Sadržaj

1. Opis problema

Cilj napisanog algoritma, koji će detaljno biti objašnjen u ovom tekstu, jeste nova dvostepena, rekurzivna M-rubusna estimacija parametara AR modela signala. Ovaj metod daje izuzetne rezultate kada je u pitanju estimacija parametara signala pod uticajem nestacionarnog i impulsivnog šuma.

Napisani algoritam je zasnovan na teorijskoj diskusiji iz knjige „Robusna digitalna obrada govornog signala“ . U pomenutoj knjizi je predložen kompletan matematički aparat potreban za pisanje ovog algoritma i njegovo testiranje u odabranom programskom jeziku.

Za razvoj ovog algoritma je odabran programski paket MATLAB pri čemu su sve funkcije korišćene u algoritmu pisane od strane autora algoritma.

Sam algoritam se sastoji od nekoliko koraka:

1. generisanje signala i njegovih parametara,
2. propuštanje signala kroz robusni Kalmanov filtar (predfiltriranje),
3. računanje estimacije parametara pomoću promenljivog faktora zaboravljanja,
4. grafičko upoređivanje rezultata.

U ovom radu neće biti obrađivana teorija koja je korišćena za dobijanje rezultata već samo realizacija ove teorije na test signalima.

2. Generisanje test signala i njihovih parametara

Program je koncipiran tako da se različite celine algoritma izvršavaju zasebno u posebno napisanim funkcijama kako bi se lako mogli menjati pojedini delovi programa. Kod je pisan tako da se kao ulazni podatak programa (signal) može dovesti signal bilo kog reda pri čemu jedino treba definisati dužinu i red signala. Svi proračuni u algoritmu se prilagođavaju redu signala pa se tako jedan program može koristiti za analizu signala bilo kog reda.

Za analizu efikasnosti rada algoritma su korišćena dva test signala. Oba test signala su drugog reda. Napisan je algoritam i za modeliranje trećeg test signala osmog reda ali nismo uspeli da nađemo adekvatne parametre za taj test signal pa se ne dobijaju dobra rešenja.

Prvi test signal je kosinusoida sa jednom promenom frekvencije na 800-tom koraku.
Algoritam koji je korišćen za modeliranje ovog signala je sledeći:

```
%% Generisem test signal 1 i parametre tog signala
function [duz,t,p,teta,y]=test_signal_1()
% Definisanje parametara AR modela signala
f(1,1:800)=0.2;
f(1,801:1000)=0.4;
duz=1000;                     % duzina signala
t=1:duz;                      % vremenski interval
arg(1,t)=2*pi*f(1,t);
p=2;                          % red AR modela signala
teta_1(1,t)=-2*cos(arg(1,t)); % prvi parametara AR modela
teta_2=1;                     % drugi parametar AR modela
teta=zeros(p,duz);            % alokacija memorije
for i=1:duz
    teta(:,i)=[teta_1(1,i); teta_2];
end

% Definisem pocetne uslove AR modela (y(k-1) i y(k-2))
y(1,1:2)=cos(arg(1,1:2));

% Definisem AR model signala preko parametara AR modela
for i=3:duz % prve dve vrednosti su pocetne vrednosti
    y(1,i)=-teta_1(1,i)*y(1,i-1)-teta_2*y(1,i-2);
end
end
```

Na početku funkcije za test signal 1 je definisan izvestan broj promenljivih koje će se koristiti u daljem programu (duz,t,p,teta,y). Ulazni parametri za ovu funkciju ne postoje tako da je za argument funkcije stavljeno samo „()". Izlazni parametri funkcije su promenljive koje se koriste u daljem programu (globalne promenljive) pa ih zato navodimo u izlazu funkcije (duz,t,p,teta,y).

Pošto su u MATLAB-u sve promenljive predstavljene matricama, velika pažnja je posvećena pravilnom izboru dimenzija matrica kako bi algoritam mogao da radi. Na nekim mestima je vršeno i transponovanje određenih matrica da bi promenljive bile upotrebljive u kompletnom algoritmu.

Dosta je spora bilo oko dimenzija matrica koje su korišćene u delu algoritma koji je zadužen za računanje promenljivog faktora zaboravljanja gde su često odabrane trodimenzionalne matrice da bi vrednosti bile korektno predstavljene. Prve dve dimenzije (indeksa) predstavljaju broj vrsta i kolona a treća dimenzija predstavlja redni broj p dimenzione matrice (gde je p red signala).

Test signal 2 je modeliran na sličan način. U stvari, jedina razlika je u definisanju promene frekvencije obzirom da je signal 2 komplikovaniji od prvog pa tako ima osam promena vrednosti frekvencije. Treći test signal ima istu promenu frekvencije u vremenu kao i drugi test signal s tom razlikom što su dimenzije matrica morale biti ažurirane.

Promena frekvencije za drugi i treći test signal je definisana na sledeći način:

```
f(1,1:800)=0.1;
for i=800:1100
    f(1,i)=f(1,i-1)+0.001;
end
f(1,1101:1350)=0.4;
f(1,1350:1650)=0.1;
for i=1650:1800
    f(1,i)=f(1,i-1)+0.002;
end
f(1,1801:2000)=0.4;
f(1,2001:2200)=0.1;
f(1,2201:2600)=0.25;
```

Algoritam koji je korišćen za definisanje trećeg test signala je sledeći:

```
%% Generisem test signal 3 (signal osmog reda) i parametre tog signala
function [duz,t,p,teta,y]=test_signal_3()
duz=2600; % duzina signala
t=1:duz; % vremenski interval
f=zeros(1,duz);

f(1,1:800)=0.1;
for i=800:1100
    f(1,i)=f(1,i-1)+0.001;
end
f(1,1101:1350)=0.4;
f(1,1350:1650)=0.1;
for i=1650:1800
    f(1,i)=f(1,i-1)+0.002;
end
f(1,1801:2000)=0.4;
f(1,2001:2200)=0.1;
f(1,2201:2600)=0.25;

arg(1,t)=2*pi*f(1,t);
% Red AR modela signala
p=8;

% Definisem parametre teta za test signal 3
teta_1=zeros(1,duz);
for i=1:duz
    teta_1(1,i)=-2*cos(arg(1,i)); % prvi parametar AR modela
end
teta_2=0.24;
teta_3=0.22;
teta_4=0.17;
teta_5=-0.2;
teta_6=0.15;
```

```
teta_7=0.25;
teta_8=0.1;
% Spajam sve parametre u jednu matricu (teta)
teta=zeros(p,duz);
for i=1:duz
    teta(:,i)=[teta_1(1,i); teta_2; teta_3; teta_4;...
               teta_5; teta_6; teta_7; teta_8];
end

% Definisem pocetne uslove AR modela (y(k-1) i y(k-2)... y(k-8))
y(1,1:p)=-2*cos(arg(1,1:p));

% Definisem AR model signala preko parametara AR modela
for i=(p+1):duz % prve dve vrednosti su pocetne vrednosti
    y(1,i)=-teta_1(1,i)*y(1,i-1)-teta_2*y(1,i-2)-teta_3*y(1,i-3)...
           -teta_4*y(1,i-4)-teta_5*y(1,i-5)-teta_6*y(1,i-6)-teta_7*y(1,i-7)...
           -teta_8*y(1,i-8);
end
end
```

Kada smo modelirali sva tri test signala mogli smo da pređemo na sledeći korak, Kalmanovo predfiltriranje.

3. Robusni Kalmanov filtar

Iako u MATLAB-u već postoji ugrađen Kalmanov filtar koji se može pozvati kao funkcija mi smo se odlučili na pisanje sopstvenog algoritma iz nekoliko razloga. Pošto smo lično pisali algoritam za Kalmanov filtar mogli smo da imamo uvid u bilo koji deo procesa, prema tome mogli smo bliže da posmatramo uticaj promene određenih parametara na filtraciju signala. Drugi razlog je mnogo važniji. U MATLAB-u ne postoji funkcija koja ima ulogu robusnog Kalmanovog filtra. Zbog toga smo robustifikaciju kao i ostatak algoritma napisali koristeći se pomenutom literaturom.

Kalmanova filtracija je takođe izvedena u zasebnoj funkciji. Prvi korak u ovoj funkciji je bio učitavanje jednog od prethodno modeliranih signala.

U funkciji za robusnu Kalmanovu filtraciju se vrši izbor test signala:

```
%% Generisem test signal p-tog reda i parametre tog signala
if izbor==1
    % Test signal 1
    [duz,t,p,teta,y]=test_signal_1();
elseif izbor==2
    % Test signal 2
    [duz,t,p,teta,y]=test_signal_2();
elseif izbor==3
    % Test signal 3
    [duz,t,p,teta,y]=test_signal_3();
end
```

Vrednost promenljive *izbor* se vrši u glavnom programu o čemu će biti reči dalje u tekstu. Nakon generisanja (uvoza) željenog signala potrebno je generisati beli Gausovski šum koji ćemo dodati na modelirani signal.

Generisanje Gausovskog šuma je izvedeno u zasebnoj funkciji.

Signal je pod uticajem dve vrste šuma: nominalnog i kontaminiranog Gausovskog šuma. Nominalni Gausovski šum ima nultu srednju vrednost dok kontaminirani ima vrednost 0.1. Odnosi varijansi šumova su definisani sa $\sigma_0^2 / \sigma = 10$ pri čemu smo usvojili da je $\sigma_0^2 = 0.7$.

Iako je u literature odnos nominalnog i kontaminiranog Gausovskog šuma definisan pomoću

$$p(e_k) = (1-\varepsilon) * N(0,\sigma^2) + \varepsilon * N(0,\sigma_0^2)\,, \quad 0 < \varepsilon \le 1,$$

mi smo se odlučili za sledeći pristup.

Prilikom svakog koraka (dužine test signala) generišemo jedan nasumični broj od 0 do 1. Ukoliko je taj broj veći od definisanog odnosa (promenljiva *odnos* se definiše takođe na početku glavnog programa) na signal u datom koraku dodajemo kontaminirani Gausovski šum, a ukoliko je manji dodajemo nominalni Gausovski šum. Šum smo na ovaj način dodavali signalu kako bi u zavisnosti od veličine promenljive *odnos* mogli da kontrolišemo broj outlier-a u signalu i da pratimo kvalitet estimacije u zavisnosti od tipa šuma.

Algoritam kojim je definisam Gausovski šum je sledeći:

```
% Definisem kontaminirani, beli Gausovski sum
sr_vr_kont_suma=0.1;
sigma_0=sqrt(0.7);
varijansa_kont_suma=sigma_0;
kont_beli_sum=sr_vr_kont_suma+varijansa_kont_suma.*randn(duz,1);
% Definisem nominalni, beli Gausovski sum
sr_vr_nom_suma=0;
sigma_1=sigma_0/sqrt(10);
varijansa_nom_suma=sigma_1;
nom_beli_sum=sr_vr_nom_suma+varijansa_nom_suma.*(2*(rand(duz,1)-0.5));
% Kombinujem ova dva suma u zavisnosti od izabranog odnosa
beli_sum=zeros(duz,1);
for i=1:duz
    nasum_br=rand(1);
    if nasum_br<=(1-odnos)
        beli_sum(i,1)=nom_beli_sum(i,1);
    else
        beli_sum(i,1)=kont_beli_sum(i,1);
    end
end
```

Sada je potrebno izračunati varijansu dobijenog šuma kao i izvršiti skaliranje tako da odnos signal šum bude 10 decibela (SNR=10dB).

```
% Racunam varijansu dobijenog suma
varijansa_suma=var(beli_sum);
% Skaliram sum u odnosu da signal da SNR bude 10dB
[beli_sum,faktor]=skaliranje(beli_sum,y,duz);
```

Poslednji korak je dodavanje skaliranog šuma na modelirani signal kako bi dobili signal sa šumom koji je potrebno filtrirati robusnim Kalmanovim filtrom:

```
% Dodajem beli Gausovski sum na signal
y_sum=y+beli_sum;
```

Nakon definisanja šuma izračunat je parametar *Z* iz AR modela:

```
% Definisem Z izvedeno iz AR modela
Z=zeros(p,duz); % definisem unapred zbog brze alokacije memorije
for i=(p+1):duz
    for j=1:p
        Z(j,i)=-y_sum(i-j,1);
    end
end
```

Zatim smo definisali grešku predikcije:

```
% Definisem gresku predikcije (rezidual merenja)
epsilon=zeros(duz,1);
for i=1:duz
    epsilon(i,1)=y_sum(i,1)-Z(:,i)'*teta(:,i);
end
```

Ostalo je da se definišu matrice jednačina stanja sistema, konstante kao i početne vrednosti estimacije signala i matrice kovarijanse greške estimacije.

U ovom trenutku imamo generisan signal i parametre signala p-tog reda kao i početne vrednosti promenljivih (matrica), dakle napravili smo sve preduslove za algoritam Kalmanove filtracije.

Jednačine koje definišu Kalmanov filtar su rekurzivne i međusobno zavisne jer se ažuriraju istovremeno tako da su morale biti napisane sve odjednom (u istom koraku). U MATLAB-u indeksi za promenljive počinju od broja 1 tako da je početni korak svih promenljivih prvi korak. Iz tog razloga iteracija petlje u kojoj se računa Kalmanova filtracija počinje od drugog koraka.

Kao rezultat Kalmanove filtracije dobijamo *z_hat* predikciju izlaznog signala sistema. Grafički prikazana rešenja estimacija i ostalih bitnih parametara kompletnog algoritma biće predstavljena i analizirana kasnije u drugom delu ovog teksta.

Algoritam Kalmanove filtracije je izveden na sledeći način:

```
%% Algoritam Kalmanove filtracije
for k=2:duz
% ********** AZURIRANJE VREMENA **********
    % Predikcija stanja sistema
    x_line(:,:,k)=F(:,:,k-1)*x_hat(:,:,k-1);
    % Matrica kovarijanse greske predikcije
    M(:,:,k)=F(:,:,k-1)*P(:,:,k-1)*(F(:,:,k-1)')+Q;
    % Predikcija izlaza sistema
% ********** AZURIRANJE MERENJA **********
    % Matrica kovarijanse reziduala
    s(k,1)=sqrt(H*M(:,:,k)*(H')+R);
    % Tezinska forma
    [omega_pom]=omega_kalman(k,v,s);
    omega(k,1)=omega_pom;
    % Matrica Kalmanovog pojacanja
    K(:,:,k)=omega(k,1)*M(:,:,k)*(H')*(s(k,1)^(-1));
    % Matrica kovarijanse greske estimacije
    P(:,:,k)=(eye(p)-K(:,:,k)*H)*M(:,:,k);
    % Procena stanja
    x_hat(:,:,k)=x_line(:,:,k)+K(:,:,k)*v(k,1);
    % Sum ulaza sistema
    v(k,:)=z(k,:)-H*x_hat(:,:,k);
    % Predikcija izlaza sistema
    z_hat(k,1)=H*x_hat(:,:,k);
end
```

Na kraju ove funkcije (*Kalmanova_filtracija*) dodali smo opciju da se filtrirani signal oslabi kako bi se rezultati mogli grafički efikasnije uporediti. Ova opcija može se i isključiti postavljanjem promenljive *dailine* na nultu vrednost. Postavljanje ove promenljive na vrednost 1 uključuje ovu opciju.

```
% Slabljenje filtriranog signal (ako je tako odabrano u glavnom programu)
if dailine==1
    z_hat=z_hat*faktor;
end
```

Sledi opis funkcija za računanje promenljivog faktora zaboravljanja preko generalisane greške predikcije i diskriminacione funkcije.

4. Generalisana greška predikcije

Usvojili smo da je izlazni signal iz Kalmanovog filtra ulazni signal u funkciji koja računa generalisanu grešku predikcije (GGP). Na početku ove funkcije definišemo početne vrednosti promenljivih koje ćemo koristiti u daljem radu algoritma kao i vrednosti konstanti predloženih u teorijskoj analizi. Takođe, računamo ponovo matricu koja nosi informaciju o prethodnim rekurzijama signala (*Z_novo*) i matricu greške estimacije (*epsilon_novo*).

```
%% Generalisana greska predikcije - pocetni uslovi i potrebne promenljive
% Izlazni signal nakon Kalmanove predfiltracije je sada ulazni signal
y_novo=z_hat;
% Definisem faktor skaliranja
d=median(abs(y_novo-median(y_novo)))/0.6745;
% Definisem Z
Z_novo=Z;
for i=(p+1):duz
    for j=1:p
        Z_novo(j,i)=-y_novo(i-j,1);
    end
end
% Definisem pocetno epsilon i teta_hat
teta_hat=teta;
epsilon_novo=epsilon;
Ma=5;
for i=1:(Ma+1)
    epsilon_novo(i,1)=y_novo(i,1)-Z_novo(:,i)'*teta_hat(:,i);
end
% Definisem pocetno sigma_hat
sigma_hat=0.22*ones(duz,1);
% Definisem neke konstante za izracunavanje lambda
N_max=500;
lambda_min=0.75;
lambda_max=0.998;
E=zeros(duz,1);
omega_novo=omega; % izjednacavam ih zbog pocetnih vrednosti
N=zeros(duz,1);
lambda=zeros(duz,1);
```

Kao i kod Kalmanove filtracije i jednačine koje definišu generalisanu grešku predikcije su rekurzivne i međusobno zavisne pa se i one ažuriraju u svakom koraku.

```
%% Generalisana greska predikcije - algoritam
for k=(Ma+1):duz
% Opet definisem gresku predikcije (sa novim vrednostima)
epsilon_novo(k,1)=y_novo(k,1)-Z_novo(:,k)'*teta_hat(:,k-1);
% Racunam tezinsku formu (omega_novo)
[omega_pom_2]=omega_ggp(k,epsilon_novo,d,y_novo,Z_novo,teta_hat);
omega_novo(k,1)=omega_pom_2;
% Racunam estimiranu vrednost varijanse suma
sigma_hat(k,1)=((k-1)*sigma_hat(k-1,1)+(epsilon_novo(k,1)^2)*omega_novo(k,1))/k;
% Racunam Extended Prediction Error (EPE)
suma_1=0;
for i=1:Ma;
    suma_1=suma_1+epsilon_novo(k-i,1)^2;
end
E(k,1)=suma_1/Ma;
% Duzina memorije
N(k,1)=((sigma_hat(k,1)^2)*N_max)/E(k,1);
% Racunam promenljivi faktor zaboravljanja (VFF)
lambda(k,1)=max(1-E(k,1)/(sigma_hat(k,1)*N_max),lambda_min);
% Ogranicavam minimalnu i maksimalnu vrednost VFF-a
if lambda(k,1)>lambda_max
    lambda(k,1)=lambda_max;
elseif lambda(k,1)<lambda_min
    lambda(k,1)=lambda_min;
end
M(:,:,k)=P(:,:,k-1)/lambda(k,1);
```

```
pom_1=M(:,:,k)*Z_novo(:,k)*omega_novo(k,1); % pomocna promenljiva 1
pom_2=1+Z_novo(:,k)'*pom_1;                 % pomocna promenljiva 2

% Matrica pojacanja
K(:,:,k)=pom_1/pom_2;
% Definisem pomocnu konstantu
C=0.11;
% Matrica kovarijanse greske estimacije
P(:,:,k)=C*eye(p)-K(:,:,k)*(Z_novo(:,k)')*M(:,:,k);
% Estimirana vrednost parametara
teta_hat(:,k)=teta_hat(:,k-1)+K(:,:,k)*epsilon_novo(k,1);
end
```

Kao rezultat algoritma generalisane greške predikcije dobija se estimacija parametara AR modela (*teta_hat*). Na osnovu tih parametara generišemo signal da bi uporedili ne samo parametre već i izgled samog signala koji se dobija na osnovu estimiranih parametara.

```
% Modeliram signal na osnovu estimiranih parametara
z_novo=y_novo;
for i=3:duz
    z_novo(i,1)=-teta_hat(1,i)*z_novo(i-1,1)-teta_hat(2,i)*z_novo(i-2,1);
end
```

Treba primetiti da u algoritmu GGP-a figuriše konstanta C koja utiče na estimaciju parametra. Ovu konstantu treba birati tako da estimacija bude optimalna. Ova vrednost je u početku podešena na vrednost 1.

5. Diskriminaciona funkcija

Razlika između generalisane greške predikcije i diskriminacione funkcije je u načinu računanja promenljivog faktora zaboravljanja. Prema tome, razlika u algoritmu između ove dve funkcije je u par linija koda koji definiše računanje lambda.

Pošto diskriminaciona funkcija ima drugačiji algoritam za računanje lambda potrebno je usvojiti drugačije konstante:

```
% Definisem neke konstante za izracunavanje lambda
N_min=20;
N_max=500;
D_min=0;
D_max=1;
D=zeros(duz,1);
lambda_min=1-1/N_min;
lambda_max=1-1/N_max;
omega_novo=omega; % izjednacavam ih zbog pocetnih vrednosti
lambda=zeros(duz,1);
% Odredjujem velicinu prozora kod diskriminacione funkcije
I=50;
```

Razlika u algoritmu u odnosu na generalisanu grešku predikcije je sledeća:

```
% Racunam diskriminacionu f-ju preko logaritma f-je verodostojnosti
D(k,1)=L(k-I+1,k+I,epsilon_novo)-L(k-I+1,k,epsilon_novo)-L(k+1,k+I,epsilon_novo);
% Modifikujem D preko trenda diskriminacione funkcije
[D]=trend_D(k,I,D,epsilon_novo);
% Racunam lambda preko diskriminacione funkcije
lambda(k,1)=((lambda_max-lambda_min)/(D_min-D_max))*(D(k,1)-D_max)+lambda_min;
% Racunam maksimum diskriminacione funkcije
D_max=max(D(1:k,1));
```

Pri računanju diskriminacione funkcije D preko logaritma funkcije verodostojnosti koristili smo promenljivu L čije smo računanje izveli u zasebnoj funkciji.

```
% Logaritam funkcije verodostojnosti
suma_eps=0;
for i=a:b
    suma_eps=suma_eps+epsilon_novo(i,1)^2;
end
rez=(b-a+1)*log((1/(b-a+1))*suma_eps);
```

U funkciji za računanje diskriminacione funkcije smo koristili unapređeni MGLR pošto je običan MGLR imao takve promene lambda kao da signal nikada nije stacionaran. Dodavanjem dodatnih uslova preko funkcije *trend_D* dobili smo zadovoljavajuće promene lambda, takve da je lambda maksimalno kada je signal stacionaran a minimalno kada signal ima nagle promene.

Algoritam za funkciju trend je realizovan na sledeći način:

```
% Funkcija za biranje intervala u kome VFF ima nagle promene
n_1=k-round(I/4)+1; % donja granica
n_2=k+round(I/4);   % gornja granica
prag=80;            % vrednost praga

% Definisem lokalne ekstremume
alfa_min=min(D(n_1:n_2,1));
alfa_max=max(D(n_1:n_2,1));
delta_alfa=alfa_max-alfa_min;
if (delta_alfa<=prag)
    % Ako nastaju nagle promene u lokalu tada ne diram D
    D(k,1)=L(k-I+1,k+I,epsilon_novo)-L(k-I+1,k,epsilon_novo)-L(k+1,k+I,epsilon_novo);
elseif (delta_alfa>prag)
    % Ako ne nastaju nagle promene u lokalu signal je priblizno stacionaran
    D(k,1)=alfa_min;
end
```

U stvari, funkcija *trend_D* utvrđuje da li je u prozoru sa granicama [n_1,n_2] došlo do nagle promene signala. Ako jeste tada se D računa preko logaritamske funkcije verodostojnosti, ukoliko nije došlo do nagle promene tada je D jednako minimalnoj vrednosti D iz tog intervala.

Prag koji odlučuje kada će promena biti tretirana kao nagla, tj. kada će se ignorisati a kada uzeti u obzir se menja na početku same funkcije promenom vrednosti promenljive *prag*.

U dosadašnjim funkcijama (*Kalmanova_filtracija*,*GGP*,*D_fja*) se pojavljivala funkcija omega (*omega_kalman*,*omega_ggp*,*omega_dfja*) koju smo koristili za proračun težinske forme preko Huberove funkcije rezultata (*Huber's score function*). Ova funkcije u sva tri slučaja ima sličnu formu ali je bilo potrebno napisati tri različite verzije zbog različitih promenljivih koje u njoj figurišu.

Primer težinske forme računate kod Kalmanove filtracije je:

```
if v(k,:)~=0
    arg=v(k,:)/s(k,1);
    psi=min(abs(arg),k)*sign(arg);
    omega_pom=psi/arg;
else
    omega_pom=1;
end
```

6. Glavni program

Glavni program je mesto na kome smo pozivali sve navedene funkcije. Kalmanova filtracija za odabrani signal je izvršena jednom pa je računanje promenljivog faktora zaboravljanja na dva načina rađeno za isti filtrirani signal. To je urađeno da bi se uporedila efikasnost ova dva načina računanja VFF-a.

U glavnom programu se vrši izbor test signala , odnosa između nominalnog i kontaminiranog Gausovskog šuma kao i izbor da li će se signal pojačavati i slabiti u pojedinim delovima koda da bi se rezultati "lepše" iscrtali i mogli bolje uporediti.

```
% Biram test signal (1 za prvi, 2 za drugi i 3 za treci test signal)
izbor=2;
% Definisem odnos nominalnog i kontaminiranog Gausovskog suma
odnos=0.1;
% Biram da li cu da ukljucujem pojacanja i slabljenja signala (1-ON,0-OFF)
dailine=1;
% Pozivam funkciju za Kalmanovu filtraciju
[p,Z,y,y_sum,z_hat,teta,epsilon,t,duz,omega,beli_sum]=Kalmanova_filtracija(izbor,odnos,dailine
);
% Pozivam funkciju za estimaciju preko diskriminacione funkcije
[teta_hat_1,z_novo_1,lambda_1]=D_fja(p,Z,z_hat,teta,epsilon,duz,omega);
% Pozivam funkciju za estimaciju preko generalisane greske predikcije
[teta_hat_2,z_novo_2,lambda_2]=GGP(p,Z,z_hat,teta,epsilon,duz,omega);
```

Promenom vrednosti promenljive *izbor* zapravo se vrši izbor signala za testiranje. Ova promenljiva može da uzme tri vrednosti kao što je to opisano u samom kodu. Što se tiče promenljive odnos, za nju je već rečeno da utvrđuje odnos nominalnog i kontaminiranog Gausovskog šuma u signalu. Ukoliko je odabrana vrednost kao na prikazanom delu algoritma, tada je 90% šuma nominalni Gausovski šum dok je ostalih 10% kontaminirani Gausovski šum.

Promenljiva *dailine* može i da se postavi na vrednost nula jer je ova promenljiva uvedena u algoritam samo zbog toga da bi se dobili pregledniji grafici za analizu rezultata ali ona nikako bitno ne utiče na rad samog algoritma za estimaciju parametara.

7. Analiza rezultata

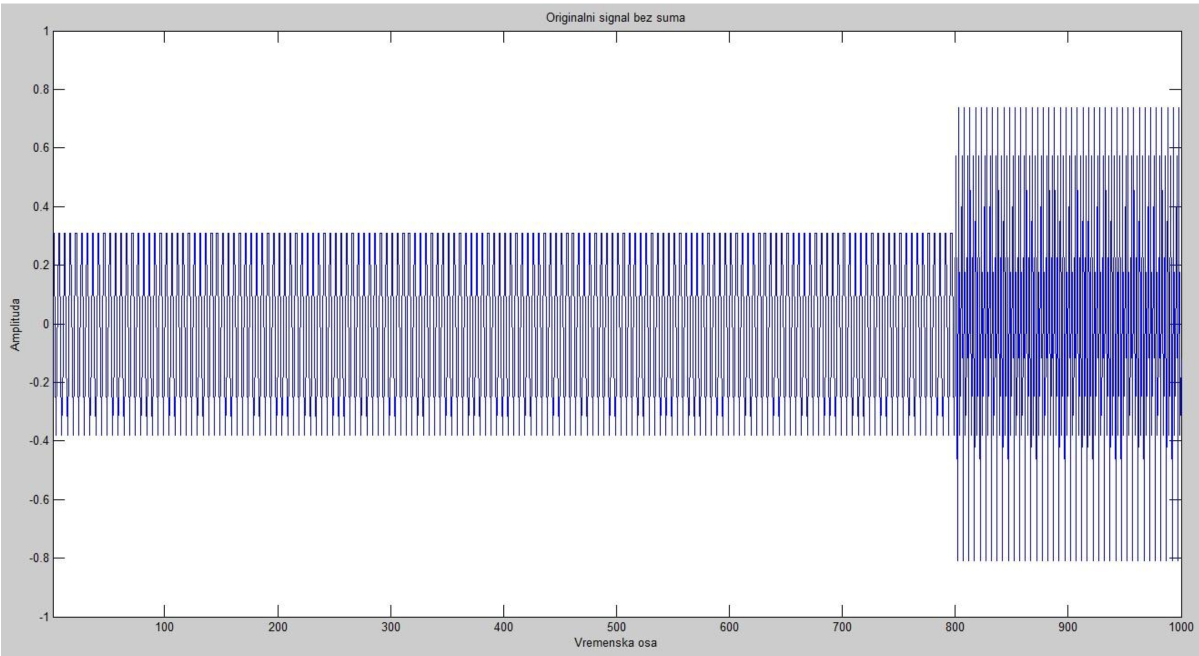
7.1. Kalmanova predfiltracija

Na samom početku analize prikazaćemo izgled prvog i drugog test signala sa i bez šuma koje ćemo kasnije koristiti u Kalmanovom predfiltru. Treći signal nije analiziran jer su parametri nepodesni. Nepodesni su jer se dobija signal koji ima amplitudu 10^150 pa se tako prilikom računa manji brojevi gube pa dobijamo deljenje sa nulom što rezultira u nepostojećoj estimaciji parametara.

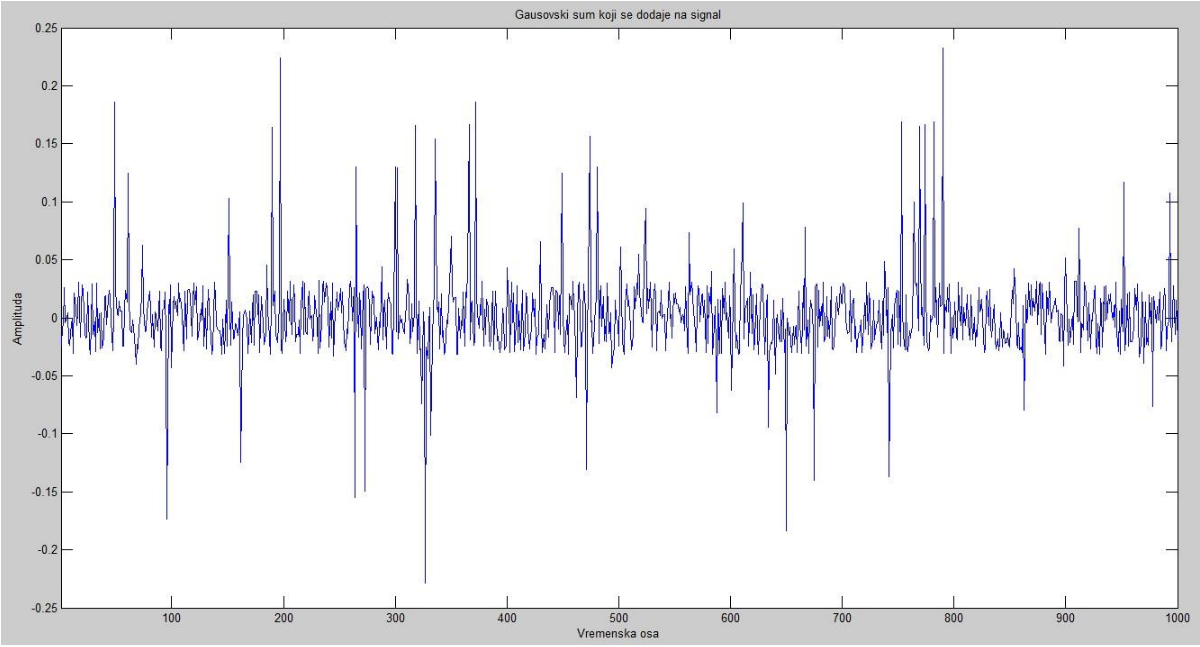
Za prikazani Gausovske šumove i rezultate Kalmanove filtracije usvojeno je da u belom Gausovskom šumu postoji 90% nominalnog i 10% kontaminiranog šuma, gde je srednja vrednost kontaminiranog šuma 0.1, a nominalnog 0. Varijansa kontaminiranog šuma je menjana od slučaja do slučaja kako bi se dobili bolji rezultati. Konkretno, za prvi signal je odabrana varijansa 0.7, dok je za drugi test signal birana varijansa kontaminiranog šuma 0.1.

Filtrirani signal na izlazu iz Kalmanovog filtra je onoliko oslabljen koliko je šum oslabljen (skaliran) da bi se rezultati mogli efikasnije (grafički) uporediti.

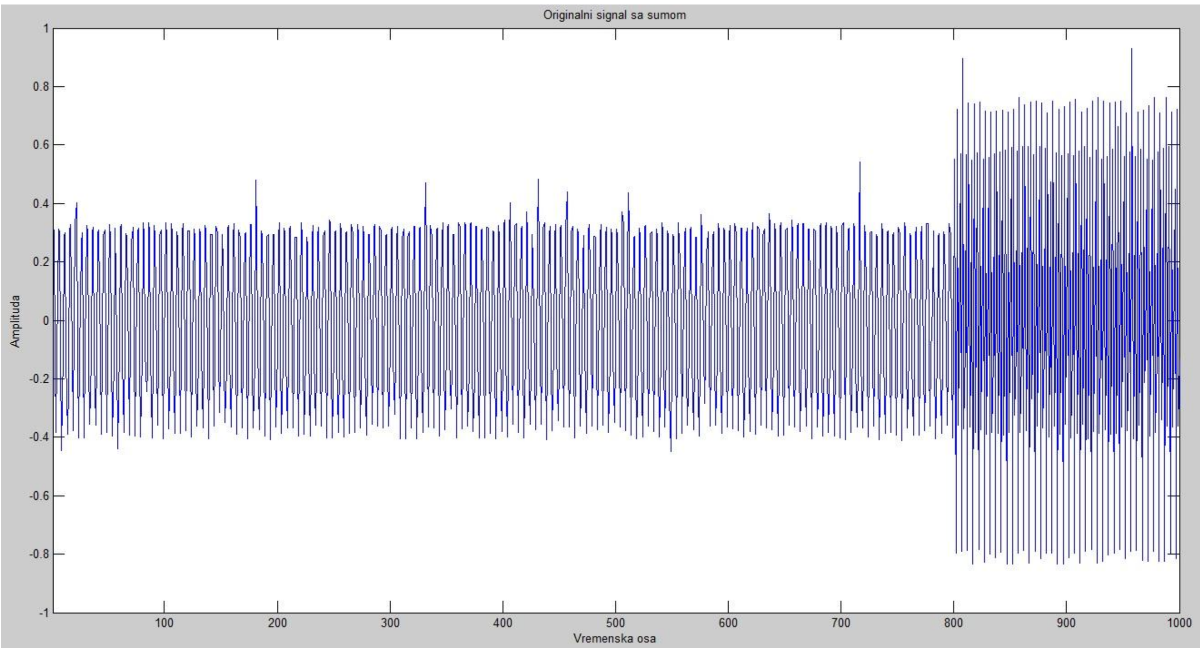
Za rezultantni Gausovski šum je upotrebljena funkcija skaliranja kako bi odnos signal/šum bio 10dB.



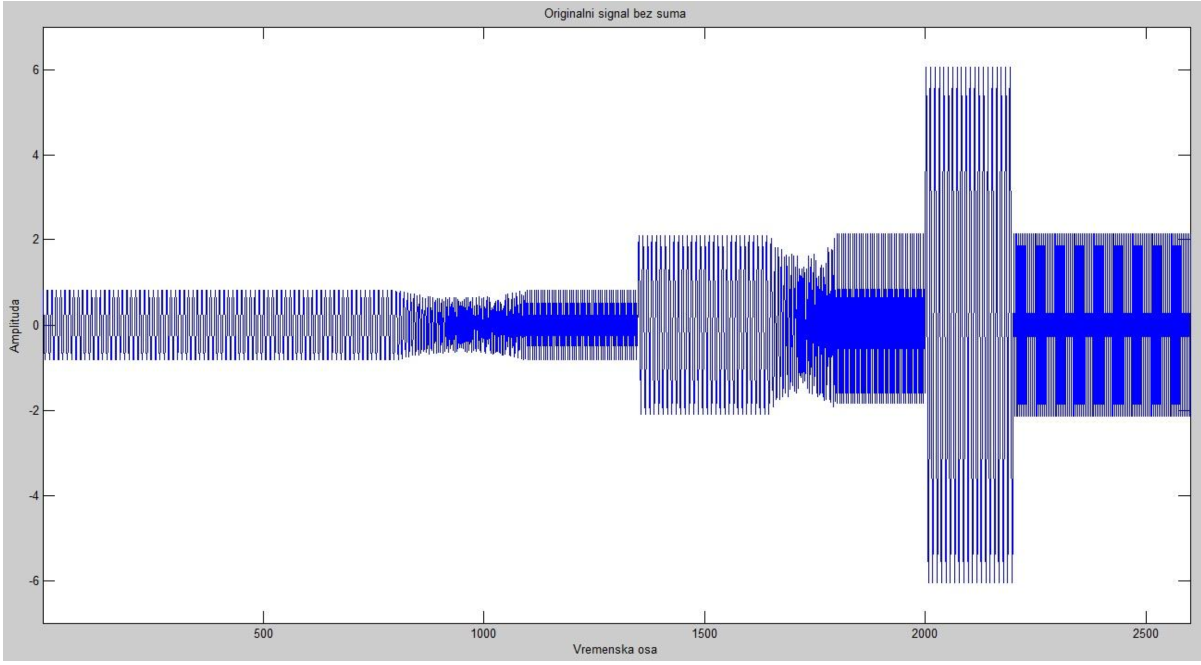
Slika 1: Originalni test signal 1 bez šuma.



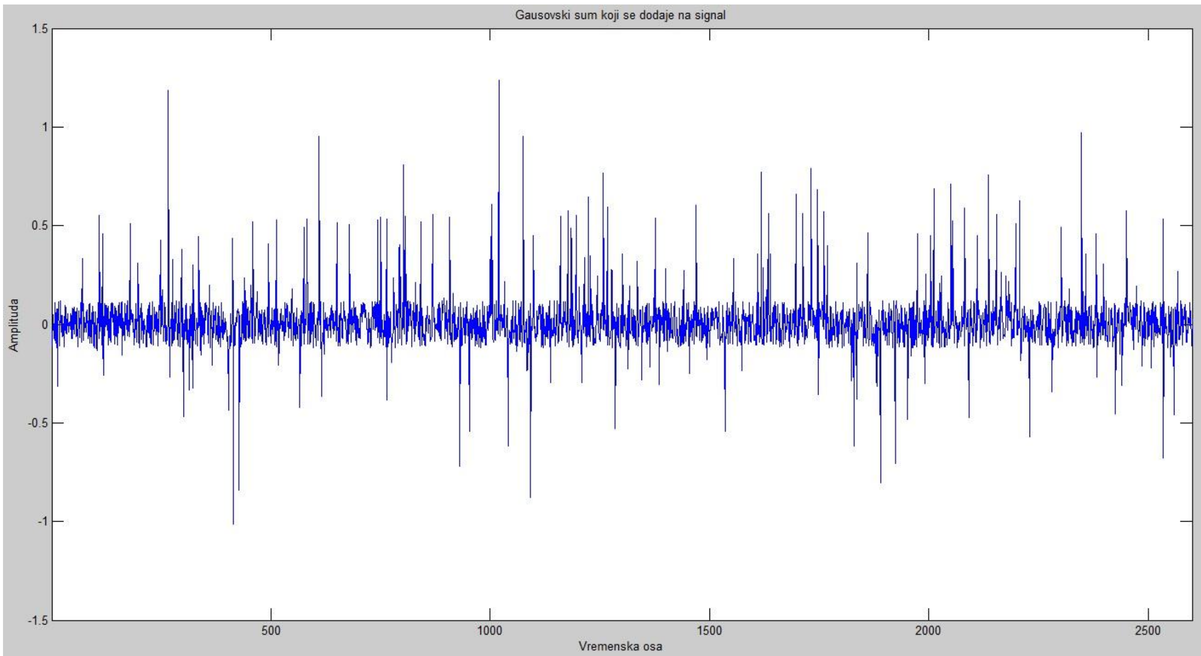
Slika 2: Gausovski šum (nominalni + kontaminirani) za prvi test signal.



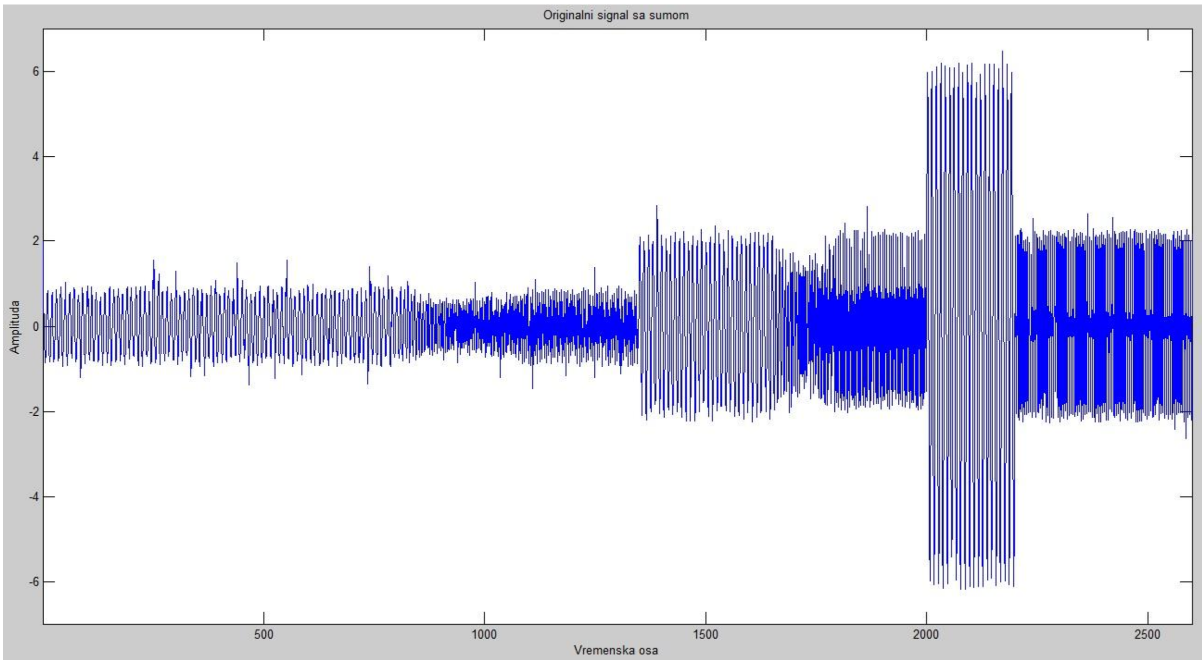
Slika 3: Prvi test signal sa Gausovskim šumom.



Slika 4: Originalni test signal 2 bez šuma.

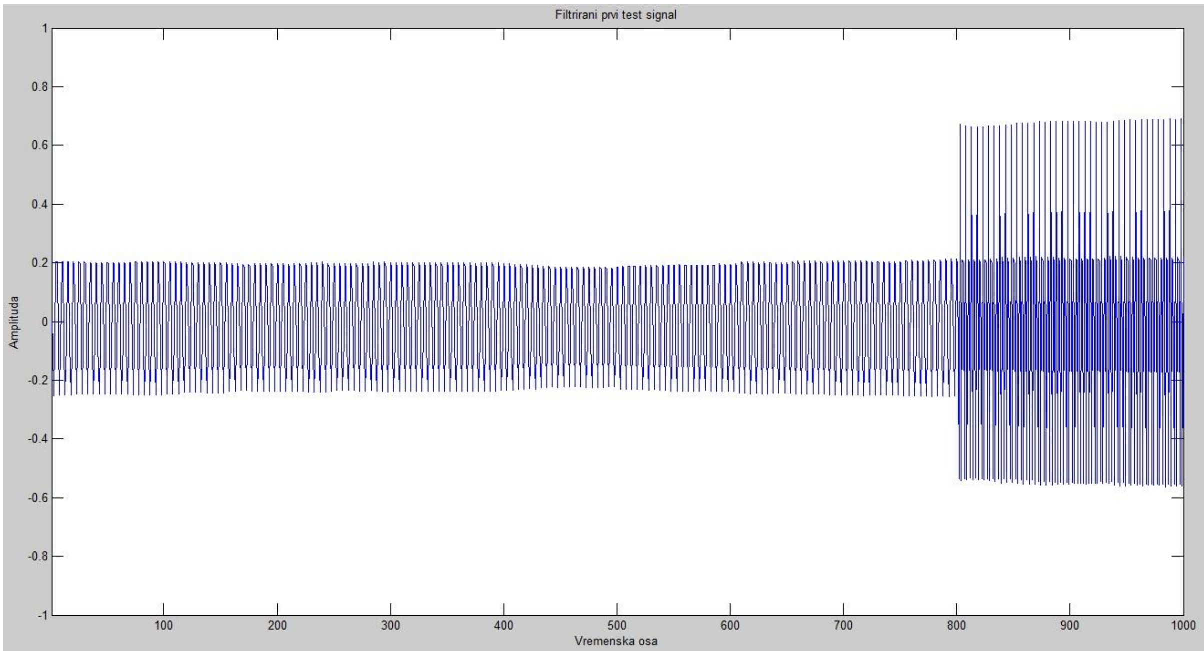


Slika 5: Gausovski šum (nominalni + kontaminirani) za drugi test signal.

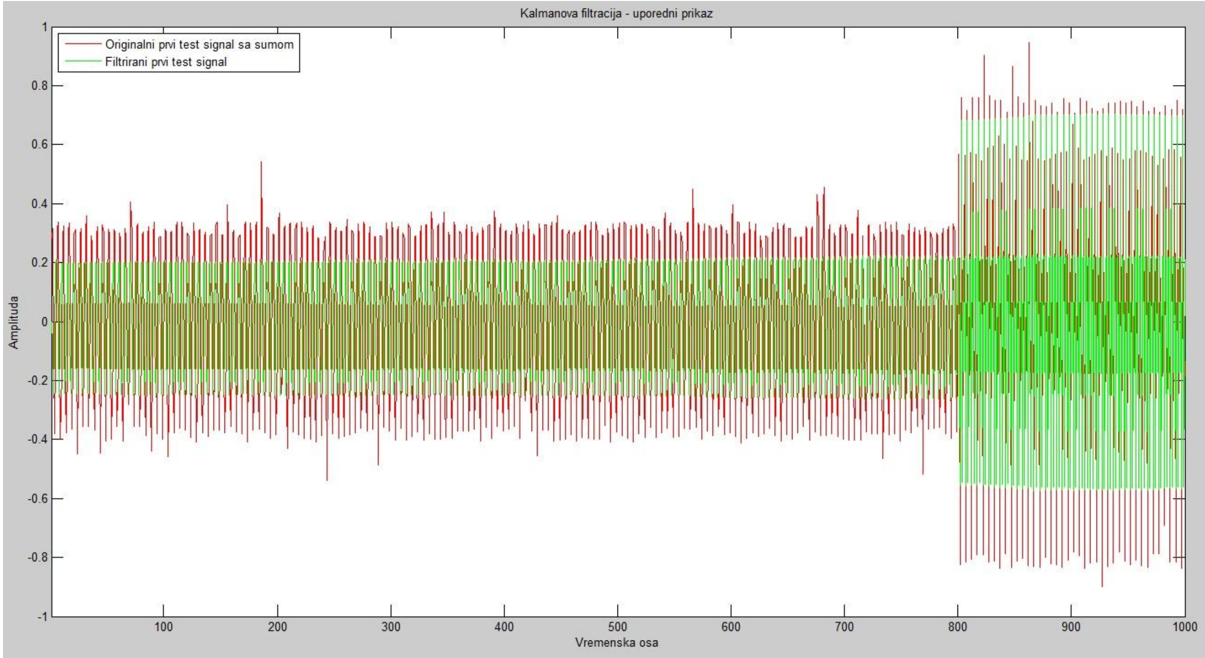


Slika 6: Drugi test signal sa Gausovskim šumom.

Kalmanova predfiltracija za prvi test signal je prikazana na slikama 7 i 8. Na slici 7 je prikazan izgled signala na izlazu iz Kalmanovog filtra dok slika 8 prikazuje praćenje filtriranog signala (promenu signala u kojoj se nalazi informacija) u odnosu na originalni signal sa šumom.

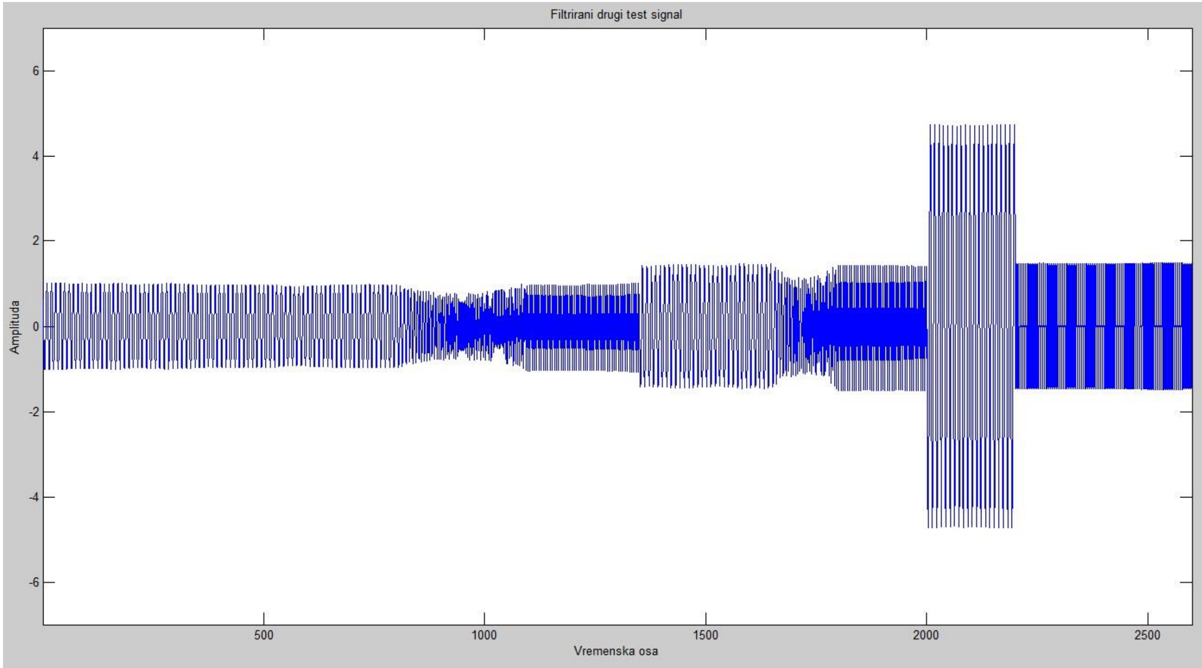


Slika 7: Izgled prvog test signala nakon Kalmanove filtracije.

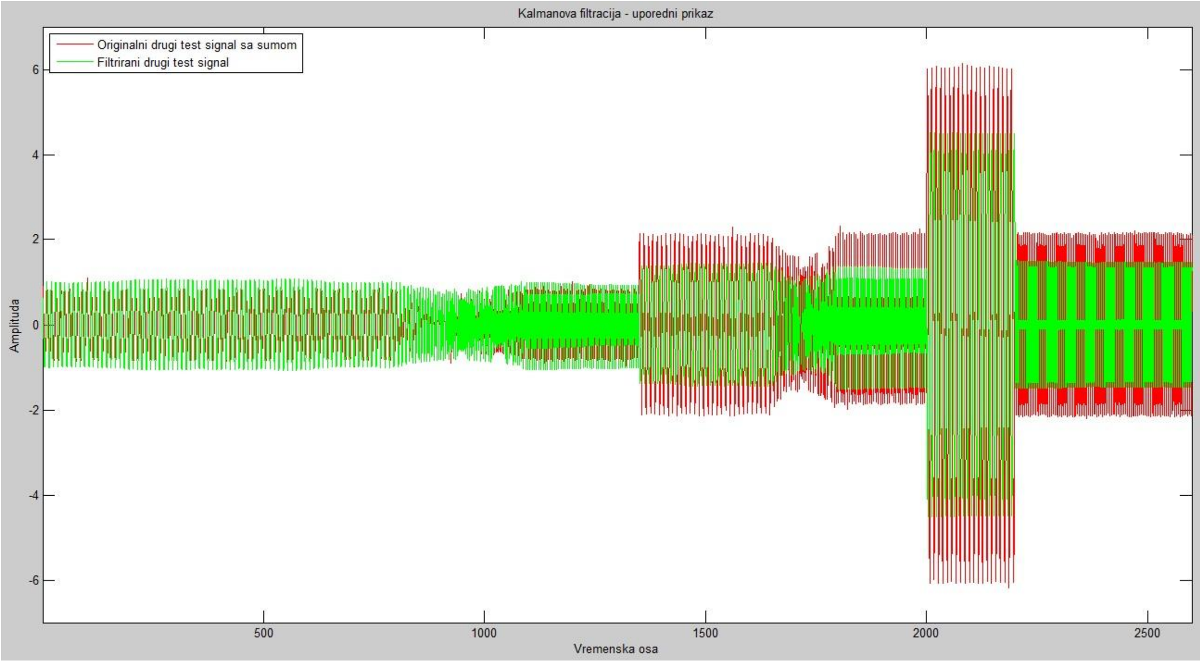


Slika 8: Filtrirani signal (zeleni) u odnosu na originalni signal sa šumom (crveni).

Isti grafici samo za drugi test signal su prikazani na sledećim slikama.



Slika 9: Izgled drugog test signala nakon Kalmanove filtracije.



Slika 10: Filtrirani signal (zeleni) u odnosu na originalni signal sa šumom (crveni).

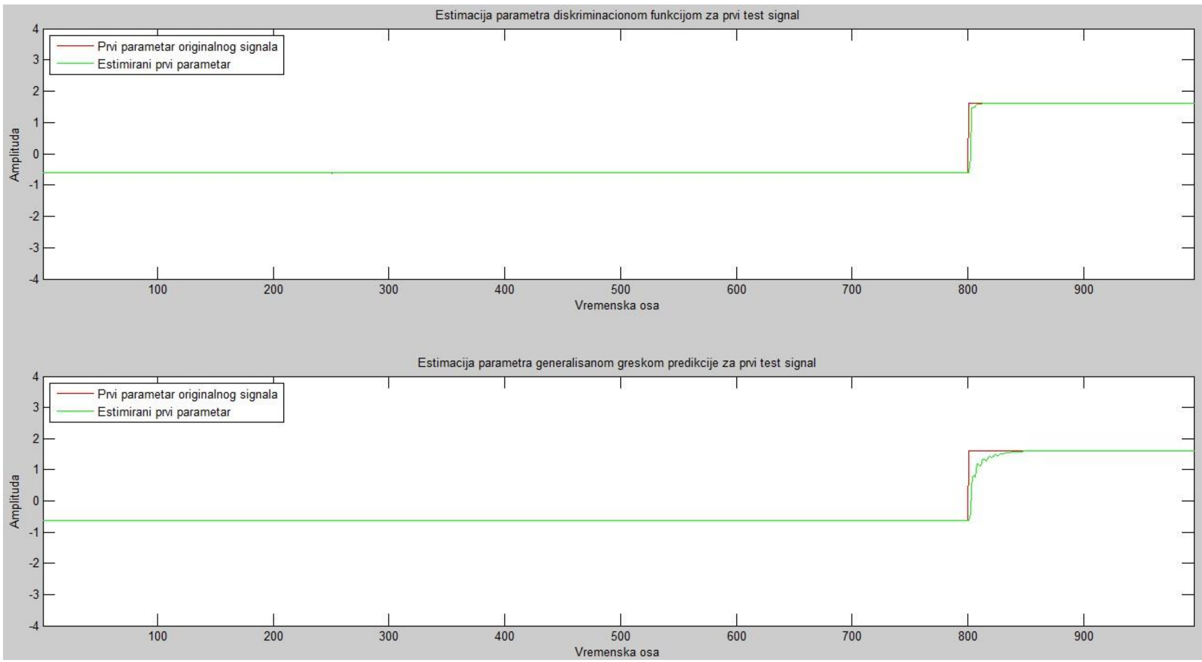
7.2. Promenljivi faktor zaboravljanja
- prvi test signal -

Kao što smo već napomenuli, estimacija parametara pomoću promenljivog faktora zaboravljanja diskriminacionom funkcijom i generalisanom greškom predikcije je rađena za isti filtrirani signal kako bi se mogla preciznije uporediti efikasnost pomenutih algoritama.

Kod oba analizirana test signala prvi parametar zavisi od vremena (nije konstanta) dok je drugi parametar konstantan (pošto su oba signala drugog reda imamo samo dva parametra koja treba estimirati).

Prvo ćemo analizirati prvi test signal, tj. analiziraćemo estimaciju oba parametra dobijenu generalisanom greškom predikcije i diskriminacionom funkcijom.

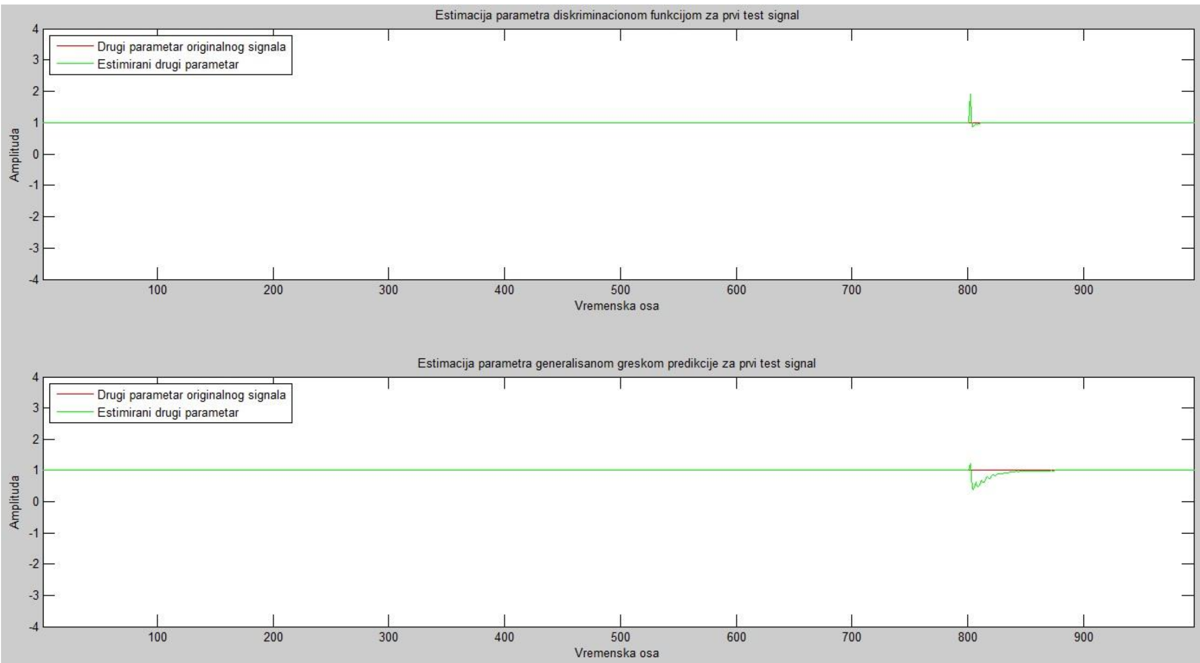
Dobijene estimacije ćemo prikazivati uporedo za oba pomenuta metoda računanja VFF-a. Sve slike su sačinjene od dva grafika pri čemu prvi predstavlja rezultate dobijene diskriminacionom funkcijom a drugi generalisanom greškom predikcije.



Slika 11: Estimacija prvog parametra na oba načina za prvi test signal.

Na slici 11 možemo primetiti da računanje promenljivog faktora zaboravljanja preko diskriminacione funkcije daje bolji rezultat od računanja istog preko generalisane greške predikcije. Test signal 1 ima samo jednu promenu frekvencije a samim tim i samo jednu promenu prvog parametra u vremenu pa tako on nije toliko interesantan za analizu.

Ipak, dužni smo prikazati i estimaciju drugog parametra za prvi test signal (slika 12).



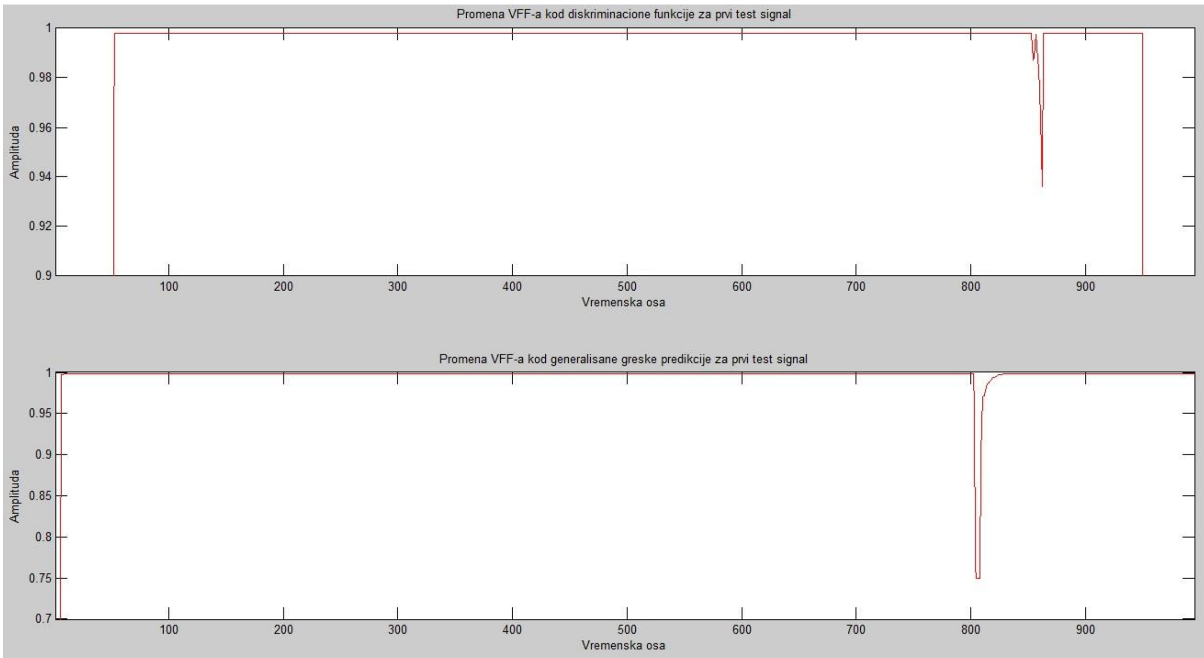
Slika 12: Estimacija drugog parametra na oba načina za prvi test signal.

Kao što je očekivano, i u slučaju estimacije drugog parametra diskriminaciona funkcija daje bolje rezultate.

U slučaju prvog test signala dobijamo iste rezultate (istu estimaciju) kada povećamo prisustvo kontaminiranog šuma u signalu. Estimacija je ista čak i kada je šum u potpunosti kontaminiran (100% kontaminiranog i 0% nominalnog šuma).

Iako se estimacija pogoršala neznatno kada je u ukupnom šumu prisutan i kontaminirani Gausovski šum i dalje se ipak može reći da je estimacija dobra.

Promena faktora zaboravljanja (lambda u algoritmu) za prvi test signal je prikazana na sledećem grafiku.



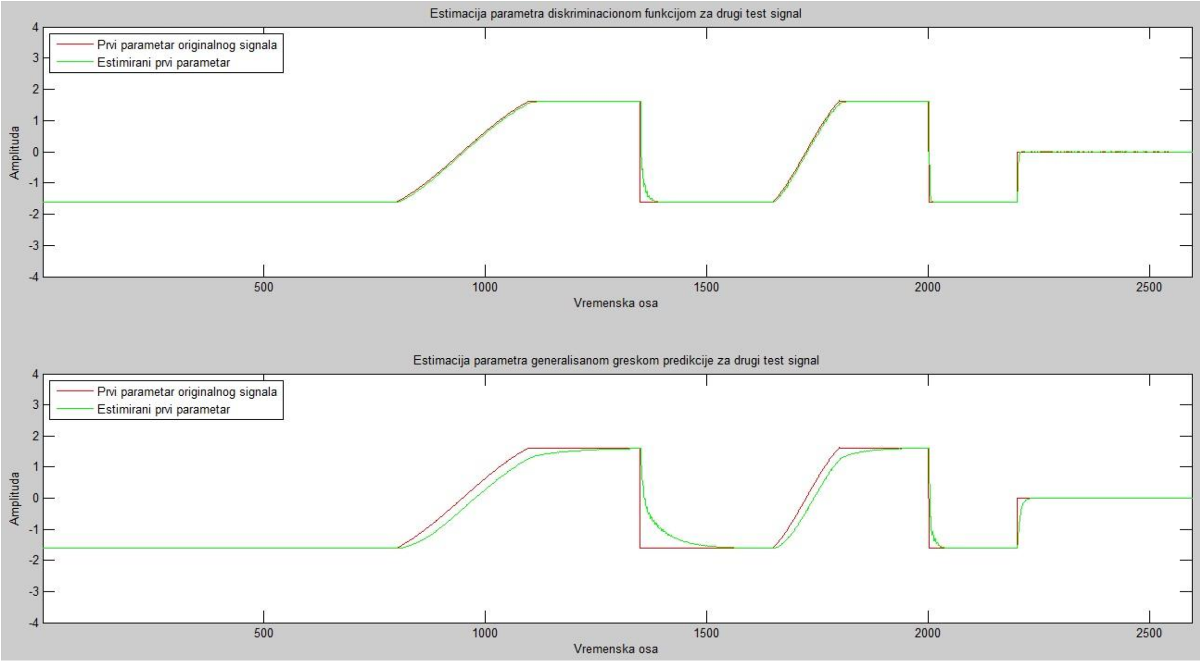
Slika 13: Promena VFF-a za diskriminacionu f-ju i gpp za prvi test signal.

Kod prikaza VFF-a za diskriminacionu funkciju primetno je da ne postoji vrednost za prvih i poslednjih 50 odbiraka. Do ovoga dolazi usled načina računanja VFF-a kod D funkcije jer je upravo tolika dužina prozora koji se koristi u algoritmu.

- drugi test signal -

Drugi signal je kompleksniji zbog više promena frekvencije pa samim tim i više promena prvog parametra AR modela signala.

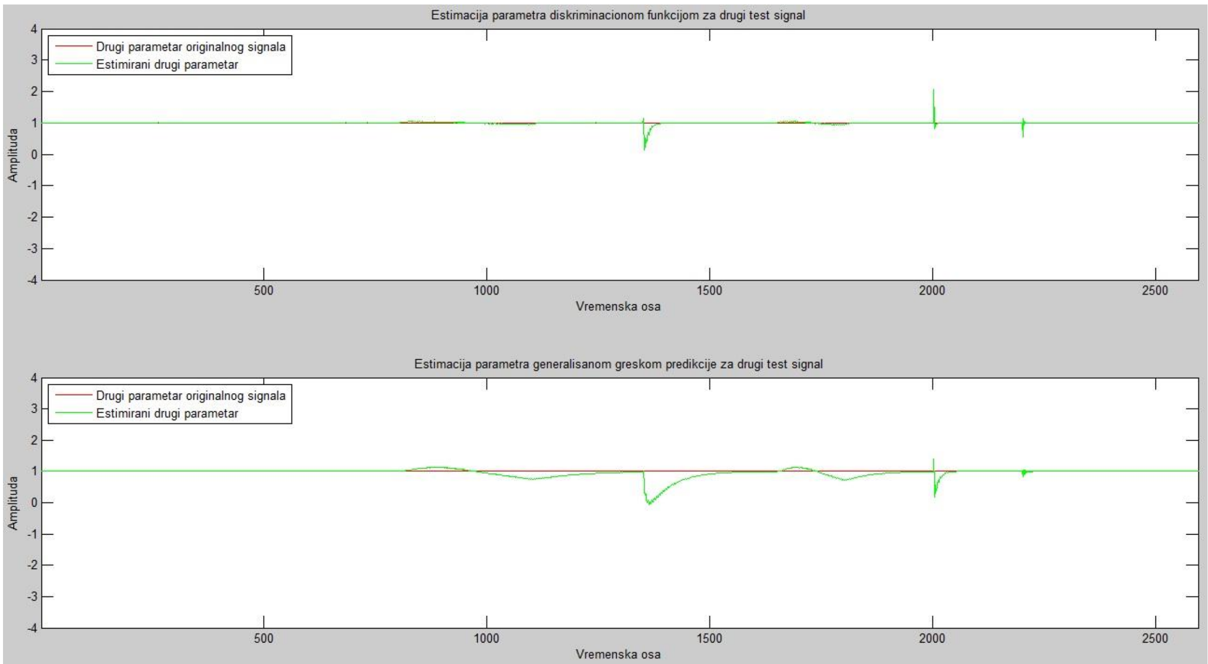
Sledeći grafici su rezultati dobijeni kada je Gausovski šum formiran od 90% nominalnog i 10% kontaminiranog Gausovskog šuma.



Slika 14: Estimacija prvog parametra na oba načina za drugi test signal.

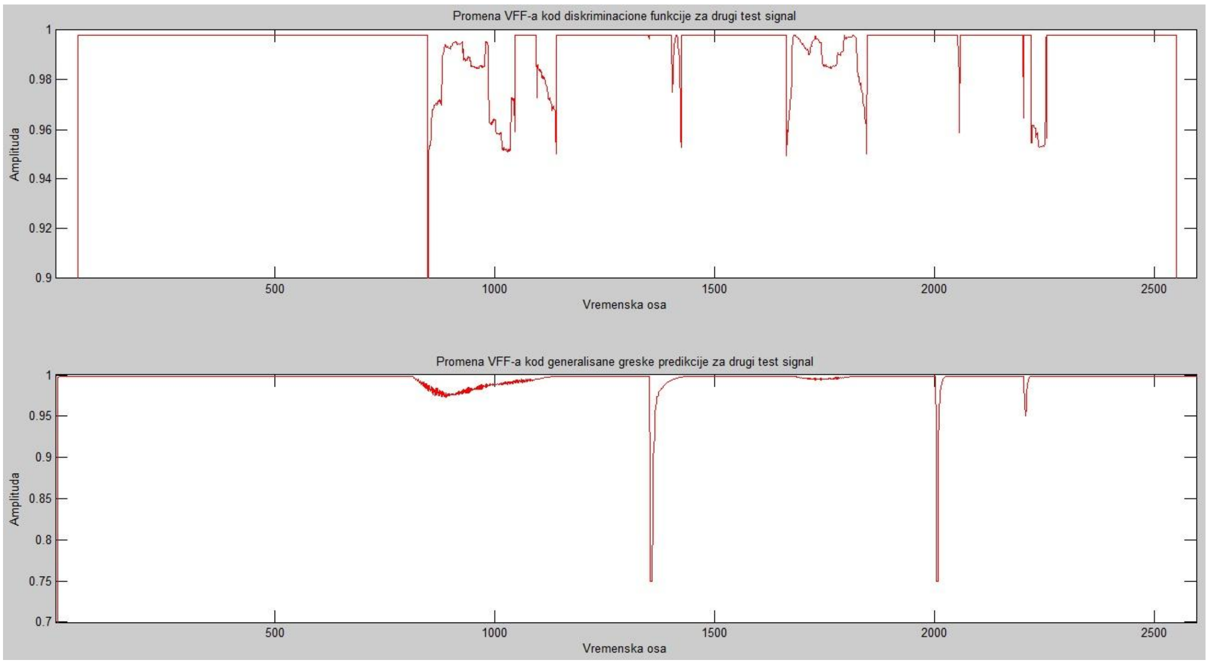
Estimacija prvog parametra AR modela za drugi test signal pokazuje jasnije razliku između efikasnosti dva analizirana algoritma. Kao što možemo videti sa slike diskriminaciona funkcija daje mnogo bolju estimaciju od estimacije generalisane greške predikcije.

Estimacija drugog parametra daje slične rezultate.



Slika 15: Estimacija drugog parametra na oba načina za drugi test signal.

Promene VFF-a za drugi test signal su prikazane na sledećoj slici.



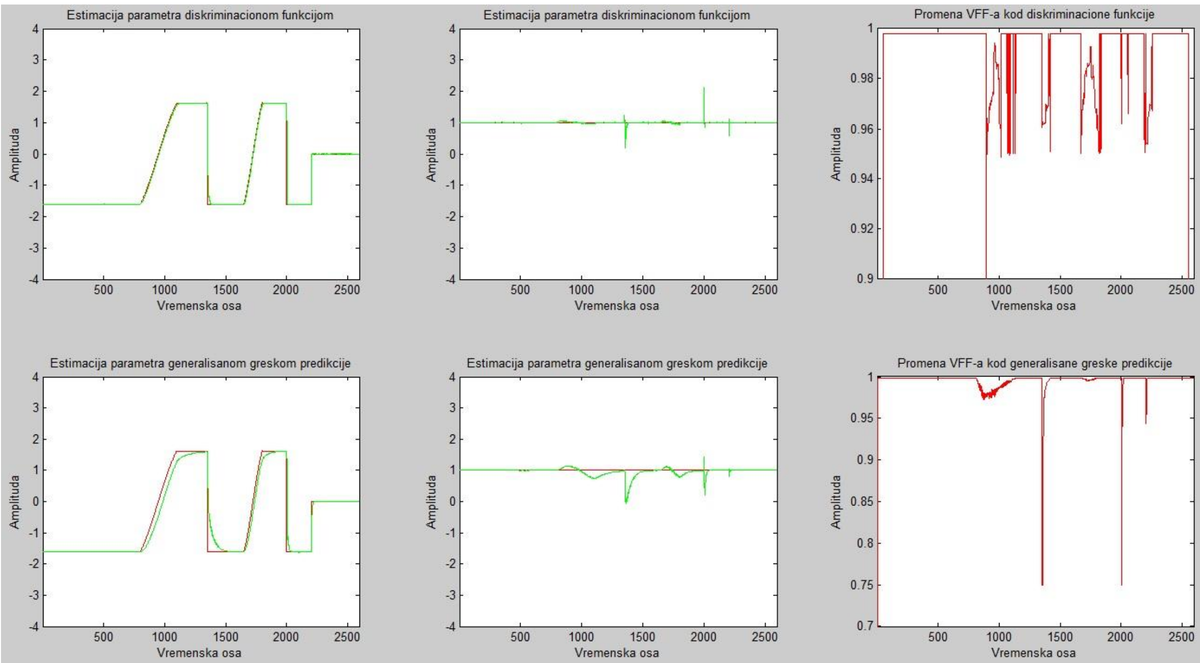
Slika 16: Promena VFF-a za diskriminacionu f-ju i gpp za drugi test signal.

7.3. Estimacija pod različitim uticajem šuma

U predloženom algoritmu postoji nekoliko vrednosti konstanti koje se mogu menjati pa time i uticati na kvalitet estimacije. Međutim, estimacija signala ipak najviše zavisi od uticaja šuma na signal. Logično je da se estimacija pogoršava kada je šum obilat naglim skokovima. Ovaj odeljak je posvećen dokazivanju ove tvrdnje.

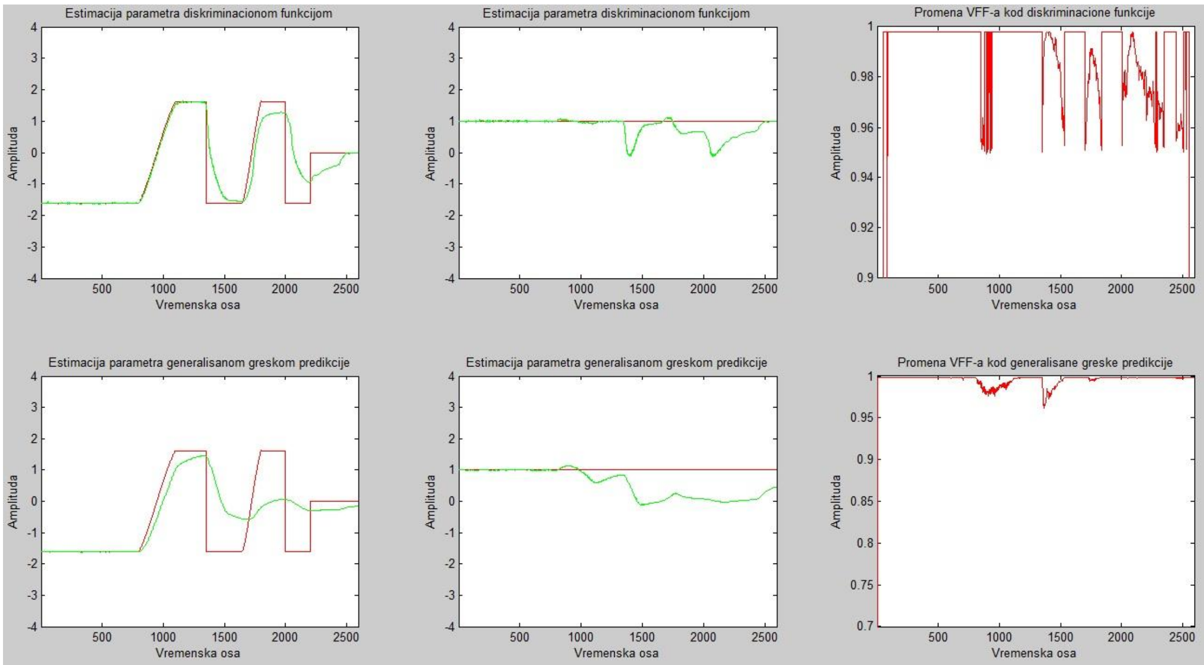
Posmatraćemo samo drugi test signal jer se na njemu „lepše“ vidi uticaj šuma na estimaciju. Pogledajmo prvo rezultate estimacije kada je Gausovski šum „sastavljen“ od 80% nominalnog i 20% kontaminiranog šuma. Vrednosti varijanse su iste kao i u prethodnom poglavlju, odnosno varijansa kontaminiranog šuma je 0.1.

Kako bi izlaganje bilo skladnije od sada ćemo rezultate estimacije za oba načina računanja VFF-a za oba parametra kao i promene VFF-a prikazivati na jednoj slici.



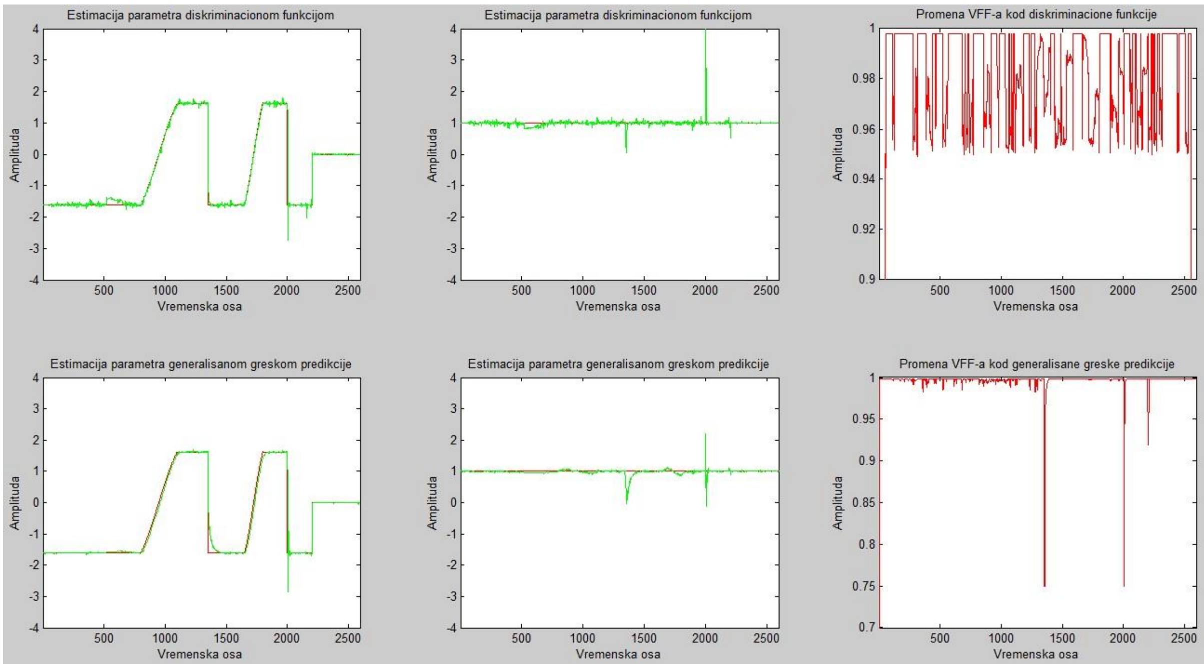
Slika 17: Estimacija parametara i promena VFF-a za 20% kontaminiranog šuma.

Sa slike 17 vidimo da se veoma slični rezultati dobijaju i za 20% kontaminiranog Gausovskog šuma. U stvari, rezultati su podjednako dobri dok god kontaminirani Gausovski šum ne čini barem 40% ukupnog šuma! U tom slučaju se dobija nekvalitetna estimacija (slika 18).



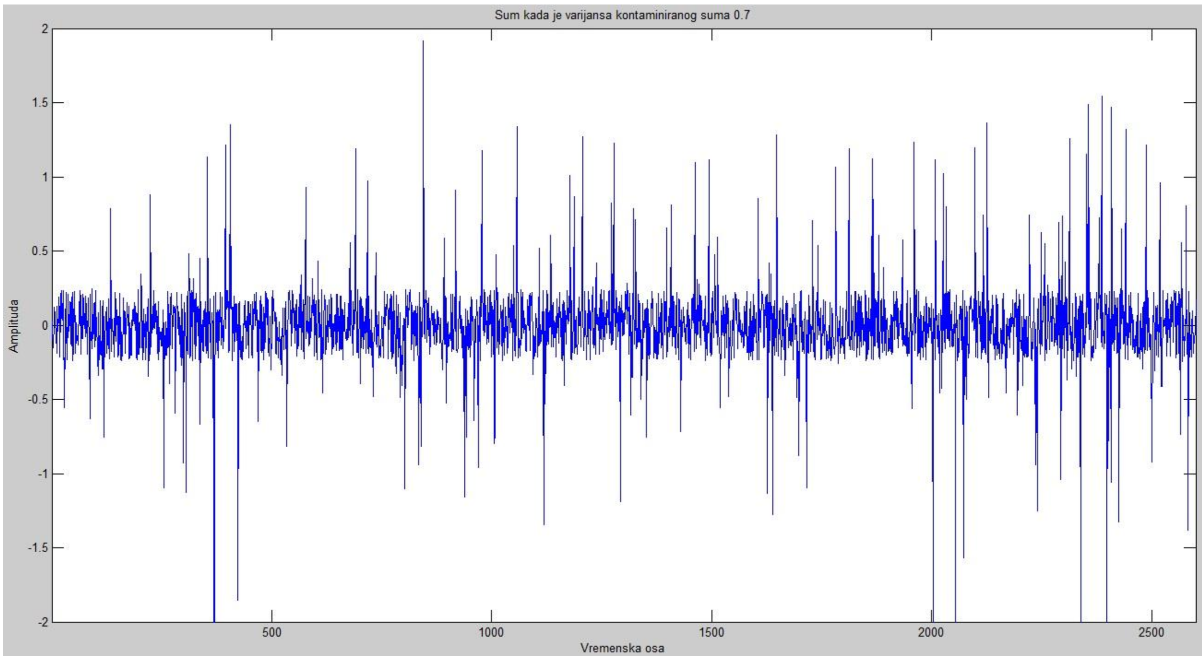
Slika 18: Estimacija parametara i promena VFF-a za 40% kontaminiranog šuma.

Pogledajmo šta se dešava kada je u ukupnom šumu prisutno svega 10% kontaminiranog šuma ali je varijansa kontaminiranog šuma sada 0.7, za razliku od 0.1 koliko je bilo u dosadašnjim testovima.



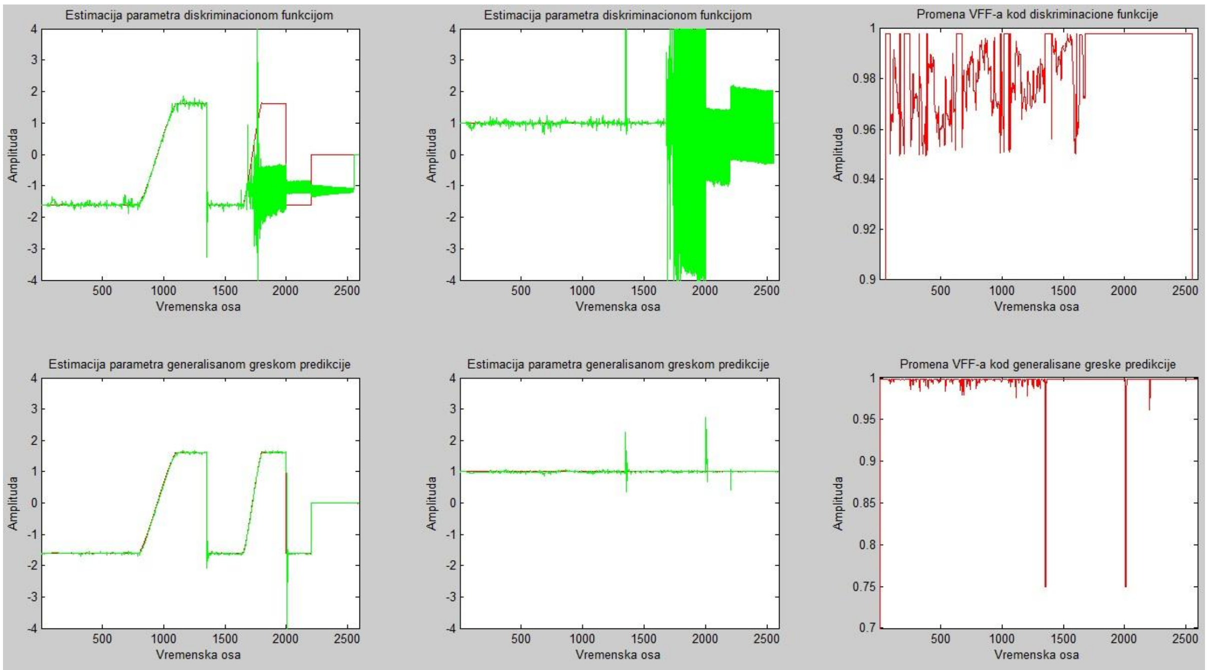
Slika 19: Estimacija parametara i promena VFF-a za 10% kontaminiranog šuma kada je varijansa kontaminiranog šuma 0.7.

Na slici 19 se jasno vidi da su se rezultati iskvarili kada je varijansa kontaminiranog šuma 0.7. Dobijeni rezultat je logičan obzirom da su sada amplitude pikova rezultantnog Gausovskog šuma veće (slika 20).



Slika 20: Izgled Gausovskog šuma kada je varijansa kontaminiranog šuma 0.7.

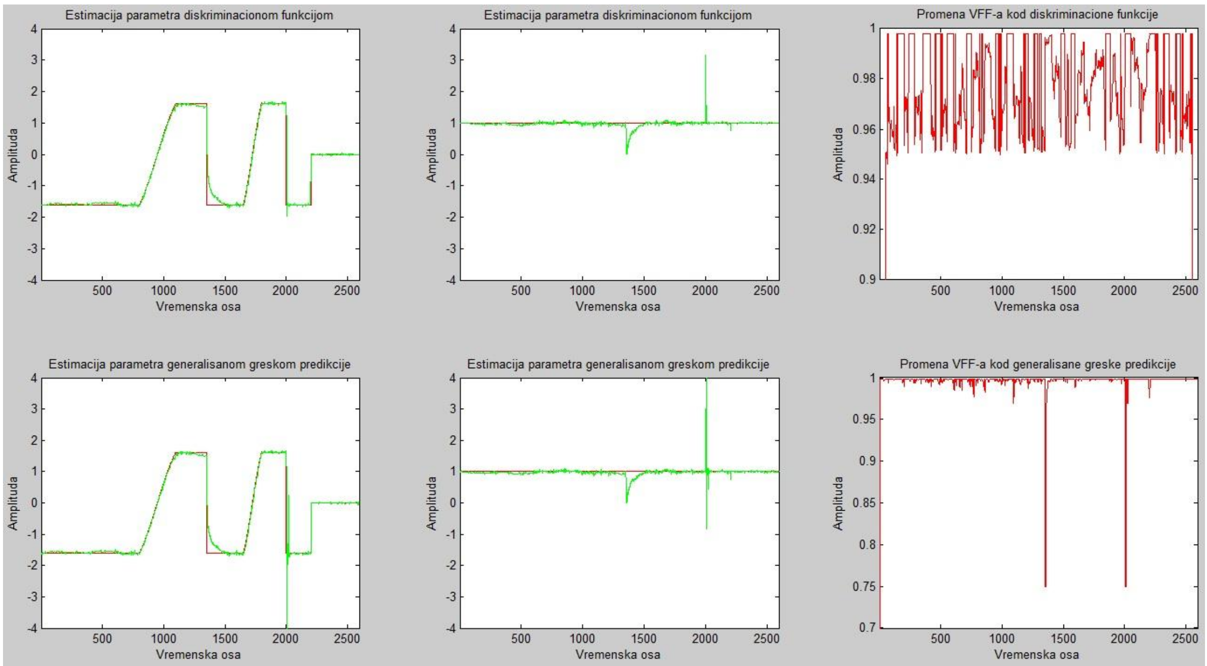
Pri velikim vrednostima varijanse kontaminiranog šuma dovoljno je da kontaminirani šum čini svega 20% od ukupnog Gausovskog šuma pa da estimacija bude potpuno neupotrebljiva (slika 21).



Slika 21: Neispravna estimacija za kada je varijansa kontaminiranog šuma 0.7
i kada kontaminirani šum čini 20% od ukupnog Gausovskog šuma.

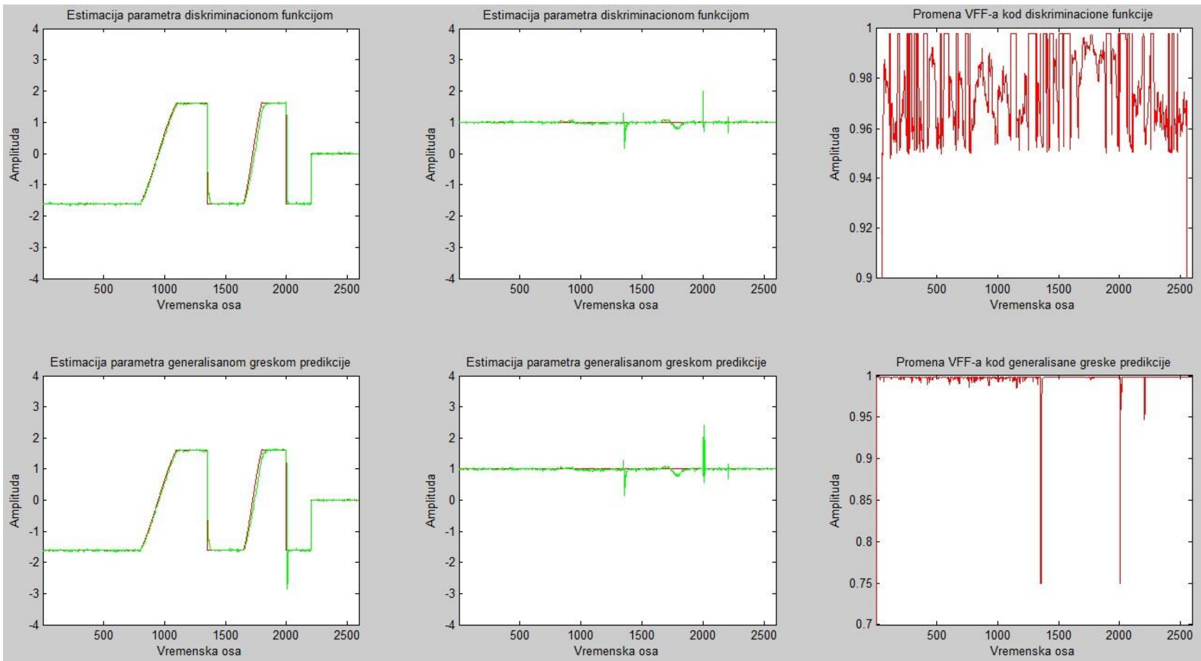
Ipak sa slike 21 možemo videti da je estimacija pomoću generalisane greške predikcije skoro ispravna. To nam govori da i pored ovakvog uticaja šuma estimacija može biti ispravna ukoliko se parametri podese drugačije.

Pogledajmo estimaciju za iste vrednosti Gausovskog šuma. Jedina razlika je promena konstante C u algoritmu za računanje VFF-a preko diskriminacione funkcije za koju mesto vrednosti 1 usvajamo vrednost 0.11.



Slika 22: Popravljena estimacija promenom vrednosti konstante C=1 na C=0.11.

Optimalnu estimaciju za opisani Gausov šum nalazimo za C=0.5 (slika 23).



Slika 23: Optimalna estimacija promenom vrednosti konstante C=0.11 na C=0.5.

Dakle, analizom test signala 2 pokazali smo da je algoritam zaista efikasan kada je signal pod uticajem šuma obilatog pikovima. Pisanjem programa za ovaj algoritam, takvog da je na ulaz sistema moguće dovesti signal bilo kog reda i sa bilo kojim brojem promena ili amplitudom, stvorili smo efikasnu alatku koja je od velike pomoći za dalje testiranje efikasnosti algoritma pod različitim uticajima šuma i za različite vrednosti konstanti sa različitim signalima.

Na žalost, nismo imali ispravan model test signala reda govornog signala (10 red) pa da testiramo algoritam i u ovom slučaju. Ipak, efikasnost koda je proverena, tako da sa sigurnošću možemo reći da će program pravilno testirati i govorni signal ukoliko je model (parametri AR modela) ispravan.

Kompletan kod algoritma se nalazi u dodatku ovog teksta.

8. Dodatak

8.1. funkcija *glavni program*

```
% Biram test signal (1 za prvi, 2 za drugi i 3 za treci test signal)
izbor=2;
% Definisem odnos nominalnog i kontaminiranog Gausovskog suma
odnos=0.2;
% Biram da li cu da ukljucujem pojacanja i slabljenja signala (1-ON,0-OFF)
dailine=0;
% Definisem pomocne konstante koje uticu na estimaciju
C1=0.5; % konstanta za diskriminacionu funkciju
C2=0.5; % konstanta za generalisanu gresku predikcije
% Pozivam funkciju za Kalmanovu filtraciju
[p,Z,y,y_sum,z_hat,teta,epsilon,t,duz,omega,beli_sum]=Kalmanova_filtracija(izbor,odnos,dailine
);
% Pozivam funkciju za estimaciju preko diskriminacione funkcije
[teta_hat_1,z_novo_1,lambda_1]=D_fja(p,Z,z_hat,teta,epsilon,duz,omega,C1);
% Pozivam funkciju za estimaciju preko generalisane greske predikcije
[teta_hat_2,z_novo_2,lambda_2]=GGP(p,Z,z_hat,teta,epsilon,duz,omega,C2);
```

8.2. funkcija *Kalmanova_filtracija*

```
function
[p,Z,y,y_sum,z_hat,teta,epsilon,t,duz,omega,beli_sum]=Kalmanova_filtracija(izbor,odnos,dailine
)
% Robustifikovani Kalmanov filtar

%% Generisem test signal drugog reda i parametre tog signala
if izbor==1
    % Test signal 1
    [duz,t,p,teta,y]=test_signal_1();
elseif izbor==2
    % Test signal 2
    [duz,t,p,teta,y]=test_signal_2();
elseif izbor==3
    % Test signal 3
    [duz,t,p,teta,y]=test_signal_3();
end

% Racunam sum koji utice na signal
[beli_sum,varijansa_suma]=gausov_sum(duz,odnos);
% Skaliram sum u odnosu da signal da SNR bude 10dB
[beli_sum,faktor]=skaliranje(beli_sum,y,duz);
% Dodajem beli Gausovski sum na signal
y_sum=y+beli_sum;

% Definisem Z izvedeno iz AR modela
Z=zeros(p,duz); % definisem unapred zbog brze alokacije memorije
for i=(p+1):duz
    for j=1:p
        Z(j,i)=-y_sum(i-j,1);
    end
end

% Definisem gresku predikcije (rezidual merenja)
epsilon=zeros(duz,1);
for i=1:duz
    epsilon(i,1)=y_sum(i,1)-Z(:,i)'*teta(:,i);
end

% Inicijalizujem matrice jednacina stanja sistema
F=zeros(p,p,duz);
for i=1:duz
    for j=1:p
        prva_vrsta(i,j)=-teta(j,i);
    end
    dodajem=horzcat(eye(p-1),zeros((p-1),1));
    F(:,:,i)=vertcat(prva_vrsta(i,:),dodajem);
end
G=zeros(p,1);
G(1,1)=1;
% sum AR modela
```

```
u(1,t)=beli_sum;
% ulazni sum
v=(G*u)';
% varijansa merenog suma
sigma_u=varijansa_suma;
% kovarijansa merenog suma
Q=sigma_u^2*G*(G');
R=0;
H=zeros(1,p);
H(1,1)=1;
% pocetna vrednost ulaza modela stanja
xo_hat=ones(p,1);
x_hat(:,:,1)=xo_hat;
% Pocetna vrednost kovarijanse greske estimacije
P=zeros(p,p,1);

%% Algoritam Kalmanove filtracije
omega=zeros(duz,1);
x_line=zeros(p,1,duz);
M=zeros(p,p,duz);
s=zeros(duz,1);
K=zeros(p,1,duz);
z_hat=zeros(duz,1);
z=y_sum;

for k=2:duz
% ********** AZURIRANJE VREMENA **********
    % Predikcija stanja sistema
    x_line(:,:,k)=F(:,:,k-1)*x_hat(:,:,k-1);
    % Matrica kovarijanse greske predikcije
    M(:,:,k)=F(:,:,k-1)*P(:,:,k-1)*(F(:,:,k-1)')+Q;
    % Predikcija izlaza sistema
% ********** AZURIRANJE MERENJA **********
    % Matrica kovarijanse reziduala
    s(k,1)=sqrt(H*M(:,:,k)*(H')+R);
    % Tezinska forma
    [omega_pom]=omega_kalman(k,v,s);
    omega(k,1)=omega_pom;
    % Matrica Kalmanovog pojacanja
    K(:,:,k)=omega(k,1)*M(:,:,k)*(H')*(s(k,1)^(-1));
    % Matrica kovarijanse greske estimacije
    P(:,:,k)=(eye(p)-K(:,:,k)*H)*M(:,:,k);
    % Procena stanja
    x_hat(:,:,k)=x_line(:,:,k)+K(:,:,k)*v(k,1);
    % Sum ulaza sistema
    v(k,:)=z(k,:)-H*x_hat(:,:,k);
    % Predikcija izlaza sistema
    z_hat(k,1)=H*x_hat(:,:,k);
end
% Slabljenje filtriranog signal (ako je tako odabrano u glavnom programu)
if dailine==1
    z_hat=z_hat*faktor;
end
end
```

8.3. funkcija *test_signal_1*

```
%% Generisem test signal 1 i parametre tog signala
function [duz,t,p,teta,y]=test_signal_1()
% Definisanje parametara AR modela signala
f(1,1:800)=0.2;
f(1,801:1000)=0.4;
duz=1000;                      % duzina signala
t=1:duz;                       % vremenski interval
arg(1,t)=2*pi*f(1,t);
p=2;                           % red AR modela signala
teta_1(1,t)=-2*cos(arg(1,t)); % prvi parametara AR modela
teta_2=1;                      % drugi parametar AR modela
teta=zeros(p,duz);             % alokacija memorije
for i=1:duz
    teta(:,i)=[teta_1(1,i); teta_2];
end
```

```
% Definisem pocetne uslove AR modela (y(k-1) i y(k-2))
y(1,1:2)=cos(arg(1,1:2));

% Definisem AR model signala preko parametara AR modela
for i=3:duz % prve dve vrednosti su pocetne vrednosti
    y(1,i)=-teta_1(1,i)*y(1,i-1)-teta_2*y(1,i-2);
end
% Transponujem matricu AR modela da bi dobio odgovarajuce dimenzije matrice
y=y';
end
```

8.4. funkcija *test_signal_2*

```
%% Generisem test signal 2 i parametre tog signala
function [duz,t,p,teta,y]=test_signal_2()
duz=2600; % duzina signala
t=1:duz; % vremenski interval
f=zeros(1,duz);

f(1,1:800)=0.1;
for i=800:1100
    f(1,i)=f(1,i-1)+0.001;
end
f(1,1101:1350)=0.4;
f(1,1350:1650)=0.1;
for i=1650:1800
    f(1,i)=f(1,i-1)+0.002;
end
f(1,1801:2000)=0.4;
f(1,2001:2200)=0.1;
f(1,2201:2600)=0.25;

arg(1,t)=2*pi*f(1,t);

% Definisem parametre teta za test signal 2
teta_1=zeros(1,duz);
for i=1:duz
    teta_1(1,i)=-2*cos(arg(1,i)); % prvi parametar AR modela
end
teta_2=1;                         % drugi parametar AR modela

% Red AR modela signala
p=2;

% Spajam oba parametra (teta_1 i teta_2) u jednu matricu
teta=zeros(p,duz);
for i=1:duz
    teta(:,i)=[teta_1(1,i); teta_2];
end

% Definisem pocetne uslove AR modela (y(k-1) i y(k-2))
y(1,1:p)=cos(arg(1,1:p));

% Definisem AR model signala preko parametara AR modela
for i=(p+1):duz % prve dve vrednosti su pocetne vrednosti
    y(1,i)=-teta_1(1,i)*y(1,i-1)-teta_2*y(1,i-2);
end

% Transponujem matricu AR modela da bi dobio odgovarajuce dimenzije matrice
y=y';
end
```

8.5. funkcija *test_signal_3*

```
%% Generisem test signal 3 (signal osmog reda) i parametre tog signala
function [duz,t,p,teta,y]=test_signal_3()
duz=2600; % duzina signala
t=1:duz; % vremenski interval
f=zeros(1,duz);

f(1,1:800)=0.1;
for i=800:1100
    f(1,i)=f(1,i-1)+0.001;
end
f(1,1101:1350)=0.4;
f(1,1350:1650)=0.1;
for i=1650:1800
    f(1,i)=f(1,i-1)+0.002;
end
f(1,1801:2000)=0.4;
f(1,2001:2200)=0.1;
f(1,2201:2600)=0.25;

arg(1,t)=2*pi*f(1,t);
% Red AR modela signala
p=8;

% Definisem parametre teta za test signal 3
teta_1=zeros(1,duz);
for i=1:duz
    teta_1(1,i)=-2*cos(arg(1,i)); % prvi parametar AR modela
end
teta_2=0.24;
teta_3=0.22;
teta_4=0.17;
teta_5=-0.2;
teta_6=0.15;
teta_7=0.25;
teta_8=0.1;

% Spajam sve parametre u jednu matricu (teta)
teta=zeros(p,duz);
for i=1:duz
    teta(:,i)=[teta_1(1,i); teta_2; teta_3; teta_4;...
               teta_5; teta_6; teta_7; teta_8];
end

% Definisem pocetne uslove AR modela (y(k-1) i y(k-2)... y(k-8))
y(1,1:p)=-2*cos(arg(1,1:p));

% Definisem AR model signala preko parametara AR modela
for i=(p+1):duz % prve dve vrednosti su pocetne vrednosti
    y(1,i)=-teta_1(1,i)*y(1,i-1)-teta_2*y(1,i-2)-teta_3*y(1,i-3)...
           -teta_4*y(1,i-4)-teta_5*y(1,i-5)-teta_6*y(1,i-6)-teta_7*y(1,i-7)...
           -teta_8*y(1,i-8);
end

% Transponujem matricu AR modela da bi dobio odgovarajuce dimenzije matrice
y=y';
end
```

8.6. funkcija *gausov_sum*

```
function [beli_sum,varijansa_suma]=gausov_sum(duz,odnos)
% Funkcija za odredjivanje ukupnog suma koji deluje na signal

% Definisem kontaminirani, beli Gausovski sum
sr_vr_kont_suma=0.1;
sigma_0=sqrt(0.7);
varijansa_kont_suma=sigma_0;
kont_beli_sum=sr_vr_kont_suma+varijansa_kont_suma.*randn(duz,1);
% Definisem nominalni, beli Gausovski sum
sr_vr_nom_suma=0;
sigma_1=sigma_0/sqrt(10);
```

```
varijansa_nom_suma=sigma_1;
nom_beli_sum=sr_vr_nom_suma+varijansa_nom_suma.*(2*(rand(duz,1)-0.5));
% Kombinujem ova dva suma u zavisnosti od izabranog odnosa
beli_sum=zeros(duz,1);
for i=1:duz
    nasum_br=rand(1);
    if nasum_br<=(1-odnos)
        beli_sum(i,1)=nom_beli_sum(i,1);
    else
        beli_sum(i,1)=kont_beli_sum(i,1);
    end
end
varijansa_suma=var(beli_sum);
end
```

8.7. funkcija *skaliranje*

```
% Skaliranje amplitude gausovog belog suma
function [beli_sum,faktor]=skaliranje(beli_sum,y,duz)
max_amp_sig=max(y); % maksimalna amplituda signala
max_amp_sum=max(beli_sum); % maksimalna amplituda suma
dozv_max_amp_sum=max_amp_sig/(sqrt(10)); % dozvoljeni maksimum za sum
% ako je maksimalna amplituda suma veca od dozvoljene
if max_amp_sum>dozv_max_amp_sum
    faktor=dozv_max_amp_sum/max_amp_sum;
    for i=1:duz
        % umanjujem svaki odbirak suma za vrednost izracunatog faktora
        beli_sum(i,1)=beli_sum(i,1)*faktor;
    end
% ako je maksimalna amplituda suma manja od dozvoljene
elseif max_amp_sum<dozv_max_amp_sum
    faktor=max_amp_sum/dozv_max_amp_sum;
    for i=1:duz
        % uvecavam svaki odbirak suma za vrednost izracunatog faktora
        beli_sum(i,1)=beli_sum(i,1)*faktor;
    end
end
```

8.8. funkcija *omega_kalman*

```
function [omega_pom]=omega_kalman(k,v,s)
% Funkcija za pronalazenje omega za azuriranje merenja u Kalmanovom
% algoritmu
if v(k,:)~=0
    arg=v(k,:)/s(k,1);
    psi=min(abs(arg),k)*sign(arg);
    omega_pom=psi/arg;
else
    omega_pom=1;
end
end
```

8.9. funkcija *D_fja*

```
function [teta_hat,z_novo,lambda]=D_fja(p,Z,z_hat,teta,epsilon,duz,omega,C1)
% Estimacija parametara pomocu diskriminacione f-je za signal drugog reda

%% Diskriminaciona funkcija - pocetni uslovi i potrebne promenljive
% Odredjujem velicinu prozora kod diskriminacione funkcije
I=50;
% Izlazni signal nakon Kalmanove predfiltracije je sada ulazni signal
y_novo=z_hat;
% Definisem faktor skaliranja
d=median(abs(y_novo-median(y_novo)))/0.6745;
% Definisem Z
Z_novo=Z;
for i=(p+1):duz
    for j=1:p
        Z_novo(j,i)=-y_novo(i-j,1);
    end
```

```
end
% Definisem pocetno epsilon i teta_hat
teta_hat=teta;
epsilon_novo=epsilon;
for i=1:I
    epsilon_novo(i,1)=y_novo(i,1)-Z_novo(:,i)'*teta_hat(:,i);
end
% Definisem neke konstante za izracunavanje lambda
N_min=20;
N_max=500;
D_min=0;
D_max=1;
D=zeros(duz,1);
lambda_min=1-1/N_min;
lambda_max=1-1/N_max;
omega_novo=omega; % izjednacavam ih zbog pocetnih vrednosti
lambda=zeros(duz,1);

%% Diskriminaciona funkcija - algoritam
M=zeros(p,p,duz);
K=zeros(p,1,duz);
P=zeros(p,p,duz);
for k=(1+I):(duz-I)
% Opet definisem gresku predikcije (sa novim vrednostima)
epsilon_novo(k,1)=y_novo(k,1)-Z_novo(:,k)'*teta_hat(:,k-1);
% Racunam tezinsku formu (omega_novo)
[omega_pom_2]=omega_dfja(k,epsilon_novo,d,y_novo,Z_novo,teta_hat);
omega_novo(k,1)=omega_pom_2;
% Racunam diskriminacionu f-ju preko logaritma f-je verodostojnosti
D(k,1)=L(k-I+1,k+I,epsilon_novo)-L(k-I+1,k,epsilon_novo)-L(k+1,k+I,epsilon_novo);
% Modifikujem D preko trenda diskriminacione funkcije
[D]=trend_D(k,I,D,epsilon_novo);
% Racunam lambda preko diskriminacione funkcije
lambda(k,1)=((lambda_max-lambda_min)/(D_min-D_max))*(D(k,1)-D_max)+lambda_min;
% Racunam maksimum diskriminacione funkcije
D_max=max(D(1:k,1));
M(:,:,k)=P(:,:,k-1)/lambda(k,1);
pom_1=M(:,:,k)*Z_novo(:,k)*omega_novo(k,1); % pomocna promenljiva 1
pom_2=1+Z_novo(:,k)'*pom_1;                 % pomocna promenljiva 2
% Matrica pojacanja
K(:,:,k)=pom_1/pom_2;
% Pozivam pomocnu konstantu
C=C1;
% Matrica kovarijanse greske estimacije
P(:,:,k)=C*eye(p)-K(:,:,k)*(Z_novo(:,k)')*M(:,:,k);
% Estimirana vrednost parametara
teta_hat(:,k)=teta_hat(:,k-1)+K(:,:,k)*epsilon_novo(k,1);
end

% Modeliram signal na osnovu estimiranih parametara
z_novo=y_novo;
for i=(p+1):duz
    z_novo(i,1)=0;
    for k=1:p
        z_novo(i,1)=z_novo(i,1)-teta_hat(k,i)*z_novo(i-k,1);
    end
end
end
```

8.10. funkcija *omega_dfja*

```
function [omega_pom_2]=omega_dfja(k,epsilon_novo,d,y_novo,Z_novo,teta_hat)
% Racunam omega za diskriminacionu funkciju (dfja)
arg=epsilon_novo(k,1)/d;
psi=min(abs(arg),k)*sign(arg);
if (y_novo(k,1)~=Z_novo(:,k)'*teta_hat(:,k))
    omega_pom_2=psi/arg;
else
    omega_pom_2=1;
end
end
```

8.11. funkcija *L*

```
function [rez]=L(a,b,epsilon_novo)
% Logaritam funkcije verodostojnosti - koristi se za racunanje
% diskriminacione funkcije
suma_eps=0;
for i=a:b
    suma_eps=suma_eps+epsilon_novo(i,1)^2;
end
rez=(b-a+1)*log((1/(b-a+1))*suma_eps);
end
```

8.12. funkcija *trend_D*

```
function [D]=trend_D(k,I,D,epsilon_novo)
% Funkcija za biranje intervala u kome VFF ima nagle promene
n_1=k-round(I/4)+1; % donja granica
n_2=k+round(I/4);   % gornja granica
prag=80;            % vrednost praga

% Definisem lokalne ekstremume
alfa_min=min(D(n_1:n_2,1));
alfa_max=max(D(n_1:n_2,1));
delta_alfa=alfa_max-alfa_min;
if (delta_alfa<=prag)
    % Ako nastaju nagle promene u lokalu tada ne diram D
    D(k,1)=L(k-I+1,k+I,epsilon_novo)-L(k-I+1,k,epsilon_novo)-L(k+1,k+I,epsilon_novo);
elseif (delta_alfa>prag)
    % Ako ne nastaju nagle promene u lokalu signal je priblizno stacionaran
    D(k,1)=alfa_min;
end
end
```

8.13. funkcija *GGP*

```
function [teta_hat,z_novo,lambda]=GGP(p,Z,z_hat,teta,epsilon,duz,omega,C2)
% Estimacija parametara pomocu generalisane greske predikcije za signal
% drugog reda

%% Generalisana greska predikcije - pocetni uslovi i potrebne promenljive
% Izlazni signal nakon Kalmanove predfiltracije je sada ulazni signal
y_novo=z_hat;
% Definisem faktor skaliranja
d=median(abs(y_novo-median(y_novo)))/0.6745;
% Definisem Z
Z_novo=Z;
for i=(p+1):duz
    for j=1:p
        Z_novo(j,i)=-y_novo(i-j,1);
    end
end
% Definisem pocetno epsilon i teta_hat
teta_hat=teta;
epsilon_novo=epsilon;
Ma=5;
for i=1:(Ma+1)
    epsilon_novo(i,1)=y_novo(i,1)-Z_novo(:,i)'*teta_hat(:,i);
end
% Definisem pocetno sigma_hat
sigma_hat=0.22*ones(duz,1);
% Definisem neke konstante za izracunavanje lambda
N_max=500;
lambda_min=0.75;
lambda_max=0.998;
E=zeros(duz,1);
omega_novo=omega; % izjednacavam ih zbog pocetnih vrednosti
N=zeros(duz,1);
lambda=zeros(duz,1);

%% Generalisana greska predikcije - algoritam
M=zeros(p,p,duz);
```

```
K=zeros(p,1,duz);
P=zeros(p,p,duz);
for k=(Ma+1):duz
% Opet definisem gresku predikcije (sa novim vrednostima)
epsilon_novo(k,1)=y_novo(k,1)-Z_novo(:,k)'*teta_hat(:,k-1);
% Racunam tezinsku formu (omega_novo)
[omega_pom_2]=omega_ggp(k,epsilon_novo,d,y_novo,Z_novo,teta_hat);
omega_novo(k,1)=omega_pom_2;
% Racunam estimiranu vrednost varijanse suma
sigma_hat(k,1)=((k-1)*sigma_hat(k-1,1)+(epsilon_novo(k,1)^2)*omega_novo(k,1))/k;
% Racunam Extended Prediction Error (EPE)
suma_1=0;
for i=1:Ma;
    suma_1=suma_1+epsilon_novo(k-i,1)^2;
end
E(k,1)=suma_1/Ma;
% Duzina memorije
N(k,1)=((sigma_hat(k,1)^2)*N_max)/E(k,1);
% Racunam promenljivi faktor zaboravljanja (VFF)
lambda(k,1)=max(1-E(k,1)/(sigma_hat(k,1)*N_max),lambda_min);
% Ogranicavam minimalnu i maksimalnu vrednost VFF-a
if lambda(k,1)>lambda_max
    lambda(k,1)=lambda_max;
elseif lambda(k,1)<lambda_min
    lambda(k,1)=lambda_min;
end
M(:,:,k)=P(:,:,k-1)/lambda(k,1);
pom_1=M(:,:,k)*Z_novo(:,k)*omega_novo(k,1); % pomocna promenljiva 1
pom_2=1+Z_novo(:,k)'*pom_1;                 % pomocna promenljiva 2
% Matrica pojacanja
K(:,:,k)=pom_1/pom_2;
% Pozivam pomocnu konstantu
C=C2;
% Matrica kovarijanse greske estimacije
P(:,:,k)=C*eye(p)-K(:,:,k)*(Z_novo(:,k)')*M(:,:,k);
% Estimirana vrednost parametara
teta_hat(:,k)=teta_hat(:,k-1)+K(:,:,k)*epsilon_novo(k,1);
end

% Modeliram signal na osnovu estimiranih parametara
z_novo=y_novo;
for i=(p+1):duz
    z_novo(i,1)=0;
    for k=1:p
        z_novo(i,1)=z_novo(i,1)-teta_hat(k,i)*z_novo(i-k,1);
    end
end
end
```

8.14. funkcija *omega_ggp*

```
function [omega_pom_2]=omega_ggp(k,epsilon_novo,d,y_novo,Z_novo,teta_hat)
% Racunam omega za generalisanu gresku predikcije (ggp)
arg=epsilon_novo(k,1)/d;
psi=min(abs(arg),k)*sign(arg);
if (y_novo(k,1)~=Z_novo(:,k)'*teta_hat(:,k))
    omega_pom_2=psi/arg;
else
    omega_pom_2=1;
end
end
```