VEŽBA 5 – IIR filtri

Potrebno predznanje

- Poznavanje programskog jezika C
- Urađena Vežba 1 – Uvod u Digitalnu Obradu Signala
- Urađena Vežba 4 – FIR filtri
- Odslušana predavanja iz predmeta OAiS DSP 1 na temu Diskretni sistemi
- Odslušana predavanja iz predmeta OAiS DSP 1 na temu FIR filtri
- Odslušana predavanja iz predmeta OAiS DSP 1 na temu IIR filtri

Šta će biti naučeno tokom izrade vežbe

U okviru ove vežbe naučićete:

- Šta su to diskretni filtri sa beskonačnim odzivom
- Koje su prednosti i mane filtara sa beskonačnim odzivom
- Kakve su performanse i računarski zahtevi filtara sa beskonačnim odzivom u odnosu na filtere sa konačnim odzivom.
- Šta su to *all-pass* filtri
- Kako se realizuje *half-band* strukutra, i primenu ovakve strukture u algoritmima za skremblovanje govora

Motivacija

Diskretnim FIR filtrima možemo, teoretski, ukloniti proizvoljne delove spektra (nepoželjne spektralne komponente) uz propuštanje ostatka spektra (željene spektralne komponente). Na deo spektra koji se nalazi između željenih i nepoželjnih frekvencija uglavnom ne možemo uticati pa nam je cilj da je taj deo spektra što manji. Sa povećanjem zahteva za preciznom kontrolom opsega frekvencija koje se propuštaju i koje se potiskuju brzo raste složenost FIR filtra. U određenim slučajevima nije moguće praktično realizovati FIR filter zbog potrebe za velikim brojem operacija u jedinici vremena i čuvanjem velikog broja ulaznih odbiraka iz prošlosti (memorija filtra). Ako nije neophodno sačuvati fazu filtriranog signala moguće je koristiti filtere sa beskonačnim impulsnim odzivom (IIR). Kod IIR filtera osim odbiraka ulaznog signala pri filtriranju koriste se i odbirci prethodno filtriranog signala. Time se postiže manja kompleksnost nego kod FIR filtera (broj operacija i veličina interne memorije filtra) uz potencijalnu nestabilnost filtra (zbog povratne sprege) i nelinearnu faznu karakteristiku.

1 TEORIJSKE OSNOVE

1.1 IIR strukture

Za razliku od filtara sa konačnim impuslnim odzivom, funkcija prenosa kod IIR filtara sadrži i direktni (a-parametri) i rekurzivni (b-parametri) deo. Prenosna funkcija IIR filtra u *z*-transformaciji data je sa:

$$H_z(z) = \frac{\sum_{k=0}^{L-1} a(k) \cdot z^{-k}}{1 + \sum_{k=1}^{M} b(k) \cdot z^{-k}} .$$

Data jednačina sadrži nule (nule polinoma u brojiocu) i polove (nule polinoma u imeniocu). U vremenskom domenu je IIR filter opisan sa diferencijalnom jednačinom:

$$y(n) = \sum_{k=0}^{L-1} a(k) \cdot x(n-k) - \sum_{k=1}^{M} b(k) \cdot y(n-k) ,$$

gde su *x(n)* odbirci ulaznog signala a *y(n)* odbirci izlaznog signala. Impulsni odziv IIR filtra *h(n)* je generalno beskonačan i može se pronaći na dva načina kada su poznati koeficijenti filtra (a i b):

- Deljenjem polinoma u z-transformaciji:

$$H_z(z) = \frac{\sum_{k=0}^{L-1} a(k) \cdot z^{-k}}{1 + \sum_{k=1}^{M} b(k) \cdot z^{-k}} = \sum_{n=0}^{\infty} h(n) \cdot z^{-n} ;$$

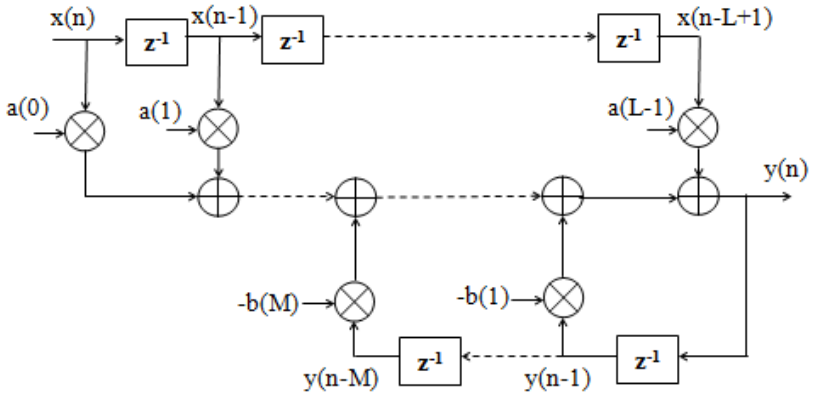
- Korišćenjem diferencijalne jednačine za *d*-impuls na ulazu:

$$x(n) = \delta(n) \Rightarrow y(n) = \sum_{k=0}^{L-1} a(k) \cdot x(n-k) - \sum_{k=1}^{M} b(k) \cdot y(n-k) \Rightarrow h(n) = y(n) .$$

Osnovni nedostaci IIR filtara su:

- Stabilnost filtra se mora kontrolisati (polovi moraju biti unutar jediničnog kruga) jer IIR filtar može biti nestabilan (ako ima polove van jediničnog kruga)
- Impulsni odziv je beskonačan i nije simetričan, pa fazna karakteristika nije linearna. Drugim rečima, IIR filtar nema konstantno grupno kašnjenje (dodatno se mora voditi računa da fazna karakteristika makar u propusnom opsegu bude približno linearna).

Velika prednost IIR filtara je što se zahtevana specifikacija najčešće može ispuniti sa značajno manjim brojem koeficijenata (L+M) nego što bi zahtevala FIR realizacija.



Slika 1 - Prva kanonična forma IIR filtra

1.2 Implementacija IIR filtra

Implementacija IIR filtra u direktnoj formi upotrebom programskog jezika C može se izvršiti na sličan način kao i prikazana implementacija FIR filtra u prethodnoj vežbi. Parametrizovana funkcija koja predstavlja IIR filtar data je sa:

```
Int16 iir_basic(Int16 input, Int16* a_coeffs, Int16* x_history, Uint16 n_a_coeff,
Int16* b_coeffs, Int16* y_history, Uint16 n_b_coeff)
{
    Int16 i;
    Int32 ret_val;

    for (i = n_a_coeff - 2; i >= 0; i--)
    {
        x_history[i + 1] = x_history[i];
    }
    for (i = n_b_coeff - 2; i >= 0; i--)
    {
        y_history[i + 1] = y_history[i];
    }

    x_history[0] = input;

    ret_val = 0;
    for (i = 0; i < n_a_coeff; i++)
    {
        ret_val = _smac(ret_val, a_coeffs[i], x_history[i]);
    }

    ret_val = (ret_val >> 15) * b_coeffs[0];
    for (i = 1; i < n_b_coeff; i++)
    {
        ret_val = _smas(ret_val, b_coeffs[i], y_history[i]);
    }

    y_history[0] = (Int16)(ret_val >> 16);
    return y_history[0];
}
```

Parametri funkcije za filtriranje su:

- *input* - odbirak ulaznog signala
- *a_coeffs* - niz sa koeficijentima koji se nalaze u brojiocu polinoma
- *x_history* - niz koji predstavlja memoriju u kojoj će se čuvati *L* poslednjih ulaznih odbiraka
- *n_a_coeff* - broj *a* koeficijenata
- *b_coeffs* - niz sa koeficijentima koji se nalaze u imeniocu polinoma
- *y_history* - niz koji predstavlja memoriju u kojoj će se čuvati *M* poslednjih izlaznih vrednosti
- *n_b_coeff* - broj *b* koeficijenata

U okviru funkcije vrši se pomeranje elemenata u memorijskim nizovima za skladištenje prethodnih ulaznih i izlaznih odbiraka (*x_history* i *y_history*) za jedno mesto udesno. Potom se vrednost ulaznog odbirka postavlja na prvo mesto *x_history* niza. Sledi računanje konvolucije između koeficijenata koji se nalaze u brojiocu i ulaznih odbiraka kao kod FIR filtra. Izračunata vrednost se koristi kao trenutna vrednost izlaza *y(n)*, i množi se sa koeficijentom *b(0)*. Nakon toga vrši se oduzimanje proizvoda koeficijenata imenioca i prethodnih vrednosti izlaza iz sistema. Za računanje ovih vrednosti koristi se MAS operacija (*Multiply and Substract*). MAS operacija slična je MAC operaciji, razlika je u tome što se vrši oduzimanje rezultata množenja od akumulatorske vrednosti umesto sabiranja. Za izvršenje MAS operacije takođe postoji ugrađena kompajlerska funkcija, i ona je data sa:

- `Int32 _smas(Int32 src1, Int16 src2, Int16 src3)`

Izračunata vrednost se pomera u desno za 16 mesta kako bi se dobavilo gornjih 16 bita rezultata kao i kod FIR filtra. Dobijena vrednost se smešta u memorijski niz za pamćenje prethodnih izlaza iz sistema, na prvo mesto i potom vraća kao povratna vrednost funkcije.

Implementacija IIR filtra upotrebom kružnog bafera realizuje se na isti način kao i kod FIR filtra. U slučaju IIR filtra koriste se dva kružna bafera, jedan za pamćenje prethodnih ulaza i drugi za pamćenje prethodnih izlaza iz sistema. Realizacija IIR filtra upotrebom kružnog bafera data je na sledećem primeru:

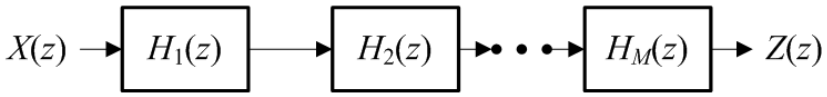
```
Int16 iir_circular(Int16 input, Int16* a_coeffs, Int16* x_history, Uint16* x_state,
Uint16 n_a_coeff, Int16* b_coeffs, Int16* y_history, Uint16* y_state, Uint16
n_b_coeff)
{
    int i, x_state_l, y_state_l;
    Int32 ret_val;

    x_state_l = *x_state;
    y_state_l = *y_state;
    x_history[x_state_l] = input;
    if (++x_state_l >= n_a_coeff)
    {
        x_state_l = 0;
    }

    ret_val = 0;
    for (i = n_a_coeff - 1; i >= 0; i--)
    {
        ret_val = _smac(ret_val, a_coeffs[i], x_history[x_state_l]);
```

```
        if (++x_state_l >= n_a_coeff)
        {
            x_state_l = 0;
        }
    }

    ret_val = (ret_val >> 15) * b_coeffs[0];
    for (i = n_b_coeff - 1; i >= 1; i--)
    {
        if (++y_state_l >= n_b_coeff)
        {
            y_state_l = 0;
        }
        ret_val = _smas(ret_val, b_coeffs[i], y_history[y_state_l]);
    }
    if (++y_state_l >= n_b_coeff)
    {
        y_state_l = 0;
    }

    y_history[y_state_l] = (Int16)(ret_val >> 16);
    if (++y_state_l >= n_b_coeff)
    {
        y_state_l = 0;
    }

    *x_state = x_state_l;
    *y_state = y_state_l;
    return  (Int16)(ret_val >> 16);
}
```

1.3 Kaskadna realizacija IIR filtara

Kod IIR filtara realizovanih u direktnoj formi sa povećanjem reda dolazi do povećanja opsega koeficijenata. Samim tim dolazi do povećanja greške koja nastaje kvantizacijom koeficijenata, kao i greške koja nastaje kao posledica zaokruživanja prilikom računanja međurezultata. Iz tog razloga, u praksi se mnogo češće koristi kaskadna realizacija IIR filtara.

Kaskadna realizacija IIR strukture se dobija tako što se funkcija prenosa predstavi kao proizvod funkcija prenosa nižeg reda, tipično prvog i drugog.



Slika 2 – Kaskadna veza filtara

Kod IIR sistema faktorizacija funkcija prenosa u proizvod faktora drugog reda vrši se na sledeći način:

$$H(z) = \prod_{k=1}^{N_s} H_k(z) = \prod_{k=1}^{N_s} \frac{a_{0k} + a_{1k}z^{-1} + a_{2k}z^{-2}}{b_{0k} + b_{1k}z^{-1} + b_{2k}z^{-2}}$$

gde je $N_s = [\max(L, M) + 1]/2$.

U slučajevima kada je $L < M$, u brojiocu nekih parcijalnih funkcija prenosa koeficijenti a_{1k} i a_{2k} će biti jednaki nuli. Ako je $L > M$ kod određenih sekcija koeficijenti b_{1k} i b_{2k} će biti jednaki nuli, tj. sekcija je FIR tipa. Ako je red sistema neparan, tj. ako je max(L, M) neparan broj, jedna od sekcija je prvog reda sa koeficijentima $a_{2k} = b_{2k} = 0$. Međutim, zbog modularnosti se obično sve sekcije realizuju kao kompletne sekcije drugog reda, pri čemu neki od koeficijenata mogu biti jednaki nuli. U opštem slučaju, koeficijenti u brojiocu i imeniocu parcijalnih funkcija prenosa su realni brojevi, što kao posledicu ima da se konjugovano kompleksni parovi polova ili nula uvek grupišu i realizuju istom sekcijom.

Osnovna komponenta kod kaskadne realizaijce IIR filtra jeste sekcija drugog reda (*biquad*). Sekcija drugog reda sadrži dve nule i dva pola.

$$H(z) = \frac{a_{0k} + a_{1k} z^{-1} + a_{2k} z^{-2}}{b_{0k} + b_{1k} z^{-1} + b_{2k} z^{-2}}$$

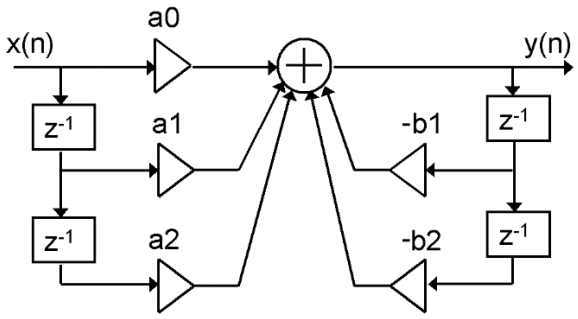
Najčešće se koeficijenti sekcije drugog reda normalizuju tako da vrednost koeficijenta *b0* bude jednaka 1. Time se olakšava implementacija operacija filtriranja.

$$H(z) = \frac{a_{0k} + a_{1k} z^{-1} + a_{2k} z^{-2}}{1 + b_{1k} z^{-1} + b_{2k} z^{-2}}$$

Prednost kaskadne realizacije jeste smanjenje reda veličina koeficijenata IIR filtara, kao i reda veličina međurezultata množenja što znatno umanjuje grešku kvantizacije tih vrednosti.

1.4 Kaskadna realizacija IIR filtra upotrebom programskog jezika C

Kao što je objašnjeno, svi filtri višeg reda realizuju se kaskadnom vezom sekcija drugog reda. Implementacija sekcije drugog reda može se izvršiti na nekoliko načina. Prvi način jeste tzv. prva direktna forma predstavljena na slici 3. Prva direktna forma podrazumeva direktnu implementaciju jednačine prenosa *H(z)* ove sekcije. Ova implementacija podrazumeva pamcenje prethodna dva ulazna odbirka i prethodna dva izlazna odbirka iz sistema, svaki odbirak se množi odgovarajućim koeficijentom i rezultat se akumulira.



Slika 3 - Prva direktna forma *biquad* filtra

Ovakav pristup se najčešće koristi kod celobrojne implementacije IIR filtara. Sa ovakvom strukturom se postiže manja greška zbog preciznosti. Dovoljno je kao akumulator koristiti tip veće preciznosti.

Kod celobrojne implementacije koeficijenti filtra su predstavljeni kao brojevi u opsegu [-1, 1), skalirani na opseg celobrojnog tipa koji se koristi. Prilikom ditazjna filtra (računanja koeficijnata) pokazalo se da se dobija značajno bolja prenosna karakteristika ukoliko su koeficijenti *a1* I *b1* u opsegu [-2, 2). Jedan način da se to omogući jeste da se funkcija za filtriranje implementira tako da umesto koeficijenata *a1* i *b1* očekuje da joj se proslede polovine vrednosti ovih koeficijenata. Unutar tela funkcije množenje odgovarajućih odbiraka tim koeficijentima vršiće se dva puta.

Celobrojna implementacija IIR sekcije drugog reda u prvoj direktnoj formi data je na primeru ispod. Funkciji se kao parametri prosleđuju:

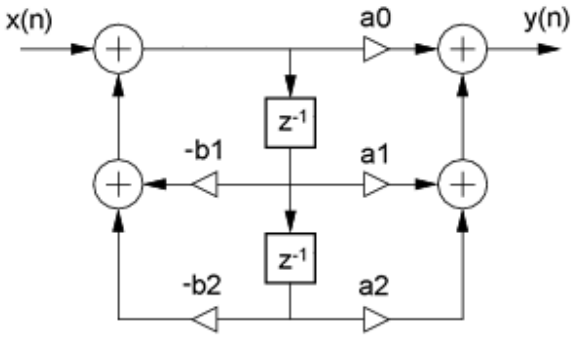
- *input* – vrednost ulaznog odbirka
- *coefficients* – niz sa koeficijentima filtra (*a0, a1, a2, b0, b1, b2*)
- *x_history* - memorija za pamćenje prethodnih ulaznih odbiraka (dužine 2)
- *y_history* - memorija za pamćenje prethodnih izlaznih odbiraka (dužine 2)

```
Int16 second_order_IIR(Int16 input, Int16* coefficients, Int16* x_history, Int16*
y_history) {
    Int32 accum = 0;

    accum = _smac(accum, coefficients[0], input);        /* A0 * x(n)     */
    accum = _smac(accum, coefficients[1], x_history[0]); /* A1/2 * x(n-1) */
    accum = _smac(accum, coefficients[1], x_history[0]); /* A1/2 * x(n-1) */
    accum = _smac(accum, coefficients[2], x_history[1]); /* A2 * x(n-2)   */
    accum = _smas(accum, coefficients[4], y_history[0]); /* B1/2 * y(n-1) */
    accum = _smas(accum, coefficients[4], y_history[0]); /* B1/2 * y(n-1) */
    accum = _smas(accum, coefficients[5], y_history[1]); /* B2 * y(n-2)   */

    accum >>= 16;

    y_history[1] = y_history[0];    /* y(n-2) = y(n-1) */
    y_history[0] = (Int16) (accum); /* y(n-1) = y(n)   */
    x_history [1] = x_history [0];  /* x(n-2) = x(n-1) */
    x_history [0] = input;          /* x(n-1) = x(n)   */

    return ((Int16) accum);
}
```

Prvi deo funkcije predstavlja množenje odbiraka odgovarajućim koeficijentima. Nakon toga vrši se skaliranje rezultata (uzimanje gornjih 16 bita promenljive accum). Poslednji korak predstavlja pomeranje odbiraka u memoriji, kako bi memorijski nizovi bili spremni za sledeću iteraciju filtriranja.

Podelimo tačke sumiranja na dva dela, levi koji sadrži ulazne odbirke pomnožene sa *a* koeficijentima, i desni deo koji sadrži izlazne odbirke pomnožene *b* koeficijentima. Zamenom mesta te dve polovine, memorijski nizovi skladište iste vrednosti, pa ih je moguće zameniti jednim memorijskim nizom. Na taj način dobija se **Druga direktna forma** *biquad* filtra. Ovakva implementacija štedi 2 memorijske lokacije u odnosu na prvu direktnu formu, međutim nepogodna je za celobrojnu implementaciju zbog veće mogućnosti prekoračenja opsega prilikom množenja. Druga direktna forma češće se koristi prilikom

implementacije IIR filtra drugog reda koristeći aritmetiku pokretnog zareza. Jedna implementacija data je na primeru ispod.



Slika 4 – Druga direktna forma *biquad* filtra

```
float second_order_IIR_f(float input, float* coefficients, float* history) {
    float accum = 0;
    float temp = 0;

    accum = input;
    accum -= coefficients[4]*history[0];
    accum -= coefficients[5]*history[1];

    temp = accum;

    accum = coefficients[0] * accum;
    accum += coefficients[1] * history[0];
    accum += coefficients[2] * history[1];

    history[1] = history[0];
    history[0] = temp;

    return accum;
}
```

Kaskadna realizacija filtra višeg reda prikazana je na primeru filtra četvrtog reda. Funkcija započinje smeštanjem ulaznog odbirka u promenljivu *temp*. Potom se 2 puta ponavlja petlja za filtriranje što predstavlja 2 povezana filtra drugog reda. Telo petlje za filtriranje jednako je prethodno prikazanoj implementaciji filtra drugog reda u prvoj direktnoj formi. Kao ulaz se koristi promenljiva *temp.* Nakon izvršenog filtriranja, poslednji korak u petlji jeste kopiranje trenutne vrednosti izlaza u promenljivu temp, kako bi ta vrednost predstavlja ulaz u sledeći filtar (sledeću iteraciju petlje).

```
Int16 fourth_order_IIR(Int16 input, Int16 coefficients[][6], Int16 x_history[][2],
        Int16 y_history[][2]) {
    Uint32 accum;
    Int16 temp;
    Uint16 stage;

    temp = input;
```

```
    for ( stage = 0 ; stage < 2 ; stage++)
    {
        accum = 0;
        accum = _smac(accum, coefficients[stage][0], temp);
        accum = _smac(accum, coefficients[stage][1], x_history[stage][0]);
        accum = _smac(accum, coefficients[stage][1], x_history[stage][0]);
        accum = _smac(accum, coefficients[stage][2], x_history[stage][1]);
        accum = _smas(accum, coefficients[stage][4], y_history[stage][0]);
        accum = _smas(accum, coefficients[stage][4], y_history[stage][0]);
        accum = _smas(accum, coefficients[stage][5], y_history[stage][1]);

        accum >>= 16;

        y_history[stage][1] = y_history[stage][0];
        y_history[stage][0] = (Int16) (accum);
        x_history[stage][1] = x_history[stage][0];
        x_history[stage][0] = temp;

        temp = (Int16)accum;
    }
    return (Int16)accum;
}
```

1.5 *All-pass* filtri

All-pass filtar predstavlja specifičnu strukturu IIR filtra koja se koristi kao gradivni modul raznih sistema za obradu signala. Nazvana je *all-pass IIR* iz razloga što vrši modifikaciju samo faze spektralnih komponenti ulaznog signala ne menjajući njihove amplitude. *All-pass* struktura se postiže definisanjem koeficijenata filtra tako da je polinom u brojiocu iste dužine kao i polinom u imeniocu P(z), i da su koeficijenti polinoma u brojiocu jednaki koeficijentima imenioca poređanim obrnutim redosledom.

$$H_z = \frac{\sum_{k=0}^{M-1} b(M-k)\cdot z^{-k} + z^{-M}}{1+\sum_{k=1}^{M} b(k)\cdot z^{-k}} = \frac{P(1/z)}{P(z)} \quad P(z) = 1+\sum_{k=1}^{M} b(k)\cdot z^{-k}$$

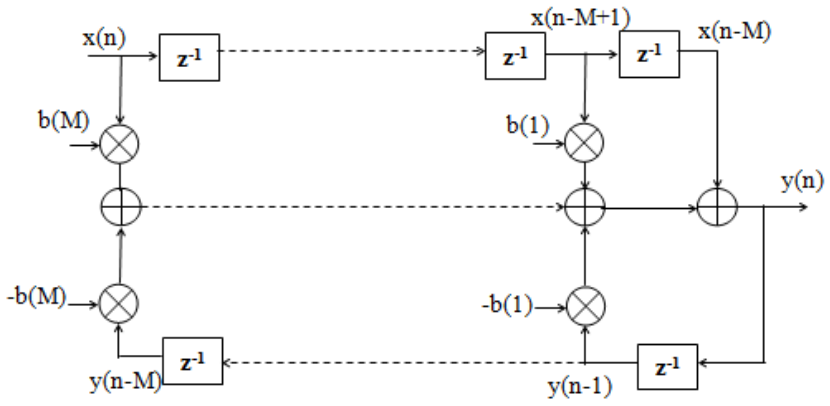
Amplitudna karakteristika *all-pass* filtra data je jediničnom:

$$A(f) = \left|H_z\left(e^{2j\pi f}\right)\right| = \frac{\left|P\left(e^{-2j\pi f}\right)\right|}{\left|P\left(e^{2j\pi f}\right)\right|} = 1 .$$

Fazna karakteristika je predstavljena sa:

$$\varphi(f) = \arg\left\{H_z\left(e^{2j\pi f}\right)\right\} = \arg\left\{P\left(e^{-2j\pi f}\right)\right\} - \arg\left\{P\left(e^{2j\pi f}\right)\right\} = -2\cdot\arg\left\{P\left(e^{2j\pi f}\right)\right\}.$$

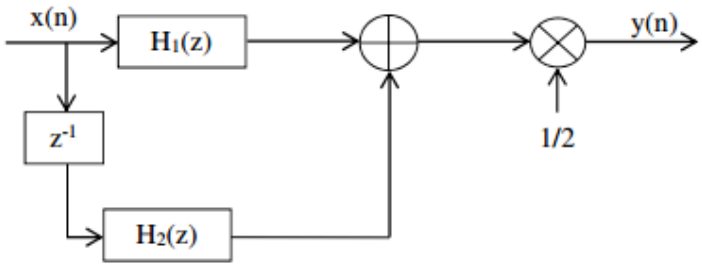
Amplitude ostaju nepromenjene, a faze spektralnih komponenti se menjaju u zavisnosti od frekvencije spektralne komponente.



Slika 5 - All-pass IIR struktura

1.6 *Half-band* struktura

Upotrebom IIR filtara može se izvršiti projektovanje specifične strukture nazvane *half-band* filtri (HB). Ova struktura se sastoji od paralelne veze dva *all-pass* (AP) filtra (*H1* i *H2*), pričemu je ulaz jednog pomeren u vremenu. Ideja je da AP filtri imaju slične fazne karakteristike u propusnom opsegu i fazne karakteristike pomerene za π u nepropusnom opsegu, tako da se njihovi izlazi sabiraju u propusnom opsegu, a oduzimaju u nepropusnom. Ova struktura je posebno pogodna za graničnu učestanost propusnog opsega na sredini osnovnog intervala f_g=0,25.



Slika 6 - *Half Band* struktura

Prenosna karakteristika HB filtra u *z*-transformaciji data je sa:

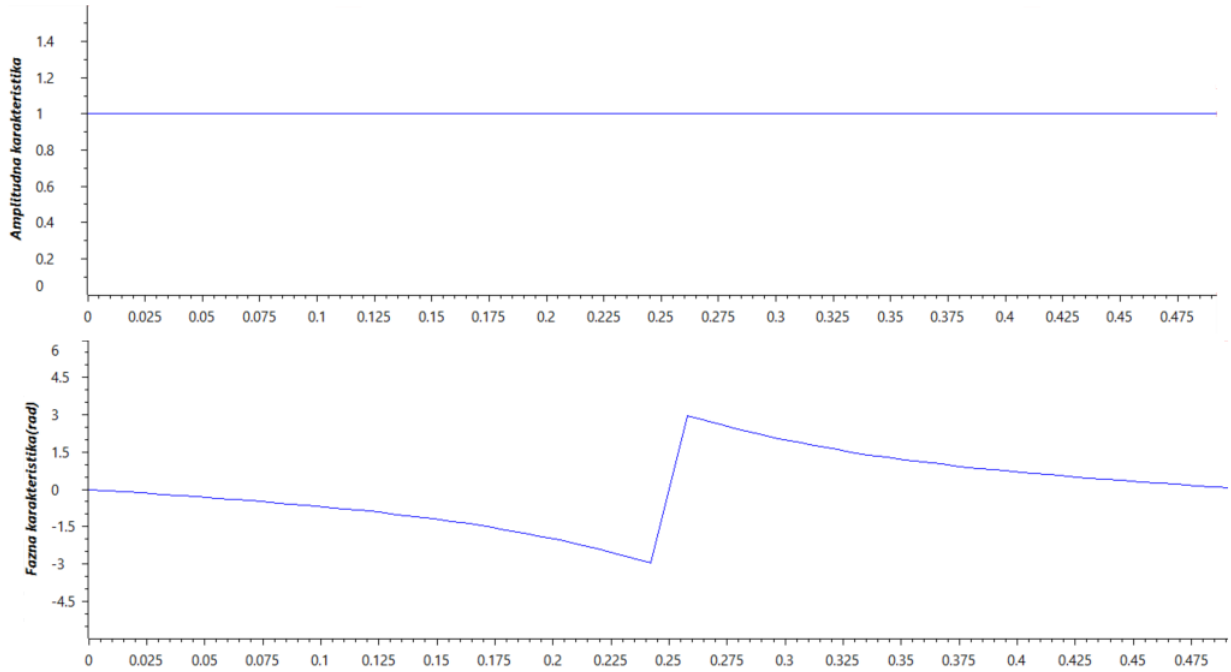
$$H_z(z) = \frac{1}{2}\left(H_1(z) + z^{-1}\cdot H_2(z)\right)$$

$$H_1(z) = \frac{\sum_{k=0}^{M-1} a(M-k)\cdot z^{-2k} + z^{-2M}}{1 + \sum_{k=1}^{M} a(k)\cdot z^{-2k}}$$

$$H_2(z) = \frac{\sum_{k=0}^{M-1} b(M-k)\cdot z^{-2k} + z^{-2M}}{1 + \sum_{k=1}^{M} b(k)\cdot z^{-2k}}$$

Lako je proveriti da ovakva struktura ima zahtevane vrednosti NF filtra za tri učestanosti:

$$\begin{aligned} f = 0 \Rightarrow z = 1 \quad & \Rightarrow |H_z(z)| = 1 \\ f = f_g = 1/4 \Rightarrow z = j \quad & \Rightarrow |H_z(z)| = \sqrt{2}/2 = 1/\sqrt{2}\ (-3\ dB)\,. \\ f = 1/2 \Rightarrow z = -1 \quad & \Rightarrow |H_z(z)| = 0 \end{aligned}$$



Slika 7 - Prenosna karakteristika *all-pass* filtra drgog reda

2 ZADACI

2.1 Zadatak 1

1. Uvući projektni zadatak 1 u radni prostor
2. Povezati *line out* izlaz vašeg računara na *line in* ulaz na razvojnoj ploči.
3. Na računaru reprodukovati datoteku *signal1.wav* (ulazna datoteka jednaka je ulaznoj datoteci iz prošlog zadatka.)
4. Filtrirati ulazni signal na levom kanalu upotrebom nisko frekventnog (NF) IIR filtra drugog reda (funkcija **`second_order_IIR`**). Koeficijenti filtra su dati u okviru zaglavlja *IIR_low_pass_filters.h*
5. Pokrenuti program, zaustaviti izvršenje upotrebom tačke prekida.
6. Iscrtati vrednosti izlaznih signala u vremenskom i frekventnom domenu, pre i nakon filtriranja.
7. Umesto signala sa AD konvertora dovesti na ulaz filtra delta impuls u trajanju od 128 odbiraka.
8. Prikazati Impulsni odziv filtra koristeći alat *Graph -> Single Time* i amplitudnu i faznu prenosnu karakteristiku koristeći alat *Graph -> FFT Magnitude Phase* nad dobijenim izlaznim signalom.

2.2 Zadatak 2

1. U okviru projektnog zadatka 1 data je funkcija koja predstavlja IIR filtar četvrtog reda implementiran kaskadnom vezom dva IIR filtra drugog reda:
   - `Int16 fourth_order_IIR(Int16 input, Int16 coefficients[][6], Int16 x_history[][2], Int16 y_history[][2])`
2. Filtrirati ulazni signal na desnom kanalu upotrebom niskopropusnog IIR filtra četvrtog reda. Koeficijenti filtra su dati u okviru zaglavlja *IIR_low_pass_filters.h*
3. Uporediti rezultate filtriranja sa signalom filtriranim korišćenjem filtra drugog reda.
4. Uporediti rezultate filtriranja sa rezultatima filtriranja istog signala sa istom graničnom frekvencijom upotrebom FIR filtra 32-og reda. (Vežba 7)
5. Uporediti kompleksnost i zahteve FIR I IIR filtra koji daju približne rezultate. (Broj koeficijenata, broj množenja, memorijski zahtevi).
6. Umesto signala sa AD konvertora dovesti na ulaz filtra delta impuls u trajanju od 128 odbiraka.
7. Prikazati Impulsni odziv filtra koristeći alat *Graph -> Single Time* i amplitudnu i faznu prenosnu karakteristiku koristeći alat *Graph -> FFT Magnitude Phase* nad dobijenim izlaznim signalom.
8. Ponoviti ceo postupak za visoko frekventni i pojasni filtar (koeficijenti se nalaze u zaglavljima *IIR_high_pass_filters.h* i *IIR_band_pass_filters.h*).

2.3 Zadatak 3

1. Uključiti zaglavlje sa koeficijentima *all pass* filtra drugog reda.
2. Filtrirati ulazni signal na levom kanalu upotrebom IIR filtra drugog reda koristeći koeficijente za *all pass* filtar (*allpass_2nd_order_1*)
3. Pokrenuti program, zaustaviti izvršenje upotrebom tačke prekida.
4. Prikazati amplitudnu i **faznu** karakterisku ulaznog i izlaznog signala upotrebom alata *Tools -> Graph -> FFT Magnitude Phase*.
5. Uporediti vrednosti i komentarisati rezultate.
6. Umesto signala sa AD konvertora dovesti na ulaz filtra delta impuls u trajanju od 128 odbiraka.

7. Prikazati Impulsni odziv filtra koristeći alat *Graph -> Single Time* i amplitudnu i faznu prenosnu karakteristiku koristeći alat *Graph -> FFT Magnitude Phase* nad dobijenim izlaznim signalom.

2.4 Zadatak 4

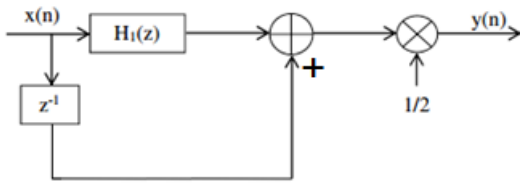
1. Implementirati funkciju:
   - `Int16 Nth_order_IIR(Int16 input, Int16 coefficients[][6], Int16 x_history[][2], Int16 y_history[][2], Int16 order)`

2. Koja sadrži implementaciju IIR filtra N-tog predstavljenu kaskadnom vezom N/2 filtera drugog reda. Prosleđeni parametar *order* određuje red filtra.
3. Ispravnost funkcije proveriti poređenjem impulsnog odziva i prenosne karakteristike filtra za *order*=4 sa prenosnom karakteristikom i impulsnim odzivom filtra realizovanog sa *fourth_order_IIR*, pri čemu je potrebno koristiti iste koeficijente.
4. Filtrirati ulazni signal na levom kanalu upotrebom nisko propusnog (NF) IIR filtra četvrtog reda, a na desnom kanalu upotrebom nisko propusnog IIR filtra šestog reda. Uporediti rezultate filtriranja.

2.5 Zadatak 5

1. U okviru funkcije:
   - `Int16 halfband(Int16 input, Int16* x_history, Int16* y_history, Int16 lowpass)`

Implementirati strukturu datu na slici:



2. Za filtriranje koristiti *all-pass* filtar iz zadatka 3.
3. Parametar *lowpas* označava da li je u pitanju niskopropusni ili visokopropusni filtar. Ukoliko je vrednost ovog parametra različita od 0, potrebno je sabrati zakasneli odbirak sa izlazom iz filtra, a ukoliko je jednaka nuli, potrebno ga je oduzeti.
4. Nakon filtriranja zakasneli odbirak se nalazi u x memorijskom nizu na lokaciji z_x[1], tak oda nije potrebno praviti dodatni memorijski niz za kašnjenje.

3 Zaključak

U okviru ove vežbe upozali ste se strukturom digitalnih filtara sa beskonačnim odzivom. Uvideli ste da su prednosti ovakvih filtara postizanje znatno boljih performansi za filtre istog reda u odnosu na filtre sa konačnim odzivom. Upoznali ste se sa nedostacima ovih filtara, koji ujedno i predstavljaju razloge zbog kog nemaju uvek prednost u odnosu na filtere sa konačnim odzivom (iako su znazno efikasniji). Naučili ste kako je moguće iskoristiti nelinearnu faznu karakteristiku filtara sa beskonačnim odzivom za promenu fazne karakteristike signala bez promene amplitude. Na primeru realnog sistema za obradu zvuka pokazana je upotreba ovakvih filtara.