Ciele cvičenia

V rámci cvičenia sa budeme zaoberať číslicovými filtrami s nekonečnou impulzovou odozvou (IIR – Infinite Impulse Response). Opíšeme základné štruktúry týchto filtrov: priamu formu I. a II., kaskádne a paralelné zapojenie. Vysvetlíme použitie bilineárnej transformácie pri návrhu filtrov a naučíme sa pracovať so špecializovaným nástrojom SPTOOL® programu MATLAB®, určeným pre rýchly a jednoduchý návrh číslicových filtrov. Na záver cvičenia, s využitím nástroja SPTOOL navrhneme IIR filter podľa zadania, implementujeme ho na vývojovej doske TMS320C6713 DSK a zmeriame jeho parametre.

1. Úvod

Ako už vieme, správanie číslicových filtrov opisujeme diferenčnými rovnicami. V prípade filtrov IIR má táto rovnica nasledujúci všeobecný rekurzívny tvar

$$\begin{aligned} y(n) &= \sum_{k=0}^{N} a_k x(n-k) - \sum_{j=1}^{M} b_j y(n-j) = \\ &= a_0 x(n) + a_1 x(n-1) + \ldots + a_n x(n-N) - b_1 y(n-1) - \ldots - b_M y(n-M) \end{aligned} \tag{10.1}$$

Vidíme, že postupnosť výstupných vzoriek y(n) závisí nielen od postupnosti vstupných vzoriek x(n), x(n-1), x(n-2),..., x(n-N), ale aj od postupnosti minulých výstupných vzoriek y(n-1), y(n-2),..., y(n-M). IIR fltre majú teda zavedenú spätnú väzbu.
Ak budeme predpokladať, že všetky počiatočné podmienky v rovnici (10.1) sú rovné nule, potom po z-transformácii tejto rovnice dostávame

$$\begin{aligned} Y(z) &= a_0 X(z) + a_1 z^{-1} X(z) + \ldots + a_N z^{-N} X(z) - \\ &- b_1 z^{-1} Y(z) - b_2 z^{-2} Y(z) - \ldots - b_M z^{-M} Y(z) \end{aligned} \tag{10.2}$$

Nech v rovnici (12.2) N=M, potom prenosová funkcia H(z) bude

$$H(z) = \frac{Y(z)}{X(z)} = \frac{a_0 + a_1 z^{-1} + \ldots + a_N z^{-N}}{1 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_M z^{-M}} = \frac{N(z)}{D(z)} \tag{10.3}$$

kde N(z) a D(z) reprezentujú polynómy čitateľa a menovateľa prenosovej funkcie. Ak vynásobíme čitateľa aj menovateľa z^N, dostávame

$$H(z) = \frac{a_0 z^N + a_1 z^{N-1} + \ldots + a_N}{z^N + b_1 z^{N-1} + b_2 z^{N-2} + \ldots + b_N} = C \prod_{i=1}^{N} \frac{z - z_i}{z - p_i} \tag{10.4}$$

Výraz (10.4) predstavuje prenosovú funkciu s N nulami a N pólmi. Ak by sme všetky koeficienty b_j v rovnici (10.4) položili rovné nule, dostali by sme prenosovú funkciu s N pólmi v počiatku z-roviny, ktorá by reprezentovala FIR filter. Aby bol systém stabilný, musia ležať všetky póly vo vnútri jednotkovej kružnice. Odtiaľ pre podmienku stability IIR filtra vyplýva, že absolútna hodnota všetkých pólov jeho prenosovej funkcie musí byť menšia ako 1, teda

1. ak $|P_j| < 1$, potom $h(n) \rightarrow 0$, keď $n \rightarrow \infty$, a IIR filter je stabilný

2. ak $|P_j|>1$, potom $h(n) \rightarrow \infty$, keď $n \rightarrow \infty$, a IIR filter je nestabilný

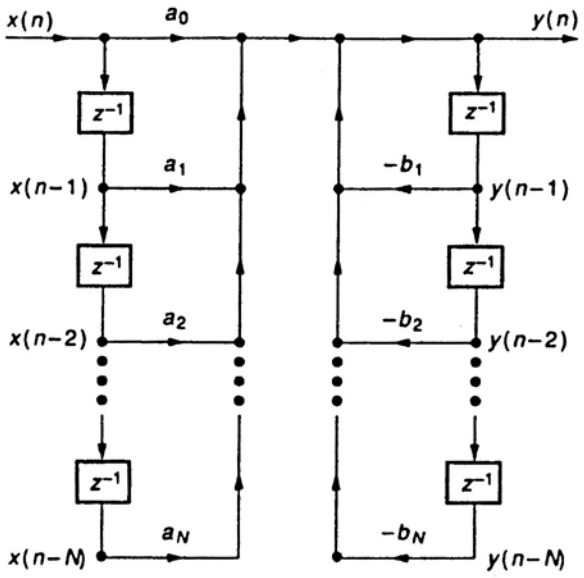
Ak $|P_j|=1$, systém je na hranici stability a správa sa ako oscilátor. Ak sa na jednotkovej kružnici nachádzajú viacnásobné póly, jedná sa o nestabilný systém. Ak sú všetky koeficienty b_j rovné nule dostávame nerekurzívny stabilný systém - FIR filter.

2. Štruktúra IIR filtra

Existuje niekoľko štruktúr, ktoré umožňujú realizáciu IIR filtrov. Označujeme ich príznačne: priama forma I. a priama forma II. Priame formy II. je možné transponovať a kombinovať v paralelnej alebo kaskádovej štruktúre tak, aby sme získali filtre vyšších rádov. Tieto štruktúry a ich kombinácie teraz postupne opíšeme.

Priama forma I.

Ak sa rozhodneme realizovať IIR filter presne podľa výrazu pre jeho prenos (10.3), bude jednoznačne daný štruktúrou podľa obr.1. Pre filter N-tého rádu bude mať štruktúra 2N oneskorovacích členov, reprezentovaných pri z-transformácii výrazom z^{-1}.



Obr.1. Štruktúra IIR filtra – priama forma I.

Priama forma II.

Predchádzajúca štruktúra IIR filtra síce bezpochyby umožňuje jeho implementáciu, je však výpočtovo náročná z hľadiska počtu oneskorovacích členov a sumátorov. Preto sa v praxi oveľa častejšie využíva priama forma II., ktorú vidíme na obr.2. Je zrejmé, že vyžaduje o polovicu menej oneskorovacích členov ako priama forma I., teda ich počet presne zodpovedá rádu filtra. Už menej zrejmé je, že aj táto štruktúra predstavuje IIR filter s prenosovou charakteristikou podľa (10.3). Našou úlohou bude ukázať, že tomu tak v skutočnosti je. Pre tento účel zavedieme pomocnú premennú U(z), definovanú vzťahom

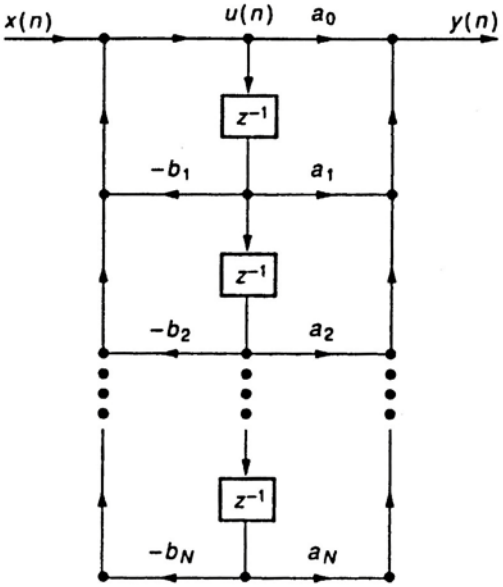
$$U(z) = \frac{X(z)}{D(z)} \tag{10.5}$$

kde D(z) je polynóm v menovateli prenosovej funkcie IIR filtra (10.3). Zo vzťahov (10.3) a (9.5) dostávame

$$Y(z) = \frac{N(z)X(z)}{D(z)} = N(z)U(z) = U(z)\left(a_0 + a_1 z^{-1} + a_2 z^{-2} + ... + a_N z^{-N}\right) \tag{10.6}$$

kde N(z) je polynóm čitateľa prenosovej funkcie IIR filtra (10.3). Zo vzťahu (10.5) dostávame

$$X(z) = U(z)D(z) = U(z)\left(1 + b_1 z^{-1} + b_2 z^{-2} + ... + b_N z^{-N}\right) \tag{10.7}$$



Obr.2. Štruktúra IIR filtra – priama forma II.

Inverznou transformáciou vzťahu (9.7) dostávame

$$x(n) = u(n) + b_1 u(n-1) + b_2 u(n-2) + ... + b_N u(n-N) \tag{10.8}$$

Z predchádzajúceho vzťahu pre premennú u(n) vyplýva

$$u(n) = x(n) - b_1 u(n-1) - b_2 u(n-2) - ... - b_N u(n-N) \tag{10.9}$$

Inverznou transformáciou vzťahu (10.6) dostávame

$$y(n) = a_0 u(n) + a_1 u(n-1) + a_2 u(n-2) + ... + a_N u(n-N) \tag{10.10}$$

Existujú samozrejme, ďalšie štruktúry vhodné pre aplikácie v oblasti adaptívnych filtrov, rozoznávania hlasu, atď. Tieto sú však obvykle výpočtovo náročnejšie než priama forma II. a vyžadujú viac operácií násobenia. Ich výhodou je však nižšia citlivosť na efekt kvantovania.

Kvantizačná chyba súvisiaca s koeficientami IIR filtra závisí od veľkosti zmeny pozície núl a pólov jeho prenosovej funkcie v komplexnej rovine. Zmena pozície jedného pólu závisí od pozície všetkých ostatných pólov. Aby sme túto závislosť pólov minimalizovali, navrhujeme obvykle IIR filtre ako kaskádu štruktúr druhého rádu na báze priamej formy II.

Zadanie

1. Preštudujte si tteoretický úvod k problematike.
2. Prostredníctvom špecializovaného nástroja SPTOOL (signal Processing Toolbox) programu MATLAB navrhnite jeden z nasledujúcich IIR filtrov podľa zadania
   a. dolnopriepustný filter - lowpass (N = 10, f_c = 1,5kHz),
   b. hornopriepustný filter - highpass (N = 10, f_c = 1,5kHz),
   c. pásmová zádrž - bandstop (N = 10, f_1 = 1,5kHz, f_2 = 2,5kHz),
   d. pásmová priepusť -bandpass (N = 10, f_1 = 1,5kHz, f_2 = 2,5kHz),

   Zvlnenie v priepustnom pásme uvažujte 3dB, tlmenie uvažujte 50dB.
3. Získané koeficienty IIR filtra spracujte do textového súboru *typ_filtra.cof*.
4. V prostredí CCS otvorte projekt *iir.pjt*, upravte súbor *iir.c* tak, aby vkladacia direktíva *#include* vložila do programu súbor s vašimi koeficientami. Na prilinkovanie súboru s koeficientmi použite "Scan All Dependencies" v hlavnom menu.
5. Projekt uložte a vykonajte kompiláciu. Výsledný objektový súbor *iir.out* uložte do pamäte procesora a spustite program. Pomocou osciloskopu a funkčného generátora odmerajte jeho charakteristiky.

Poznámky k zadaniu

K úlohe 2)

1. V príkazovom okne Matlabu napíšeme "sptools" a spustíme rovnomenný program rpe interaktívny návrh číslicových filtrov.
2. V úvodnom okne "startup" zvolíme položku "New Design", ćím otvoríme okno "Filter Designer", v ktorom zvolíme typ filtra a algoritmu, jeho parametre a vzorkovaciu frekvenciu (f_s = 8kHz)
3. Po zadaní parametrov filtra uskutočníme jeho výpočet
4. Po skončení výpočtu prejdeme do okna "startup", kde vyberieme "Edit→ Name" a zmeníme štandardné meno súbora podľa typu filtra (lp, hp, bs, bp). Vyberieme "File → Export → export to workspace" , čím uložíme Struktúru navrhovaného filtra pre ďalšie použitie v matlabe.
5. V príkazom okne Matlabu zadáme nasledujúce príkaze (napr. pre LP filter)

   ```
   [z,p,k] = tf2zp(lp.tf.num, lp.tf.den);
   coef = zp2sos(z,p,k);
   coef = round(coef*2^15);
   ```

   Podrobnú špecifikáciu týchto funkcií nájdete v časti HELP programu Matlab, teda popis len stručný: Funckia *tf2zp* realizuje výpočet núl (z) pólov (p) a konštánt (k) prenosovej funkcie. Tieto výsledky ďalej spracujeme pomosou *zp2sos*, ktorá vypočíta koeficienty filtra realizovaného sekciami druhého rádu. Napokon získané koeficient ešte normujeme 32768 = 2^15 a zaokrúhlime na najbližšie prirodzené číslo, keďže procesor pracuje len s celými číslami a nie zlomkami.

K úlohe 3)

Vytvorte textový súbor podľa nasledujúcej predlohy
Upozornenie:
Koeficient [a, b] získané z matlabu, zodpovedajú koeficientom [b, a] v tomto súbore.

```
#define stupen 5

int a [stupen][3]={                    // koeficienty citatela
        {a10, a11, a12}                // prvy stupen
        {a20, a21, a22}                // druhy stupen
        {a30, a31, a32}                // treti stupen
        {a40, a41, a42}                // stvrty stupen
        {a50, a51, a52}                // piaty stupen
        };

int b [stupen][3]={                    // koeficienty citatela
        {b11, b12}                     // prvy stupen
        {b21, b22}                     // druhy stupen
        {b31, b32}                     // treti stupen
        {b41, b42}                     // stvrty stupen
        {b51, b52}                     // piaty stupen
        };
```

K úlohe 4)

Preštudujte výpis súboru *iir.c*:

```
//IIR.c  IIR filter using cascaded Direct Form II
//Coefficients a's and b's correspond to b's and a's from MATLAB

#include "DSK6713_AIC23.h"                  //codec-DSK support file
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;          //set sampling rate
#include "bs1750.cof"           //BS @ 1750 Hz coefficient file
short dly[stages][2] = {0};    //delay samples per stage

interrupt void c_int11()          //ISR
{
 short   i, input;
 int     un, yn;

 input = input_sample();          //input to 1st stage
 for (i = 0; i < stages; i++)     //repeat for each stage
 {
  un=input-((b[i][0]*dly[i][0])>>15) - ((b[i][1]*dly[i][1])>>15);

  yn=((a[i][0]*un)>>15)+((a[i][1]*dly[i][0])>>15)+((a[i][2]*dly[i][1])>>15);

  dly[i][1] = dly[i][0];           //update delays
  dly[i][0] = un;                  //update delays
```

```
  input = yn;                    //intermediate output->input to next stage
 }
  output_sample((short)yn);  //output final result for time n
  return;                    //return from ISR
}

void main()
{
  comm_intr();               //init DSK, codec, McBSP
  while(1);                  //infinite loop
}
```