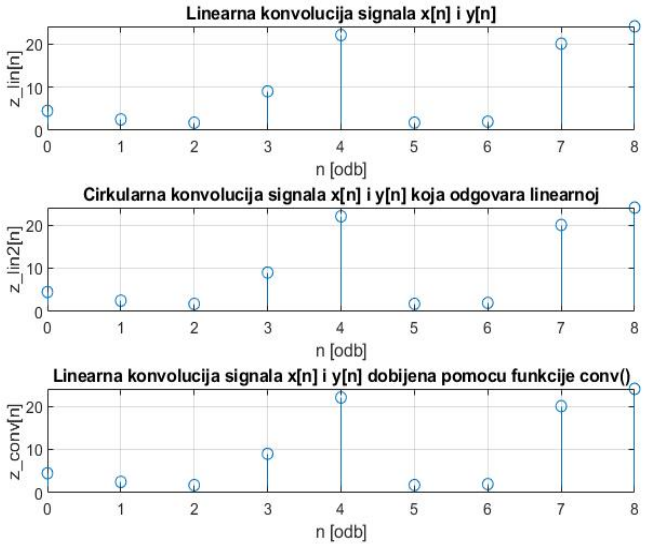
Sada možemo računarski implementirati linearnu konvoluciju signala **x** i **y**. Primetimo da iterator i se kreće kroz vektor signala **x**, stoga taj vektor ne moramo produžavati u vremenu, odnosno, popunjavati nulama, dok iterator j nadmašuje dužinu vektora **y**, stoga taj signal-vektor treba dopuniti nulama tako da je njegova nova dužina $N_1 + N_2 - 1$, što je učinjeno i sačuvano u vektoru **y2**. Potom je implementirana dvostruka suma iz poslednjeg izraza kroz dve *for*-petlje i rezultat smešten u vektor **z_lin**, a zatim iskoristili ugrađenu funkciju ***conv()*** da bismo proverili da li su saglasne dobijene vrednosti. Važna napomena je da, pošto numeracija vektora u Matlab-u počinje od jedinice, a ne od nule (kako odgovara u teoriji signala i samom izrazu linearne konvolucije), u kodu potrebno je granice sumiranja pomeriti za jedan.

Najzad je potrebno odrediti cirkularnu konvoluciju u onoliko tačaka tako da je ona jednaka linearnoj konvoluciji. Po definiciji, potreban broj tačaka je $N = N_1 + N_2 - 1$, samim tim, naše vektore **x** i **y** treba dopuniti nulama toliko da im dužina bude $N = N_1 + N_2 - 1$. U prethodnom delu zadatka, to je već i urađeno i sačuvano u vektoru **y2**, dok vektor **x** nije bilo potrebno dopunjavati nulama. Sada je to neophodno, pa je urađeno i sačuvano u vektoru **x2**. Dalje je implementacija potpuno analogna kao prvom delu zadatka.



Slika 4: Grafici linearne konvolucije signala $x[n]$ i $y[n]$ i cirkularne konvolucije istih signala u $P + P - 1 = 9$ tačaka

Dodatno, prilaže se grafik koji ilustruje tri različita načina dobijanja linearne konvolucije signala $x[n]$ i $y[n]$.



Slika 5: Grafici linearne konvolucije signala $x[n]$ i $y[n]$ i cirkularne konvolucije istih signala u $P + P - 1 = 9$ tačaka, kao i linearne konvolucije dobijene funkcijom ***conv()***

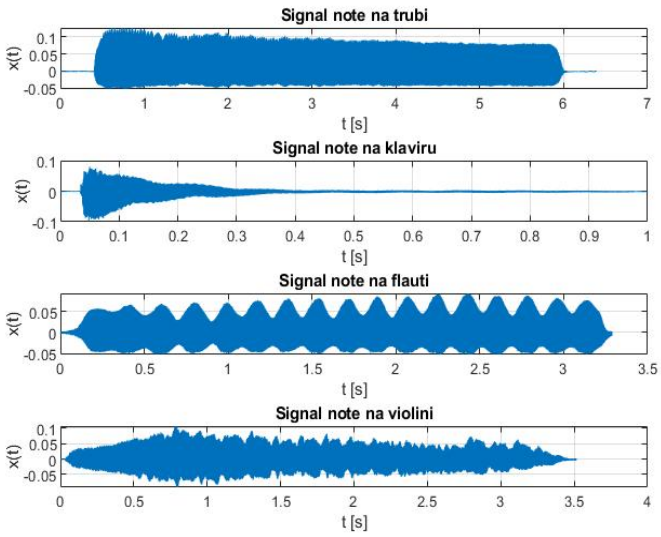
```
clear all
close all
clc

P = mod(0021,4) + 4;
Q = mod(0+0+2+1,4);
R = mod(0021+2019,3);
S = mod(0021,3);

%% odredjivanje diskretnih signala x i y

x = zeros(1,P);
y = zeros(1,P);

for i=0:(P-1)
    if (i < floor(P/2))
        x(i+1) = sin(i) + 2*cos(2*i) + P/2;
        y(i+1) = (-1)^i + mod(i,2);
    else
        x(i+1) = 2^i;
        y(i+1) = i - P/2;
    end
end

%% iscrtavanje diskretnih signala x i y u P tacaka

n = 0:(P-1);
figure(1)
subplot(2,1,1);
stem(n,x);
xlabel('n [odb]'); ylabel('x[n]'); title('Signal x[n]'); grid on;
subplot(2,1,2);
```

```
33 stem(n,y);
34 xlabel('n [odb]'); ylabel('y[n]'); title('Signal y[n]'); grid on;
35
36 %% ciklicna konvolucija u P+2 (5+2=7) tacaka
37
38 x1 = [x zeros(1,(P+2)-length(x))]; %dopunjavanje nulama
39 y1 = [y zeros(1,(P+2)-length(y))];
40
41 z = zeros(1,(P+2));
42
43 for k = 0:((P+2)-1)
44     z(k+1) = x1*transpose(y1(mod(k:-1:k-((P+2)-1),(P+2))+1));
45 end
46
47 z_cconv = cconv(x1,y1,(P+2));
48
49 n1 = 0:(P+1);
50 figure(2)
51 stem(n1,z);
52 xlabel('n [odb]'); ylabel('z[n]'); title...
53     ('Cirkularna konvolucija signala x[n] i y[n] u P+2 tacaka'); grid on;
54
55 %% linearna konvolucija
56 N = length(x)+length(y)-1;
57
58 y2 = [y zeros(1,N-length(y))];
59
60 z_lin = zeros(1,N);
61
62 for i=1:N
63     for j=1:length(x)
64         if(i-j+1>0)
65             z_lin(i)=z_lin(i)+x(j)*y2(i-j+1);
66         end
67     end
68 end
69
70 z_lin_conv = conv(y,x);
71
72 n_lin = 0:(N-1);
73 figure(3)
74 subplot(2,1,1)
75 stem(n_lin,z_lin);
76 xlabel('n [odb]'); ylabel('z\_lin[n]'); title...
77     ('Linearna konvolucija signala x[n] i y[n]'); grid on;
78
79 %% cirkularna konvolucija koja odgovara linearnoj
80
81 x2 = [x zeros(1,N-length(x))];
82
83 z_lin2 = zeros(1,N);
84
85 for k = 0:(N-1)
86 z_lin2(k+1) = sum(x2.*y2(mod(k:-1:k-(N-1) ,N)+1));
87 end
88
89 z_lin_cconv = cconv(x2, y2, N);
90
```

```
figure(3)
subplot(2,1,2)
stem(n_lin,z_lin2);
xlabel('n [odb]'); ylabel('z\_lin2[n]'); title...
    ('Cirkularna konvolucija signala x[n] i y[n] u P+P-1 tacaka'); grid on;

%% provera
figure(4)
subplot(2,1,1)
stem(n1,z);
xlabel('n [odb]'); ylabel('z[n]'); title...
('Cirkularna konvolucija signala x[n] i y[n] u P+2 tacaka'); grid on;
subplot(2,1,2)
stem(n1,z_cconv);
xlabel('n [odb]'); ylabel('z\_cconv[n]'); title...
('Cirkularna konvolucija signala x[n] i y[n] u P+2 tacaka dobijena pomocu funkcije cco

figure(5)
subplot(3,1,1)
stem(n_lin,z_lin);
xlabel('n [odb]'); ylabel('z\_lin[n]'); title...
('Linearna konvolucija signala x[n] i y[n]'); grid on;
subplot(3,1,2)
stem(n_lin,z_lin2);
xlabel('n [odb]'); ylabel('z\_lin2[n]'); title...
('Cirkularna konvolucija signala x[n] i y[n] koja odgovara linearnoj'); grid on;
subplot(3,1,3)
stem(n_lin,z_lin_cconv);
xlabel('n [odb]'); ylabel('z\_conv[n]'); title...
('Linearna konvolucija signala x[n] i y[n] dobijena pomocu funkcije conv()'); grid on;
```

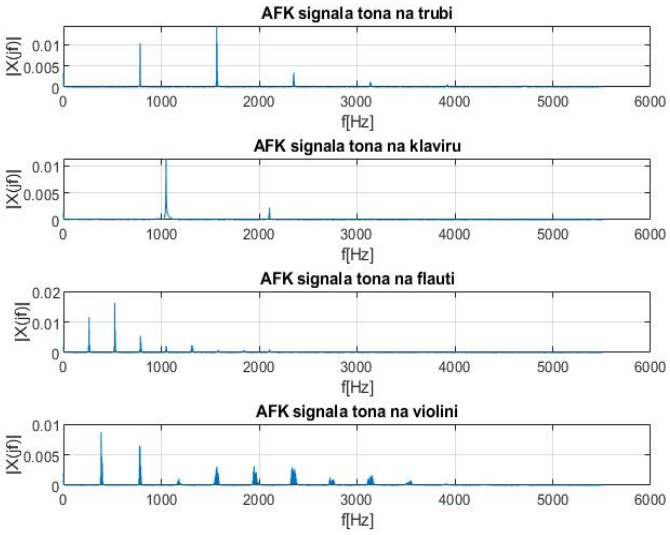
2 Zadatak 2

a) Za $Q = 3$, vremenski oblici sva četiri signala su dati u nastavku.

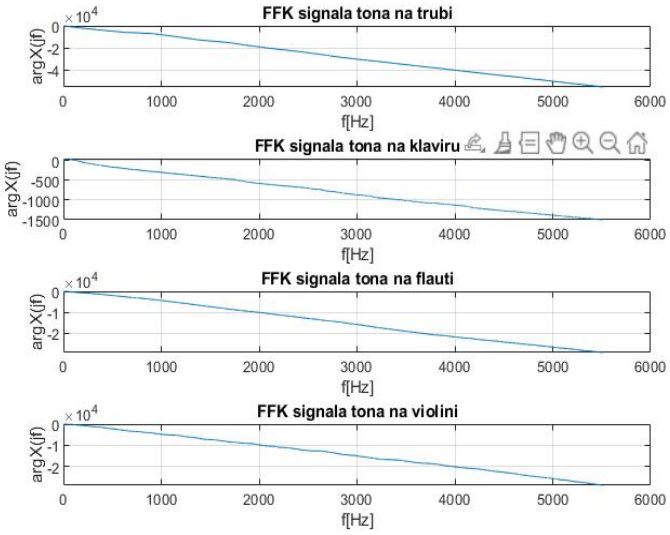


Slika 6: Vremenski oblici signala tona na trubi, klaviru, flauti i violini respektivno

b) Amplitudske frekvencijske karakteristike i fazne frekvencijske karakteristike sva četiri signala su dati u nastavku.



Slika 7: Amplitudske frekvencijske karakteristike tona na trubi, klaviru, flauti i violini respektivno



Slika 8: Fazne frekvencijske karakteristike tona na trubi, klaviru, flauti i violini respektivno

c) U ovom delu zadatka traže se frekvencije prvih pikova za svaki ton, a zatim da se uporede te dobijene vrednosti frekvencija sa frekvencijama tonova iz tabele date u zadatku.

Ovaj deo zadatka je odrađen tako što se iz *main* programa zadatka **zad2.m** četiri puta pozivala funkcija ***pikovi()***, čiji parametri su **X** (što je vektor vrednosti amplitudske karakteristike signala), **Fs** (što je frekvencija odabiranja) i **N** (što je odgovarajići stepen dvojke koji označava broj tačaka u kojima se radi FFT), a povratna vrednost te funkcije je vektor od dva elementa, od kojih je prvi element frekvencija prvog pika u spektru, a drugi element je prosečno rastojanje između susednih pikova u spektru **X**.

Sada kada smo za sva četiri signala tona (na instrumentima truba, klavir, flauta i violina) pozvali funkciju ***pikovi()***, za svaki signal smo uveli nove pomoćne promenljive koje služe zbog kasnijeg tabelarnog pregleda. Frekvencije prvih pikova smo zaokružili na celobrojne vrednosti korišćenjem funkcije ***round()***.

1×4 table

truba_freq_first_pick	klavir_freq_first_pick	flauta_freq_first_pick	violina_freq_first_pick
785	1050	263	386

Slika 9: Zaokružene vrednosti frekvencija prvih pikova za sva četiri instrumenta

Frekvencija trube od 785 Hz je najbliža vrednosti od 784 Hz, što odgovara noti G5. Frekvencija klavira od 1050 Hz je najbliža vrednosti od 1047 Hz, što odgovara noti C6. Frekvencija flaute od 263 Hz je najbliža vrednosti od 261 Hz, što odgovara noti C4. Frekvencija violine od 386 Hz je najbliža vrednosti od 392 Hz, što odgovara noti G4. Uočavamo da dobijene vrednosti frekvencija prvih pikova imaju malo odstupanje od konkretnih vrednosti frekvencija nota (najveće odstupanje je kod violine i iznosi 6 Hz).

d) U nastavku su tabelarno prikazana i uprosečena rastojanja između susednih pikova u spektrima sva četiri signala.

1×4 table

truba_average_freq	klavir_average_freq	flauta_average_freq	violina_average_freq
784.88	1057.8	262.96	391.48

Slika 10: Uprosečena rastojanja susednih pikova za sva četiri instrumenta

```
clear all;
close all;
clc;

%% prikazivanje u vremenskom domenu

[x1, Fs1] = audioread('truba_4.wav');
t1 = 0:(1/Fs1):((length(x1)-1)/Fs1);
figure(1)
subplot(4,1,1);
plot(t1,x1);
xlabel('t [s]'); ylabel('x(t)'); title('Signal note na trubi'); grid on;

[x2, Fs2] = audioread('klavir_4.wav');
t2 = 0:(1/Fs2):((length(x2)-1)/Fs2);
figure(1)
subplot(4,1,2);
plot(t2,x2);
xlabel('t [s]'); ylabel('x(t)'); title('Signal note na klaviru'); grid on;

[x3, Fs3] = audioread('flauta_4.wav');
t3 = 0:(1/Fs3):((length(x3)-1)/Fs3);
figure(1)
subplot(4,1,3);
plot(t3,x3);
xlabel('t [s]'); ylabel('x(t)'); title('Signal note na flauti'); grid on;

[x4, Fs4] = audioread('violina_4.wav');
t4 = 0:(1/Fs4):((length(x4)-1)/Fs4);
figure(1)
subplot(4,1,4);
plot(t4,x4);
xlabel('t [s]'); ylabel('x(t)'); title('Signal note na violini'); grid on;

%% amplitudske i fazne karakteristike

N1 = 2^nextpow2(length(x1));
X = fft(x1,N1)/length(x1);
f1 = 0:(Fs1/N1):(Fs1/2);
X1 = abs(X(1:(N1/2)+1));
X1(2:(N1/2)+1) = 2*X1(2:(N1/2)+1);
Xphase1 =unwrap(angle(X(1:(N1/2)+1)));
figure(2)
subplot(4,1,1)
plot(f1,X1);
title('AFK signala tona na trubi');
xlabel('f[Hz]'); ylabel('|X(jf)|'); grid on;
figure(3)
subplot(4,1,1);
plot(f1,Xphase1);
title('FFK signala tona na trubi');
xlabel('f[Hz]'); ylabel('arg{X(jf)}'); grid on;

N2 = 2^nextpow2(length(x2));
X = fft(x2,N2)/length(x2);
f2 = 0:(Fs2/N2):(Fs2/2);
X2 = abs(X(1:(N2/2)+1));
X2(2:(N2/2)+1) = 2*X2(2:(N2/2)+1);
```

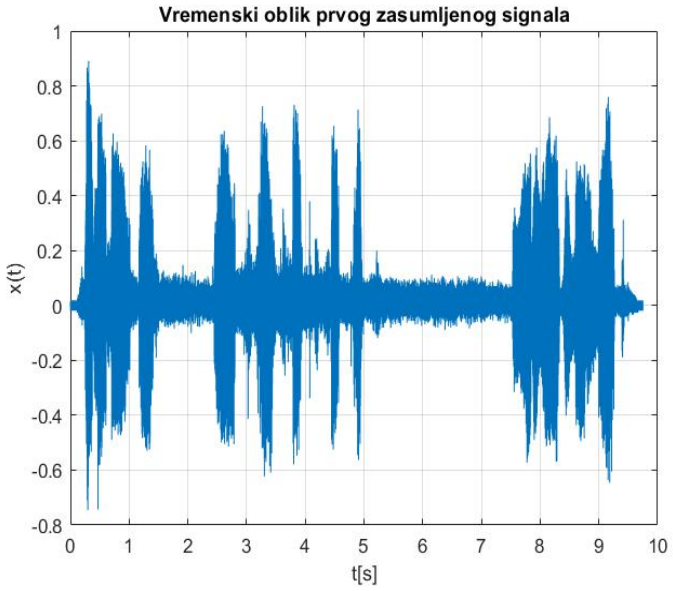
```
59  Xphase2 =unwrap(angle(X(1:(N2/2)+1)));
60  figure(2)
61  subplot(4,1,2)
62  plot(f2,X2);
63  title('AFK signala tona na klaviru');
64  xlabel('f[Hz]'); ylabel('|X(jf)|'); grid on;
65  figure(3)
66  subplot(4,1,2);
67  plot(f2,Xphase2);
68  title('FFK signala tona na klaviru');
69  xlabel('f[Hz]'); ylabel('arg{X(jf)}'); grid on;
70
71  N3 = 2^nextpow2(length(x3));
72  X = fft(x3,N3)/length(x3);
73  f3 = 0:(Fs3/N3):(Fs3/2);
74  X3 = abs(X(1:(N3/2)+1));
75  X3(2:(N3/2)+1) = 2*X3(2:(N3/2)+1); % X1(f) = X2(-f) + X2(f)
76  Xphase3 =unwrap(angle(X(1:(N3/2)+1)));
77  figure(2)
78  subplot(4,1,3)
79  plot(f3,X3);
80  title('AFK signala tona na flauti');
81  xlabel('f[Hz]'); ylabel('|X(jf)|'); grid on;
82  figure(3)
83  subplot(4,1,3);
84  plot(f3,Xphase3);
85  title('FFK signala tona na flauti');
86  xlabel('f[Hz]'); ylabel('arg{X(jf)}'); grid on;
87
88  N4 = 2^nextpow2(length(x4));
89  X = fft(x4,N4)/length(x4);
90  f4 = 0:(Fs4/N4):(Fs4/2);
91  X4 = abs(X(1:(N4/2)+1));
92  X4(2:(N4/2)+1) = 2*X4(2:(N4/2)+1);
93  Xphase4 =unwrap(angle(X(1:(N4/2)+1)));
94  figure(2)
95  subplot(4,1,4)
96  plot(f4,X4);
97  title('AFK signala tona na violini');
98  xlabel('f[Hz]'); ylabel('|X(jf)|'); grid on;
99  figure(3)
100 subplot(4,1,4);
101 plot(f4,Xphase4);
102 title('FFK signala tona na violini');
103 xlabel('f[Hz]'); ylabel('arg{X(jf)}'); grid on;
104
105 %% odredjivanje frekvencija prvih pikova
106
107 pom1 = pikovi(X1,Fs1,N1);
108 truba_freq_first_pick = round(pom1(1));
109 truba_average_freq = pom1(2);
110
111 pom2 = pikovi(X2,Fs2,N2);
112 klavir_freq_first_pick = round(pom2(1));
113 klavir_average_freq = pom2(2);
114
115 pom3 = pikovi(X3,Fs3,N3);
116 flauta_freq_first_pick = round(pom3(1));
```

```
flauta_average_freq = pom3(2);

pom4 = pikovi(X4,Fs4,N4);
violina_freq_first_pick = round(pom4(1));
violina_average_freq = pom4(2);

table(truba_freq_first_pick ,klavir_freq_first_pick ,...
    flauta_freq_first_pick, violina_freq_first_pick)

table(truba_average_freq,klavir_average_freq ,...
    flauta_average_freq, violina_average_freq)
```

```
function f = pikovi(X,Fs,N)

[all_peaks, all_locations] = findpeaks(X,'MinPeakDistance',200);
peaks = [];
locations = [];
for i=1:length(all_peaks)
    if (all_peaks(i) > 0.03*(max(all_peaks))) %kriterijum biranja pikova
        peaks = [peaks all_peaks(i)]; %vrednosti amplituda
        locations = [locations all_locations(i)]; %vrednosti pozicija u nizu
    end
end

locations = locations*(Fs/N); % skaliranje pozicija pikova u Hz

%u vektoru locations se nalaze neke vrednosti u Hz koje su jako bliske
%jedna drugoj, stoga je potrebno te bliske vrednosti usrednjiti i
%sacuvati u nekom novom vektoru valid_locations

pom = [];
valid_locations = [];
for i=1:length(locations)
    if (i == 1)
        pom = [locations(i)];
    else
        if (locations(i) - locations(i-1) > 100)
            valid_locations = [valid_locations sum(pom)/length(pom)];
            pom = [locations(i)];
        else
            pom = [pom locations(i)];
        end
    end
end
valid_locations = [valid_locations sum(pom)/length(pom)];

distances = diff(valid_locations); %rastojanja izmedju susednih pikova
average_distance = sum(distances)/length(distances); %uprosecena rastojanja
number_peaks = length(valid_locations);
for i=1:number_peaks
    if(locations(i) > 20) %treba nam prvi pik, a ne DC komponenta
        first_pick = locations(i);
        break
    end
end
f = [first_pick average_distance];
end
```

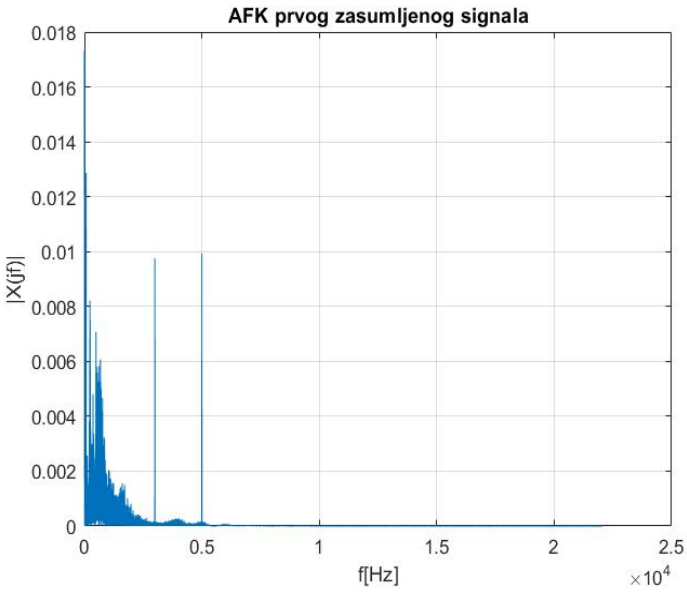
3 Zadatak 3

a) Za parametar $R = 0$, vremenski oblik signala *forrest_ gump_ zasumljen1.wav* je prikazan na sledećem grafiku.



Slika 11: Vremenski oblik signala *forrest_ gump_ zasumljen1.wav*

b) Na amplitudskoj faznoj karakteristici signala datoj ispod, uočavaju se dve sinusoidalne komponente šuma na frekvencijama od 3 kHZ i 5 kHz.



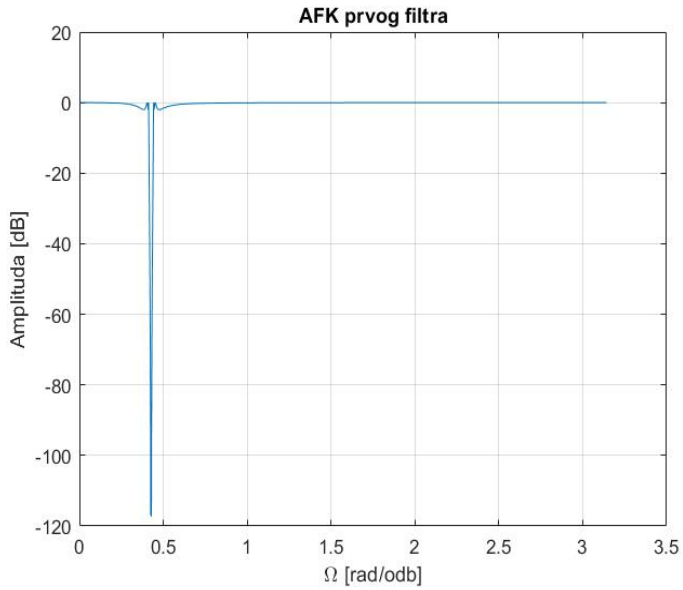
Slika 12: Amplitudska frekvencijska karakteristika signala *forrest_ gump_ zasumljen1.wav*

c) Pošto su uočene dve sinusoidalne komponente šuma, a potrebno je projektovati digitalni IIR filtar za svaku komponentu, jasno je da su nam potrebna dva filtra **nepropusnika učestanosti** (na engleskom *bandstop* filtri). Za moj parametar $R = 0$, u pitanju su Čebiševljevi filtri prvog reda.

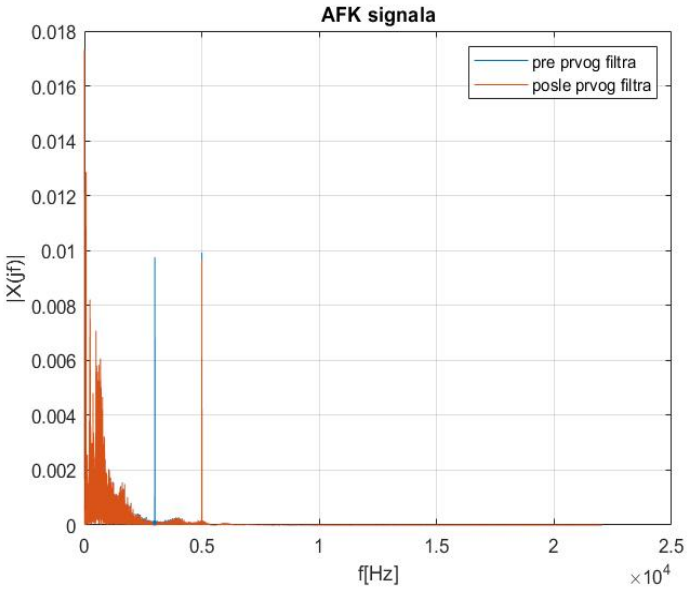
d) Za uklanjanje prve sinusoidalne komponente koja se nalazi na frekvenciji od 3 kHz, odabran je Čebišljev filtar prve vrste koji ne propušta frekvencije u intervalu $[2950, 3050]$ Hz, a sve frekvencije manje od 2950 Hz i sve frekvencije veće od 3050 Hz propušta. Najpre su u vektorima **Wp1** i **Ws1** dodate vrednosti ovih frekvencija podeljenih sa polovinom vrednosti frekvencije odabiranja **Fs**,

pošto u narednoj liniji koda, kada se određuje red filtra **n1** pomoću funkcije ***cheby1ord()*** kojoj se kao parametri šalju granične vrednosti frekvencija **Wp1** i **Ws1**, ali iz intervala $[0, 1]$. Nakon toga, pomoću funkcije ***cheby1()***, kojoj se prosleđuju parametri **n1**, **Rp** (što označava maksimalno dozvoljeno slabljenje signala u propusnom opsegu), **Wp1**, ali i string ***stop*** koji označava da je u pitanju *bandstop* filtar, se određuju koeficijenti polinoma filtra **a1** i **b1**, koji se koriste kao parametri u pozivu funkcije ***freqz()***. Pored njih, kao parametri se šalje i broj tačaka $(\frac{N}{2} + 1)$ u kojima je potrebno odrediti frekvencijski odziv kao i pomenuta frekvencija odabiranja **Fs** koja služi da se dobije vektor frekvencija umesto vektora učestanosti pozivom funkcije ***freqz()***.

Nakon toga, pomoću funkcije ***filter()*** kojoj se prosleđuju parametri **a1**,**b1** i vremenski oblik signala **x1** dobijamo filtrirani signal **x2**.

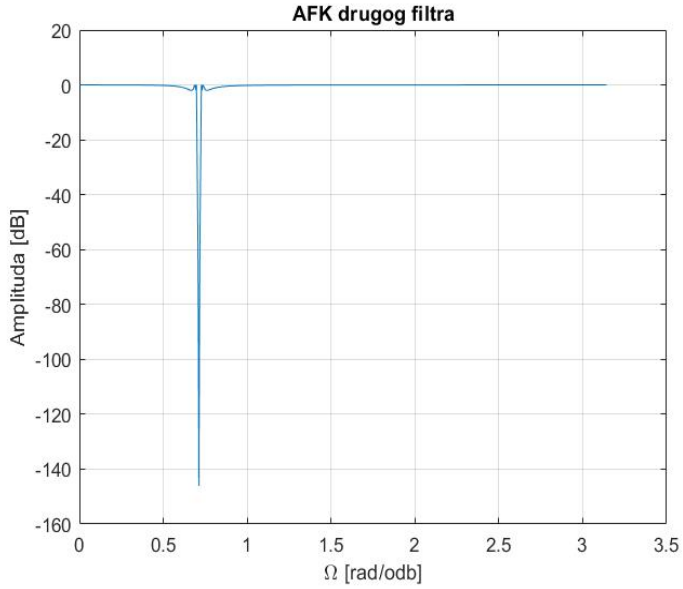


Slika 13: Amplitudska frekvencijska karakteristika prvog filtra

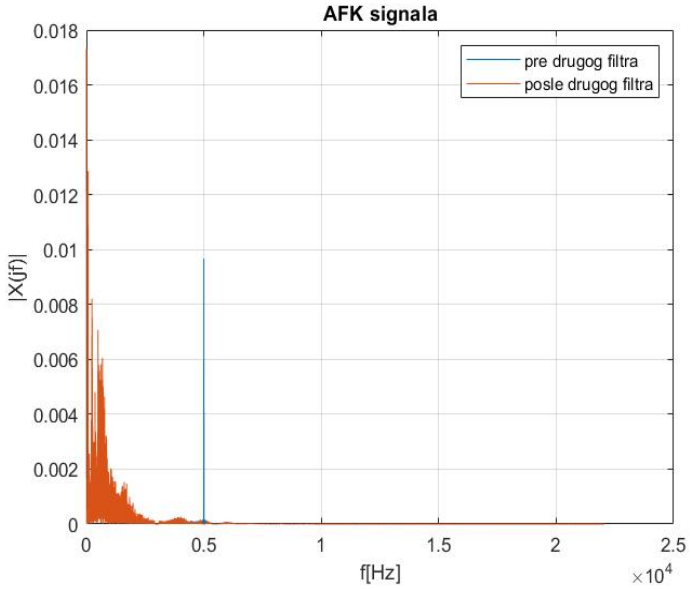


Slika 14: Amplitudska frekvencijska karakteristika signala pre i posle prvog filtra

Potpuno analogno se kreira drugi IIR filtar koji uklanja sinusoidalni šum na frekvenciji od 5 kHz.

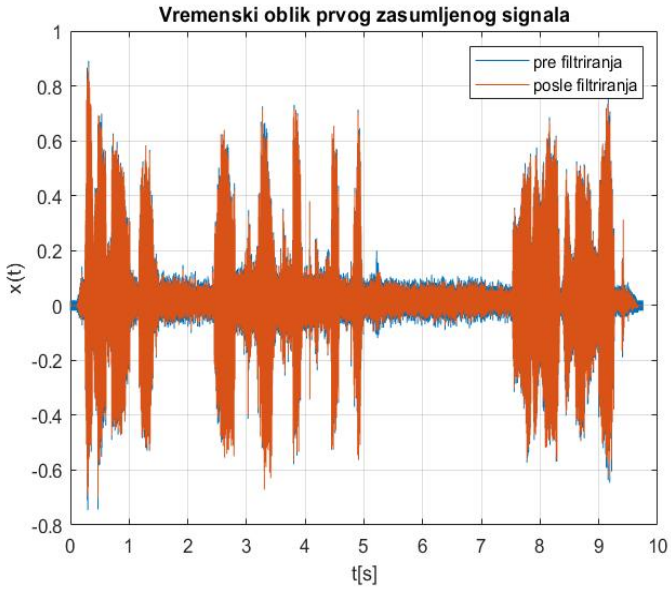


Slika 15: Amplitudska frekvencijska karakteristika drugog filtra



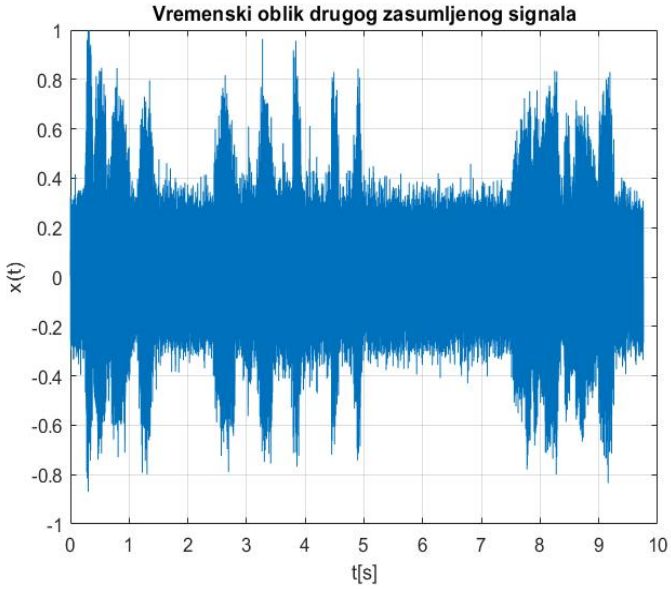
Slika 16: Amplitudska frekvencijska karakteristika signala pre i posle drugog filtra

Na samom kraju, sačuvan je dvostruko filtrirani audio signal i preslušavanjem je potvrđeno uklanjanje šuma.



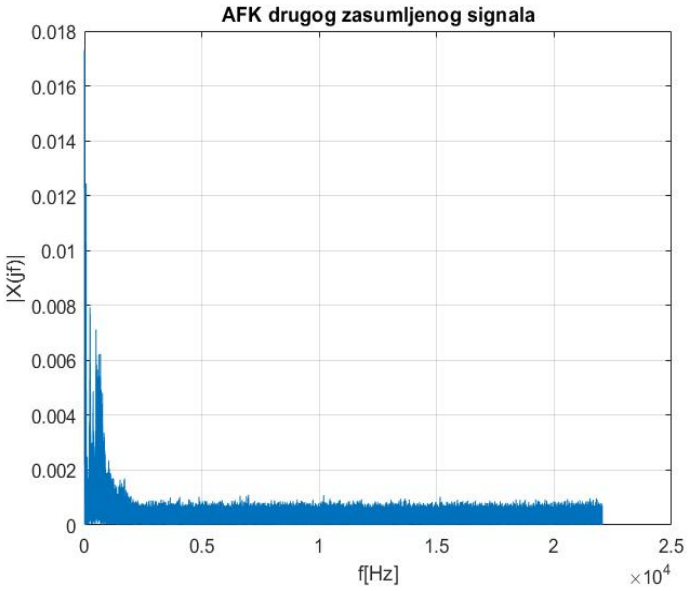
Slika 17: Vremenski oblici zašumljenog signala *forrest_ gump_ zasumljen1.wav* i isfiltriranog signala *isfiltrirani1.wav*

e) Sada je potrebno projektovati analogni IIR filtar (takođe Čebiševljev prve vrste), potom ga diskretizovati (za parametar $R = 0$ bilinearnom transformacijom) i takvim filtrom filtrirati drugi zašumljeni signal *forrest_ gump_ zasumljen2.wav*.



Slika 18: Vremenski oblik zašumljenog signala *forrest_ gump_ zasumljen2.wav*

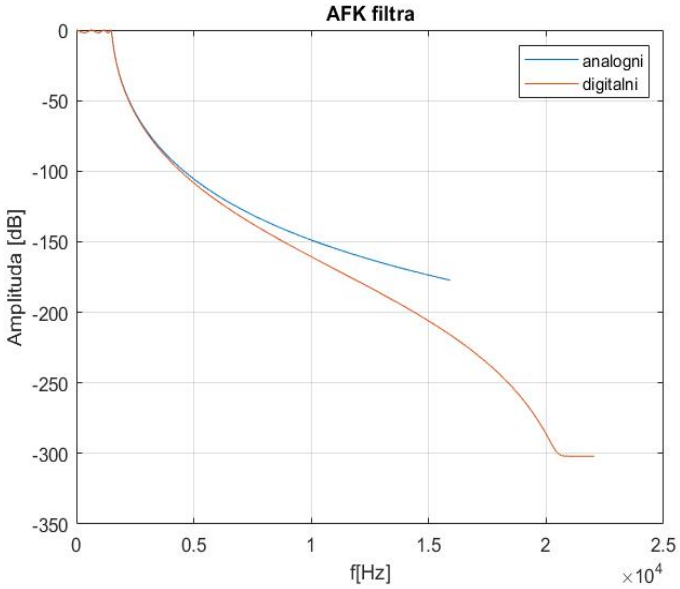
Najpre, iscrtani su vremenski i frekvencijski oblici zašumljenog signala na potpuno identičan način kao i u prethodnom delu zadatka. Uočavamo da se šum javlja gotovo uniformno na svim učestanostima većim od 2 kHz. Zaključujemo da nam je potreban NF filtar čiju graničnu frekvenciju biram da je 2 kHz. Slično kao u prethodnom delu zadatka, potrebno je odrediti granične vrednosti učestanosti (ne)propuštanja **Wpb** i **Wsb**, jer je u pitanju NF filtar. Kao i u prethodnom delu zadatka, koristeći funkciju ***cheby1ord()*** određuje se red filtra, a koristeći funkciju ***cheby1()*** dobijamo vrednosti koeficijenata polinoma filtra **ab** i **bb**. Jedina razlika u odnosu na prvi deo ovog



Slika 19: Amplitudska frekvencijska karakteristika zašumljenog signala *forrest_gump_zasumljen2.wav*

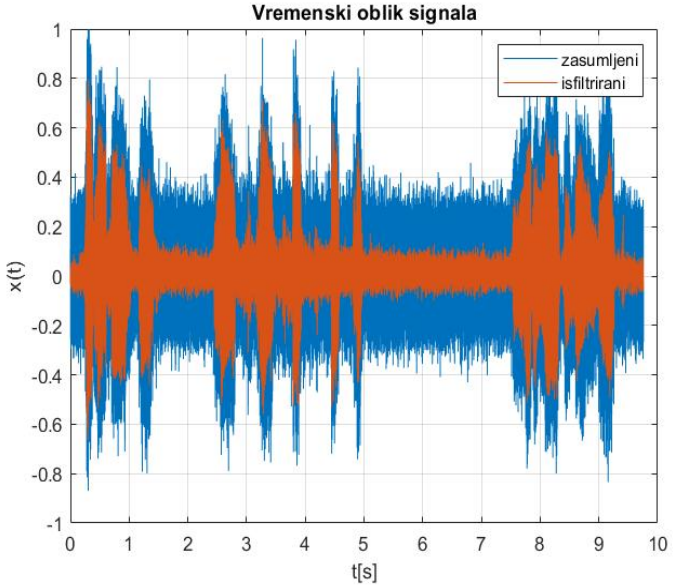
trećeg zadatka jeste ta što, pored svih parametara u tom delu, prilikom projektovanja analognog filtra, parametrima funkcija ***cheby1ord()*** i ***cheby1()*** dodaje se i string ***s***, koji označava da je u pitanju analogni filtar. Naredni korak se razlikuje minimalno u odnosu na prvi deo zadatka, pošto umesto funkcije ***freqz()*** se poziva funkcija ***freqs()*** kojom se dobija frekvencijski odziv analognog filtra.

Sada se funkcijom ***bilinear()*** određuju koeficijenti polinoma digitalizovanog filtra **az** i **bz** od koeficijenata polinoma analognog filtra **ab** i **bb**. Ponovo, kao u prvom delu zadatka, koristeći funkciju ***freqz()*** se određuje frekvencijski odziv digitalizovanog filtra.

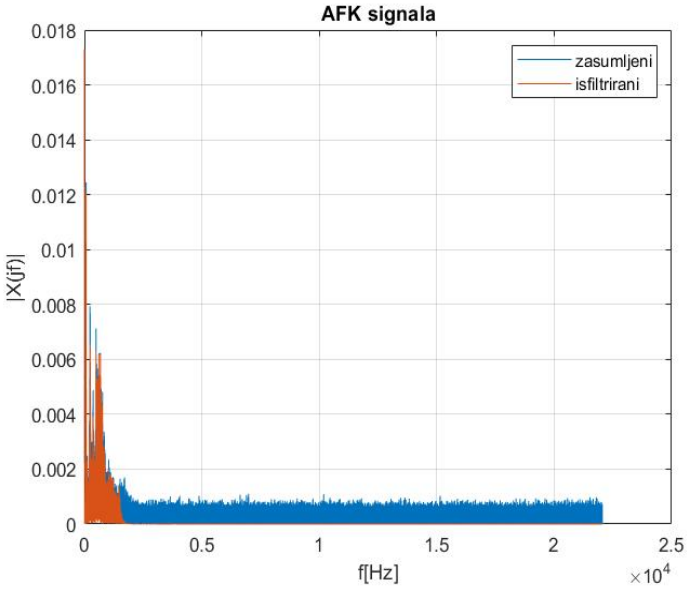


Slika 20: Amplitudske frekvencijske karakteristike analognog i digitalnog filtra

Na samom kraju, odrađena je i filtracija zašumljenog signala projektovanim, digitalnim filtrom i preslušavanjem filtriranog audio snimka, čulno se opaža smanjenje prisutnosti šuma, ali i dosta dublji ton zvuka, što je i očekivano, jer je korišćen NF filtar, koji propušta niske frekvencije, a ne propušta visoke frekvencije zvuka.



Slika 21: Vremenski oblici zašumljenog signala *forrest_ gump_ zasumljen2.wav* i isfiltriranog signala



Slika 22: Amplitudska frekvencijska karakteristika zašumljenog signala *forrest_ gump_ zasumljen2.wav* i isfiltriranog signala *isfiltrirani2.wav*

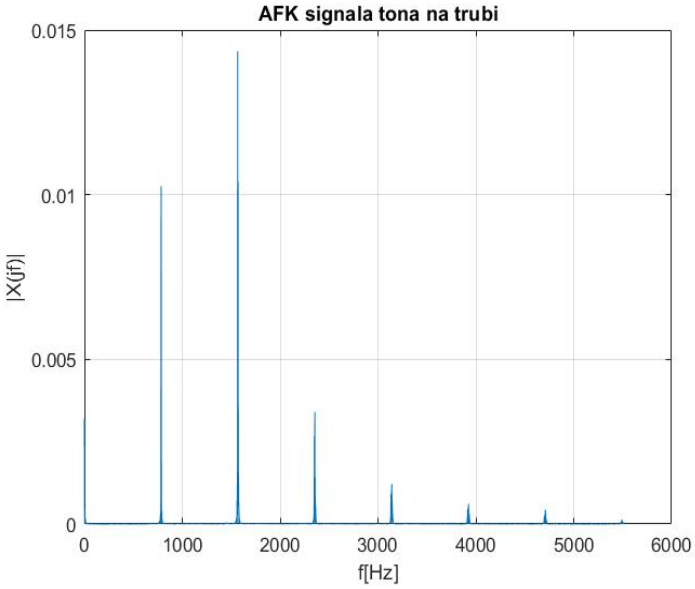
```
%% a - iscrtavanje u vremenskom domenu
clear all
close all
clc

[x, Fs] = audioread('forrest_gump_zasumljen1.wav');
t = 0:1/Fs:(length(x)-1)/Fs;

figure(1)
plot(t,x);
xlabel('t[s]'); ylabel('x(t)');
title('Vremenski oblik prvog zasumljenog signala'); grid on;

%% b - iscrtavanje AFK

N = 2^nextpow2(length(x));
f1 = 0:Fs/N:Fs/2;
X = fft(x,N)/length(x);
X1 = abs(X(1:N/2+1));
X1(2:N/2+1) = 2*X1(2:N/2+1);

figure(2)
plot(f1,X1);
xlabel('f[Hz]'); ylabel('|X(jf)|');
title('AFK prvog zasumljenog signala'); grid on;

%% c - prvi filtar - uklanjanje sinusoidalnog suma na 3kHz

Rp = 2;
Rs = 40;
Wp1 = [2900 3100]/(Fs/2);
Ws1 = [2950 3050]/(Fs/2);
[n1, Wn1] = cheb1ord(Wp1, Ws1, Rp, Rs);
[b1, a1] = cheby1(n1, Rp, Wp1, 'stop');
[h1, w1] = freqz(b1, a1, N/2+1);

figure(3)
plot(w1, 20*log10(abs(h1)));
xlabel('\Omega [rad/odb]'); ylabel('Amplituda [dB]');
title('AFK prvog filtra'); grid on;

%filtriranje signala prvim filtrom

y1 = filter(b1, a1, x);
Y = fft(y1,N)/length(y1);
Y1 = abs(Y(1:N/2+1)); %abs funkcija ne radi
Y1(2:N/2+1) = 2*Y(2:N/2+1);

%% c - drugi filtar - uklanjanje sinusoidalnog suma na 5kHz

Wp2 = [4900 5100]/(Fs/2);
Ws2 = [4950 5050]/(Fs/2);
[n2, Wn2] = cheb1ord(Wp2, Ws2, Rp, Rs);
[b2, a2] = cheby1(n2, Rp, Wp2, 'stop');
[h2, w2] = freqz(b2, a2, N/2+1);

figure(4)
plot(w2, 20*log10(abs(h2)));
```

```
59 xlabel('\Omega [rad/odb]'); ylabel('Amplituda [dB]');
60 title('AFK drugog filtra '); grid on;
61
62 % filtriranje filtriranog signala drugim filtrom
63
64 y2 = filter(b2, a2, y1);
65 Y = fft(y2,N)/length(y2);
66 Y2 = abs(Y(1:N/2+1));
67 Y2(2:N/2+1) = 2*Y(2:N/2+1);
68
69 figure(5)
70 plot(t,x);
71 hold on;
72 plot(t,y2);
73 xlabel('t[s]'); ylabel('x(t)'); grid on;
74 title('Vremenski oblik prvog zasumljenog signala');
75 legend('pre filtriranja','posle filtriranja');
76
77 figure(6)
78 plot(f1,X1);
79 xlabel('f[Hz]'); ylabel('|X(jf)|');
80 title('AFK signala'); grid on;
81 hold on;
82 plot(f1,abs(Y1)); %prilikom odredjivanja vektora Y1, koji bi zbog
83                   %abs() funkcije trebao biti nenegativan
84                   %(sto nije tako)
85                   %da bih postigao zeljenu apsolutnost prilikom iscrtavanja
86                   %grafika, u plot() naredbi dodatno "abs"-ujem vektor Y1
87                   %ista stvar se desava i kasnije kod vektora Y2
88 legend('pre prvog filtra','posle prvog filtra');
89
90 figure(7)
91 plot(f1,abs(Y1));
92 xlabel('f[Hz]'); ylabel('|X(jf)|');
93 title('AFK signala'); grid on;
94 hold on;
95 plot(f1,abs(Y2));
96 legend('pre drugog filtra','posle drugog filtra');
97
98 audiowrite('isfiltriran1.wav', y2, Fs);
99 %% e - drugi zasumljeni signal
100
101 [x2,Fs2] = audioread('forrest_gump_zasumljen2.wav');
102
103 t2 = 0:1/Fs2:(length(x2)-1)/Fs2;
104 figure(8)
105 plot(t2,x2);
106 xlabel('t[s]'); ylabel('x(t)'); grid on;
107 title('Vremenski oblik drugog zasumljenog signala');
108 %hold on;
109
110 N2 = 2^nextpow2(length(x2));
111 f2 = 0:Fs2/N2:Fs2/2;
112 Xb = fft(x2,N2)/length(x2);
113 Xb1 = abs(Xb(1:N2/2+1));
114 Xb1(2:N2/2+1) = 2*Xb(2:N2/2+1);
115
116 figure(9)
```

```
117 plot(f2,abs(Xb1));
118 xlabel('f[Hz]'); ylabel('|X(jf)|');
119 title('AFK drugog zasumljenog signala'); grid on;
120 %hold on;
121
122 %analogni filtar
123 Wpb = 1500*2*pi;
124 Wsb = 2000*2*pi;
125 [nb, Wnb] = cheb1ord(Wpb, Wsb, Rp, Rs,'s');
126 [bb, ab] = cheby1(nb, Rp, Wpb, 's');
127 [hb, wb] = freqs(bb, ab, N2/2+1);
128
129 figure(10)
130 plot(wb/(2*pi),20*log10(abs(hb)));
131 grid on;
132 hold on;
133
134 [bz, az] = bilinear(bb, ab, Fs2);
135 [hz, fz] = freqz(bz, az, N2/2+1, Fs2); %digitalni
136
137 figure(10)
138 plot(fz, 20*log10(abs(hz)));
139 xlabel('f[Hz]'); ylabel('Amplituda [dB]');
140 title('AFK filtra'); grid on;
141 legend('analogni','digitalni');
142 grid on;
143
144 yb = filter(bz, az, x2);
145 Yb = fft(yb,N2)/length(yb);
146 Yb1 = abs(Yb(1:N2/2+1));
147 Yb1(2:N2/2+1) = 2*Yb1(2:N2/2+1);
148
149 figure(11)
150 plot(t2,x2);
151 hold on;
152 plot(t2,yb);
153 xlabel('t[s]'); ylabel('x(t)'); grid on;
154 title('Vremenski oblik signala');
155 legend('zasumljeni','isfiltrirani');
156
157 figure(12)
158 plot(f2,abs(Xb1));
159 hold on;
160 plot(f2,Yb1);
161 xlabel('f[Hz]'); ylabel('|X(jf)|'); grid on;
162 title('AFK signala');
163 legend('zasumljeni','isfiltrirani');
164
165 audiowrite('isfiltriran2.wav', yb, Fs2);
```

4 Zadatak 4

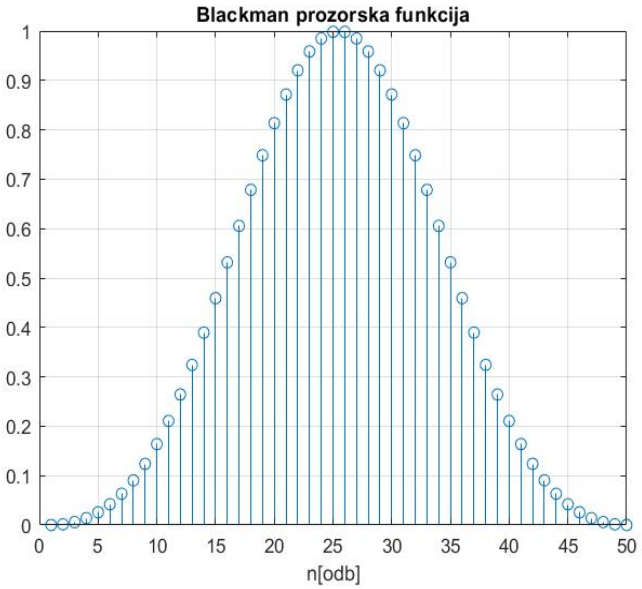
a) Prvi signal iz drugog zadatka je *truba_4.wav*. Amplitudska fazna karakteristika tog signala data je ispod.



Slika 23: Amplitudska frekvencijska karakteristika signala *truba_4.wav*

Neophodno je isfiltrirati dati signal FIR filtrom tako da se propusti samo prvi pik iz amplitudske karakteristike signala tona. Uočavamo da se prvi pik nalazi na frekvenciji oko 750 Hz. Stoga će se projektovati FIR filtra propusnik učestanosti (na engleskom *bandpass filter*) sa opsegom frekvencija $[500, 1000]$ Hz. Pošto nam se ostavlja sloboda projektovanja FIR filtra, odlučio sam da projektujem filtar 49. reda (u kodu je ova promenljiva označena sa **n**). Ono što nam nije sloboda izbora jeste prozorska funkcija, a za moj parametar $S = 0$ to je *Blackman* prozorska funkcija.

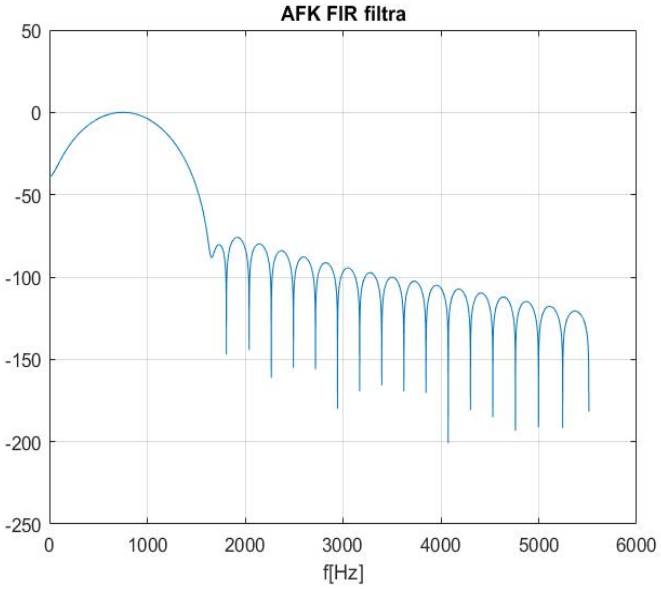
Pomenuto je da je u kodu već označen red filtra sa **n**, samim tim, naredbom ***blackman()*** formira se prozorska funkcija u **n+1** (pozivni argument funkcije ***blackman()***) tačaka.



Slika 24: *Blackman* prozorska funkcija u 30 tačaka

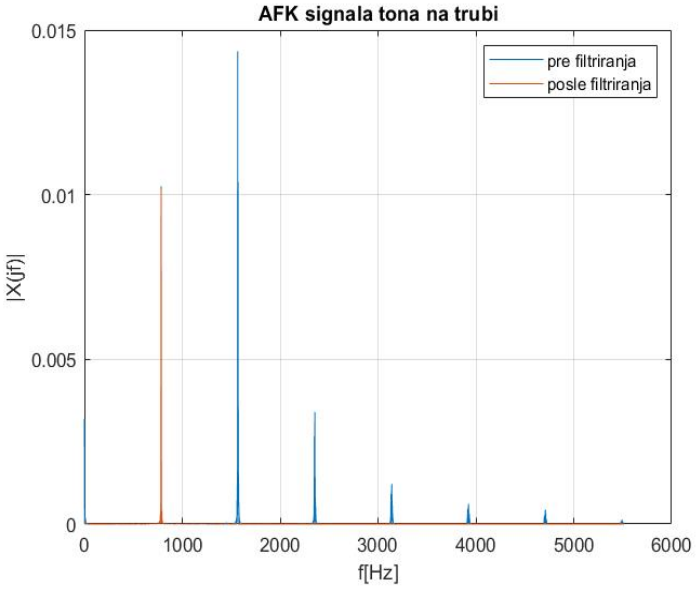
Nakon toga, deklarisan je i opseg propusnih frekvencija **Wn** koji je podeljen polovinom frekven-

cije odabiranja **Fs**, iz istog razloga kao i kod digitalnih IIR filtara - kako bi se vrednosti granica opsega frekvencija prevele u interval $[0, 1]$. Zatim se koeficijenti polinoma filtra u brojiocu **b** dobijaju pomoću ugrađene funkcije ***fir1()***, čiji su pozivni parametri red filtra **n**, propusni opseg **Wn** i prozorska funkcija **window**, a potom se frekvencijski odziv dobija na potpuno isti način kao i u prethodnom zadatku, pomoću funkcije ***freqz()*** i iscrtavanje amplitudske frekvencijske karakteristike tog FIR filtra se odvija na potpuno isti način kao što je to bio slučaj kod IIR filtara.



Slika 25: Amplitudska frekvencijska karakteristika FIR filtra propusnika učestanosti

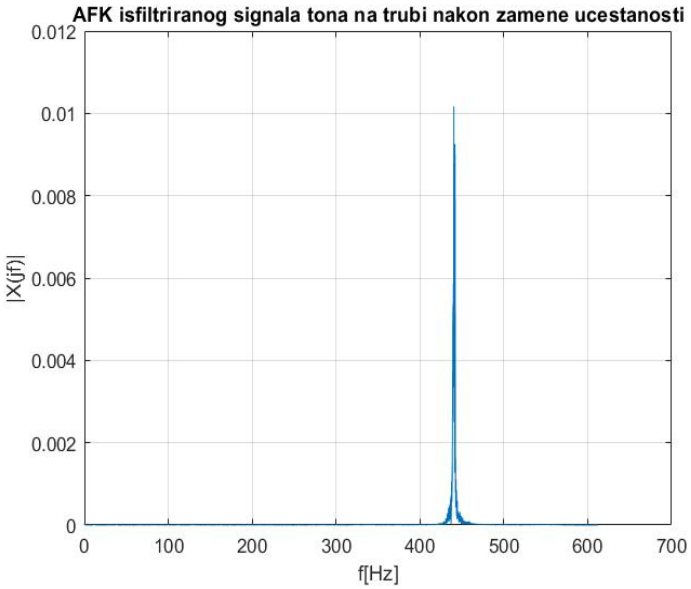
Takođe, i filtracija signala se vrši pomoću iste funkcije ***filter()***.



Slika 26: Amplitudska frekvencijska karakteristika signala *truba_4.wav* pre i posle filtriranja

Isfiltrirani signal zapisan u vektoru **y1** je sačuvan u *audio* formatu *isfiltriran3.wav* i primećuje se razlika u odnosu na *audio* zapis *truba_4.wav*.

b) U prethodnom delu zadatka filtrirani signal sadrži samo jednu sinusoidu na učestanosti nešto manjoj od 800 Hz. Pri tome, učestanost odabiranja signala iz .wav fajla je **Fs** i iznosi 11025 Hz. Da bi se dobio efekat zamene učestanosti, neophodno je odabrati dati signal u vremenu nekom novom frekvencijom odabiranja **Fs2** koja ne zadovoljava uslov odabiranja ($\frac{Fs_2}{2} < f_{gr}$). Pošto je jedini pik na učestanosti od oko 800 Hz (što se može interpretirati kao minimalna vrednost granične učestanosti), jasno je da nova frekvencija odabiranja **Fs_new** mora biti manja od 1600 Hz. Konkretno u zadatku, za **Fs_new** uzeta je vrednost 1125 Hz, što je zapravo vrednost koja se dobije deljenjem **Fs** sa 9. Tim povodom, najpre je vršeno odabiranje u vremenu isfiltriranog signala **y** jednom *for*-petljom tako što se u novi vektor **y1** upisivao svaki deveti element vektora **y**. Nakon toga, istim postupkom kao i u prethodnim zadacima odrađena je Furijeova transformacija signala **y1** i određena je nova frekvencijska osa **f1**.



Slika 27: Amplitudska frekvencijska karakteristika filtriranog signala *truba_4.wav* nakon zamene učestanosti

Poslednji grafik ilustruje efekat zamene učestanosti, jer je prethodnom programskom manipulacijom dobijena nova amplitudska frekvencijska karakteristika koja takođe ima samo jedan pik, ali sada na drugačijoj frekvenciji. Ta nova frekvencija pika iznosi približno 440 Hz, što je u skladu sa matematičkim proračunom.

Na samom kraju, preslušavanjem audio snimka *isfiltriran4.wav* primećuje se značajna razlika u tonu u odnosu na signal *isfiltriran3.wav* u pogledu dubine tona.

```
clear all
close all
clc
%% a - AFK signala tona
[x, Fs] = audioread('truba_4.wav');
N = 2^nextpow2(length(x));
X = fft(x,N)/length(x);
f = 0:(Fs/N):(Fs/2);
X1 = abs(X(1:(N/2)+1));
X1(2:(N/2)+1) = 2*X(2:(N/2)+1); % X1(f) = X2(-f) + X2(f)

figure(1)
plot(f,abs(X1));
title('AFK signala tona na trubi');
xlabel('f[Hz]'); ylabel('|X(jf)|'); grid on;

%% projektovanje FIR filtra
n = 49;
window = blackman(n+1);
Wn = [500 1000]/(Fs/2);
b = fir1(n, Wn, window);
a = 1;
[hz, fz] = freqz(b, a, N/2+1, Fs);

figure(2)
stem(window);
xlabel('n[odb]');
title('Blackman prozorska funkcija'); grid on;

figure(3)
plot(fz, 20*log10(abs(hz)));
xlabel('f[Hz]');
title('AFK FIR filtra'); grid on;

%% filtriranje signala

y = filter(b, a, x);
Y = fft(y,N)/length(y);
Y1 = abs(Y(1:N/2+1));
Y1(2:N/2+1) = 2*Y1(2:N/2+1);

figure(4)
plot(f,abs(X1));
hold on;
plot(f,abs(Y1));
title('AFK signala tona na trubi');
xlabel('f[Hz]'); ylabel('|X(jf)|'); grid on;
legend('pre filtriranja','posle filtriranja');

audiowrite('isfiltriran3.wav', y, Fs);
%% b
y1 = [];
for i=1:length(y)
    if (mod(i,9) == 0)
        y1 = [y1 y(i)];
    end
end

```

```
% fft novoodabranog signala y1
Fs_new = Fs/9;
N1 = 2^nextpow2(length(y1));
Y_new = fft(y1,N1)/length(y1);
Y1_new = abs(Y_new(1:(N1/2)+1));
Y1_new(2:(N1/2)+1) = 2*Y_new(2:(N1/2)+1);

f1 = 0:(Fs_new/N1):(Fs_new/2); %nova frekvencijska osa

figure(5)
plot(f1,abs(Y1_new));
title('AFK isfiltriranog signala tona na trubi nakon zamene ucestanosti');
xlabel('f[Hz]'); ylabel('|X(jf)|'); grid on;
grid on;

audiowrite('isfiltriran4.wav', y1, Fs_new);
```