

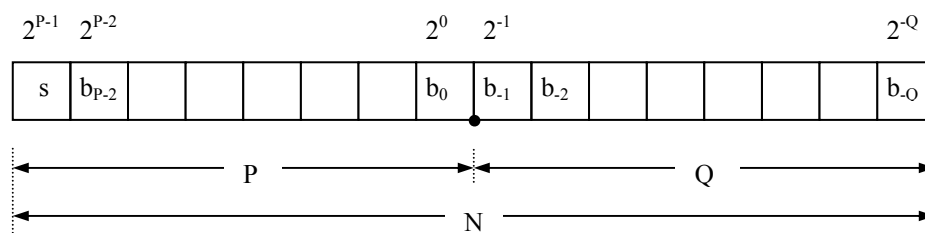
6. A véges szóhosszúság

A digitális jelfeldolgozó processzorok (DSP-k) sokféle szempont szerint osztályozhatók. Az egyik ilyen szempont lehet az, hogy a bennük lévő aritmetikai egység milyen elven működik, nevezetesen: fixpontos vagy lebegőpontos a műveletvégzés. Egy másik szempont lehet az, hogy az adatbusz (és a hozzá kapcsolódó memória) milyen széles (16,24,32 bit). A lebegőpontos processzorok utasítás készletében általában megtalálhatók a fixpontos műveletvégzés utasításai, míg fordítva ez nem áll fenn.

Ebben a fejezetben a **fixpontos aritmetikai egységek** kérdéseivel fogunk foglalkozni, feltételezve, hogy az adatbusz 16 bit széles. Természetesen eredményeink értelemszerűen más adatbusz szélességekre is érvényesek maradnak.

6.1. A kettes komplement kód

Fixpontos aritmetikában általánosan elterjedt a kettes komplement kód használata (Two's Complement code = TC). Ez amiatt van így, mivel a teljes bináris összeadó-kivonó áramkörök (ALU-k) ezen kód esetén adják előjel helyesen az eredményt.



6.1. ábra A kettes komplement kód helyértékei

A kódot két mennyiséggel jellemezhetjük: N a szóhosszúság, Q a törtrész bitjeinek a száma. Hangsúlyozzuk, hogy ebből a két adatból csak az N a processzor jellemző, a Q a mindenkor programozó adatstruktúra választásától függ.

A program dokumentálásakor a szokásos jelölés pld: 16 TC Q15 ami 16 bites kettes komplement kódot jelent, melyben a tört rész 15 biten kerül ábrázolásra

A kettes komplement kódban az ' x ' mennyiséghez az alábbi definíció szerint rendeljük hozzá a kódot:

$$x = -s2^{P-1} + \sum_{k=0}^{P-2} b_k 2^k + \sum_{k=-1}^{-Q} b_k 2^k = -s2^{P-1} + \sum_{k=-Q}^{P-2} b_k 2^k \quad (6.1.)$$

ahol az ' s ' előjelbit (sign bit) definíciója:

$$s = \begin{cases} 0 & \text{ha } x \geq 0 \\ 1 & \text{ha } x < 0 \end{cases} \quad (6.2.)$$

Véges szóhosszúság esetén mindig van egy legnagyobb és egy legkisebb szám, ami az adott kódban még ábrázolható:

$$x_{\min} \leq x \leq x_{\max} \quad (6.3.)$$

Kettes komplementes kódban ezek:

$$x_{\max} = 2^{P-1} - 2^{-Q} = 2^{N-Q-1} - 2^{-Q} = (2^{N-1} - 1)2^{-Q} = (2^{N-1} - 1)q \quad (6.4.)$$

$$x_{\min} = -2^{P-1} = -2^{N-Q-1} = -2^{N-1}2^{-Q} = -2^{N-1}q \quad (6.5.)$$

ahol q a legalacsonyabb helyérték (LSB) értéke:

$$q = 2^{-Q} \quad (6.6.)$$

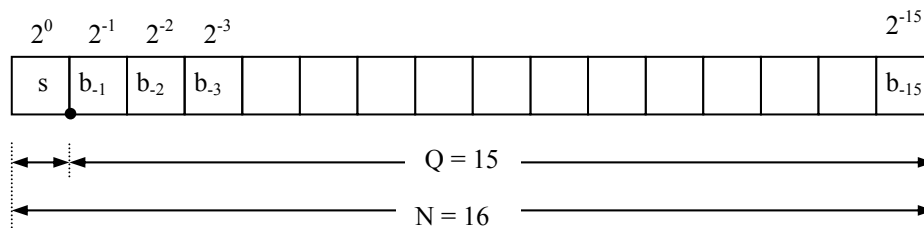
Minden kettes komplementes kódú tört szám kifejezhető az LSB egész számú többszöröseként és gyakran így hivatkozunk azok értékére:

$$x = -s2^{P-1} + \sum_{k=-Q}^{P-2} b_k 2^k = \left(-s2^{N-1} + \sum_{k=0}^{N-2} b_k 2^k \right) 2^{-Q} = Xq \quad (6.7.)$$

A fixpontos aritmetikákban a leggyakrabban használt formátum a: **16 TC Q 15**. Ebben a formátumban a számok a:

$$\begin{aligned} -1 \leq x \leq 1 - 2^{-15} \\ -32768 \leq X \leq +32767 \end{aligned} \quad (6.8.)$$

intervallumban helyezkednek el.



6.2. ábra A 16 TC Q 15 formátum helyértékei

Példaként határozzuk meg két szám kódját:

$$\begin{aligned} x_1 = 0.625_{10} &= 0 * 2^0 + 0.5_{10} + 0.125_{10} = 0 * 2^0 + 1 * 2^{-1} + 0 * 2^{-2} + 1 * 2^{-3} = \\ &= 0101\ 0000\ 0000\ 0000_{TC} = 5000h\ (Q15) \end{aligned}$$

$$x_2 = -0.625_{10} = -1 * 2^0 + \sum_{k=-1}^{-15} b_k 2^k \quad (s = 1)$$

amiből:
$$\sum_{k=-1}^{-15} b_k 2^k = 1 - 0.625_{10} = 0.375_{10} = 0.25_{10} + 0.125_{10} = 0 * 2^{-1} + 1 * 2^{-2} + 1 * 2^{-3}$$

$$x_2 = 1011\ 0000\ 0000\ 0000_{TC} = B000h\ (Q15)$$

Egy másik módszerrel:

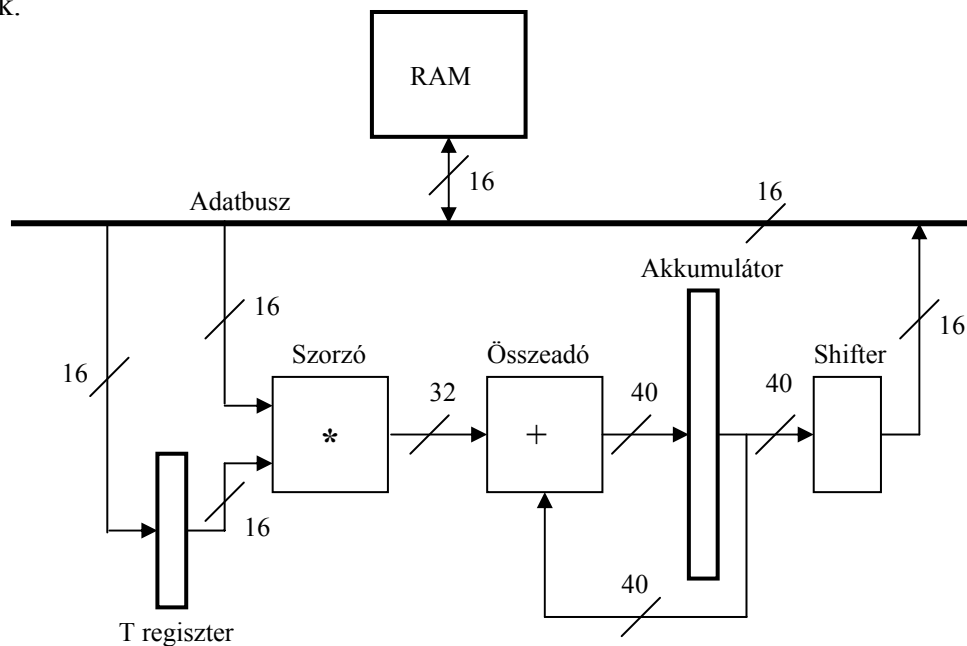
$$\begin{array}{r} x_1 = 0101\ 0000\ 0000\ 0000 \\ \bar{x}_1 = 1010\ 1111\ 1111\ 1111 \\ + \quad 0000\ 0000\ 0000\ 0001 \\ \hline x_2 = 1011\ 0000\ 0000\ 0000 \end{array}$$

6.2. A fixpontos aritmetikai egység

A digitális jelfeldolgozó processzorok aritmetikai egységét úgy alakították ki, hogy a (6.9.) alakú differencia egyenleteket a lehető legnagyobb hatékonysággal legyenek képesek kiszámítani.

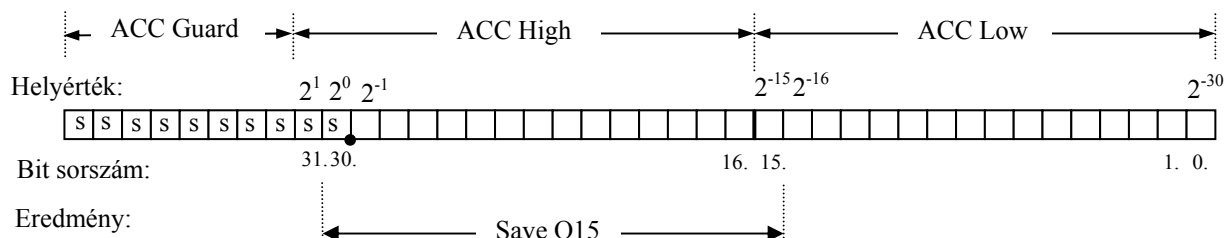
$$y(n) = \sum_{k=0}^M a_k x(n-k) + \sum_{k=1}^N (-b_k) y(n-k) \quad (6.9.)$$

A fenti összefüggés szerint szorzatokat kell összeadni. Az aritmetikai egység egy ciklusban végezze ezeket a műveleteket úgy, hogy miközben egy szorzást végez, az alatt az idő alatt az előző szorzatot akkumulálja. (Pipe-line architektúra: az aritmetikai fokozatok regiszterek között helyezkednek el.) A 6.3. ábrán egy tipikus DSP aritmetikai egységet láthatunk.



6.3. ábra Fixpontos DSP aritmetikai egysége

Elemezzük a helyértékek szempontjából a fenti aritmetikai egységet! Tételezzük fel, hogy minden szám (a konstansok és a jelminták) egyformán Q 15 formátumúak. Ekkor a szorzatot tartalmazó akkumulátor formátuma a 6.3. ábrának megfelelő lesz.



6.4. ábra Az akkumulátor formátuma, ha az operandusok Q 15 formátumúak

Ha az eredményt is Q 15 formátumban kívánjuk eltárolni (mert a következő ütemben már ez az adat is bemenő adat lesz), láthatóan az ACC(30...15) helyi értékeket kell a "Shifter"-en keresztül a memóriába elmenteni.

Ha a bemenő adatok formátuma más, akkor az eredmény helyi érték kiosztása is megváltozik. Minden formátumra a programozónak kell megválasztania az eredmény mentésének módját, ugyanis ezen a lépésen múlik az eredmény formátuma.

Mivel a mentés nem terjed ki a teljes akkumulátorra, felmerül a kérdés: jelent-e problémát az akkumulátorban maradt (nem elmentett) bitek elhagyása?

Ha az algoritmus és a hozzá illeszkedő adatformátumok jól lettek megtervezve, akkor az eredmény “belefér” az elmentett adat formátumába, azaz nem lép fel túlcsoordulás. A kettes komplement kódjának sajátossága, hogy az előjelbit “balra kiterjesztődik”. Példánkban ez azt jelenti, hogy a 30-ik és a 31-ik bit azonos lesz. Tehát ha az akkumulátor legmagasabb helyi érték bitjét (az MSB-t, ami a mindenkori előjelbit) nem mentjük el, az nem okoz gondot. Ha túlcsoordulás lép fel, a két bit különböző lesz, így az elmentett adat dűrva torzítást szenved (ellenkező előjelű lesz!).

Itt jegyezzük meg, ha a 32 bites akkumulátorba 16 bites adatot töltünk be egy adott pozícióba, akkor az előjel kiterjesztés balra automatikusan végrehajtódik, míg jobbra zérusokkal tölti fel a processzor az akkumulátort (amivel nem változtatja meg a szám értékét, függetlenül attól, hogy a szóhosszúság megváltozott).

Más a helyzet az akkumulátor alsó helyi értékein hagyott bitekkel. Szándékunkkal ellentétben a program nem a (6.1.) egyenlet szerinti eredményt fogja kiszámítani, hanem a (6.10.) szerinti mennyiséget.

$$y'(n) = y''(n) + e(n) = \sum_{k=0}^M a_k x(n-k) + \sum_{k=1}^N (-b_k) y''(n-k) \quad (6.10.)$$

A fenti összefüggésben $y'(n)$ a 32 bites akkumulátor tartalmát, $y''(n)$ az elmentett eredményt és $e(n)$ az akkumulátorban hagyott mennyiséget jelöli. A különbség abból is adódik, hogy már a bemenő adatok sem azonosak, mivel a korábbi ütemekben történt mentésekben az $y''(n-k)$ minták sem voltak pontosak. Az alsó bitek elhagyásából (a jel újra kvantálásából) adódó hiba részletes analizisével majd a 8. Fejezetben fogunk foglalkozni.

Ha az eredmény kiszámítása után rögtön elmentjük az akkumulátorból a formátumnak megfelelő biteket, akkor azt mondjuk, hogy az eredményt csonkoltuk (*truncation*). Az akkumulátorban maradt hiba jel ilyenkor mindig pozitív:

$$\text{Csonkolás:} \quad 0 \leq e(n) < +q \quad (6.11.)$$

A hibajelre egyenletes eloszlást feltételezve, a hiba várható értéke:

$$E\{e(n)\} = \frac{q}{2} \quad (6.12.)$$

A (6.10.)-ből

$$y''(n) = y'(n) - e(n) \quad (6.13.)$$

és így az eredmény várható értéke zérustól különböző (negatív) érték lesz, ha $E\{y'(n)\} = 0$

$$E\{y''(n)\} = E\{y'(n) - e(n)\} = E\{y'(n)\} - E\{e(n)\} = -\frac{q}{2} \quad (6.14)$$

Ha a mentés előtt a $q/2$ mennyiséget hozzáadjuk az akkumulátorhoz (Q 15 formátumban a 14. bit pozícióhoz egy bináris 1-et adunk, 2^{-16} helyérték, lásd 6.4. ábra), akkor beszélünk kerekítésről (rounding). Ekkor az $y''(n)$ mentett adat várható értéke meg fog egyezni az $y'(n)$ várható értékével, azaz a hiba várható értéke zérus lesz.

Kerekítés esetén:

$$-\frac{q}{2} \leq e(n) < +\frac{q}{2} \quad (6.15.)$$

Szokásos eljárás még az u.n. magnitúdó csonkolás (magnitude truncation), amikor is a kiszámított mintát a zérus irányába eső kvantálási lépcsőre kerekítjük. Pozitív minta esetén ez csonkolás alkalmazását, negatív minta esetén pedig a fentebb leírt kerekítési eljárás alkalmazását jelenti.

Ekkor:

$$-q < e(n) < +q \quad (6.16.)$$

6.2. A véges szóhosszúság hatása a szűrők átviteli karakterisztikájára

A véges szóhosszúság nem teszi lehetővé, hogy a realizált szűrőkben az elvileg pontos együtthatók értékeit állítsuk be, mivel azok pontos számábrázolásához általában végtelen szóhosszúság kellene. Ha csak az együtthatók pontatlanságát tekintjük, akkor a hatás az lesz, hogy a szűrő átviteli karakterisztikája egy "kicsit más lesz". Célunk ezen eltérés megbecslése.

1. FIR szűrők

Jelöljük a tervezési eljárás során meghatározott (az elvileg pontosnak tekintett) együtthatók értékét h_k^0 -val. A formátum megválasztása után ki kell számítanunk a véges szóhosszúságnak megfelelő értéket:

$$h_k = q \left[\text{round} \left(\frac{h_k^0}{q} \right) \right] \quad (6.17.)$$

ahol a $\text{round}(\cdot)$ függvény a legközelebbi egészre történő kerekítést jelöli. A k -ik együttható kerekítésénél elkövetett hiba:

$$e_k = h_k - h_k^0 \quad (6.18.)$$

melyre

$$-\frac{q}{2} \leq e_k < \frac{q}{2} \quad (6.19.)$$

Számítsuk ki az elvileg pontos és a valóságos együtthatókkal felépített FIR szűrő átviteli karakterisztikáinak különbségét, illetve ennek abszolút értékét.

$$\begin{aligned} E(\omega) &= |H(\omega) - H^0(\omega)| = \left| \sum_{k=0}^{N-1} h_k e^{-j\omega kT} - \sum_{k=0}^{N-1} h_k^0 e^{-j\omega kT} \right| = \left| \sum_{k=0}^{N-1} (h_k - h_k^0) e^{-j\omega kT} \right| = \\ &= \left| \sum_{k=0}^{N-1} e_k e^{-j\omega kT} \right| \leq \sum_{k=0}^{N-1} |e_k| |e^{-j\omega kT}| = \sum_{k=0}^{N-1} |e_k| \leq \sum_{k=0}^{N-1} \frac{q}{2} = \frac{q}{2} N \end{aligned} \quad (6.20.)$$

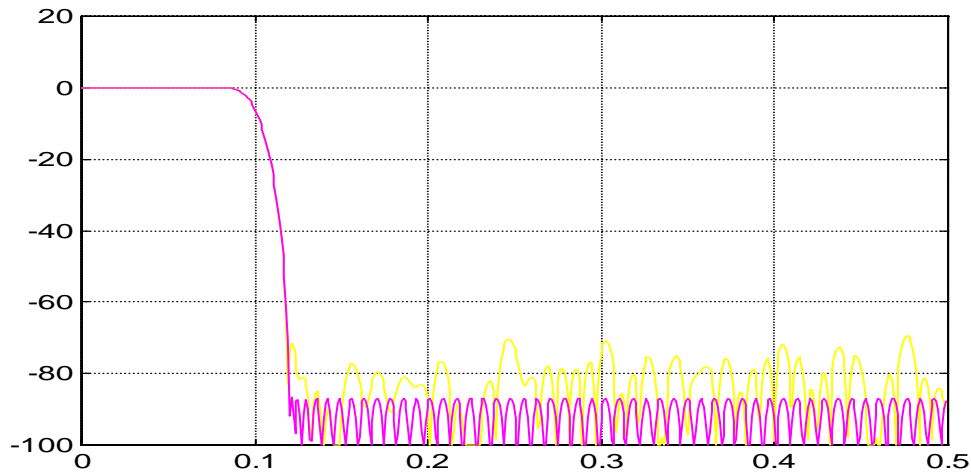
A (6.20.) relációban a legrosszabb esetet (worst case) tételeztük fel, mely mellett:

$$E(\omega) = |H(\omega) - H^0(\omega)| \leq \frac{q}{2} N \quad (6.21.)$$

Tételezzük fel, hogy a szűrő fokszáma $N = 64 = 2^6$ és a választott formátum Q 15 ($q = 2^{-15}$). Ebben az esetben a maximális hiba:

$$20 \log(E) \leq 20 \log\left(\frac{2^{-15}}{2} 2^6\right) = 20 \log(2^{-10}) = 20 \log\left(\frac{1}{1024}\right) \approx -60 \text{ dB}$$

értéket is elérhet, ami meglehetősen nagy.



6.5. ábra 64-ed fokú FIR szűrő átviteli karakterisztikája ideális és kerekített együtthatókkal

2. Direkt struktúrájú IIR szűrők együttható érzékenysége

IIR szűrők esetében az együtthatók pontatlanságának az átviteli karakterisztikára gyakorolt hatását meglehetősen nehéz kiértékelni a sok paraméter miatt. Ezért az IIR szűrők esetében az átviteli függvény pólusainak és zérusainak együttható érzékenységét fogjuk vizsgálni. Ha ez az érzékenység nagy, akkor ez azt jelenti, hogy a realizált szűrő átviteli függvénye jelentősen eltérhet a specifikációt kielégítő, elvileg pontos függvénytől.

Legyen az IIR szűrő transzfer függvénye:

$$H(z) = \frac{A(z)}{B(z)} \quad (6.22.)$$

Ahol a számláló:

$$A(z) = \sum_{k=0}^N a_k z^{-k} = a_0 \prod_{m=1}^N (1 - z_m z^{-1}) \quad (6.23.)$$

és a nevező:

$$B(z) = 1 + \sum_{k=1}^N b_k z^{-k} \quad (6.24.)$$

gyöktényezős alakban:

$$B(z) = \prod_{i=1}^N (1 - p_i z^{-1}) \quad (6.26.)$$

A számításokat az átviteli függvény pólusaira fogjuk elvégezni (A zérusokra a számítások teljesen azonosak).

A pólusokat a (6.24.) egyenlet gyökeiként határozhatjuk meg. A gyökök az együtthatóknak a függvényei (gyökmegoldó képlet):

$$p_i = f(b_1, b_2, \dots, b_k, \dots, b_N) \quad (6.28.)$$

Az elvi pontos együtthatókból az elvi pontos gyökök határozhatóak meg:

$$p_i^0 = f(b_1^0, b_2^0, \dots, b_k^0, \dots, b_N^0) \quad (6.29.)$$

A fenti N változós skalár függvény megváltozását közelíthetjük a teljes differencia kiszámításával

$$\Delta p_i = p_i - p_i^0 \cong \sum_{k=1}^N \frac{\partial p_i}{\partial b_k} \Delta b_k \quad (6.30.)$$

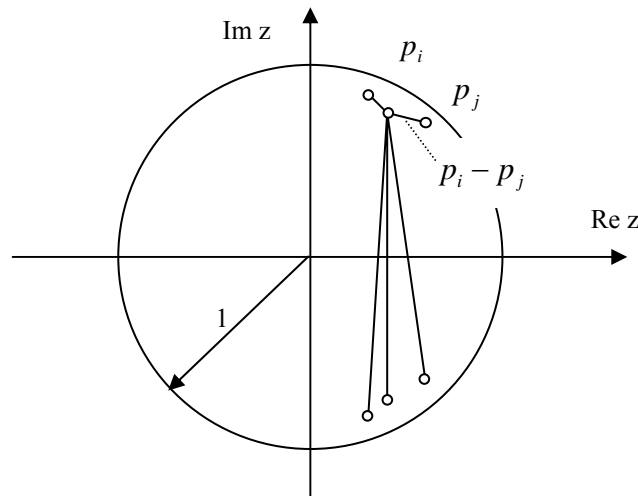
ahol: $\Delta b_k = b_k - b_k^0$

A gondot az jelenti, hogy a gyökmegoldó képlet magasabb fokú polinomokra nem ismert, ezért közvetlenül nem tudjuk a (6.30.)-ban szereplő parciális deriváltakat kiszámítani. A probléma megkerülhető az implicit függvény deriválási szabályának alkalmazásával, mely szerint:

(6.31.)

$$\frac{\partial p_i}{\partial b_k} = \frac{\frac{\partial B(z)}{\partial b_k} \Big|_{z=p_i}}{\frac{\partial B(z)}{\partial p_i} \Big|_{z=p_i}} = \frac{z^{-k} \Big|_{z=p_i}}{-z^{-1} \prod_{\substack{j=1 \\ j \neq i}}^N (1 - p_j z^{-1})} \Big|_{z=p_i} = - \frac{p_i^{-k}}{p_i^{-1} \prod_{\substack{j=1 \\ j \neq i}}^N (1 - p_j p_i^{-1})} = - \frac{p_i^{N-k}}{\prod_{\substack{j=1 \\ j \neq i}}^N (p_i - p_j)}$$

A (6.31.)-ben a számláló deriváltját a (6.24.)-ből, a nevezőét a (6.25.)-ből számítottuk ki.



6.6. ábra Keskenysávú szűrő pólusainak elhelyezkedése a “Z” síkon

Az egyes együtthatók érzékenységének (6.31.)-ben megadott végső formuláját tekintve láthatjuk, hogy a nevezőben az i -ik gyöknek a többi gyöktől vett távolságainak szorzata szerepel. Keskenysávú szűrő esetében a gyökök közel helyezkednek el, így magas foksám esetén a nevező extrém kicsi, vagyis az érzékenység igen nagy.

Az együtthatók pontatlansága és a nagy érzékenység miatt a pólus elmozdulása olyan nagy lehet, hogy a valóságos pólus az egységkörön kívülre kerül, a szűrő instabillá válik.

A fenti gondolatmenet eredményeként egy általános megállapítást tehetünk: Direkt struktúrában ne realizáljunk magas foksámú, keskenysávú szűrőt, mert ha stabil is marad a szűrő, az átviteli karakterisztika jelentősen el fog térni a tervezett karakterisztikától. Magas foksámú IIR szűrőt ezért más struktúrákban (például első és másodfokú alaptagok kaszkád kapcsolásával) realizálunk, mikor is egy pólus helyzetét csak az alaptag egy (vagy kettő) együtthatójának pontossága befolyásolja.

3. Másodfokú alaptag együttható érzékenysége

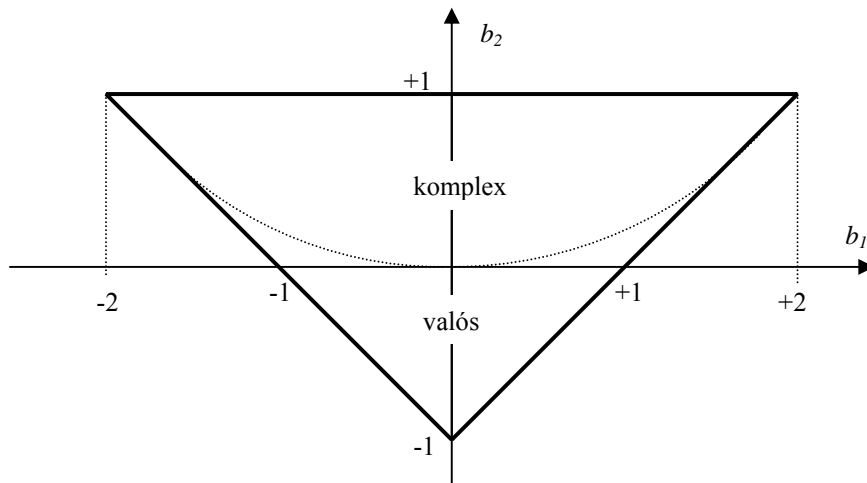
A fenti gondolatmenetből következően kiemelt fontosságú a másodfokú alaptag, illetve annak érzékenysége. Ezért ezt külön is megvizsgáljuk. A (6.30.) és a (6.31.) egyenletekből az együtthatók kerekítése következtében történő pólus elmozdulások értéke:

$$\Delta p_1 = \frac{p_1}{p_2 - p_1} \Delta b_1 + \frac{1}{p_2 - p_1} \Delta b_2 \quad \text{és} \quad \Delta p_2 = \Delta p_1^* \quad (6.32.)$$

Másodfokú esetben a gyökmegoldó képlet ismert, így könnyen ellenőrizhető a (6.32.) formula.

$$p_{1,2} = -\frac{b_1}{2} \pm \sqrt{\left(\frac{b_1}{2}\right)^2 - b_2} = -\frac{b_1}{2} \pm j\sqrt{b_2 - \left(\frac{b_1}{2}\right)^2} \quad (6.33.)$$

A fokozat akkor marad stabil, ha $|p_1| < 1$ és $|p_2| < 1$



6.7. ábra Másodfokú alaptag stabilitási tartománya az együtthatók síkján