# 7. VJEŽBA

## Efekti kvantizacije kod IIR filtara

Sagledavanje svih efekata kod projektiranja i realizacije IIR filtara nije potpuno ako se detaljno ne prouče i efekti koji nastaju uslijed ograničene točnosti procesora ili sklopovlja kojima se takav filtar realizira. U prošlim vježbama je pored različitih realizacija analiziran i efekt kvantizacije koeficijenata filtra. Međutim, treba voditi računa o tome da i ulazni i izlazni signal kao i stanja filtra također moraju biti kvantiziranih iznosa. U općenitom slučaju, broj bita za prikaz koeficijenata i broj bita za prikaz signala i stanja ne mora biti isti.

Efekti kvantizacije stanja došli su do izražaja u prethodnim vježbama pri realizaciji filtara na DSP procesoru. Ovisno o realizaciji, kvantizacija se provodila na 16 (jednostruka preciznost) ili 32 bita (dvostruka preciznost), a samim tim bila je definirana i točnost filtriranja signala. Pogreška je promatrana u usporedbi s rezultatima filtriranja u MATLAB-u koji radi s puno većom točnošću bez kvantizacije stanja, ali koristi iste kvantizirane koeficijente.

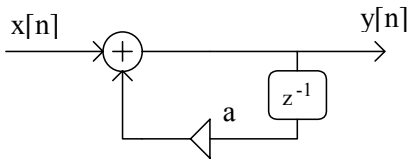
Greška nastala uslijed kvantizacije ulaznog signala se relativno lako može analizirati, jer grešku kvantizacije možemo promatrati kao signal pogreške koji se pribraja ulazu i koji se zatim filtrira jednako kao i sam signal. Nešto je složenija analiza grešaka nastalih kvantizacijom stanja unutar filtra, a pojave vezane uz to biti će opisane u nastavku.

### *Zero input limit cycle*

Ako na stabilne vremenski diskretne IIR sisteme realizirane aritmetikom beskonačne preciznosti djelujemo pobudom koja u određenom trenutku padne u nulu i ostane u nuli, tada se odziv takvih sistema asimptotski približava nuli. Za istu takvu pobudu i isti sistem, ali realziran aritmetikom konačne preciznosti, odziv može nastaviti beskonačno oscilirati iako je pobuda ostala u nuli. Takva pojava se naziva **'zero input limit cycle'**, a posljedica je ili nelinearnosti kvantizacije u povratnoj vezi ili prekoračenja dinamike kod zbrajanja u two's complement aritmetici.

- **Limit cycle uslijed zaokruženja ili odsijecanja pri kvantizaciji**

Ova pojava je vrlo složena i teško se analizira općenito, pa je možemo promotriti na jednostavnom primjeru sistema prvog reda, opisanog jednadžbom diferencija (7.1) i blok shemom (Slika 7.1).




Slika 7.1. Blok shema IIR strukture 1. Reda.

$$y[n] = a \cdot y[n-1] + x[n] \ , \ |a| < 1 \tag{7.1}$$

Pretpostavimo da dužina registra za pohranu koeficijenta $a$, ulaza $x[n]$ i varijable stanja $y[n-1]$, iznosi 4 bita (3 bita + 1 bit predznak). Zbog registara konačne dužine, 7-bitni produkt $a \cdot y[n-1]$ se mora ili zaokružiti ili odsjeći na 4 bita prije sumacije s $x[n]$ i pohrane u

registar stanja. Na primjer, pretpostavimo sistem s a = 0.5 = 0∧100, početnim stanjem jednakim nuli i pobudom $x[n] = 7/8 \cdot \delta[n] = (0\text{^}111) \cdot \delta[n]$. Da bismo pronašli $y[1]$, množimo $y[0]$ s *a* i dobije se 7-bitni broj 0∧011 | 100 koji treba zaokružiti na 4-bitnu vrijednost. Ovaj broj, 7/16, je točno na pola između dva 4 bitna kvantizacijska nivoa, 3/8 i 4/8. Ako u takvim slučajevima zaokružujemo na prvi veći broj, tada se zaokruženjem dobiva 0∧100 = 0.5. Pošto je pobuda od $n = 1$ na dalje jednaka nuli, vrijednost kvantiziranog produkta je identična izlaznoj vrijednosti $y[n]$. Ponavljanjem izvođenja jednadžbe diferencija, dobiva se y[2] = 0∧ 010 = 0.25 i y[3] = 0∧001 = 0.125. U oba ova slučaja donji dio produkta je jednak nuli pa zaokruženje nije potrebno. No, kod traženja $y[4]$ zaokruženjem $a \cdot y[3]$=0∧000 | 100 dobiva se 0∧001 = 0.125, što se dobiva i za sve daljnje uzorke izlaza za $n > 3$. Vidljivo je da iako je ulaz pao u nulu, izlaz ostaje zauvijek na vrijednosti 0.125. Kada bi koeficijent *a* imao vrijednost $a = -0.5$, ponavljanjem gornjeg postupka može se opaziti da se na izlazu javljaju oscilacije, tj. uz nulu na ulazu, izlaz naizmjence poprima vrijednost 0.125 i -0.125. Oba ova slučaja se tretiraju kao pojava zero input limit cycle.

U prethodnom primjeru je razmatran slučaj kvantizacije s funkcijom zaokruženja na najbliži cijeli broj (rounding). Slična ponašanja javljaju se i kod kvantizacije odsijecanjem, odnosno odsijecanjem simetričnim prema nuli.

- **Limit cycle uslijed prekoračenja dinamike**

Promatrajmo sistem drugog reda opisan jednadžbom diferencija

$$y[n] = a_1 \cdot y[n-1] + a_2 \cdot y[n-2] + x[n] \tag{7.2}$$

Kvantizacija je posljedica zaokruženja na 4 bita. Do prekoračenja dinamike može doći kod zbrajanja two's complement vrijednosti zaokruženih produkata. Npr. neka je $a_1$=0.75=0∧110 i $a_2$=-0.75=1∧010, ulazni signal $x[n]$ neka je nula za $n \geq 0$ i početna stanja y[-1]=0.75=0∧110 i y[-2]=-0.75=1∧010. Izlaz za $n = 0$ dobiva se kao

$$y[0] = 0 \wedge 110 \times 0 \wedge 110 + 1 \wedge 010 \times 1 \wedge 010$$

tj. ako izračunamo produkte:

$$y[0] = 0 \wedge 100 | 100 + 0 \wedge 100 | 100$$

što nakon zaokruženja daje

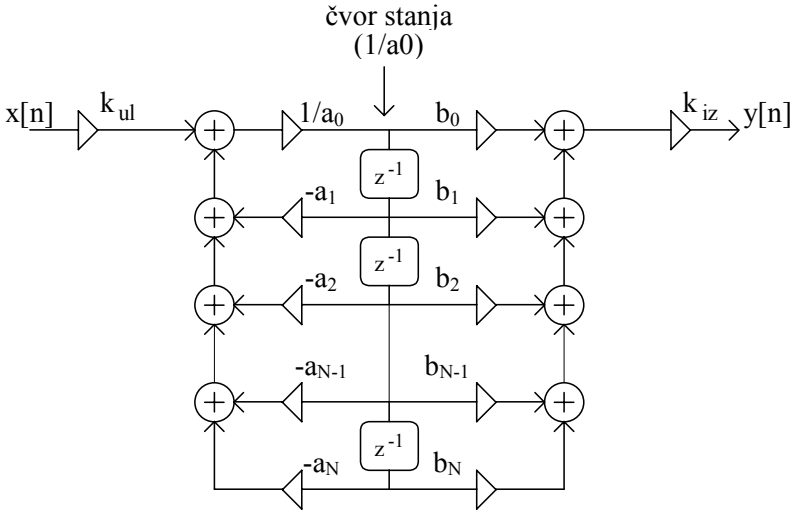
$$y[0] = 0 \wedge 101 + 0 \wedge 101 = 1 \wedge 010 = -0.75$$

Vidljivo je da two's complement sumacijom dva velika pozitivna broja dobivamo negativni broj, što predstavlja značajnu pogrešku. Ponavljanjem jednadžbe diferencije dobivamo y[1] = 0.75 koji je dobiven sumacijom dva velika negativna broja. Očito je da će izlaz nastaviti oscilirati između 0.75 i -0.75 tako dugo dok god je na ulazu nula.

Ovakva pojava je vrlo neugodna jer uzrokuje oscilacije gotovo maksimalnom amplitudom iako je na ulazu nula. Ona se može spriječiti tako da se umjesto klasične two's complement aritmetike koristi aritmetika sa zasićenjem (saturation). Njenom primjenom, suma dva velika pozitivna broja koja bi u normalnom slučaju rezultirala negativnim brojem, zamjenjuje se s maksimalnim pozitivnim brojem. Analogno, suma dva velika negativna broja, zamjenjuje se s maksimalnim negativnim brojem.

## *Direktna realizacija*

Za ilustraciju programske realizacije direktne filtarske strukture dan je odsječak funkcije napisane za programski paket MATLAB, koja u potpunosti simulira rad jednog cjelobrojnog procesora. U odsječku se pozivaju funkcije za zbrajanje i množenje 'intadd.m' i 'intmult.m' koje simuliraju rad s brojevima fiksne preciznosti, a blok shema samog filtra prikazana je slikom (Slika 7.2).




Slika 7.2. Blok shema IIR filtra za direktnu realizaciju.

**Funkcija 'INTADD.M'**

```
function sum=intadd(s1,s2)          % funkcija za zbrajanje sa zasićenjem
s=s1+s2;                            % zbroji s1 i s2
if (s<mns),                         % ako je rezultat manji od najmanjeg
                                    % negativnog broja
      s=mns;                        % zamijeni ga s max negativnim brojem
      l=1;                          % označi da se desilo zasićenje
end;
if (s>mxs),                         % ako je rezultat veći od najvećeg
                                    % pozitivnog broja
      s=mxs;                        % zamijeni ga s max pozitivnim brojem
      l=1;                          % označi da se desilo zasićenje
end;
sum=s;                              % rezultat je jednak s
```

**Funkcija 'INTMULT.M'**

```
function prd=intmult(s,c,round_type)% funkcija za množenje i kvantizaciju
                                    % produkta
if (c==mxc),                        % ako je koeficijent jednak 1,
      p=s;                          % produkt je jednak signalu s
else, if (c==mnc),                  % inače, ako je koeficijent jednak -1,
            p= -s;                  % produkt iznosi -s
      else,                         % za sve ostale slučajeve,
            p=s*c;                  % nađi produkt signala i koeficijenta
            if (round_type==0),     % ako je round_type = 0
                  p=floor(p/mxc);   % odsijeci produkt na broj bita signala
            else, if (round_type==1),     % inače, ako je round_type = 1
                        p=fix(p/mxc);     % simetrično odsijeci oko nule
                  else,                   % inače,
                        p=round(p/mxc);   % zaokruži na najbližu frakciju
                  end;
            end;
      end;
end;
if (p==mxs+1),                % ako je kao produkt dobivena vrijednost 1
      p=mxs;                  % zamijeni je s max pozitivnom vrijednošću
      l=1;                    % označi da se dogodilo zasićenje
end;
prd=p;                        % rezultat množenja je u prd
```

**Programski odsječak za direktnu realizaciju**

```
ce=intmult(cc,scal(1),rt);                % pomnoži signal ulaz. pojačalom
sum=0;                                    % obriši sumu za nazivnik
for cv=N:-1:1,                            % od N-tog do prvog stanja
      pr=intmult(st(cv),-naz(cv+1),rt);   % pomnozi stanje i koeficijent ai
      sum=intadd(sum,pr);                 % akumuliraj u sumu
end;
sum=intadd(sum,ce);                       % dodaj skalirani signal
mn=mxc/naz(1);                            % vrijednost 1/a0
sh=round(log10(mn)/log10(2));             % nađi broj shift-anja
ca=intshift(sum,sh,rt);                   % prema potrebi shiftaj --> time
                                          % se dobije signal u čvoru 1/a0
sum=0;                                    % obriši sumu za brojnik
for cv=N:-1:1,                            % od N-tog do prvog stanja
      pr=intmult(st(cv),br(cv+1),rt);     % pomnozi stanje i koeficijent bi
      sum=intadd(sum,pr);                 % akumuliraj u sumu
end;
pr=intmult(ca,br(1),rt);                  % čvor (1/a0) * b(0)
sum=intadd(sum,pr);                       % dodaj i njega u sumu
for cv=N:-1:2,                            % napravi pomak stanja
      st(cv)=st(cv-1);                    % u i-to ubaci (i-1)-vo stanje
end;
st(1)=ca;                                 % prvo stanje poprima vrijednost
                                          % čvora 1/a0
% množenje signala  s izlaznim pojačalom kiz
iz=0;
um=scal(2);                               % vrijednost izlaznog pojačala
bs=floor(um/mxc);                         % cijeli dio pojačala
um=rem(um,mxc);                           % decimalni dio pojačala
if (bs~=0),                               % ako postoji cijeli dio
      for bss=1:bs,                       % napravi uzastopnu sumaciju
            iz=intadd(iz,sum);
      end;
end;
izf=intmult(sum,um,rt);                   % produkt s decimalnim dijelom
iz=intadd(izf,iz);                        % zbroji cijeli i decimalni dio
```

Ulazne varijable u gore prikazani programski odsječak su:

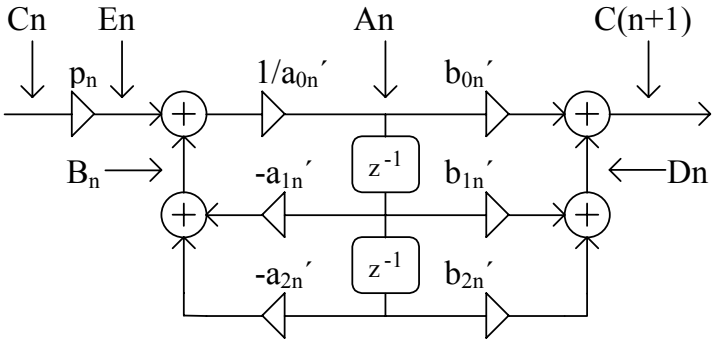
- **N** je red IIR filtra;
- **cc** je vrijednost uzorka ulaznog signala;
- **br** je redak s **(N+1)**-im elementom koji predstavljaju cjelobrojne koeficijente brojnika prijenosne funkcije **H(z);**
- **naz** je također redak iste dimenzije, a sadrži koeficijente nazivnika;
- **scal** je stupac ili redak s dva elementa koji predstavljaju vrijednosti ulaznog i izlaznog pojačala;
- **mxc** je $2^{\text{brbit}-1}$ i predstavlja frakcionalnu vrijednost 1, gdje je **brbit** broj bita za frakcionalne koeficijente;
- **st** je redak s **N** elemenata i sadrži stanja filtra;
- **rt** je varijabla koja sadrži način kvantizacije produkata i proslijeđuje se funkciji intmult;

Na kraju izvođenja ovog odsječka varijabla **iz** predstavlja izlaz izlaznog pojačala filtra, a to je ujedno i jedan izlazni uzorak cijelog filtra. Uzastopnim ponavljanjem ovog odsječka za sve uzorke ulaznog signala dobiva se ukupni odziv filtra.

## *Kaskadna realizacija*

Za ilustraciju programske realizacije kaskadne filterske strukture dan je odsječak funkcije također napisane za MATLAB, koja u potpunosti simulira rad jednog cjelobrojnog

procesora. U odsječku se pozivaju iste funkcije za zbrajanje i množenje kao kod primjera direktne realizacije. Blok shema jedne kaskade filtra s ucrtanim čvorovima prikazana je na slikom (Slika 7.3)




Slika 7.3. Blok shema jedne kaskade IIR filtra.

**Programski odsječak za kaskadnu realizaciju**

```
for cas=1:brcas+1,                    % za sve kaskade i zadnje pojačalo
      cec=0;
      um=scal(cas);                   % vrijednost ulaznog pojačala
      bs=floor(um/mxc);               % cijeli dio multiply-era
      um=rem(um,mxc);                 % decimalni dio
      if (bs~=0),                     % ako postoji cijeli dio
            for bss=1:bs,             % napravi uzastopnu sumaciju
                  cec=intadd(cec,cc);
            end;
      end;
      cef=intmult(cc,um,rt);          % nađi produkt s decim. dijelom
      ce=intadd(cef,cec);             % zbroji cijeli i decimalni dio
      if (cas<(brcas+1)),             % ako to nije pojačalo na izlazu
                                      % iz filtra, tada radi i biquad
            cpb2=intmult(st(cas,2),-naz(cas,3),rt);   % x(k-2) * -a(2)
            cpb1=intmult(st(cas,1),-naz(cas,2),rt);   % x(k-1) * -a(1)
            cb=intadd(cpb1,cpb2);                     % čvor b
            ca=intadd(cb,ce);                         % čvor a
            if (naz(cas,1)==mxc/2),             % ako je a(0)=0.5
                  ca=intadd(ca,ca);             % pomnozi čvor a s dva
            end;
            cpd2=intmult(st(cas,2),br(cas,3),rt);     % x(k-2) * b(2)
            cpd1=intmult(st(cas,1),br(cas,2),rt);     % x(k-1) * b(1)
            cd=intadd(cpd1,cpd2);                     % čvor d
            cpc=intmult(ca,br(cas,1),rt);             % čvor a * b(0)
            cc=intadd(cpc,cd);                        % čvor c
            st(cas,2)=st(cas,1);                % x(k-2) = x(k-1)
            st(cas,1)=ca;                       % x(k-1) = čvor a
      end;
end;
```

Ulazne varijable u ovaj programski odsječak su:

- **brcas** je broj kaskada IIR filtra;
- **cc** je inicijalno vrijednost uzorka ulaznog signala;
- **br** je matrica s **brcas** redaka i 3 stupca, a predstavlja cjelobrojne koeficijente brojnika pojedinih kaskada;
- **naz** je također matrica **brcas** × 3, a sadrži koeficijente nazivnika;
- **scal** je stupac s (**brcas**+1) redaka s vrijednostima ulaznih pojačala u pojedinu kaskadu, a zadnji element stupca sadrži vrijednost izlaznog pojačala;
- **mxc** je $2^{\text{brbit}-1}$ i predstavlja frakcionalnu vrijednost 1, gdje je **brbit** broj bita za frakcionalne koeficijente;
- **st** je matrica s **brcas** redaka i dva stupca i sadrži stanja po pojedinim kaskadama;

- **rt** je varijabla koja sadrži način kvantizacije produkta i prosljeđuje se funkciji intmult.

## *Praktična vježba*

**UTJECAJ KVANTIZACIJE: SIMULACIJA U MATLABU**

1. Unutar MATLAB okruženja pokrenuti program **vj7**. Odabrati slijedeći skup parametara filtra: **niski propust**, **Chebyshev tip II**, **4. red**, **granična frekvencija 0.2**, **valovitost u području gušenja 45dB**. Odabrati **8 bita** za kvantizaciju koeficijenata.

Odabrati **12 bita** za kvantizaciju signala, i **impulsnu pobudu**. Za sve tipove kvantizacije ponoviti postupak uz isti broj bita i istu pobudu. Pogledati **Amplitudno fazno frekventnu karakteristiku**, **Simulaciju u MATLABU** i **Grešku kvantizacije stanja**. Komentirati rezultate, uz koji tip kvantizacije je greška najmanja i koliko iznosi izražena u broju bita ?

Ponoviti za sva tri tipa kvantizacije, ali uz kvantizaciju signala na **8 bita**. Koje sve pojave možete uočiti ?

Odabrati kao pobudu **step** i sva tri načina kvantizacije s **12 bita**. Kolika je maksimalna vrijednost odziva (nadvišenja) uz realna stanja, a kolika je uz cjelobrojna stanja ? Zbog čega se javlja prekoračenje dinamike ?

Odabrati kao pobudu **sinus** najprije frekvencije koja **ulazi u pass band**, a zatim i frekvencije **u stop bandu** te uz sva tri načina kvantizacije s **8 i 12 bita** analizirati dobivene rezultate. Kakav je odnos greške uz frekvenciju u pass band-u i uz frekvenciju u stop band-u ?

2. Ponoviti **kompletni postupak iz točke 1. za kaskadnu realizaciju** filtra s **rastućim Q** faktorom. Analizirati razlike između jedne i druge realizacije, te odnose maksimalnih grešaka.

3. Ponoviti gornje postupke za **Butterworth-ov filtar 6 reda i tip filtra po izboru**. Komentirati razlike u odnosu na prethodni filtar.

_____
_____
_____
_____
_____
_____
_____
_____

**UTJECAJ KVANTIZACIJE: REALIZACIJA NA DSP PROCESORU**

4. U svim primjerima ove i prethodnih vježbi provedena je konstrukcija filtra koja bi trebala neovisno o tipu realizacije, kvantizaciji koeficijenata, odnosno kvantizacije stanja, osigurati ispravan rad filtra u cijelom frekvencijskom području. Pri tome su korištene metode koje pretpostavljaju sinusni signal, maksimalne amplitude, na ulazu u sistem i promjenom njegove frekvencije kroz cijelo područje traže se maksimalne vrijednosti pojedinih interesantnih točaka digitalnog sistema, i na osnovu njih vrše odabir iznosa ukupnog ulaznog i izlaznog pojačanja. Treba uočiti da je naglasak na ulaznom signalu koji se sastoji od jednog harmonika. U praksi su mogući signali koji se sastoje od mnoštva harmonika te kao takvi nisu obuhvaćeni gornjom analizom. Zbog toga je moguće da uz takav signal filtar izađe iz linearnog područja (dinamike ±1). To predstavlja naročiti problem kod realizacije filtra u dvostrukoj preciznosti gdje u programu nije provedena aritmetika s zasićenjem te u slučaju izlaska iz dinamike dolazi do WRAP-AROUND efekta (brojevi veći od 1 postaju negativni, a manji od -1 pozitivni). Posljedica toga je javljanje oscilacija visokih frekvencija.

U ovoj točki promatra se unaprijed izračunati niskopropusni eliptički filter 12 reda, granične frekvencije 0.05Fo, te valovitošću u propusnom području iznosa 0.01dB i gušenjem u području gušenja od 120dB. Provedena je realizacija na ADSP2181 procesoru kao kaskadna izvedba s koeficijentima i stanjima duljine 32 bita te frekvencijom otipkavanja od 48kHz. Izvršna verzija programa nalazi se u datoteci ‘sat.exe’. Pritiskom na tipku **Pokreni sat.exe** pokreće se dotičan program na DSP procesoru.

Na ulaz dovesti sinusni signal i mijenjati frekvenciju od minimalno moguće do polovine frekvencije otipkavanja. Promjena ne smije biti previše brza kako bi se filtar stigao utitravati. Opisati promjene izlaznog signala u ovisnosti o ulaznoj frekvenciji.

_____
_____
_____
_____

Ponoviti postupak uz trokutni signal na ulazu.

_____
_____
_____
_____

Ponoviti postupak uz pravokutni signal na ulazu.

_____
_____
_____
_____

Opisati dobivene razlike u ovisnosti o odabranom valnom obliku ulaznog signala.

_____

_____