



M Ű E G Y E T E M 1 7 8 2

Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Méréstechnika és Információs Rendszerek Tanszék

Audio szűrők kerekítési jelenségeinek vizsgálata

MSC DIPLOMATERV

Készítette

Horváth Kristóf Szabolcs

Konzulens

dr. Bank Balázs

2015. december 15.

Tartalomjegyzék

Kivonat	4
Abstract	5
Bevezető	6
1. Logaritmikus frekvenciafelbontású szűrők tervezése	9
1.1. Warpolt szűrők	9
1.2. Fixpólusú párhuzamos szűrők	12
2. Számábrázolás	16
2.1. Bevezetés	16
2.2. Fixpontos számábrázolás	16
2.3. Együttható kerekítés	17
2.4. Kvantálási zaj	18
2.5. Túlsordulás	19
3. Szűrőstruktúrák	21
3.1. IIR direkt form megvalósítás	21
3.2. WFIR közvetlen realizáció	22
3.3. WIIR közvetlen realizáció	23
3.4. Soros és párhuzamos realizáció	25
3.5. Másodfokú tagok	26
3.5.1. Másodfokú direkt form	26
3.5.2. Gold & Rader másodfokú tag	27
3.5.3. Kingsbury másodfokú tag	28
3.5.4. Chamberlin másodfokú tag	29
3.5.5. Zölzer másodfokú tag	30
3.5.6. WIIR	31
4. Vizsgálati módszer	33
4.1. Együttható-kerekítés	33
4.1.1. Kerekítés hatása a pólusokra	33
4.1.2. Kerekítés hatása az átvitelre	36
4.2. Processzormodell	37

4.3.	Kvantálási zaj	38
4.4.	Túlsordulás elkerülése	39
4.5.	Kvantálási zaj direkt realizációjú struktúrákban	40
4.5.1.	IIR DF1 és DF2	41
4.5.2.	IIR DF1 és DF2 transzponált	42
4.5.3.	WIIR közvetlen realizáció	43
4.6.	Zaj soros és párhuzamos struktúrákban	44
4.7.	Zaj speciális másodfokú tagokban	46
4.7.1.	Gold & Rader tag	46
4.7.2.	Kingsbury tag	47
4.7.3.	Chamberlin tag	48
4.7.4.	Zölzer tag	49
5.	Másodfokú tagok összehasonlítása	50
5.1.	Másodfokú tagok érzékenysége az együtttható-kerekítésre	50
5.1.1.	Csak pólusos tagok	51
5.1.2.	Mindentáteresztő tagok	52
5.1.3.	Valós zérusok	52
5.2.	Kvantálási zaj másodfokú tagokban	53
5.2.1.	A struktúra kiválasztásának módszere	53
5.2.2.	Optimális zajú másodfokú tag keresése	54
5.2.3.	Csak pólusos tagok	54
5.2.4.	Konjugált komplex zéruspárok	55
5.2.5.	Valós zérus	59
6.	Nagyfokszámú szűrők megvalósítása	61
6.1.	IIR	61
6.2.	WIIR, $\lambda = 0,75$	64
6.3.	WIIR, $\lambda = 0,9375$	67
6.4.	Fixpólusú párhuzamos szűrő	70
6.5.	Értékelés	72
7.	Összefoglalás	74
	Irodalomjegyzék	77
	Függelék	79
F.1.	Szűrő implementációk a processzormodellen	79

HALLGATÓI NYILATKOZAT

Alulírott *Horváth Kristóf Szabolcs*, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző, cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2015. december 15.

Horváth Kristóf Szabolcs
hallgató

Kivonat

Audio területen gyakori jelfeldolgozási feladat egy adott rendszer átvitelének modellezése, illetve kompenzálása. A problémát gyakran IIR szűrőkkel oldják meg, amik kerekítési jelenségei a mai számítógépeken – a nagy bitszámú lebegőpontos ábrázolás miatt – általában elhanyagolhatóak.

Más a helyzet azonban a beágyazott rendszereknél. A lebegőpontos DSP-k mellett megjelentek olyan, lényegesen olcsóbb mikrovezérlők, amelyek képesek SIMD utasítások végrehajtására, ezáltal alkalmasak hatékony jelfeldolgozásra is. Hátulütőjük, hogy lebegőpontos számábrázolást nem, vagy csak korlátozottan tudnak használni, sokkal inkább fixpontos aritmetikára optimalizáltak. Jelprocesszorok között is jócskán találunk 16-24-32 bites fixpontos aritmetikával működő példányokat, így különösen fontos a kvantálási jelenségek vizsgálata.

Nagy fokszámú szűrőket fixpontos számábrázolás mellett célszerű másodfokú tagok (pl. DF1, Kingsbury, Warped, stb.) soros vagy párhuzamos kapcsolataként implementálni, hogy minimalizáljuk az együtthatók kerekítéséből adódó hibát. Az így előálló struktúra eredő numerikus zaját az egyes tagok tulajdonságai fogják meghatározni. Optimális eredményre juthatunk, ha a tagok struktúráját aszerint választjuk meg, hogy a kívánt átvitelük mellett milyen a kvantálási zajuk.

Dolgozatomban bemutatom az audio területen használatos logaritmikus frekvenciafelbontású szűrők (warpolt, fixpólusú párhuzamos) tervezését és implementációját. Ismertetem a megvalósításhoz használható másodfokú struktúrákat, valamint az előforduló kvantálási jelenségeket elemzem.

Az együttható-kerekítés vizsgálatára egy analitikus és egy gyakorlati módszert is ismertetek. A numerikus zajt elméleti megfontolások alapján, a kerekítési pontok és a kimenet közötti átvitel szerint számolom. A kvantálási jelenségeket minden vizsgált másodfokú struktúra esetén különböző pólus-zérus elrendezésre megvizsgálom, majd bemutatom, hogy adott átvitelnél melyiknek lesz a legkisebb a zaja.

Dolgozatom célja, hogy soros vagy párhuzamos realizációk esetén útmutatást adjak a numerikus zaj minimalizálására. Az analízis módszerére alapozva olyan eljárást adok, amivel adott szűrőátvitel esetén kiválasztható a legkisebb zajú struktúra. Konkrét gyakorlati példákön mutatom meg, hogy a javasolt módszerrel megválasztott másodfokú tagok alkalmazásával akár több nagyságrenddel is csökkenthető a numerikus zaj.

Abstract

Modelling or equalizing a transfer function is a common task in digital signal processing. A widely used solution is the use of IIR filters, whose quantization effects – due to the high precision floating point arithmetic – can be neglected on a today’s computer.

This is not the case for embedded systems. Besides floating-point DSPs, there are inexpensive, high-performance microcontrollers, which have SIMD instructions, therefore they can be efficiently used for signal processing. Their drawbacks are that their use for floating-point arithmetic is limited since they are optimized for fixed-point calculations. In addition, there are numerous fixed-point digital signal processors, which work on 16, 24, or 32 bits of data. Therefore it is important to investigate the quantization effects.

Implementation of high-order filters is practical using serial or parallel second-order sections (such as DF1, Kingsbury, etc.), because this way the error induced by the coefficient-quantization can be minimized. The round-off noise of this structure is defined by the second-order sections. Optimal results can be obtained if the second-order sections are chosen by their numerical noise spectra.

The thesis outlines the design and implementation of special, logarithmic frequency-resolution filters for audio (warped and fixed-pole parallel filters). The filter structures are also presented and their quantization properties are analysed.

For analysing coefficient-quantization, an analytical and a practical method is given. The round-off noise is computed by a theoretical noise model, using the transfer function between the rounding point and the output of the filter. The quantization effects are evaluated at different pole-zero locations for each second-order structure, then a filter structure with minimal noise is chosen.

The purpose of this work is to give guidelines to minimize the numerical noise on serial or parallel filter realizations. Based on the analysis method, a technique is given for finding the structure with lowest noise. Practical examples are also given, and it is shown that using the proposed optimization method the noise can be reduced by several orders of magnitude.

Bevezető

Az audio területen gyakran alkalmazott IIR szűrőknél fontos szempont, hogy a számábrázolásból származó zajuk minél kisebb legyen. Az emberi hallás ugyanis akár 100 dB dinamik tartományú is lehet bizonyos frekvenciákon, így rossz struktúraválasztással a zaj hallhatóvá válhat.

A mai számítógépek processzorai duplapontosságú lebegőpontos műveletek végzését lehetővé tévő FPU-val rendelkeznek, aminek használata mellett a numerikus zaj általában elhanyagolható.

Beágyazott rendszereknél általában nem áll rendelkezésre lebegőpontos egység. A nagyteljesítményű mikrovezérlők körében ugyan találunk FPU-val rendelkező processzort (pl. ARM Cortex M4), de ezek egyrészt csak 32 bites lebegőpontos számokkal dolgoznak, másrészt egy-egy művelet végrehajtása több órajelciklust vesz igénybe, így lassúnak mondhatók. Az egyedüli kivételt az ARM Cortex A sorozat jelent, ahol megtalálhatóak lebegőpontos SIMD műveletek is (NEON utasításkészlet) [1].

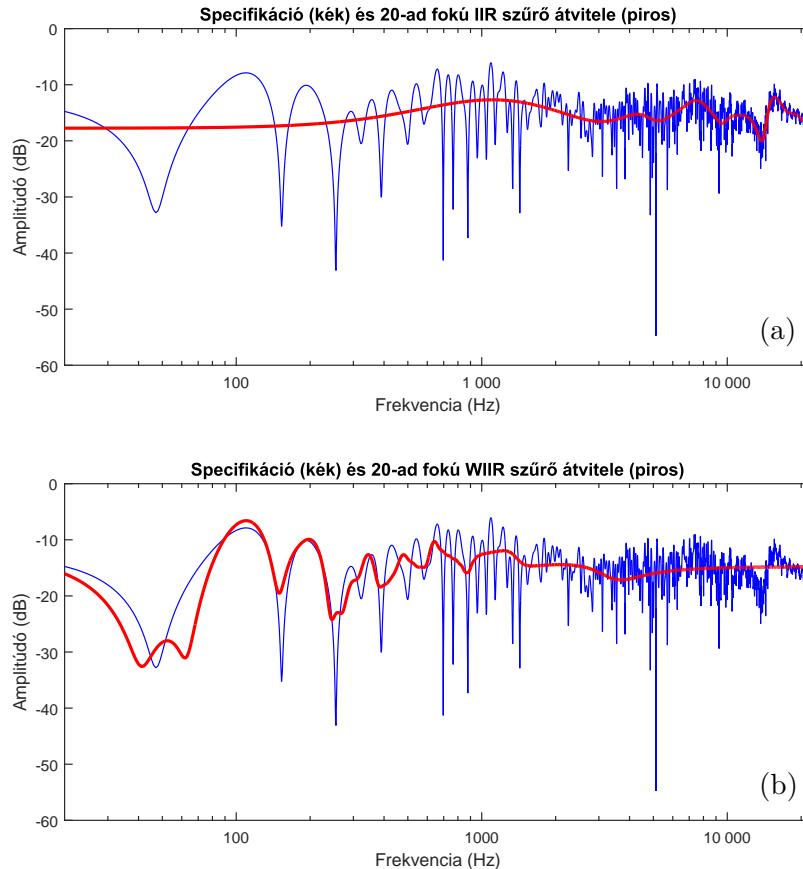
A jelprocesszorok nagy része fixpontos számábrázolással dolgozik, de mikrovezérlők között is található DSP utasításkészlettel rendelkezőket. Fontos tulajdonságuk, hogy képesek SIMD műveleteket végezni, ezáltal alkalmasak hatékony jelfeldolgozásra is. Ilyen mikrovezérlőket találunk például az ARM Cortex M, az Atmel AVR32 és a MIPS M4K sorozatban. Fixpontos jelprocesszorokat találunk például az Analog Devices Blackfin, a Freescale DSP56K, a Microchip dsPIC és a Texas Instruments TMS320C5000 sorozatában.

A vizsgált jelprocesszorok és mikrovezérlők között léteznek 16, 24, és 32 bites fixpontos aritmetikával dolgozók, így az eredményeimet ezekre a bitszámokra fogom kiértékelni. Érdekes, hogy az ARM Cortex M sorozat ugyan 32 bites, de 16 biten lényegesen gyorsabb működésre képes, mivel egy töltő utasítással 2 db adatot is behívhat a memóriából [2].

A véges pontosságú fixpontos számábrázolásból több kvantálási jelenség is származik. Az együtthatók pontatlansága tervezési időben változtatja meg a szűrő átvitelét, szélsőséges esetben instabillá is téve azt. Két szám összeszorzásakor az eredmény bitszélessége kétszer akkora lesz, mint a szorzandóké. A további műveletvégzéshez a processzor regisztereinek bitszámára kell kerekíteni a szorzatot, ez pedig a szűrő kimenetén additív zajt eredményez. Számok összeadásakor is előállhat egy nem kívánt hatás: ha az összeg nagyobb, mint az ábrázolható számtartomány, akkor túlsordulás következik be.

Az előbbi okokból továbbra is aktuális a kvantálási jelenségek vizsgálata. Az irodalom ugyan sok, numerikus szempontból előnyös struktúrát javasol [3, 4, 5], de a vizsgálatokat

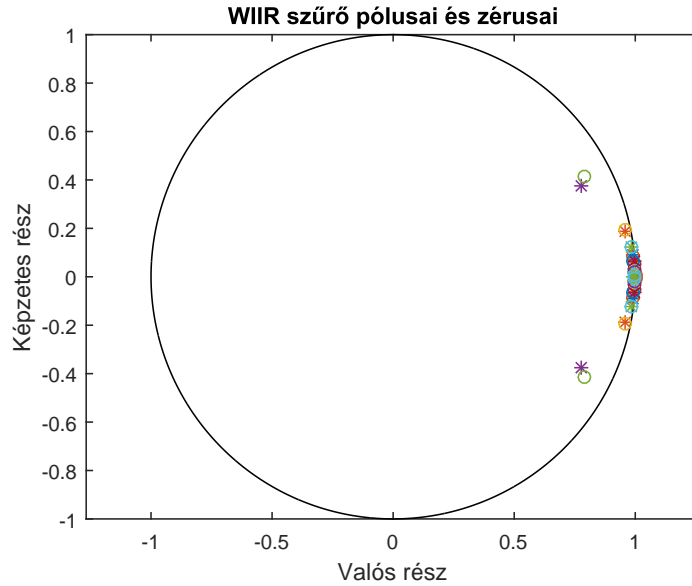
csak olyan esetekre végzik el, ahol a zérusok az origóban találhatóak. Ez pedig nem elegendő, a zérusok elhelyezkedése ugyanis jelentős hatással van a kvantálási zaj teljesítményére. A struktúrák közötti választás szempontjából fontos, hogy ismerjük az egyes szűrők számításigényét, amelyet processzormodellen implementálva adok meg. Ezzel a módszerrel pontosabb eredményt kapunk az erőforrásigényre, mint az irodalomban gyakran használt összeadás és szorzás műveletek megszámlálásával.



1. ábra. 20-ad fokú IIR (a) és WIIR (b) szűrő átvitele logaritmikus frekvenciaskálán

Audio célú alkalmazásoknál nem hanyagolható el a szűrőtervezés módja sem, az emberi fül amplitúdó- és frekvenciafelbontása ugyanis logaritmikus jellegű. A hagyományos IIR szűrőtervezési módszerek lineáris frekvenciafelbontás mellett minimalizálják az eltérést a szűrő átvitele és a specifikáció között, így alacsony frekvenciákon nem követik megfelelően a kívánt átvitelt (ld. 1. ábra). A warpolt szűrők (ld. 1.1. alfejezet) ezzel szemben nagyjából logaritmikus frekvenciaskála mellett minimalizálják a hibát [6], így kifejezetten alkalmasak audio célokra.

A warpolt szűrők mellett érdemes megemlíteni a fixpólusú párhuzamos szűrőket is [7]. A másodfokú tagokból felépített szűrő tervezésének fő jellemzője, hogy tetszőleges, előre rögzített pólusok mellett a zérusok meghatározásával modellezi a specifikációt. Amennyiben a pólusokat logaritmikus frekvenciaskála mellett egyenletesen helyezzük el, akkor a tervezett szűrő is ilyen frekvenciaskála mellett minimalizálja a szűrőátvitel és a specifikáció eltérését [8].



2. ábra. 20-ad fokú WIIR szűrő pólusai és zérusai, $\lambda = 0.95$

Warpolt szűrők implementálására léteznek speciális struktúrák, de ezek számításigényesek és nagy fokszámú problémás lehet az együtthatók ábrázolása, így felmerül az igény más struktúrákban történő megvalósításra. A warpolt és a fixpólusú párhuzamos szűrők pólusai és zérusai alacsony frekvenciákon koncentrálnak (ld. 2. ábra), emiatt olyan implementációt kell keresni, ami ebben a tartományban kvantálási szempontból is optimális.

Nagy fokszámú szűrők implementálásánál előnyös lehet a szűrő felbontása soros, illetve párhuzamos másodfokú tagokra. Az egyes tagok lehetnek különböző struktúrájúak, ezáltal lehetőség van a kvantálási jelenségek optimalizálására a pólus és zérus elhelyezkedésétől függően.

Mivel az irodalom nem vizsgálja, ezért itt hangsúlyt fektetek arra, hogy egy adott nagyfokszámú szűrő másodfokú tagokra bontásánál a kvantálási zaj szempontjából milyen struktúrában célszerű megvalósítani a tagokat. A korábbi kutatások nem tárgyalják a speciális warpolt struktúrából képezett másodfokú tagokat, így ezek vizsgálatát is elvégzem.

A dolgozat első fejezetében a warpolt és a fixpólusú párhuzamos szűrők tervezésének lépéseit ismertetem. Ezután a második fejezetben a fixpontos számábrázolást, és az alkalmazásánál fellépő kvantálási jelenségeket mutatom be. A harmadik és negyedik fejezetben a vizsgált szűrőstruktúrákat ismertetem és elemzem, valamint módszert adok az együttható-kerekítés és a kvantálási zaj vizsgálatára. Az ötödik fejezetben a vizsgálat eredményeit közlöm és javaslatot teszek egy struktúra kiválasztási módszerre, amivel optimalizálható a szűrők zaja. A hatodik fejezetben gyakorlati példák mutatnak meg, hogy az optimalizálással jelentősen csökkenthető a szűrők kvantálási zaja.

1. fejezet

Logaritmikus frekvenciafelbontású szűrők tervezése

Audio területen gyakori követelmény, hogy szűrőtervezéskor a specifikációtól való eltérést logaritmikus frekvenciatengely mellett minimalizáljuk, mivel az emberi hallás felbontása is közel logaritmikus [6]. E célra különböző módszereket dolgoztak ki, mint például a warped és a fixpólusú párhuzamos szűrők. Mivel ezek az általános jelfeldolgozási területeken nem ismertek, a fejezetben a warped és fixpólusú párhuzamos szűrők tervezési lépéseit mutatom be.

1.1. Warped szűrők

A nemlineáris frekvenciatengely mentén történő szűrőtervezést többek között frekvenciatranszformációval érhetjük el. Ennek általános lépései a lineáris frekvenciafelbontású specifikáció transzformálása, hagyományos módszerrel történő szűrőtervezés, majd a szűrő visszatranszformálása és megvalósítása.

A specifikáció frekvenciatengelyének transzformálására egyik, gyakran használt eljárás a warpolás. A módszer alapja, hogy a szűrőben megtalálható egységnyi késleltetéseket olyan taggal helyettesítjük, amely fázistolása függ a frekvenciától. Az egyes frekvencia-komponensek amplitúdójának változatlansága érdekében kézenfekvő, hogy erre a célra mindentáteresztő tagot használjunk [6]:

$$\Theta_\lambda(z^{-1}) = \frac{z^{-1} - \lambda}{1 - \lambda \cdot z^{-1}}, \quad (1.1)$$

ahol $-1 < \lambda < 1$. A λ paramétert warpolási tényezőnek nevezzük. A $z^{-1} \rightarrow \Theta_\lambda(z^{-1})$ helyettesítéssel megvalósított transzformáció az egységkört az egységkörre képezi le és igaz rá, hogy inverz transzformációja:

$$\Theta_\lambda^{-1}(z^{-1}) = \Theta_{-\lambda}(z^{-1}). \quad (1.2)$$

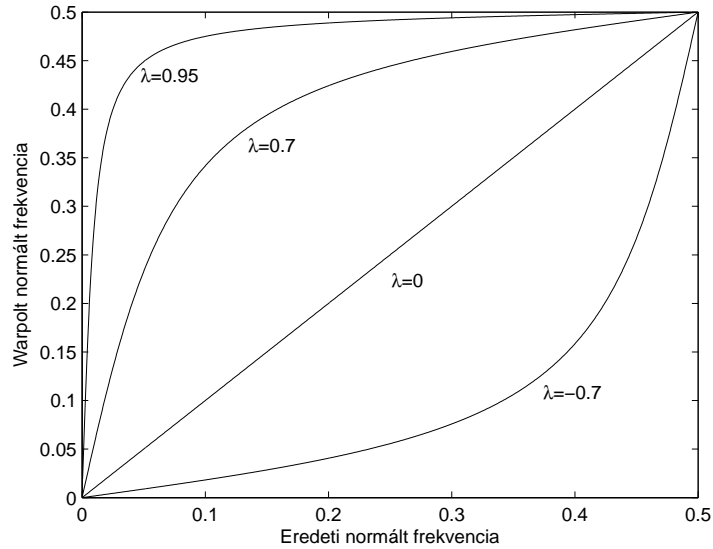
Warpolással tetszőleges jelet lehet transzformálni: a jel Z-transzformáltját eltolt Dirac-impulzusok lineáris kombinációjaként felírva, a z^k hatványok helyére pedig $\Theta_\lambda(z^k)$ -t be-

helyettesítve megkapjuk a warpolt jel Z-transzformáltját [6]. Gyakorlatilag ez azzal ekvivalens, hogy felírunk egy FIR szűrőt, melynek együtthatói (és ezzel impulzusválasza) a transzformálni kívánt jel. A szűrő késleltetéseit ezután kicseréljük λ paraméterű mindentátesztő tagokra. A warpolt jel ennek a struktúrának az impulzusválaszaként adódik.

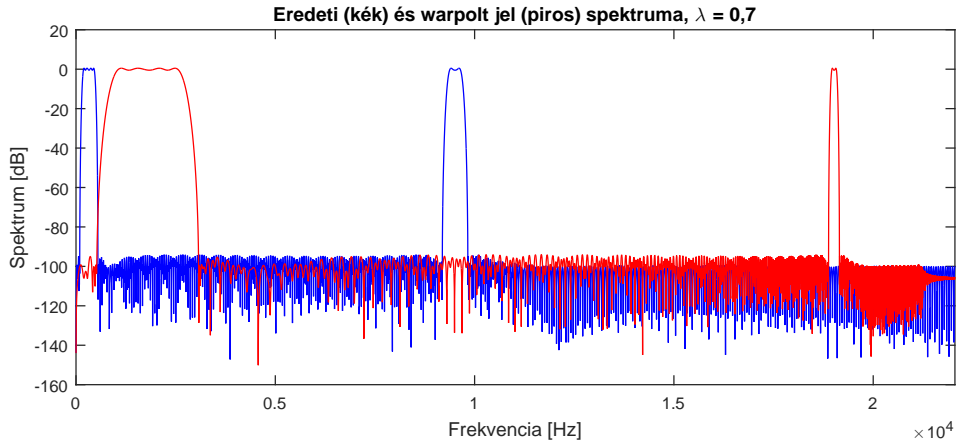
A warpolt jelek Z-transzformációjából levezethető a transzformáció frekvencia-leképezése [6]:

$$\omega' = \arctan \frac{(1 - \lambda^2) \sin(\omega)}{(1 + \lambda^2) \cos(\omega) - 2\lambda}, \quad (1.3)$$

ahol ω az eredeti, ω' a warpolt frekvencia és λ a warpolási tényező. Ennek a képletnek a kiértékelése látható az 1.1. ábrán különböző λ értékekre. Egy konkrét jel spektrumának változására az 1.2. ábra mutat példát. Vegyük észre, hogy a warpolás csak közelíti a logaritmikus frekvenciaskálára leképezést!



1.1. ábra. Warpolás hatása a frekvenciatartományban (normálás f_s -sel)



1.2. ábra. Példa: warpolás hatása a jel spektrumára

Belátható, hogy ha egy FIR/IIR szűrő átvitele $H(z^{-1})$, akkor a késleltetések mindentáteresztő tagokra cserélésével az új átvitel $H(\Theta(z^{-1}))$ lesz. A transzformáció megőrzi a fokszámot, amennyiben a pólusok és a zérusok száma megegyezik. Ilyenkor az új pólusok és zérusok a

$$\tilde{p}_i = \Theta(z)|_{z=p_i}, \quad \tilde{s}_i = \Theta(z)|_{z=s_i} \quad (1.4)$$

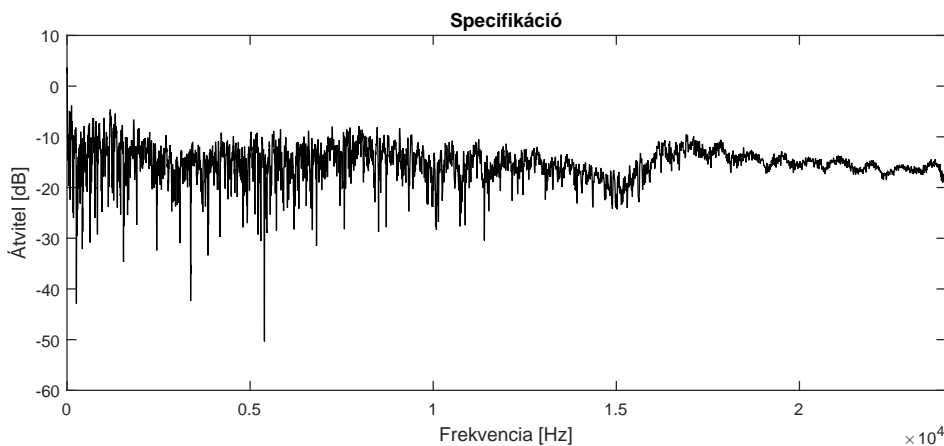
képletek szerint állnak elő [9]. Érdeemes megjegyezni, hogy a WFIR struktúrának a pólusai nem a nullában lesznek, ezáltal az impulzusválasza nem lesz véges idejű [10].

A tervezés lépései

Az alábbi műveletekkel tervezhető warpolt szűrő:

1. Specifikáció transzformálása $-\lambda$ -val,
2. Tervezés hagyományos módszerrel a transzformált impulzusválasz alapján,
3. Pólusok és zérusok transzformálása λ -val, vagy megvalósítás speciális struktúrában.

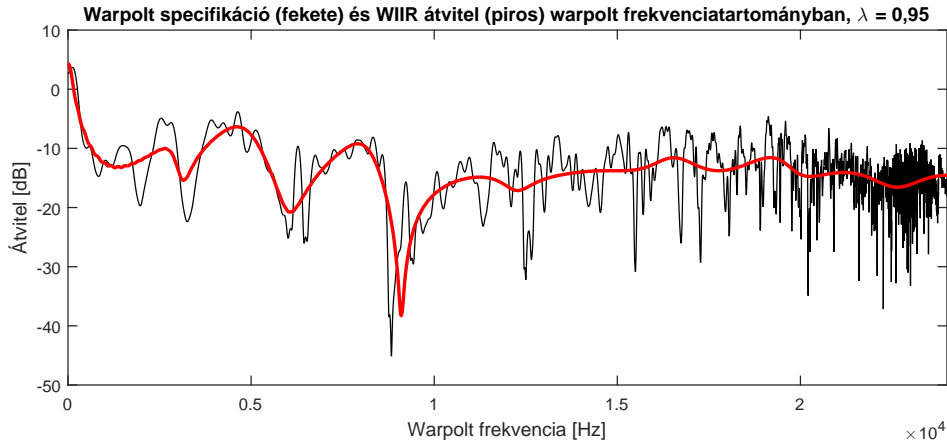
Az első lépéssel a specifikációt warpolt frekvenciatartományba transzformáljuk. Ez azt eredményezi pozitív λ mellett, hogy az eredeti átvitel alacsonyabb frekvenciájú részei szélesebb tartományt foglalnak el a warpolt átvitelben. Erre látható egy példa az 1.3–1.4. ábrákon. Az így létrejött új átvitelre kell alkalmazni a hagyományos tervezési algoritmusokat (pl. Prony módszer). A kész szűrő átvitele így alacsonyabb frekvenciákon fogja jobban közelíteni a specifikációt (lásd: 1.5. ábra).



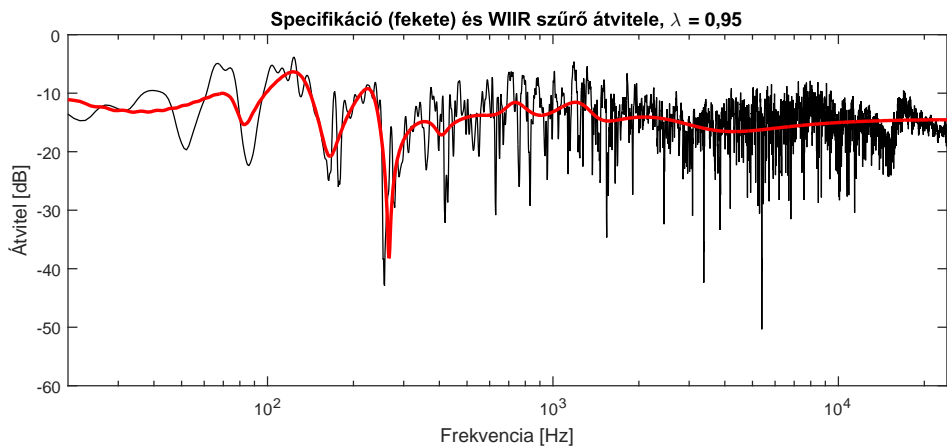
1.3. ábra. Példa: a szűrő specifikációja (lakószoba átviteli függvénye)

A tervezési algoritmusban kiadódó értékek lesznek a warpolt szűrő együtthatói, amiket közvetlenül a 3. fejezetben bemutatott speciális warpolt struktúrákban használhatunk fel, de lehetséges az együtthatók visszatranszformálása után hagyományos direkt realizációjú IIR struktúrával is megvalósítani a szűrőt. Erre azért lehet szükség, mert a speciális struktúra nagyobb számításigényű, mint az IIR direkt realizáció.

Fontos megjegyezni, hogy noha célszerűnek tűnik a warpolt szűrők direkt realizációjú IIR struktúrában történő megvalósítása, a gyakorlatban a számbázis pontatlansága



1.4. ábra. Példa: warpolt specifikáció és tervezett szűrő átvitele



1.5. ábra. Példa: specifikáció és szűrőátvitel logaritmikus frekvenciatengelyen

miatt ez nem mindig lesz működőképes: tipikusan 20-as fokszám fölött már dupla pontoságú lebegőpontos számábrázolásnál is instabillá válik a szűrő [6]. A problémát másodfokú tagokra bontással lehet elkerülni, ebben az esetben még a warpolt frekvenciatartományban kell elvégezni a szűrő felbontását, majd az egyes tagoknál külön-külön kell elvégezni a transzformációt [11].

A tervezés másik iránya is járható, miszerint egy korábban megtervezett IIR szűrőt is transzformálhatunk: a warpolt szűrő kívánt pólusait és zérusait az (1.4). képlet szerint állíthatjuk elő. Ennek a módszernek létjogosultsága akkor van, ha adott pólusokat és zérusokat szeretnénk speciális warpolt struktúrában megvalósítani.

1.2. Fixpólusú párhuzamos szűrők

A logaritmikus frekvenciafelbontás nem csak warpolt szűrőkkel közelíthető. A fixpólusú párhuzamos szűrőknél előre meghatározott pólusok mellett tervezünk, amiket akár logaritmikus frekvenciaskála szerint is elhelyezhetünk [8].

A WFIR általánosításának tekinthető Kautz szűrők jelentik a kapcsolatot a warpolt és a fixpólusú párhuzamos szűrők között [7]. Kautz szűrőknél a kívánt átvitelt ortonormált

bázisfüggvények lineáris kombinációjaként állítjuk elő, melyeket külön-külön a pólusuk határoz meg.

A fixpólusú párhuzamos szűrők esetében az átvitelt nem ortonormált bázisfüggvényekből állítjuk elő, hanem a tetszőlegesen elhelyezett pólusok átvitelét tekintjük bázisnak [7].

A tervezéshez induljunk ki a következőből: minden modellezendő $H(z^{-1}) = B(z^{-1})/A(z^{-1})$ alakú átvitelt felírhatunk az alábbi alakban:

$$H(z^{-1}) = \sum_{k=1}^K \frac{c_k}{1 - p_k \cdot z^{-1}} + \sum_{m=0}^M b_m \cdot z^{-m}, \quad (1.5)$$

ahol p_k jelöli a pólusokat, amik valós be- és kimenettel rendelkező rendszernél konjugált komplex párok. A képletben szereplő második összegzés egy FIR tag, ami $B(z^{-1})$ és $A(z^{-1})$ fokszámának egyezésekor egy konstanssá (b_0) redukálódik. Érdeemes megjegyezni, hogy az (1.5). képlet csak egyszeres pólusok esetében igaz, egyébként magasabb fokú tagok is megjelennének a képletben.

A továbbiakban helyettesítsük be az előre rögzített pólusainkat a az (1.5). képletbe! Az ismert, modellezendő átvitelre ($H(z^{-1})$) így egy lineáris kifejezés adódik, amiben c_k és b_m megválasztását legkisebb négyzetek módszerével végezhetjük [7].

Impulzusválasszal adott specifikáció esetén az alábbiak szerint kell eljárni: jelöljük u_k -val a k -adik rögzített pólus impulzusválasztát. Ekkor $u_k[n] = (p_k)^n$, ahol n a diszkrét idő. Legyen \mathbf{M} mátrix és \mathbf{p} vektor a következő:

$$\mathbf{M} = \begin{bmatrix} u_1[0] & u_2[0] & \dots & u_K[0] & \delta[0] & \delta[-1] & \dots & \delta[-M] \\ u_1[1] & u_2[1] & \dots & u_K[1] & \delta[1] & \delta[0] & \dots & \delta[1-M] \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ u_1[L] & u_2[L] & \dots & u_K[L] & \delta[L] & \delta[L-1] & \dots & \delta[L-M] \end{bmatrix}, \quad (1.6)$$

$$\mathbf{p} = [c_1, c_2 \dots c_K, b_0, b_1 \dots b_M]^T. \quad (1.7)$$

Ekkor a szűrő L hosszúságú \mathbf{h} impulzusválasza:

$$\mathbf{h} = \mathbf{M}\mathbf{p}. \quad (1.8)$$

A szűrőegyütthatók meghatározásához helyettesítsük a specifikáció impulzusválasztát (\mathbf{h}_s) \mathbf{h} helyére! Ekkor a képletben az egyedüli változó \mathbf{p} lesz, így az egyenlet mindkét oldalát balról megszorozva \mathbf{M} pszeudoinverzével, előállnak a legkisebb négyzetek szerinti szűrőegyütthatók [7]:

$$\mathbf{p}_{opt} = \mathbf{M}^+ \mathbf{h}_s \quad (1.9)$$

A valós szűrőegyütthatók érdekében össze kell vonni a konjugált komplex póluspárokat tartalmazó tagokat. Amennyiben p_k konjugált komplex párja p_{k+1} , akkor:

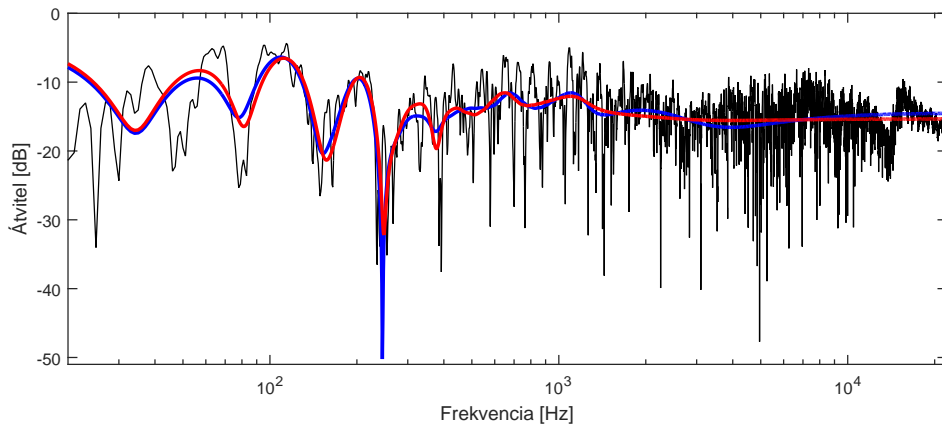
$$\frac{c_k}{1 - p_k \cdot z^{-1}} + \frac{c_{k+1}}{1 - p_{k+1} \cdot z^{-1}} = \frac{(c_k + c_{k+1}) - (c_k p_{k+1} + c_{k+1} p_k) \cdot z^{-1}}{1 - (p_k + p_{k+1}) \cdot z^{-1} + p_k p_{k+1} \cdot z^{-2}}. \quad (1.10)$$

Ezeket a másodfokú IIR tagokat és a FIR tagot kell megvalósítani.

Pólusok megválasztása

A pólusok elhelyezése alapvetően tetszőleges, de léteznek szisztematikus módszerek a megválasztásukra [8, 12, 13]. Ezek közül mutatok két példát: a warpolt skála és a logaritmusos frekvenciaskála szerinti elhelyezést.

Egy WIIR szűrő tervezésével olyan pólusokhoz jutunk, amelyek eloszlása warpolt frekvenciaskála szerint közelítően egyenletes. Ha ezekkel a pólusokkal terveztetünk egy fixpólusú párhuzamos szűrőt, az átvitel hasonló lesz a WIIR szűrőéhez. Erre mutat példát az 1.6. ábra. Látható, hogy a két szűrőátvitel közel azonos, a kisebb eltéréseket a zérusok más elhelyezése okozza.



1.6. ábra. Példa: specifikáció (fekete), 20-ad fokú WIIR (kék), $\lambda = 0,95$ és fixpólusú párhuzamos szűrő átvitele (piros), a rögzített pólusok a WIIR szűrő pólusai

Az előzőnél érdekesebb lehet, ha a pólusokat valóban logaritmusos frekvenciaskálán helyezzük el. A pólusfrekvenciák meghatározása magától értetődő: az $[f_{min}; f_{max}]$ tartományban N db pólus elhelyezése esetén az egyes pólusfrekvenciák:

$$f_k = f_{min} \cdot 10^{\frac{k-1}{N-1} \cdot \log_{10} \left(\frac{f_{max}}{f_{min}} \right)}, \quad k = 1 \dots N.$$

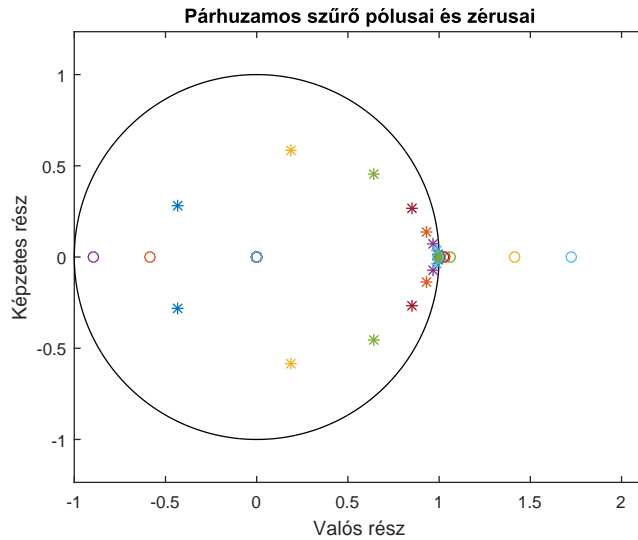
A pólusok sugarának meghatározása már nem evidens. Megoldás lehet, hogy úgy határozzuk meg a pólusok jósági tényezőjét, hogy az egymás melletti pólusok átvitele a maximumhoz képesti -3 dB-es pontban metszék egymást [8]. Ekkor:

$$\begin{aligned} \Theta_k &= \frac{2\pi f_k}{f_s}, \\ \Delta\Theta_k &= \frac{\Theta_{k+1} - \Theta_{k-1}}{2}, \quad k = 2 \dots N-1, \\ \Delta\Theta_1 &= \Theta_2 - \Theta_1, \\ \Delta\Theta_N &= \Theta_N - \Theta_{N-1}. \end{aligned}$$

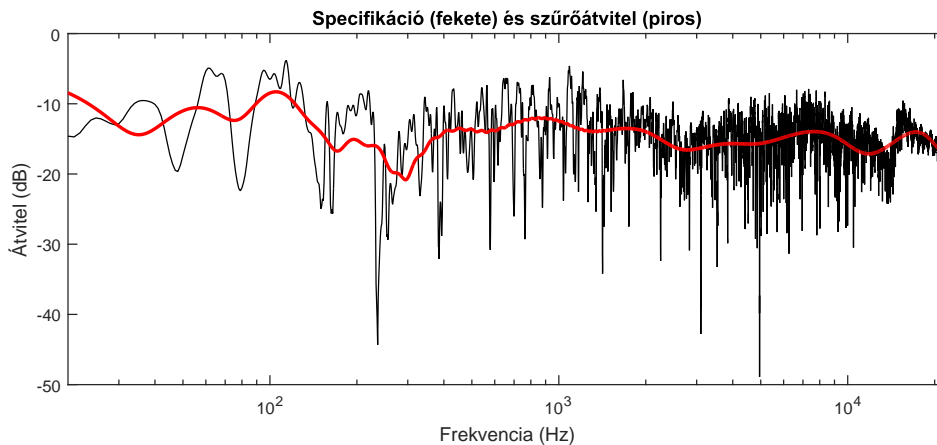
Ezekkel a jelölésekkel a pólusok:

$$p_k = e^{-\frac{\Delta\Theta_k}{2}} \cdot e^{\pm j\Theta_k}.$$

A korábbi specifikáción bemutatom ezzel a módszerrel való tervezést is: 10 db pólust helyeztem el logaritmikuskálán 30 Hz és 18 kHz között, a mintavételi frekvencia 44,1 kHz volt. Mivel konjugált komplex pólusok lesznek, a szűrő 20-ad fokú lesz. A tervezés után a pólus-zérus elrendezést az 1.7. ábra, a specifikációt és a szűrőátvitelt az 1.8. ábra mutatja. Látható, hogy a pólusok hossza a $z = 1$ -hez közeledve nő, valamint, hogy az 1.6. ábrán látható WIIR szűrővel ellentétben valóban logaritmikusképzésű.



1.7. ábra. Példa: 20-ad fokú fixpólusú párhuzamos szűrő pólus-zérus térképe, a rögzített pólusok logaritmikusképzésű elhelyezve



1.8. ábra. Példa: specifikáció (fekete), 20-ad fokú fixpólusú párhuzamos szűrő átvitel (piros), a rögzített pólusok logaritmikusképzésű elhelyezve

Látható tehát, hogy a fejezetben bemutatott módszerekkel olyan szűrők tervezhetők, amelyek az emberi hallásnak megfelelő felbontásúak.

2. fejezet

Számábrázolás

2.1. Bevezetés

Digitális szűrők implementálásánál fontos kérdés, hogy milyen aritmetikát használunk, ugyanis döntésünk érzékenyen befolyásolja a számábrázolásból származó kvantálási jelenségeket és a szűrő számításigényét.

Az asztali számítógépek, valamint a nagyteljesítményű DSP-k lebegőpontos aritmetikát alkalmaznak. A processzor ALU egységében speciális céláramkörök biztosítják, hogy egy-egy lebegőpontos művelet elvégezhető legyen egyetlen órajelciklus alatt.

Más a helyzet azonban a mikrovezérlőknél. Léteznek ugyan lebegőpontos egységgel rendelkező mikrokontrollerek (pl. ARM Cortex M4F), de ezek nem képesek egy órajelciklus alatti műveletvégzésre. Legtöbbjük azonban rendelkezik DSP utasításkészlettel, ami a gyakorlatban néhány, fixpontos aritmetikával dolgozó SIMD utasítás meglétét jelenti [14]. Ilyen módon a nagyteljesítményű mikrovezérlők hasonlóképpen alkalmazhatók, mint a fixpontos aritmetikájú jelprocesszorok.

Ebben a fejezetben az előbb ismertetett okokból a fixpontos számábrázolás mellett fellépő kvantálási jelenségeket mutatom be.

2.2. Fixpontos számábrázolás

Fixpontos számábrázolásnál a processzor egész számokkal dolgozik, kettes komplementum formában. Az egészrészt a törtrészekről elválasztó kettedespont helyét a számításaink során rögzítettnek tekintjük, ezáltal az egész és a tört helyiértékek száma is adott.

Ha az ábrázolható számtartomány a $[-2^K, 2^K)$, a bitszám pedig B , akkor a legkisebb abszolútértékű szám:

$$q = 2^{-B+K+1}. \quad (2.1)$$

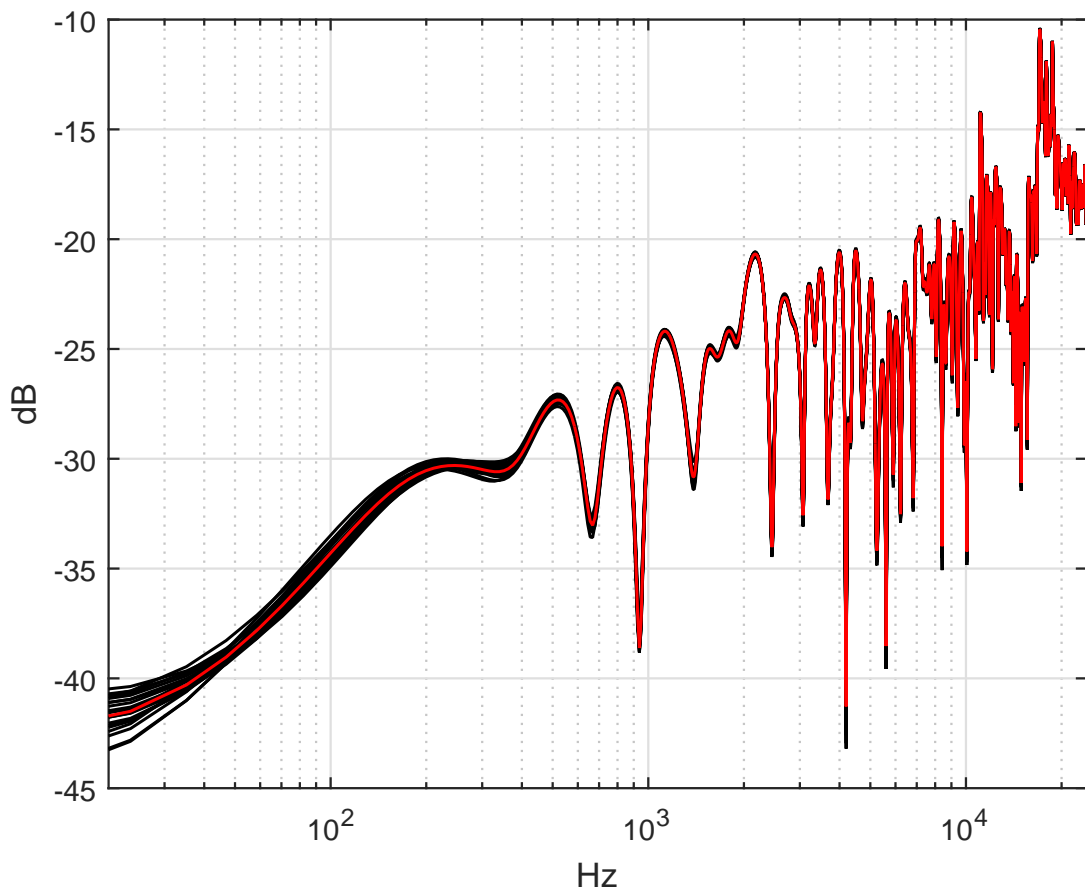
Ez lényegében véve a kvantálási küszöb. K értéke tulajdonképpen a fixpont helyét határozza meg, jelek ábrázolásánál célszerű a $K = 0$ választás, így a jelünk a $[-1, 1)$ tartományba fog esni. Érdeemes megemlíteni, hogy egy jelfeldolgozó algoritmusnál az egyes alegységek (pl. szűrők) dolgozhatnak különböző helyre tett fixponttal, de a közöttük történő információcserénél meg kell oldani a konverziót (bit shift). Szűrők implementálásánál a fixpont

helyét úgy kell megválasztani, hogy az együtthatók beleessenek a számábrázolási tartományba.

Fixpontos számábrázolásnál háromfajta kvantálási jelenséggel találkozhatunk. Az együtthatók ábrázolásából származó hiba tervezési, a kvantálási zaj és a túlcordulás pedig futási időben jelentkeznek. A fejezet további részében ezeket a jelenségeket tárgyalom.

2.3. Együttható kerekítés

Szűrők együtthatóit a memóriában B bit szélességű konstansokként tároljuk. A véges pontosságú számábrázolás az átviteli függvényre jelent korlátozást, ugyanis előfordulhat, hogy a megtervezett szűrőnk együtthatóinak fixpontosra átszámítása után az átvitel lényegesen megváltozik. Ezt mutatja a 2.1. ábra, ahol egy 200-ad fokú direkt struktúrájú IIR szűrő együttható-kerekítésre érzékenységi vizsgálatát végeztem. A lebegőponton implementált szűrő egyes együtthatóihoz véletlenszerűen a 16 bites fixpontos kvantálási küszöb felét adtam hozzá, vagy vontam le, így szimulálva a kerekítés szélsőséges eseteit. Jól látható, hogy ez a struktúra érzékeny az együttható kerekítésre: akár több decibelyit is változhat az átvitel.



2.1. ábra. 200-ad fokú IIR szűrő átvitele. Piros jelöli a dupla pontosságú lebegőpontos megvalósítás átvitelét, fekete a 16 bites fixpontosét az együttható-kerekítés szimulációjával.

Mivel az együtthatók a pólusok és zérusok elhelyezkedését is meghatározzák, a kerekítés hatására szélsőséges esetben pólusok is kikerülhetnek az egységkörön kívülre, így instabillá válhat a szűrő.

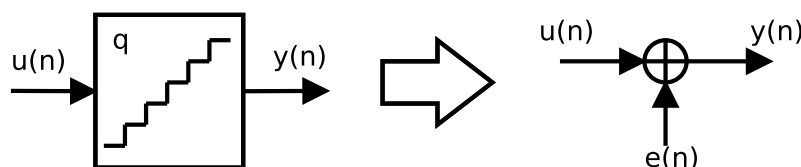
Belátható, hogy az együttható-kerekítésből származó hiba annál nagyobb, minél inkább összemérhető a kvantálási küszöb szintje (q) az együtthatóval. A jelenség hatása szűrő-struktúránként különböző lehet, emiatt a szakirodalom sokat foglalkozik olyan struktúrák keresésével, amik bizonyos körülmények között jobban teljesítenek együttható-kerekítés szempontjából [5, 15, 16, 17]. Mivel másodfokú tagokat vizsgálok, ahol az együttható-kerekítés hatása kicsi [18], a későbbiekben nem foglalkozok a jelenséggel.

2.4. Kvantálási zaj

Ha összeszorozunk két, B bit szélességű számot, akkor az eredmény $2B$ méretű lesz. Világos, hogy a $2B$ bit minden információt tartalmaz, a további műveletvégzés érdekében viszont a szorzás kimenetét B bitre kell csonkolni, ezáltal információvesztés lép fel.

Érdekes megjegyezni, hogy jelprocesszoroknál a gyártók ugyan a memória bitszélességét szokták feltüntetni bitszámként, azonban rendelkeznek ennél nagyobb pontosságú regiszterrel is, az akkumulátorral. Az akkumulátor szélessége legalább kétszer akkora, mint a memóriáé, így a regiszter szélességű szorzások eredményei biztosan adatvesztés nélkül beleférnek. Kerekítés csak akkor lép fel, amikor az akkumulátor tartalmát a memóriába írjuk. Fontos még megjegyezni, hogy az akkumulátor értékét közvetlenül nem lehet szorozni, azt előbb ki kell írni egy memória szélességű regiszterbe.

Látható, hogy a jel ilyen jellegű kvantálása egy determinisztikus, nemlineáris művelet. Ha a jelünk lényegesen nagyobb a kvantálási küszöbnél, akkor felírhatunk egy sztochasztikai modellt a kerekítésre. Ez az additív zaj modell [19].



2.2. ábra. Additív zaj modell

Kvantálásnál a kvantálási küszöb szintjének fele ($\frac{q}{2}$) a legnagyobb hiba, ami előfordulhat. A zajmodellben úgy tekintjük, hogy a kerekítési pontban a $(-\frac{q}{2}; +\frac{q}{2})$ tartományon egyenletes eloszlású, a jellel korrelálatlan fehér zajt adunk a jelhez, melynek várható értéke és szórásnégyzete [19]:

$$\mu = 0 \quad (2.2)$$

$$\sigma^2 = \frac{q^2}{12}. \quad (2.3)$$

Ilyen módon definiálva a hozzáadott zaj szórásnégyzete megegyezik a spektrális sűrűségfüggvényével. Ezt egy adott frekvenciatartományra integrálva (összegezve) kiadódik a tartomány zajteljesítménye.

Megjegyzendő, hogy ha a kvantálási ponton becsatolódó zaj áthalad a szűrő egy részén, akkor megváltozik a spektruma, így a kimeneten már színes zaj lesz. Ha kvantálás több helyen is fellép egy szűrőben, akkor ezeket független zajként összegezhjük [4].

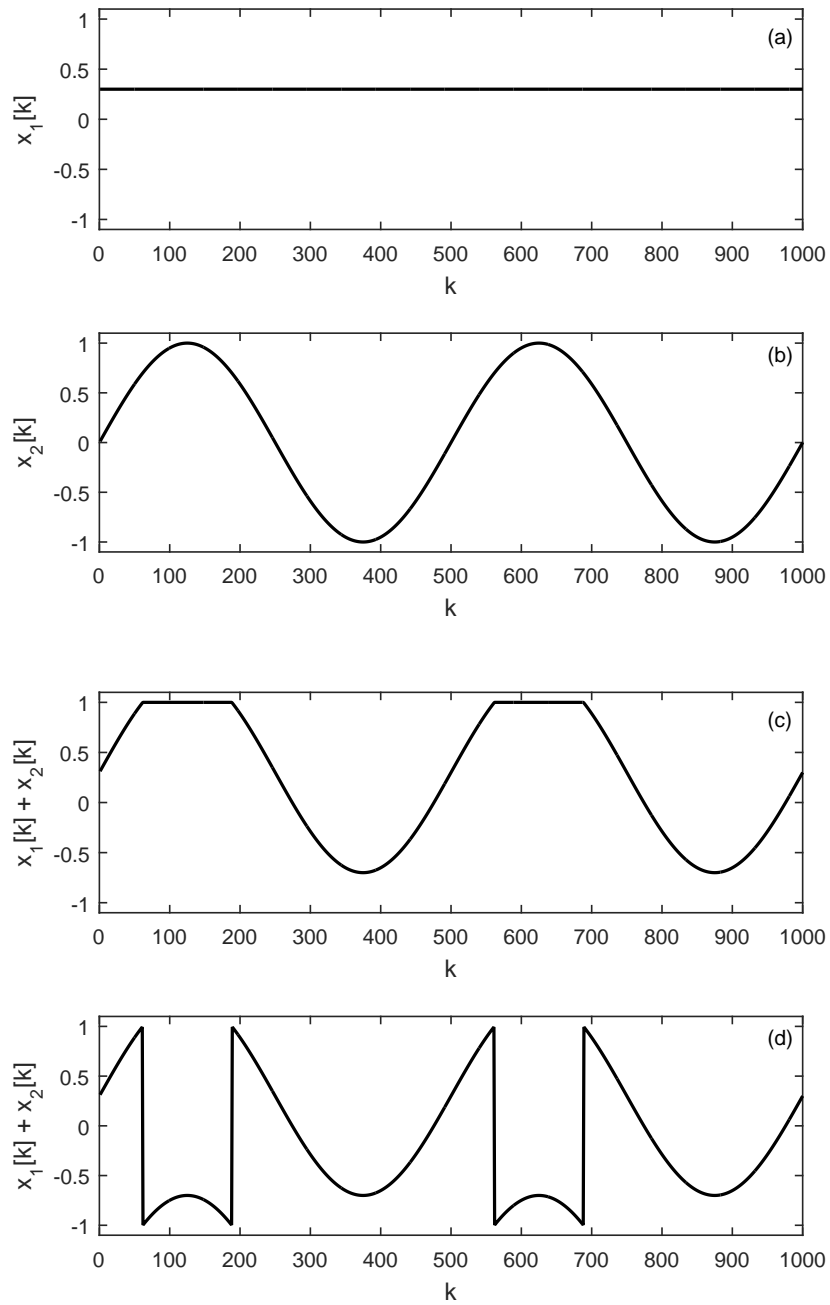
2.5. Túlcsordulás

Két, B bites szám összeadásakor az eredmény is B bites lesz. Információt tehát ideális esetben nem veszünk. Speciális esetként azonban előfordulhat, hogy az összeg kilóg a $[-2^K, 2^K)$ közötti számábrázolási tartományból. Ebben az esetben két megoldás közül választhatunk: telítés, vagy túlcsordulás.

Telítéses aritmetikát gyakran alkalmaznak jelprocesszorokban. Attól függően, hogy az összeg melyik irányba lépi túl a számábrázolás tartományát, a legnagyobb, vagy a legkisebb ábrázolható szám lesz az eredmény. Erre mutat példát a 2.3. ábra: az összeadandó jelek közül mindegyik elfér a $[-1; +1)$ számábrázolási tartományban, de az összeg már nem, így telítés, vagy túlcsordulás következik be.

Általános mikrovezérlőknél nem mindig áll rendelkezésre telítéses aritmetika. A kettes komplement számábrázolás miatt túlcsordulásnál előfordulhat, hogy noha az összeadandó számok mind pozitívak, az eredmény mégis negatív lesz. Ez egy nagyon éles ugrást eredményez a jelben (ld. 2.3.(d). ábra), IIR szűrőkben ráadásul még oszcillációt is okozhat a kimenet visszacsatolása miatt.

Látható, hogy az összeadásakor fellépő túlcsordulás vagy telítődés egy nemlineáris hatás, amit audio szűrőknél mindenképpen el kell kerülni. Erre egy módszer lehet a bemenőjel leosztása, és a kimenet felszorzása, így az átvitel nem változik. Megjegyzendő, hogy a leosztás mértéke szűrőstruktúránként más-más lehet, és a kimeneti felskálázás a kvantálási zaj teljesítményét is megnöveli.



2.3. ábra. Példa. (a)–(b): Két összeadandó jel $K = 0$ fixpont elhelyezés mellett, (c): összeg telítéssel, (d): összeg túlsordulással

3. fejezet

Szűrőstruktúrák

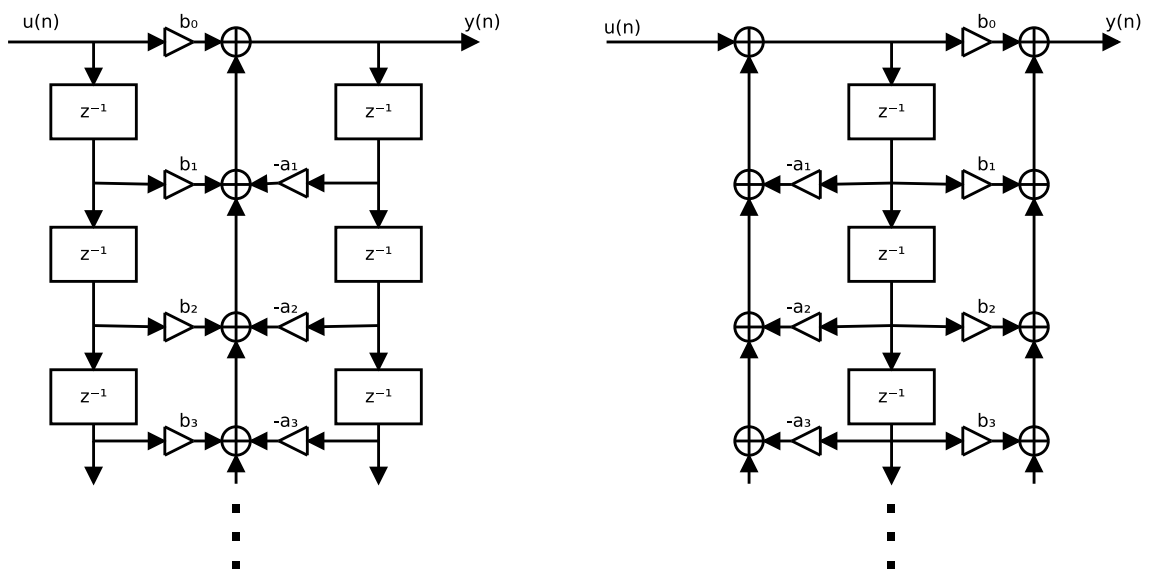
Végtelen impulzusválaszú rendszerek megvalósítására több megoldás is létezik. A racionális törtfüggvény alakú átvitel esetén adja magát, hogy direkt formában realizáljuk a szűrőt, ám nem csak ez az egyedüli megvalósítási módszer. Kvantálási jelenségek szempontjából ugyanis előnyösebbek lehetnek más struktúrák. Ebben a fejezetben ismertetem a későbbiekben vizsgált szűrőstruktúrákat.

3.1. IIR direkt form megvalósítás

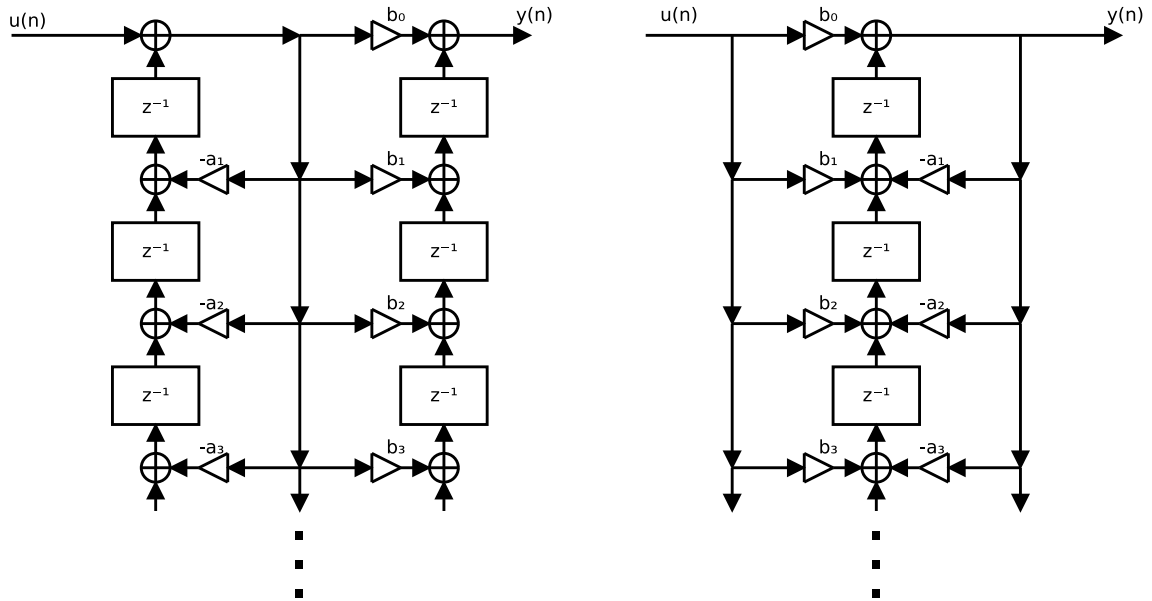
IIR szűrőt leggyakrabban direkt realizációban szokás megvalósítani. Ezek átviteli függvénye racionális törtfüggvény alakú:

$$H(z) = \frac{\sum_{k=0}^N b_k \cdot z^{-k}}{1 + \sum_{k=1}^N a_k \cdot z^{-k}}, \quad (3.1)$$

A képletben szereplő b_k és a_k paramétereket szűrőegyütthatóként felhasználva a 3.1. és 3.2. ábrák szerinti struktúrákat valósíthatjuk meg.



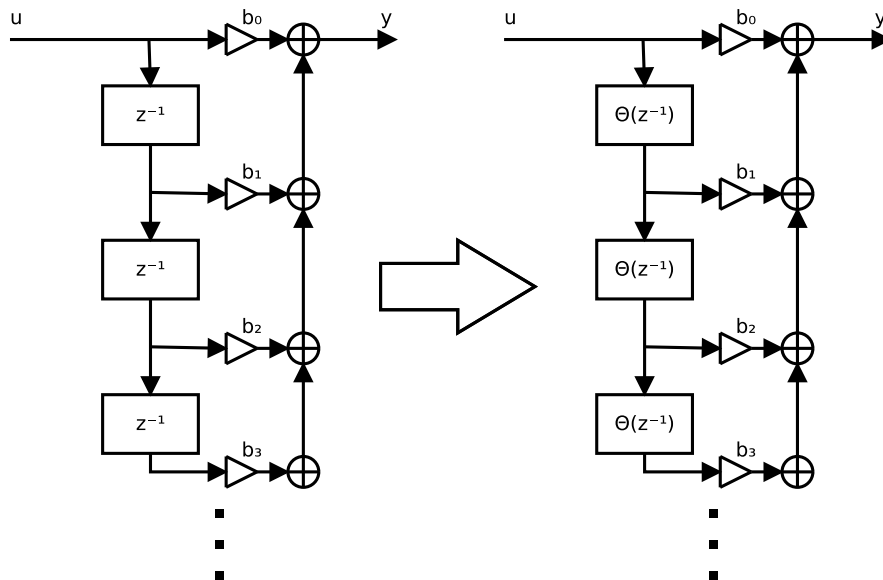
3.1. ábra. Balra: Direkt form I, jobbra: Direkt form II



3.2. ábra. Balra: DF-I transzponált, jobbra: DF-II transzponált

A direkt form realizációjú szűrők közös tulajdonsága, hogy minél nagyobb fokszámú a megvalósítani kívánt szűrő, az együttthatók annál érzékenyebbek lesznek a számábrázolásra [18]. Kedvezőbb esetben az átvitel csak kisebb-nagyobb mértékben fog eltérni a kívánttól, szélsőséges esetben azonban a szűrő instabillá válhat.

3.2. WFIR közvetlen realizáció



3.3. ábra. WFIR szűrő származtatása FIR-ből

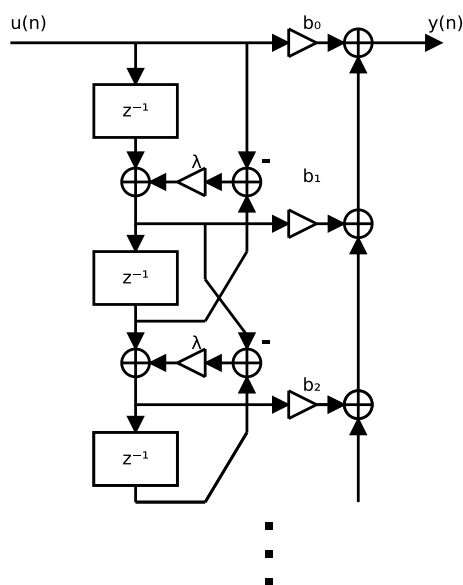
Legegyszerűbb struktúrájú warpolt szűrő. Származtatása a FIR direkt realizációjából történik mindentátesztő tagok behelyettesítésével (3.3. ábra). Implementációja a 3.4. ábrán található. Látható, hogy ebben a struktúrában vannak visszacsatolások is, ezért az im-

pulzusválasz már nem lesz időben véges. Belátható, hogy egy N -ed fokú WFIR szűrőnek N db pólusa van λ -ban.

Vegyük észre, hogy egy jel warpolását elvégző mindentáteresztő-lánc (ld. korábban) tulajdonképpen egy WFIR szűrő, a transzformált jel pedig ennek impulzusválasza.

A WFIR szűrő létjogosultsága elsősorban a tervező algoritmusok robusztusságában ke-
resendő: tulajdonképpen FIR szűrőt kell tervezni a warpolt frekvenciatartományban [6].

A későbbiekben ezt a struktúrát nem fogom vizsgálni: a pólusainak elhelyezkedése nem tetszőleges (definíció szerint λ -ban vannak), ugyanakkor számításigény szempont-
jából összemérhető a warpolt IIR-rel.



3.4. ábra. WFIR közvetlen realizáció

3.3. WIIR közvetlen realizáció

A warpolt IIR szűrő esetén nem működik az egyszerű behelyettesítéses módszer: késleltetésmentes hurok alakul ki a visszacsatoló ág miatt (ld. 3.5. ábra bal oldala). A probléma megoldásaként módosítani kell a struktúrát: erre Steiglitz adott egy egyszerű módszert [20], amivel a WIIR szűrő első késleltetőjét egy elsőfokú aluláteresztőre cserélve a többi késleltetőnél behelyettesíthető a mindentáteresztő tag.

A hurokmentesítésre alternatív megoldást Karjalainen adott [21]: az ő megoldása megőrzi az összes mindentáteresztőt. A javasolt módosított struktúra a 3.5. ábrán látható. Az új szűrőegyütthetők a 3.1. algoritmussal számíthatjuk, ahol N jelöli a fokszámot, a_i az i -edik együtthető a visszacsatoló ágban, λ a warpolási tényező, C_i az új szűrőegyütthető, S_i pedig ideiglenes változó.

Az így létrehozott struktúra nagyon hasonlít a WFIR közvetlen realizációjához, ám azzal ellentétben tetszőleges helyen lehetnek pólusai. Számításigényben nem túl nagy a különbség a két struktúra között, ezért ahol más nem indokolja, WIIR szűrőt célszerű alkalmazni.


```

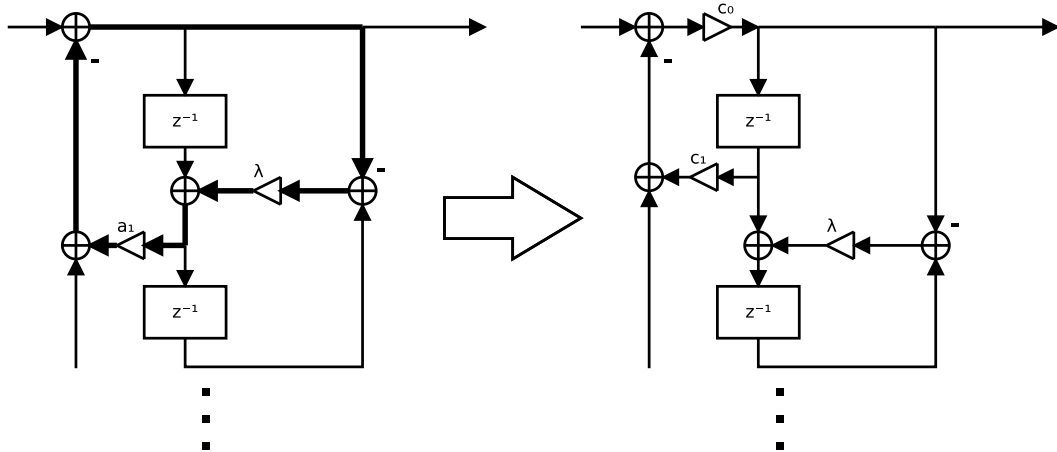

$$C_{N+1} = \lambda a_N$$


$$S_N = a_N$$

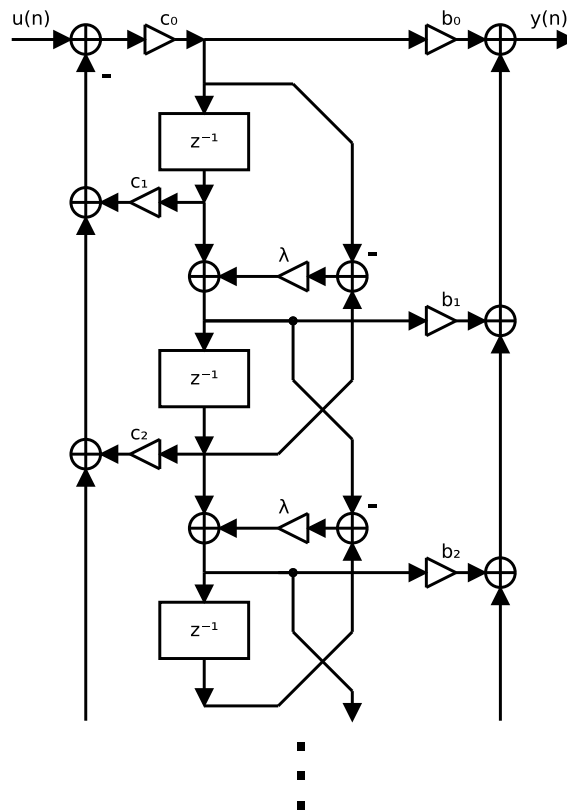
for  $i = N \dots 2$  do
  |  $S_{i-1} = a_{i-1} - \lambda S_i$ 
  |  $C_i = \lambda S_{i-1} + S_i$ 
end
 $C_1 = S_1$ 
 $C_0 = 1 - \lambda S_1$ 

```

3.1. algoritmus: Hurokmentesített WIIR együtthatóinak kiszámítása



3.5. ábra. Balra: WIIR közvetlen realizációban fellépő késleltetésmentes hurok
Jobbra: hurokmentesített verzió



3.6. ábra. WIIR módosított közvetlen realizáció

3.4. Soros és párhuzamos realizáció

A direkt form megvalósítású szűrőknél a fokszám növelésével egyre nagyobb problémát jelent az együtthatók ábrázolása (ld. 2.1. ábra) [18], így az átvitel eltérhet a kívánttól.

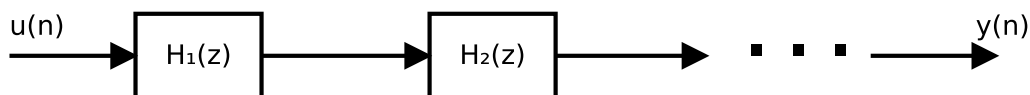
A problémára megoldást jelenthet, ha a szűrőt felbontjuk több, kisebb fokszámú tagra, majd ezeket kapcsoljuk sorosan, illetve párhuzamosan. Mivel a fokszám növelésével egyre érzékenyebbek lesznek a szűrők az együtthatók pontosságára, kézenfekvő, hogy minél kisebb tagokat alkossunk.

Soros realizációjú szűrőknél az eredő átvitel ($H(z)$) az egyes tagok átviteli függvényének szorzataként áll elő. Amennyiben egy tagnak M db pólusa és zérusa lehet, akkor:

$$H(z) = \prod_{i=1}^{\lceil N/M \rceil} H_i(z), \quad (3.2)$$

$$H_i(z) = \frac{\prod_{k=1}^M (z - z_{i,k})}{\prod_{k=1}^M (z - p_{i,k})}, \quad (3.3)$$

ahol $H_i(z)$ az i -edik tag átviteli függvénye, $z_{i,k}$ és $p_{i,k}$ pedig a tag zérusai és pólusai.



3.7. ábra. Soros topológiájú szűrő

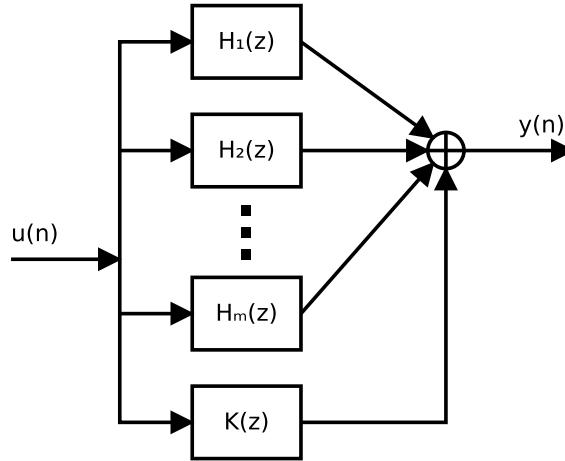
A pólusok és zérusok ismeretében ilyen módon könnyen előállítható a kívánt átvitel még akkor is, ha a megvalósítandó szűrőnek egynél nagyobb fokszámú pólusai, illetve zérusai vannak. Látható továbbá, hogy a módszerrel tetszőleges egész fokszámú tagokra bontható az átviteli függvény.

Más a helyzet a párhuzamos realizációnál. Amikor az átviteli függvényt felbontjuk racionális törtfüggvények összegére, az egyes nevezők fokszáma legalább akkora lesz, mint a legnagyobb multiplicitású pólus fokszáma. Ebből látható, hogy nem célszerű olyan szűrőt párhuzamos struktúrájává alakítani, amiben kettőnél nagyobb multiplicitású pólus található.

A tagok számlálóját a rész törtre bontás eredményeként eggyel kisebb fokszámúak lesznek, mint a nevezőjük. Azaz például másodfokú tagokra bontásnál az egyes tagoknak valós zérusai lesznek:

$$H(z) = \frac{\sum_{k=0}^N b_k \cdot z^{-k}}{1 + \sum_{k=1}^N a_k \cdot z^{-k}} = \sum_{i=1}^{\lceil N/2 \rceil} \frac{c_{0,i} + c_{1,i} \cdot z^{-1}}{1 + d_{1,i} \cdot z^{-1} + d_{2,i} \cdot z^{-2}} + K(z) = \sum_{i=1}^{\lceil N/2 \rceil} H_i(z) + K(z). \quad (3.4)$$

Az egyenletben szereplő $K(z)$ egy maradék FIR tag, ami azonos fokszámú számláló és nevező esetén csupán egy szorzótényező lesz.



3.8. ábra. Párhuzamos topológiájú szűrő

3.5. Másodfokú tagok

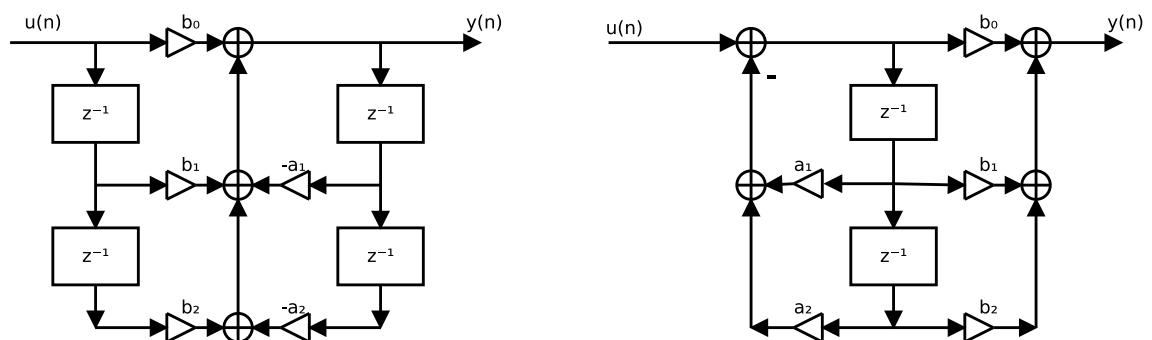
Valós szűrőegyütthatóknál konjugált komplex pólus-és zéruspárt legalább másodfokú tagban lehet realizálni. Az idők során több, különböző struktúrát alkottak meg másodfokú tagok implementálására, a fejezet további részében ezeket mutatom be.

3.5.1. Másodfokú direkt form

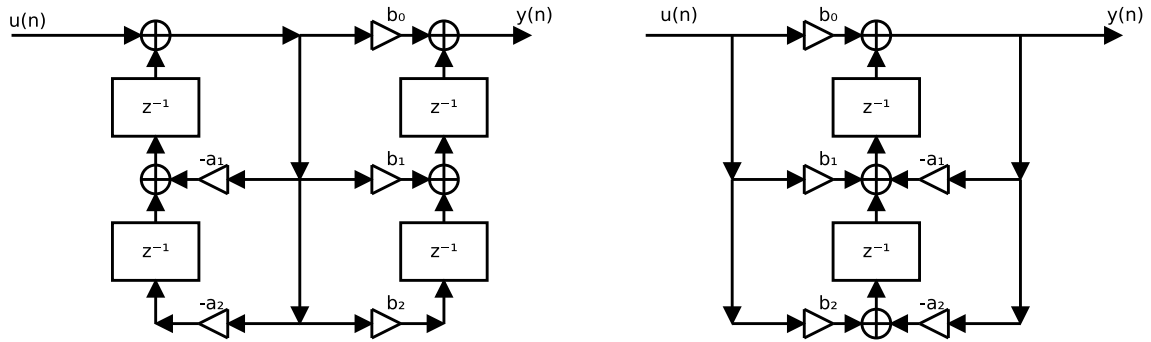
Másodfokú tagot legegyszerűbben direkt form struktúrában lehet megvalósítani. Az egyes tagok egységes átviteli függvénye:

$$H(z) = \frac{b_0 + b_1z^{-1} + b_2z^{-2}}{1 + a_1z^{-1} + a_2z^{-2}}. \quad (3.5)$$

A másodfokú direkt form realizációkat mutatják a 3.9–3.10. ábrák. A tagok közül a direkt form I és II a legkisebb számításigényűek DSP-n, mivel képesek kihasználni az architektúra speciális felépítését (MAC utasítások, cirkuláris címzés).



3.9. ábra. Balra: Direkt form I, jobbra: Direkt form II



3.10. ábra. Balra: DF-I transzponált, jobbra: DF-II transzponált

A szűrőegyütthatók és a konjugált komplex póluspár elhelyezkedésének kapcsolata:

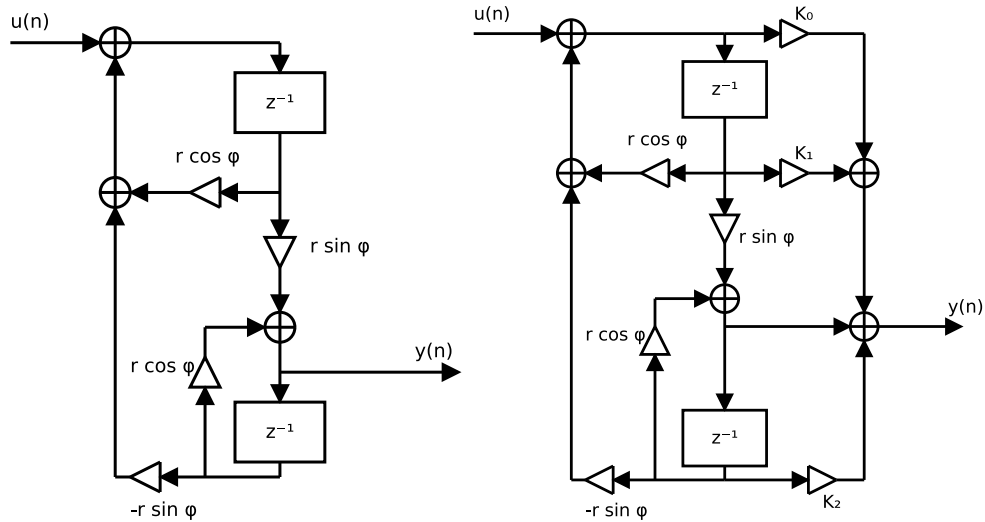
$$a_1 = -2 \cdot r_{\text{pólus}} \cdot \cos \varphi, \quad (3.6)$$

$$a_2 = r_{\text{pólus}}^2, \quad (3.7)$$

ahol $r_{\text{pólus}}$ a pólushossz, φ pedig a pólus szöge a komplex síkon.

3.5.2. Gold & Rader másodfokú tag

Gold és Rader 1967-os cikkükben [16] egy olyan szűrőstruktúrát vázolnak fel, amely kedvező kvantálási jelenségekkel rendelkezik. A cikk nem tárgyalta a tag zérusait, így a struktúrát bővítenem kellett (3.11. ábra).



3.11. ábra. Balra: Gold és Rader eredeti cikkében szereplő tag, Jobbra: zérusokkal kiegészített tag

A kiegészített tag átviteli függvénye:

$$H(z) = \frac{K_0 + (r \sin \varphi + K_1 - K_0 r \cos \varphi)z^{-1} + (K_2 r \sin \varphi - K_1 r \cos \varphi)z^{-2}}{1 - 2r \cos \varphi z^{-1} + r^2 z^{-2}} \quad (3.8)$$

ahol r és φ a pólus hossza és szöge a komplex számsíkon.

Ha az egyes szűrőegyütthatókat egy másodfokú direkt form tag átviteléből

(ld. (3.5). képlet) származtatjuk, akkor:

$$K_0 = b_0, \quad (3.9)$$

$$K_1 = b_1 + b_0 r \cos \varphi - r \sin \varphi, \quad (3.10)$$

$$K_2 = \frac{b_2 + b_1 r \cos \varphi + b_0 (r \cos \varphi)^2}{r \sin \varphi} - r \cos \varphi, \quad (3.11)$$

ahol r a konjugált komplex pólus hossza és φ a szöge. Ezeket a direkt form tag képletéből a következőképpen számolhatjuk:

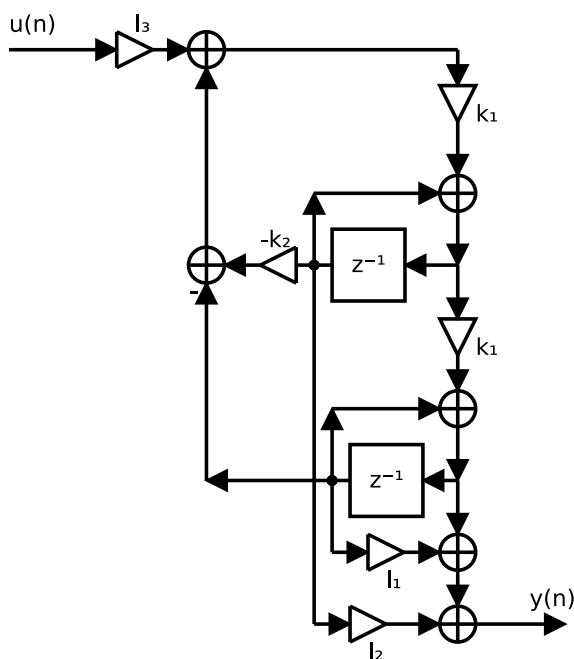
$$r = \sqrt{a_2} \quad (3.12)$$

$$\varphi = \arccos \left(\frac{-a_1}{2\sqrt{a_2}} \right). \quad (3.13)$$

A 3.11. ábrából is látható, hogy ezzel a struktúrával csak és kizárólag konjugált komplex póluspárt tartalmazó tagot lehet implementálni. A $\sin \varphi$ ugyanis valós pólusoknál nullával egyenlő, ez pedig mind tervezési időben (K_2 számításakor), mind pedig futási időben (ha nincsenek zérusok: azonosan nulla kimenet a bővítetlen struktúráján) gondot okozhat.

3.5.3. Kingsbury másodfokú tag

Kingsbury célja a struktúra megalkotásakor az volt, hogy olyan másodfokú tagot hozzon létre, amely egységkörhöz közeli pólusok esetén minél kevésbé legyen érzékeny az együtthatók kerekítésére [15]. Audio célú alkalmazásoknál gyakori, hogy a pólus hossza megközelíti az 1-et, így érdemes megvizsgálni ezt a struktúrát is. A tag blokkvázlata a 3.12. ábrán látható.



3.12. ábra. Kingsbury cikkében szereplő tag

A tag átviteli függvénye:

$$H(z) = \frac{l_3 k_1^2 + l_3 k_1^2 (l_1 + l_2) z^{-1} + l_1 l_2 l_3 k_1^2 z^{-2}}{1 - (k_1 k_2 + k_1^2 - 2) z^{-1} + (1 - k_1 k_2) z^{-2}} \quad (3.14)$$

Az egyes szűrőegyütthetők egy másodfokú direkt form tag átviteléből származtatva:

$$k_1 = \sqrt{1 + a_1 + a_2} \quad (3.15)$$

$$k_2 = \frac{1 - a_2}{k_1} \quad (3.16)$$

$$l_1 = \frac{b_1 + b_2}{b_0} \quad (3.17)$$

$$l_2 = -\frac{b_2}{b_0} k_1 \quad (3.18)$$

$$l_3 = \frac{b_0}{k_1^2} \quad (3.19)$$

Felmerül a kérdés, hogy milyen megkötések vannak a pólusok és zérusok elhelyezkedésére, ugyanis k_1 -gyel és b_0 -val is osztunk. A b_0 -ra vonatkozó korlátozás egyszerű: nem lehet nulla, azaz a másodfokú taggal nem implementálhatunk olyan szűrőt, aminek egy ütemnyi késleltetése van egyetlen valós zérus mellett.

A (3.6)-(3.7). képletek k_1 képletébe behelyettesítésével az alábbi kifejezéshez jutunk:

$$k_1 = \sqrt{1 - 2r \cos \varphi + r^2}. \quad (3.20)$$

Figyelembe véve, hogy k_1 valós, az alábbi kritériumot támaszthatjuk a pólussal szemben:

$$1 - 2r \cos \varphi + r^2 > 0. \quad (3.21)$$

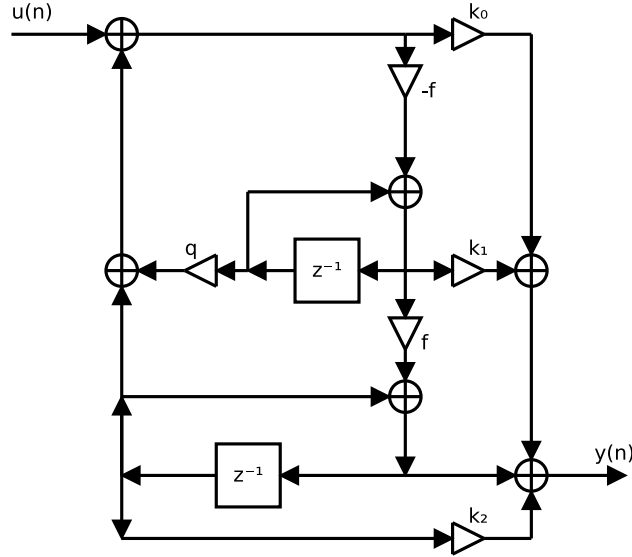
A képlet elemzéséből arra a konklúzióra juthatunk, hogy a Kingsbury-féle másodfokú tagnak nem lehet -1 -ben pólusa.

3.5.4. Chamberlin másodfokú tag

Chamberlin könyve [22] említ egy, az együtthetők kvantálására kevésbé érzékeny struktúrát. Az általa leírt megvalósítás a pólusok szempontjából ekvivalens a Kingsbury-féle struktúrával, a különbség a zérusok megvalósításában van. A tag blokkvázlata a 3.13. ábrán látható.

A tag átviteli függvénye:

$$H(z) = \frac{(k_0 - f^2 - f k_1) - (2k_0 - f k_1 + f^2 k_2) z^{-1} + k_0 z^{-2}}{1 - (2 - f q - f^2) z^{-1} + (1 - f q) z^{-2}} \quad (3.22)$$



3.13. ábra. Chamberlin könyvében szereplő tag

Az egyes szűrőegyütthatók a direkt form tag átviteléből származtatva:

$$f = \sqrt{1 + a_1 + a_2} \quad (3.23)$$

$$q = \frac{1 - a_2}{f} \quad (3.24)$$

$$k_0 = b_2 \quad (3.25)$$

$$k_1 = \frac{b_2 - b_0}{f} - f \quad (3.26)$$

$$k_2 = -\frac{b_0 + b_1 + b_2}{f^2} - 1 \quad (3.27)$$

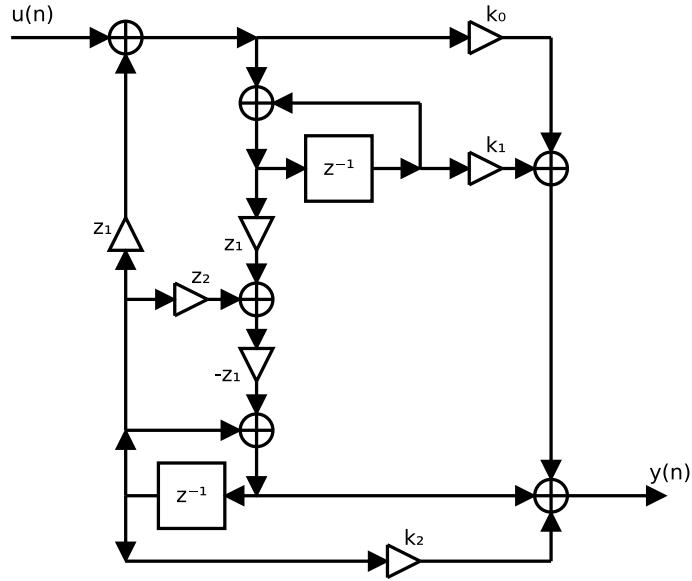
Vegyük észre, hogy f és q értéke megegyezik a Kingsbury-féle struktúra k_1 és k_2 paramétereivel. Az átvitelre tehát hasonló megállapítást tehetünk: a kívánt átvitelnek nem lehetnek -1 -ben pólusai.

3.5.5. Zölzer másodfokú tag

Zölzer több munkájában is [23, 17, 5] ismertet egy általa kifejezetten audio célokra kidolgozott másodfokú tagot. A struktúra az együttható-kerekítésre kisfrekvenciákon kevésbé érzékeny, valamint ugyanitt alacsony a numerikus zaja. A tag blokkvázlata a 3.14. ábrán látható.

A tag átviteli függvénye:

$$H(z) = \frac{(k_0 - z_1^2) + (k_1 - 2k_0 + k_0 z_1 z_2 - k_2 z_1^2)z^{-1} + (k_0 - k_0 z_1 z_2 - k_1 + k_1 z_1 z_2)z^{-2}}{1 + (z_1^3 + z_1 z_2 - 2)z^{-1} + (1 - z_1 z_2)z^{-2}} \quad (3.28)$$



3.14. ábra. Zölzer-féle másodfokú tag

A szűrőegyütthatók a direkt form átviteléből származtatva:

$$z_1 = \sqrt[3]{1 + a_1 + a_2} \quad (3.29)$$

$$z_2 = \frac{1 - a_2}{\sqrt[3]{1 + a_1 + a_2}} \quad (3.30)$$

$$k_0 = b_0 + z_1^2 \quad (3.31)$$

$$k_1 = b_0 + z_1^2 - \frac{b_2}{a_2} \quad (3.32)$$

$$k_2 = \frac{-b_2 - b_1 a_2 - b_0 a_2^2}{a_2 z_1^2} \quad (3.33)$$

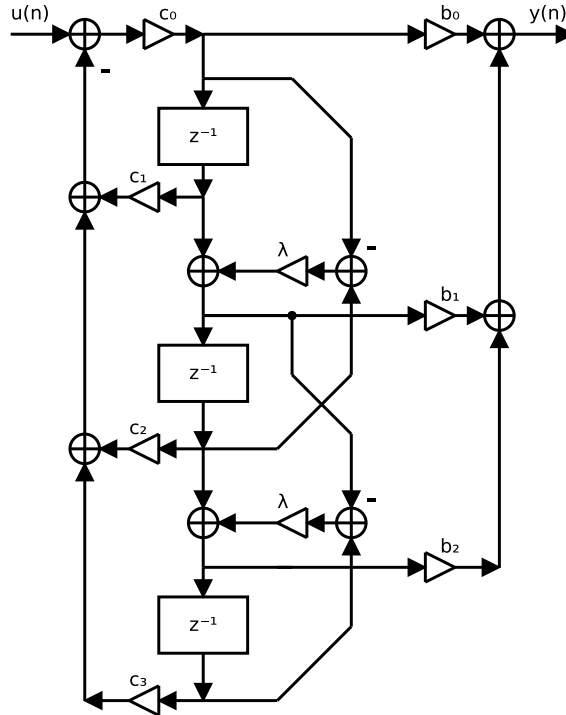
$$(3.34)$$

Belátható, hogy ennél a tagnál is vannak megkötések: z_1 és a_2 nem lehetnek nullák. Az $a_2 \neq 0$ feltételből következik, hogy a tagnak nem lehet nullában pólusa (ld. (3.7). képlet). A z_1 -re vonatkozó megkötésből levezethető, hogy a Zölzer-féle másodfokú tagnak sem lehet -1 -ben pólusa.

3.5.6. WIIR

Warpolt IIR szűrőből is képezhetünk másodfokú tagot. Az együtthatók meghatározásánál a warpolási tényező (λ) szabad paraméterként megválasztható marad.

Amennyiben adott egy megvalósítandó átvitel a (3.5). képlet szerinti formában, akkor a warpolt szűrő együtthatóit úgy kapjuk meg, hogy kiszámoljuk a transzformált frekven-



3.15. ábra. WIIR másodfokú tag

ciatartományban a szűrőegyütthetők:

$$b'_0 = \frac{b_0 + b_1\lambda + b_2\lambda^2}{1 + a_1\lambda + a_2\lambda^2}, \quad (3.35)$$

$$b'_1 = \frac{b_1 + 2b_0\lambda + 2b_2\lambda + b_1\lambda^2}{1 + a_1\lambda + a_2\lambda^2}, \quad (3.36)$$

$$b'_2 = \frac{b_2 + b_1\lambda + b_0\lambda^2}{1 + a_1\lambda + a_2\lambda^2} \quad (3.37)$$

$$a'_1 = \frac{a_1 + 2\lambda + 2a_2\lambda + a_1\lambda^2}{1 + a_1\lambda + a_2\lambda^2}, \quad (3.38)$$

$$a'_2 = \frac{a_2 + a_1\lambda + \lambda^2}{1 + a_1\lambda + a_2\lambda^2}. \quad (3.39)$$

A vesszővel jelölt warpolt szűrőegyütthetőkön végrehajtva a c_i együtthetők kiszámoló 3.1. algoritmust (24. oldal) megkapjuk a másodfokú WIIR szűrő paramétereit.

A tervezés során a λ warpolási tényező szabad változóként van jelen. Helyes megválasztásával optimalizálható a szűrőegyütthetők nagyságrendje, valamint a vizsgálatoknál látni fogjuk, hogy a numerikus zaj teljesítményére is hatással van.

4. fejezet

Vizsgálati módszer

Másodfokú tagok implementációjánál több kvantálási jelenséget figyelhetünk meg. A fejezetben az együttható-kerekítés, a numerikus zaj és a túlcsondulás vizsgálatára mutatok be módszereket.

4.1. Együttható-kerekítés

Szűrőt legtöbbször duplapontosságú lebegőpontos számábrázolás mellett tervezünk. A előállított együtthatókat ezután fixpontossá konvertáljuk, ezzel együtt meg is változtatjuk őket.

Az együtthatók kerekítésének vizsgálatára létezik analitikus módszer [18], amit ebben az alfejezetben másodfokú tagokra alkalmazok.

4.1.1. Kerekítés hatása a pólusokra

Az együtthatók kerekítésével elmozdulnak a pólusok, szélsőséges esetben az egységkörön kívülre is kerülhetnek, így a stabilitás szempontjából fontos az elmozdulás vizsgálata.

Vegyük példaként a másodfokú direkt form tagok átvitelét:

$$H(z) = \frac{b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2}}{1 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2}} = \frac{b_0 \cdot (1 - z_z \cdot z^{-1})(1 - \bar{z}_z \cdot z^{-1})}{(1 - z_p \cdot z^{-1})(1 - \bar{z}_p \cdot z^{-1})}. \quad (4.1)$$

A megvalósított pólusokat az átvitelből a következőképpen írhatjuk fel:

$$z'_p = z_p + \Delta z_p,$$

ahol z'_p a megvalósított, z_p a kvantálatlan pólus, Δz_p pedig a kerekítés hatása. Ahhoz, hogy a keresett Δz_p értékét megkapjuk, ki kell fejezni a pólusokat az átviteli függvény polinomos alakjából:

$$z_p = \frac{-a_1 + \sqrt{a_1^2 - 4a_2}}{2} \quad \bar{z}_p = \frac{-a_1 - \sqrt{a_1^2 - 4a_2}}{2} \quad (4.2)$$

Innen deriválással kaphatjuk meg az egyes együtthatók megváltozására történő elmozdu-

lást:

$$\Delta z_p \Big|_{a_i} = \frac{\partial z_p}{\partial a_i} \cdot \Delta a_i,$$

ahol Δa_i az i -edik együttható megváltozását jelöli. Ezek értékei másodfokú direkt form tagnál:

$$\Delta z_p \Big|_{a_1} = \Delta a_1 \cdot \left(-\frac{1}{2} + \frac{a_1}{2\sqrt{a_1^2 - 4a_2}} \right), \quad (4.3)$$

$$\Delta z_p \Big|_{a_2} = \Delta a_2 \cdot \frac{-2}{\sqrt{a_1^2 - 4a_2}}. \quad (4.4)$$

Innentől az egyes hatásokat össze kell adni: a pólusok legnagyobb elmozdulását keressük, ezért worst-case összegzést végzünk. Mivel azonban komplex számokkal dolgozunk, nem abszolút összeget, hanem euklideszi normát kell számolni. Azaz a pólusok maximális elmozdulása:

$$|\Delta z_p| = \sqrt{\left(\sum_{i=1}^N \left| \Re \left\{ \Delta a_i \cdot \frac{\partial z_p}{\partial a_i} \right\} \right| \right)^2 + \left(\sum_{i=1}^N \left| \Im \left\{ \Delta a_i \cdot \frac{\partial z_p}{\partial a_i} \right\} \right| \right)^2}.$$

Az együtthatók ideálistól való legnagyobb eltérése a kvantálási küszöb fele lesz ($\frac{q}{2}$), azaz:

$$|\Delta b_0| = |\Delta b_1| = |\Delta b_2| = |\Delta a_1| = |\Delta a_2| = \frac{q}{2}. \quad (4.5)$$

Vegyük észre, hogy a (4.3)–(4.4.) képletekben szereplő nevező értéke konjugált komplex póluspár esetén komplex, míg valós pólusok esetén valós. Ebből következően két esetet kell vizsgálni:

$$\left| \Delta z_p \right|_{\text{konj. komp. póluspár}} = \frac{q}{2} \cdot \sqrt{\frac{1}{4} + \frac{a_1^2 + 8|a_1| + 16}{4|a_1^2 - 4a_2|}}, \quad (4.6)$$

$$\left| \Delta z_p \right|_{\text{valós pólusok}} = \frac{q}{2} \cdot \frac{|a_1 - \sqrt{a_1^2 - 4a_2}| + 4}{2\sqrt{a_1^2 - 4a_2}}. \quad (4.7)$$

Az egyenleteket kiértékelve felírható egy $|\Delta z_p|$ sugarú kör a kerekítés előtti pólus köré, amelyen belül fog elhelyezkedni a kvantálás utáni pólus. Amennyiben ez a kör teljes egészében a komplex egységkörön belül helyezkedik el, akkor a kerekítés nem fogja instabillá tenni a szűrőt. Ha a feltétel nem áll fenn, akkor előfordulhat, hogy a megvalósítás instabil lesz.

A további vizsgálatokhoz írjuk fel a konjugált komplex póluspár hosszát (r) és szögét

(φ) , valamint a valós és képzetes részét:

$$r = \sqrt{a_2}, \quad \varphi = \arccos\left(\frac{-a_1}{2\sqrt{a_2}}\right), \quad (4.8)$$

$$\Re\{z_p\} = \frac{-a_1}{2}, \quad \Im\{z_p\} = \pm \frac{\sqrt{a_1^2 - 4a_2}}{2}. \quad (4.9)$$

A kifejezéseket összevetve a (4.6)–(4.7.) egyenletekkel látható, hogy a pólusok elmozdulása akkor lesz a legnagyobb, ha $\sqrt{a_1^2 - 4a_2}$ értéke 0-hoz közeli. Ez konjugált komplex pólus-pár esetén megegyezik a pólusok képzetes részének kétszeresével, valós pólusoknál pedig a (4.2) képlet értelmében a két pólus különbségével.

Vegyük észre, hogy kétszeres pólus esetén a (4.3)–(4.4) képletek nevezőjében szereplő $\sqrt{a_1^2 - 4a_2}$ kifejezés értéke 0, így az érzékenységek végtelenek lesznek. Ebben az esetben tehát nem használható az érzékenységszámításon alapuló módszer. Írjuk fel a szűrőegyütthatókat az alábbi alakban:

$$a_1 = a_1^* + \Delta a_1, \quad a_2 = a_2^* + \Delta a_2,$$

ahol a_1^* és a_2^* a kerekítetlen együtthatók. Ekkor a kerekített pólus kifejezése:

$$z'_p = \frac{-a_1^* - \Delta a_1 \pm \sqrt{a_1^{*2} + \Delta a_1^2 - 2\Delta a_1 \cdot a_1^* - 4a_2^* - 4\Delta a_2}}{2}$$

Vegyük figyelembe, hogy a kerekítetlen együtthatóknál $a_1^{*2} - 4a_2^* = 0$, így:

$$\Delta z_p = \frac{-\Delta a_1 \pm \sqrt{\Delta a_1^2 - 2\Delta a_1 \cdot a_1^* - 4\Delta a_2}}{2},$$

ahol $\Delta z_p = z'_p - z_p$. Innentől elágazhatunk aszerint, hogy a valós, vagy a képzetes rész megváltozását vizsgáljuk. A legnagyobb valós eltérés akkor lesz, ha a gyökjel alatti kifejezés pozitív és maximális. Ez akkor következik be, ha Δa_2 előjele negatív, Δa_1 és a_1 előjele pedig ellentétes. Ekkor kihasználva a (4.5) összefüggést:

$$|\Delta z_p|_{valós} = \frac{1}{2} \left(\frac{q}{2} + \sqrt{a_1 q + \frac{q^2}{4} + 2q} \right).$$

Hasonló megfontolásokkal a legnagyobb képzetes irányú eltéréskor Δa_2 előjele pozitív, Δa_1 és a_1 előjele pedig azonos. Ekkor a legnagyobb képzetes irányú eltérés:

$$|\Delta z_p|_{képzetes} = \frac{1}{2} \sqrt{a_1 q + \frac{q^2}{4} + 2q}.$$

Látható, hogy a stabilitás vizsgálatára analitikusan csak azt állapíthatjuk meg, hogy adott megvalósítandó pólusok esetén lehet-e instabil a kerekítés utáni szűrő. A gyakorlatban persze nem ebből érdemes kiindulni: célszerű a kerekítés után megnézni, hogy hova kerültek a pólusok, és e szerint dönteni, hogy melyik struktúrával valósítjuk meg a tagot.

4.1.2. Kerekítés hatása az átvitelre

Az együtthatók kerekítésének hatására az átvitel is megváltozik. A szűrő stabilitásának ismerete még önmagában nem elegendő, ha ugyanis az átvitel túlságosan megváltozik, akkor a szűrő nem fog megfelelni a célnak.

Az analitikus vizsgálathoz vegyük a másodfokú direkt form tag átvitelét a (4.1) képlet szerint! A megváltozott együtthatók és az átvitel a következőképpen alakul:

$$\begin{aligned} a'_i &= a_i + \Delta a_i \\ b'_i &= b_i + \Delta b_i \\ H'(z) &= H(z) + \Delta H(z) \end{aligned}$$

Egy adott együttható (c) kerekítésének hatását deriválással kaphatjuk:

$$\Delta H(z) \Big|_c = \Delta c \cdot \frac{\partial H(z)}{\partial c}.$$

Ez másodfokú direkt form tag esetében a következőképpen alakul:

$$\begin{aligned} \Delta H(z) \Big|_{b_i} &= \Delta b_i \cdot \frac{z^{-i}}{1 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2}}, \\ \Delta H(z) \Big|_{a_i} &= -\Delta a_i \cdot \frac{b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2}}{(1 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2})^2} \cdot z^{-i}. \end{aligned}$$

A továbbiakban összegezni kell a hatásokat. Amennyiben ismert, hogy az együtthatók mennyit változnak, akkor az egyszerű előjeles összegzéssel adódó átvitel pont a triviális megoldásnak felel meg:

$$H'(z) = \frac{(b_0 + \Delta b_0) + (b_1 + \Delta b_1) \cdot z^{-1} + (b_2 + \Delta b_2) \cdot z^{-2}}{1 + (a_1 + \Delta a_1) \cdot z^{-1} + (a_2 + \Delta a_2) \cdot z^{-2}}. \quad (4.10)$$

Ha a legnagyobb eltérést keressük, akkor worst-case összegzést kell végrehajtani:

$$|\Delta H(z)| = \sqrt{\left(\sum_c \left| \Re \left\{ \Delta c \cdot \frac{\partial H(z)}{\partial c} \right\} \right| \right)^2 + \left(\sum_c \left| \Im \left\{ \Delta c \cdot \frac{\partial H(z)}{\partial c} \right\} \right| \right)^2},$$

ahol c jelöli az együtthatókat. A módszert körülményessé teszi, hogy az átvitelben z egy komplex paraméter. Megoldást jelenthet, ha a $z = e^{j\omega}$ helyettesítés után különböző ω frekvenciákra kiértékeljük az egyenletet. A legnagyobb $|\Delta H(e^{j\omega})|$ értéke lesz tulajdonképpen a worst-case eltérés.

Látható, hogy az analitikus módszer bonyolult, mivel az össze struktúra minden paraméterére ki kell számítani az érzékenységet. A vizsgálat egyszerűbben is kivitelezhető: ehhez lebegőpontos számábrázolás mellett kell a fixpontos kvantálási küszöb felét hozzáadni, vagy kivonni a szűrőegyütthatókból. Az így létrejövő átvitel csak egyik szélső esete lesz

a kerekítés hatásának, emiatt minden lehetséges variációra ki kell értékelni az átviteleket. A worst-case hiba ezzel a módszerrel az eredeti és a megváltoztatott átvitelek legnagyobb különbségeként adódik.

Belátható, hogy ez utóbbi módszer K db szűrőegyüttható esetén 2^K átvitel kiértékelését eredményezi, azaz nagy foksámú szűrők ($N > 10$) esetén túl nagy számításigényűvé válik. Másodfokú tagoknál viszont használható: a legtöbb paraméterrel rendelkező warpolt tagok együtthathóinak száma 7, így még ez esetben is csak 128 db átvitelt kell kiértékelni egy adott pólus-zérus elrendezés worst-case kerekítési hibájának vizsgálatához.

4.2. Processzormodell

Ahhoz, hogy meg tudjuk határozni a kvantálási pontokat és a számítási igényt, szükségünk lesz egy egyszerűsített processzormodellre.

A nagyteljesítményű mikrovezérlők és a fixpontos DSP-k közös jellemzője a MAC (multiply and accumulate) művelet megléte. Ez egy olyan SIMD utasítás, melynél a két bemeneti érték szorzata hozzáadódik az akkumulátorhoz, ami jelprocesszoroknál egy külön entitás, mikrovezérlőknél pedig két regiszter összekapcsolásával jön létre. Mindkét esetben elmondható viszont, hogy az akkumulátor bitszélessége legalább kétszerese a regiszterekének, és további szorzást közvetlenül nem lehet végrehajtani rajta.

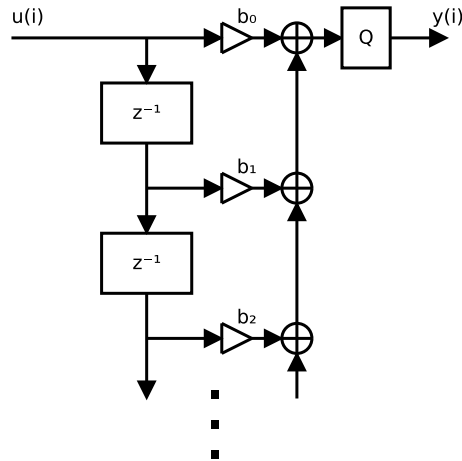
Fontos megjegyezni, hogy a memória kezelése különbözik a mikrovezérlőknél és a jelprocesszoroknál. Annak érdekében, hogy az egyes struktúrák mégis összehasonlíthatóak legyenek a számításigény tekintetében, külön memóriakezelő (Load-Store) utasításokat definiálok, melyek egy-egy konkrét processzoron lehet, hogy több utasításból valósíthatók meg, de az is lehet, hogy beépíthetők más utasításokba (pl. DSP-nél MAC automata inkrementálással és címzéssel). A definícióból következik, hogy a processzormodellen implementált szűrők számításigénye nem feltétlenül egyezik meg egy valós processzoron megvalósított szűrőével, de a becslés még mindig sokkal jobb, mint az irodalomban általánosan használt szorzások és összeadások száma.

Utasítás	Végrehajtott művelet
ADD A, B, C	$A = B + C$
SUB A, B, C	$A = B - C$
MUL A, B, C	$A = B \cdot C$
MAC Acc, B, C	$Acc = Acc + B \cdot C$
CLR Acc	$Acc = 0$
MOV B, Acc	$B = Acc$
LOAD $A, < mem >$	A betöltése memóriából
STORE $A, < mem >$	A kiírása memóriába

4.1. táblázat. Modell processzor utasításkészlet

Az utasításoknál nagybetűvel jelölöm a processzor regisztereit. Az Acc jelölés az akkumulátort jelenti, amit az egyszerűség kedvéért külön entitásnak tekintek. Akkumulátorból kerekítéssel regiszter méretű adatot kivenni a MOV utasítással, tartalmat törölni a CLR utasítással lehet.

A ciklusok szervezését nem vizsgálom. Az egyes szűrőknél csak a tényleges jelfeldolgozást végző algoritmust jelölöm, ehhez pedig a 4.1. táblázatban szereplő utasításokat használom. Egy példát mutat be a 2. algoritmus, ahol egy FIR szűrőt implementáltam. A fejezetben csak a számításigényhez szükséges utasításszámot adom meg, a részletes implementációkat a függelékben helyeztem el.



4.1. ábra. FIR szűrő

```

for  $n = 0 \dots \text{mintaszám}$  do
  CLR Acc
  for  $i = 0 \dots N$  do
    LOAD B,  $b[i]$ 
    LOAD C,  $x[i]$  /*  $x[0]$  a bemenet */
    MAC Acc, B, C
  end
  MOV A, Acc
  STORE A,  $y[n]$ 
end

```

4.1. algoritmus: Példa: N -ed fokú FIR szűrő implementációja a processzormodellen.

4.3. Kvantálási zaj

Vizsgálataim során arra keresem a választ, hogy másodfokú tagok soros, vagy párhuzamos kapcsolása esetén hogyan lehet csökkenteni a numerikus zajt. Ehhez az egyes tagok zajspektrumát vizsgálom, ha ugyanis sikerül adott pólus-zérus elrendezésnél minimális zajú tagot találni, akkor az egyes tagokból előállítva a szűrőt, az eredő zaj minimális lesz párhuzamos struktúrájánál. Kaszkád kapcsolásnál ez még nem elégséges feltétel, ott ugyanis a tagok sorrendje is hatással van a zajra.

Másodfokú tagok használatával viszonylag kicsi az együtthatók kerekítéséből származó hiba [18]. A későbbiekben azzal a feltételezéssel élek, hogy a kiválasztott szűrőtagok átvitele megfelel a célunknak, így a hangsúlyt a kvantálási zajra helyezem.

A zaj spektrumának mérésére több megoldás lehetséges. Az egyik, hogy fixponton és dupla pontosságú lebegőponton is implementáljuk a szűrőt, majd a kimeneteik különbségé-

ből periodogramot számolunk. A módszernél feltesszük, hogy a dupla pontosságú lebegő-pontos számábrázolásnál sokkal kisebb a szűrő kvantálási zaja, mint fixponton. Problémát jelent, hogy a periodogram számításával a zajspektrumnak csak egy viszonylag pontatlan becslőjét lehet megkapni, azt is jelentős számítási igény mellett. Következésképpen nem ezt a módszert célszerű tényleges vizsgálatoknál alkalmazni.

A zajspektrum számítására létezik egy elméleti módszer is. A numerikus zajt a kvantálási pontokon egyenletes eloszlású fehérzajként modellezhetjük. Amennyiben csak egy ponton kerekítünk, akkor a kvantálási pont és a kimenet közötti átvitel formálni fogja a zaj spektrumát, azaz a spektrum:

$$P_{\text{zaj}}(\omega) = \frac{q^2}{12} \cdot \left| H_{\text{zaj} \rightarrow \text{kimenet}}(e^{j\omega}) \right|^2. \quad (4.11)$$

A későbbiekben a $\frac{q^2}{12}$ konstans σ_e^2 -ként fogom jelölni. Amennyiben több kvantálási pontunk van a szűrőben, akkor az egyes zajkomponenseket összegezni kell. A kerekítéseket az egyes pontokon tekinthetjük függetlennek, így a kimeneten valószínűségi (négyzetes) összegzéssel kaphatjuk meg a zajspektrumot [3]:

$$P_{\text{zaj}}(\omega) = \sigma_e^2 \cdot \sum_{i=1}^M \left| H_i(e^{j\omega}) \right|^2. \quad (4.12)$$

Vizsgálataimban a kvantálási pont és a kimenet közötti átvitelt számítógéppel számoltattam ki: a zaj keletkezésének helyén dirac delta függvénnyel gerjesztettem a rendszert, így a kimeneten megjelenő impulzusválaszt Fourier-transzformálva megkaptam a keresett átvitelt.

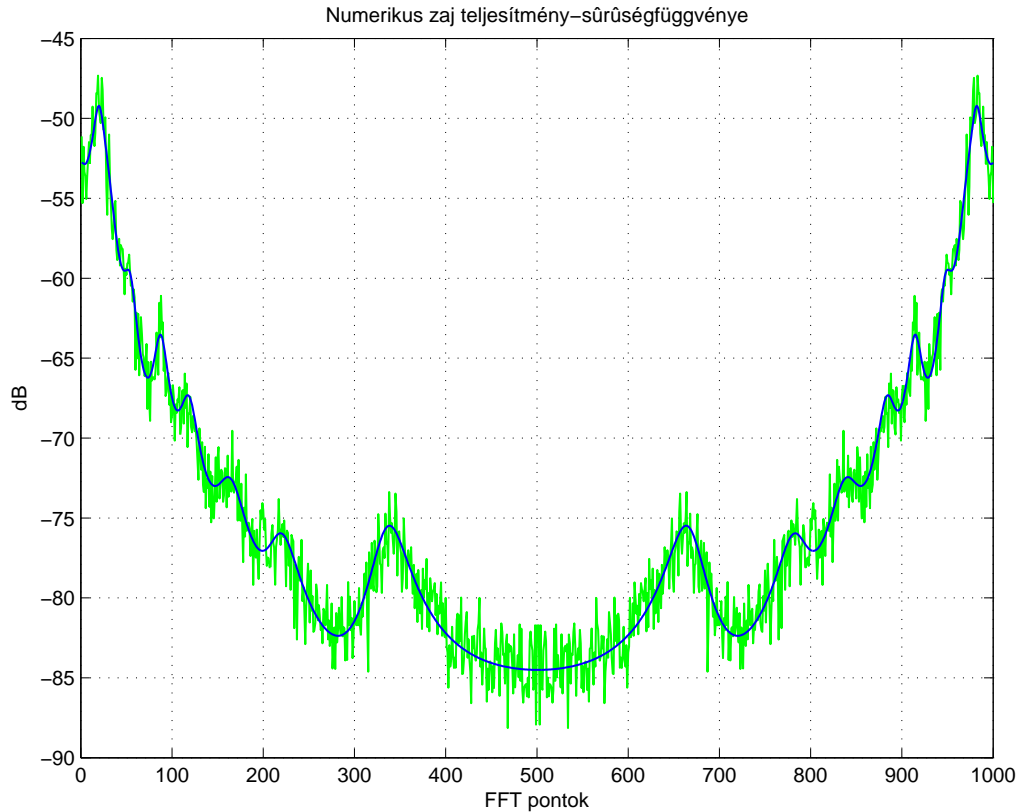
A módszer jól modellezi a valóságot. Példaként elvégeztem egy DF2 transzponált tagokból felépített soros szűrőn a zajspektrum vizsgálatát. A 4.2. ábrán látható, hogy a 25 periodogram átlaga még mindig elég nagy szórású, viszont az is látszik, hogy a periodogram várható értéke és az elméleti módszerrel kiszámított zajspektrum egybeesik.

A zajteljesítményt a spektrumból integrálással kaphatjuk meg. DFT alkalmazása esetén az integrálás helyett összegezni kell minden DFT pontra. Audio jelekről lévén szó a teljesítményt szeretnénk decibelben megadni, ehhez viszont egy referencia szint kell. Vizsgálataimban 0 dB-nek az egységnyi teljesítményt ($P = 1$) választottam, ami megegyezik az egységnyi varianciájú Gauss-zaj teljesítményének.

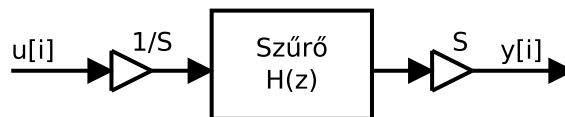
4.4. Túlcsondulás elkerülése

Ahhoz, hogy a szűrőben ne fordulhasson elő túlcsondulás, a bemenet előtt le kell skálázni a jelet, a kimeneten pedig fel kell szorozni. Ez viszont a szűrőben előálló kvantálási zajt is felszorozza a kimeneten, így fontos kérdés, hogy mennyi legyen a skálázási tényező.

A skálázás mértékével több irodalom is foglalkozik [3, 17], de pontos értéket nem tudnak adni, csak egy minimumot és egy maximumot. Az elégséges feltétel ugyanis túlságosan lecsökkenti a bemenőjelet, a szükséges teljesülése viszont nem minden jelnél garantálja a túlcsondulás elkerülését.



4.2. ábra. *DF²T* tagok soros kapcsolásából 16 biten felépített szűrő zajának periodogramja (zöld) és elméleti spektruma (kék)



4.3. ábra. Szűrő bemenetének és kimenetének skálázása

A probléma kezeléséhez abból indultam ki, hogy általános zenei jeleket jól modellezhetünk rózsazajjal. A skálázási tényező (S) meghatározásához először lebegőponton implementáltam a szűrőket, bemenetükre 1 másodpercre rózsazajt kapcsoltam és figyeltem a szűrőben előálló maximális jelértékeket. Amennyiben a maximum túllépte a fixpontos számábrázolás határát, akkor a skálázást ezzel az értékkel végeztem el. A szűrő egyes pontjai lineáris kapcsolatban vannak a bemenettel, az így elvégzett skálázással tehát nagy valószínűséggel elkerülhető a túlcsoordulás rózsazajnál és így zenei jelnél is.

4.5. Kvantálási zaj direkt realizációjú struktúrákban

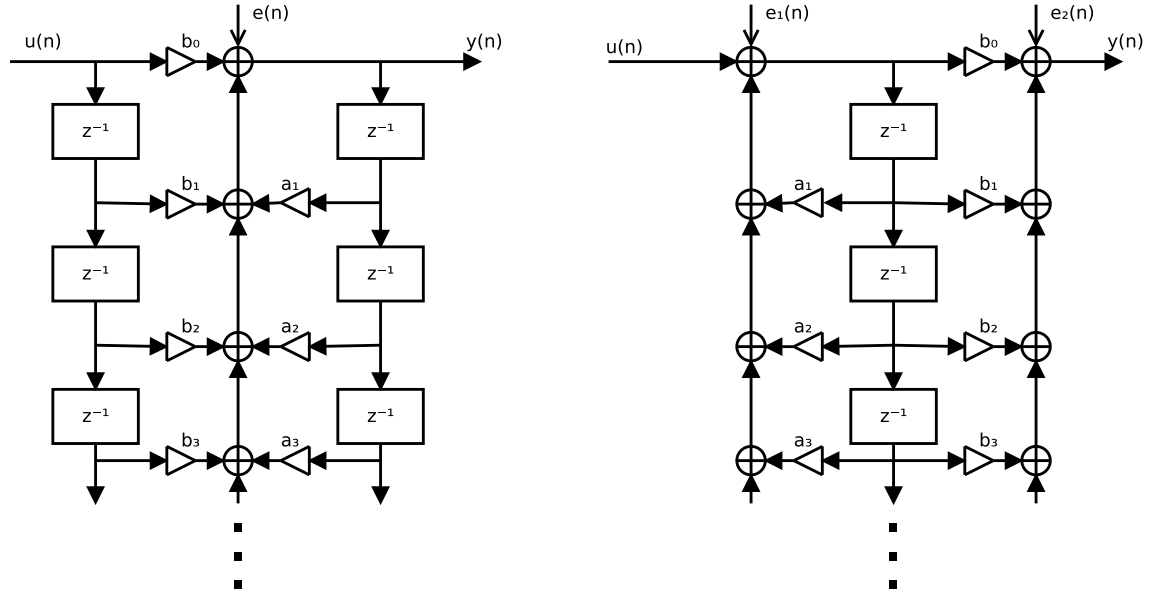
A vizsgálatoknál a hangsúlyt a másodfokú tagokból felépített szűrőkre helyezem, de célszerű elemezni a tetszőleges fokszámú tagokat is. Alapesetben ugyanis ezeket implementálnánk, de az együtttható-kerekítés, valamint a számításigény miatt mégsem ezeket választjuk. A későbbi vizsgálatoknál is hasznosak az itt leírtak, mert másodfokú tagot ezekből a struktúrákból is képezhetünk.

Az alfejezetben explicit képleteket is közléteszek a zajspektrumokra. Minden olyan ered-

ménynél, amit irodalomban találtam, jelölöm az adott képlet forrását.

4.5.1. IIR DF1 és DF2

A DF1-nél kettő, a DF2-nél csak egy ponton van kvantálás, a becsatolási pontok $e_i(n)$ -nel jelölve a 4.4. ábrán láthatók. Az implementációkat az F.1–F.2. algoritmusok tartalmazzák.



4.4. ábra. Jobbra: IIR DF1, Balra: IIR DF2

Vegyük észre, hogy míg a direkt-I realizációnál a zaj csak a visszacsatoló ágban jelenik meg, addig a direkt-II realizációban a teljes szűrőn áthalad. Ebből következik, hogy a zaj spektrális sűrűségfüggvénye [3]:

$$P_{\text{zaj,DF1}}(\omega) = \sigma_e^2 \cdot \left| \frac{1}{A(e^{j\omega})} \right|^2 \quad (4.13)$$

$$P_{\text{zaj,DF2}}(\omega) = \sigma_e^2 \cdot \left| \frac{B(e^{j\omega})}{A(e^{j\omega})} \right|^2 + \sigma_e^2 \quad (4.14)$$

A képletek alapján azt várjuk, hogy a direkt-I struktúra zaja lesz nagyobb, mert abba csak a kiemelés jelentő pólusok szólnak bele, míg a direkt-II realizációnál a zajt csillapítják a szűrő zérusai. Összességében viszont ez nem mond még semmit a jel-zaj viszonyról: az akkumulátor túlsordulása ellen le kell skálázni az átvitelt és felszorozni a kimenetet, ezáltal a zaj teljesítményét is fel kell szorozni. A skálázás értéke viszont függ magától a szűrőtől és a gerjesztéstől is.

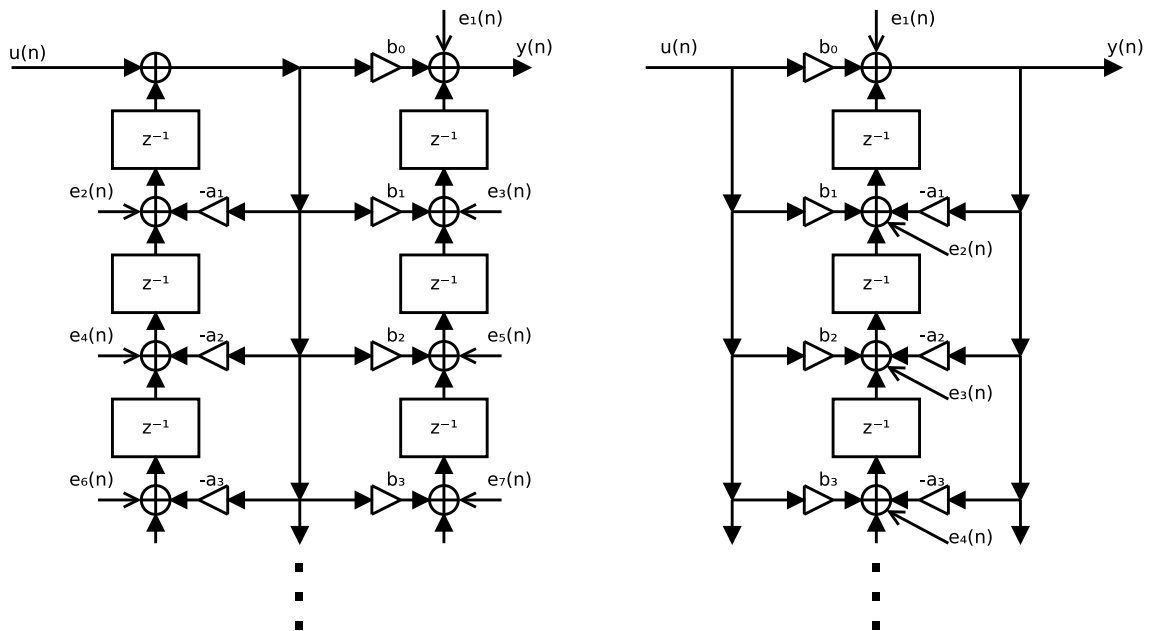
A számításgény vizsgálatánál azzal a feltételezéssel éltem, hogy az implementáció során igénybe vesszük a processzor DMA egységét, vagy cirkulárisan címzünk a késleltető tagok értékeinek léptetéséhez.

Direkt Form I		Direkt Form II	
Szükséges memória:	$2N$	Szükséges memória:	N
Akkumulátorok száma:	1	Akkumulátorok száma:	2
Töltések száma:	$4N + 2$	Töltések száma:	$3N + 1$
Tárolások száma:	2	Tárolások száma:	2
MAC utasítások:	$2N + 1$	MAC utasítások:	$2N + 1$
Összeadások:	0	Összeadások:	1
Akkumulátorból kivétel:	1	Akkumulátorból kivétel:	2
Akkumulátor törlés:	1	Akkumulátor törlés:	2
Számításigény:	$6N + 7$	Számításigény:	$5N + 9$

4.2. táblázat. N -ed fokú DF1 és DF2 szűrő erőforrásigénye

4.5.2. IIR DF1 és DF2 transzponált

A transzponált struktúrákban több helyen történik kvantálás, emiatt várható, hogy nagyobb zajjal bírnak [3]. Az implementációkat az F.3–F.5. algoritmusok tartalmazzák.



4.5. ábra. Jobbra: IIR DF1 transzponált, Balra: IIR DF2 transzponált

Vegyük észre, hogy a transzponált struktúrákban minden szorzás után kvantálni kellene az akkumulátort, azaz nem használható ki annak teljes szélessége. Célszerűbb inkább memória szélességű szorzással és összeadással dolgozni. Egy N -ed fokú direkt-I transzponált szűrőnél így $2N + 1$ helyen, egy direkt-II transzponált szűrőnél pedig $N + 1$ helyen kell kerekíteni.

A kvantálási zaj spektruma a következőképpen alakul:

$$P_{\text{zaj,DF1-T}}(\omega) = \sigma_e^2 \cdot \left(1 + \sum_{i=1}^{2N} \left| \tilde{H}_i(e^{j\omega}) \right|^2 \right), \quad (4.15)$$

$$P_{\text{zaj,DF2-T}}(\omega) = \sigma_e^2 \cdot \left(1 + \sum_{i=1}^N \left| \tilde{H}_i(e^{j\omega}) \right|^2 \right), \quad (4.16)$$

ahol $\tilde{H}_i(e^{j\omega})$ az i . kvantálási pont és a kimenet közötti átvitel.

Direkt Form I Transposed		Direkt Form II Transposed	
Szükséges memória:	$2N$	Szükséges memória:	N
Töltések száma:	$4N + 3$	Töltések száma:	$3N + 3$
Tárolások száma:	$2N + 1$	Tárolások száma:	$N + 1$
Szorzások:	$2N + 1$	Szorzások:	$2N + 1$
Összeadások:	$2N + 1$	Összeadások:	$2N + 1$
Számításigény:	$10N + 6$	Számításigény:	$8N + 6$

4.3. táblázat. N -ed fokú DF1T és DF2T szűrő erőforrásigénye

4.5.3. WIIR közvetlen realizáció

Korábban láttuk, hogy a warped IIR szűrő direkt realizációjánál nem elegendő a mindent-áteresztő tagokat behelyettesíteni, a struktúrát is át kell rendezni. A megvalósíthatóság érdekében módosított felépítés miatt a C_0 -val beszorzás előtt és után is történik kerekítés (akkumulátor tartalmát nem lehet közvetlenül szorozni). Ezt a két pontot a -1 és a 0 indexekkel, a kimeneti kvantálás pontját pedig k -val jelöltem a 4.6. ábrán. Belátható, hogy ezekből a pontokból a kimenetre vetített zajátvitel:

$$H_{e,-1}(z) = H(\Theta_\lambda(z)), \quad (4.17)$$

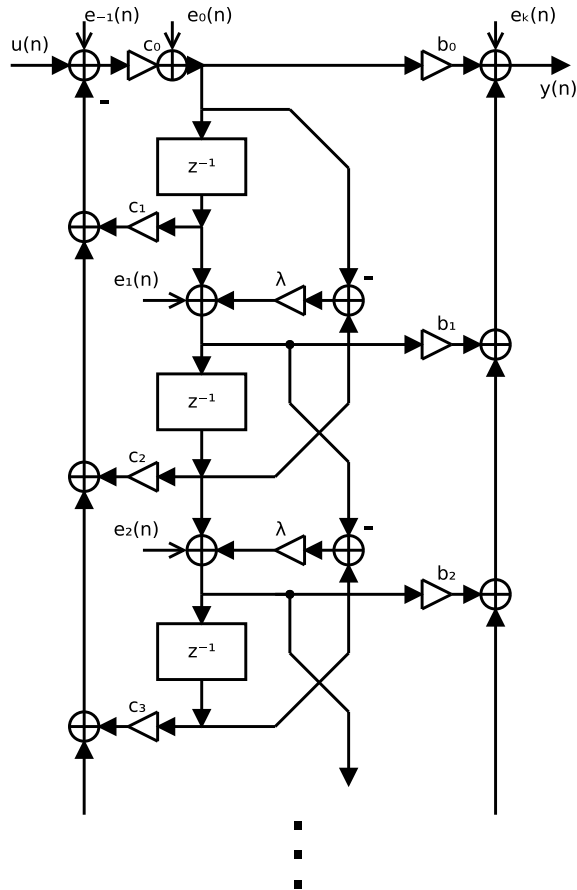
$$H_{e,0}(z) = \frac{1}{C_0} \cdot H(\Theta_\lambda(z)), \quad (4.18)$$

$$H_{e,k}(z) = 1, \quad (4.19)$$

ahol $H(\Theta_\lambda(z))$ a WIIR szűrő átvitele.

A többi pontra már nem egyszerű felírni a zajátvitelt, így a 4.3. bekezdésben ismertetett numerikus módon határozom meg a kvantálási pontok és a kimenet közötti átvitelt ($H_i(z)$). Ezekkel és a fenti átvitelekkel a kvantálási zaj spektrális teljesítmény-sűrűségfüggvénye:

$$P_{\text{zaj,WIIR}}(\omega) = \sigma_e^2 \cdot \sum_{i=-1}^N \left| H_i(e^{j\omega}) \right|^2. \quad (4.20)$$



4.6. ábra. Zaj becsatolási pontok WIIR szűrőnél

A szűrő erőforrásigénye a 4.4. táblázatban található, implementációja pedig az F.6. algoritmus.

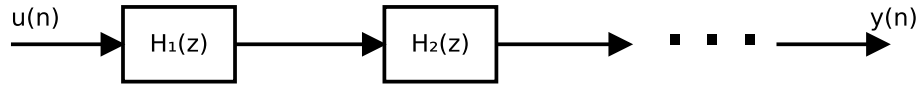
WIIR	
Szükséges memória:	$N + 1$
Akkumulátorok száma:	1
Töltések száma:	$5N + 4$
Tárolások száma:	$N + 2$
MAC utasítások:	$2N + 1$
Szorzások:	$N + 2$
Összeadások:	$2N + 1$
Akkumulátorból kivétel:	2
Akkumulátor törlés:	2
Számításigény:	$11N + 14$

4.4. táblázat. N -ed fokú WIIR szűrő erőforrásigénye

4.6. Zaj soros és párhuzamos struktúrákban

Egy tetszőleges átviteli függvény sorosá/párhuzamosá alakításakor az egyes másodfokú tagok egy-egy (konjugált komplex) póluspárt és két zérust fognak tartalmazni, ezáltal azt

várjuk, hogy ez a realizáció kevésbé lesz érzékeny az együtthatók kvantálására, mint a direkt realizáció.



4.7. ábra. Soros struktúra

A soros struktúra vizsgálatához nézzük meg egy IIR szűrő soros tagokra bontását. Az átviteli függvény a következők szerint alakul:

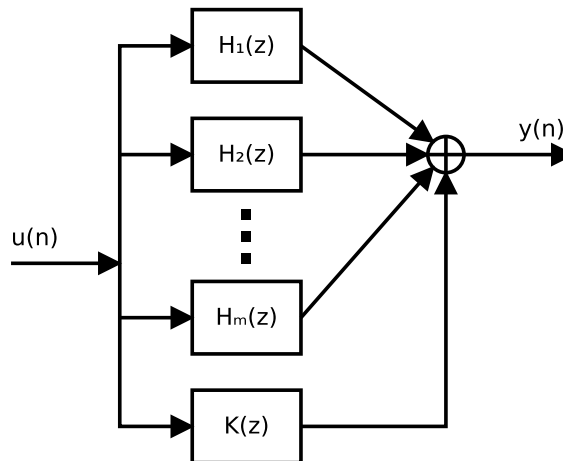
$$H(z) = \frac{\sum_{k=0}^N b_k \cdot z^{-k}}{1 + \sum_{k=1}^N a_k \cdot z^{-k}} = \prod_{i=1}^{\lceil N/2 \rceil} \frac{b_{0,i} + b_{1,i} \cdot z^{-1} + b_{2,i} \cdot z^{-2}}{1 + a_{1,i} \cdot z^{-1} + a_{2,i} \cdot z^{-2}} = \prod_{i=1}^{\lceil N/2 \rceil} H_i(z) \quad (4.21)$$

Vegyük észre, hogy a k -adik tag kimenete a $k + 1$ -edik bemenete lesz. Ez azt is jelenti, hogy ha az egyik tag erősít egy bizonyos frekvenciatartományban, úgy nem csak a hasznos jelet, hanem a kvantálási zajt is növelni fogja, emiatt nem elhanyagolható a tagok sorrendje. Ezen kívül minden egyes másodfokú tag visszacsatolásában megjelenik egy kerekítés, így ha egy N -ed fokú szűrőt IIR direkt form tagokkal valósítunk meg, a kvantálási pontok száma $M \cdot \lceil N/2 \rceil$ lesz, ahol M az egy tagban történő kerekítések száma.

A fenti megfontolások szerint a kvantálási zaj spektrális teljesítmény-sűrűségfüggvénye:

$$P_{\text{zaj,soros}}(\omega) = \sigma_e^2 \cdot \sum_{i=1}^{\lceil N/2 \rceil} \sum_{k=1}^M \left| H_{i,k}(e^{j\omega}) \cdot \prod_{l=i+1}^{\lceil N/2 \rceil} H_l(e^{j\omega}) \right|^2, \quad (4.22)$$

ahol $H_{i,k}(e^{j\omega})$ az i -edik tag k -adik pontján becsatolt zaj átvitele a tag kimenetére, $H_l(e^{j\omega})$ pedig az l -edik tag átvitele.



4.8. ábra. Párhuzamos struktúra

A párhuzamos struktúrában a soros realizációval ellentétben az i -edik tag zaja közvetlenül a kimenetre jut. Ha a tervezett szűrő fokszáma N , akkor a párhuzamos realizáció $\lceil N/2 \rceil$ db másodfokú tagot tartalmaz, valamint egy $K(z)$ maradék FIR tagot. Ha a kívánt átvitel számlálója és nevezője azonos fokszámú, akkor $K(z)$ egy konstanssal szorzást

jelent.

Ezzel a megfontolással a zaj spektrális sűrűségfüggvénye:

$$P_{\text{zaj,párhuzamos}}(\omega) = \sigma_e^2 \cdot \sum_{i=1}^{\lceil N/2 \rceil} \sum_{k=1}^{M'} \left| H_{i,k}(e^{j\omega}) \right|^2 + \sigma_e^2, \quad (4.23)$$

ahol $H_{i,k}(e^{j\omega})$ az i -edik tag k -adik pontján becsatolt zaj átvitele a tag kimenetére, M' pedig az egy tagban fellépő kerekítések száma.

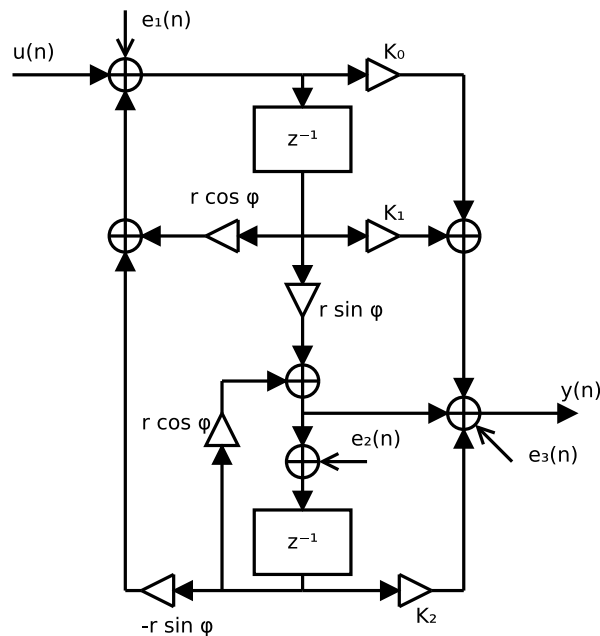
4.7. Zaj speciális másodfokú tagokban

Az előzőekben általános leírást adtam a soros és párhuzamos realizációk zajára. A képletekben szereplő zajátvitelek függenek a másodfokú tagok megvalósításától, ezért ebben az alfejezetben megvizsgálom a speciális másodfokú tagokat.

4.7.1. Gold & Rader tag

Optimális implementáció esetén 3 helyen szükséges kerekíteni. A függelékben megtalálható a pontos implementáció (F.4. algoritmus), itt csak az eredményeket közlöm. Az erőforrásigényt 4.5. táblázat tartalmazza.

Az implementációnál egy trükköt alkalmaztam. Az $e_2(n)$ -nel jelölt zaj becsatolódási ponton ugyan kimásolom az akkumulátorból (és ezzel együtt kerekítem is) a jelet, de a kerekítetlen jel továbbra is megmarad az akkumulátorban. Ezzel némileg gyorsítani lehet az algoritmust ahhoz képest, mint ha a két MAC után egyből kerekítenénk.



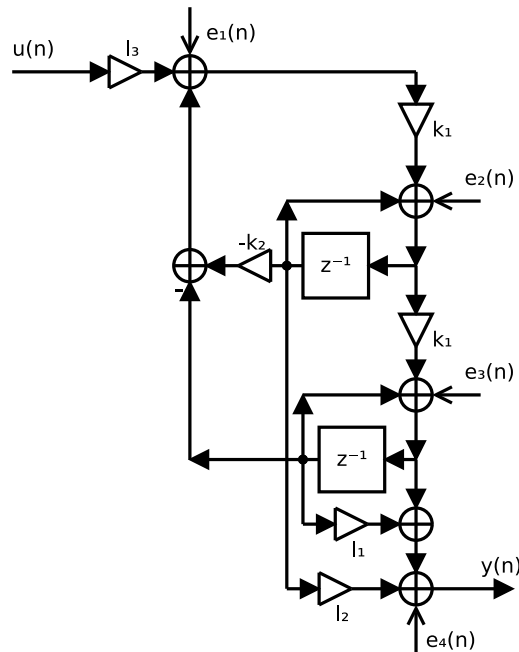
4.9. ábra. Zaj becsatolási pontok Gold & Rader másodfokú tagban

Gold & Rader	
Töltések száma:	9
Tárolások száma:	3
MAC utasítások:	7
Összeadások:	1
Akkumulátorból kivétel:	3
Akkumulátor törlés:	2
Számítási igény:	25

4.5. táblázat. Gold & Rader tag erőforrásigénye

4.7.2. Kingsbury tag

Optimális implementáció (ld. F.7. algoritmus) esetén 4 ponton szükséges kerekíteni. A struktúra sajátosságai miatt csak két helyen alkalmazható a MAC művelet: az egyik az l_3 -mal és $-k_2$ -vel szorzáskor, a másik pedig a kimeneti szorzásoknál (l_1 , l_2 -vel szorzás).



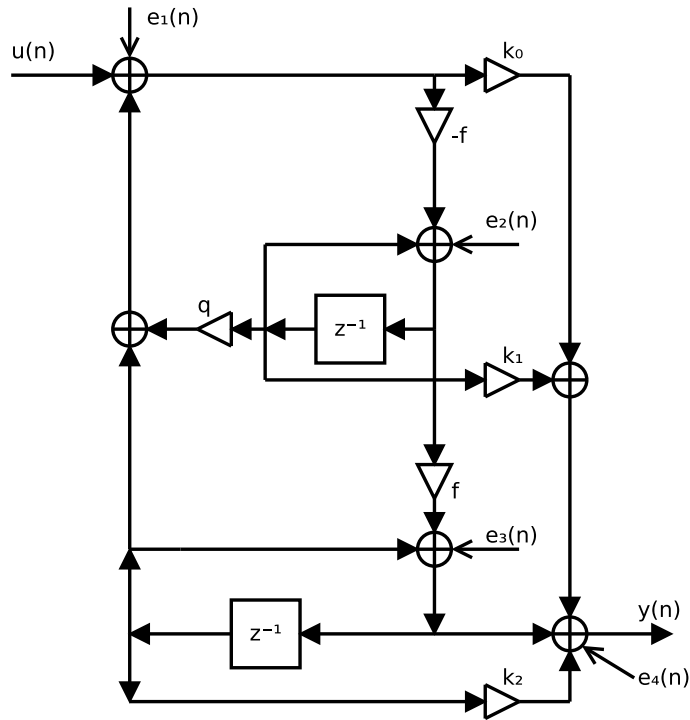
4.10. ábra. Zaj becsatolási pontok Kingsbury másodfokú tagban

Kingsbury	
Töltések száma:	8
Tárolások száma:	3
MAC utasítások:	4
Szorzások:	2
Összeadások:	4
Akkumulátorból kivétel:	2
Akkumulátor törlés:	2
Számítási igény:	25

4.6. táblázat. Kingsbury tag erőforrásigénye

4.7.3. Chamberlin tag

Optimális implementáció (ld. F.8. algoritmus) esetén 4 helyen szükséges kerekíteni. MAC műveletek csak a kimenetnél alkalmazhatók, mert a többi helyen nincs két összeadandó szorzat.



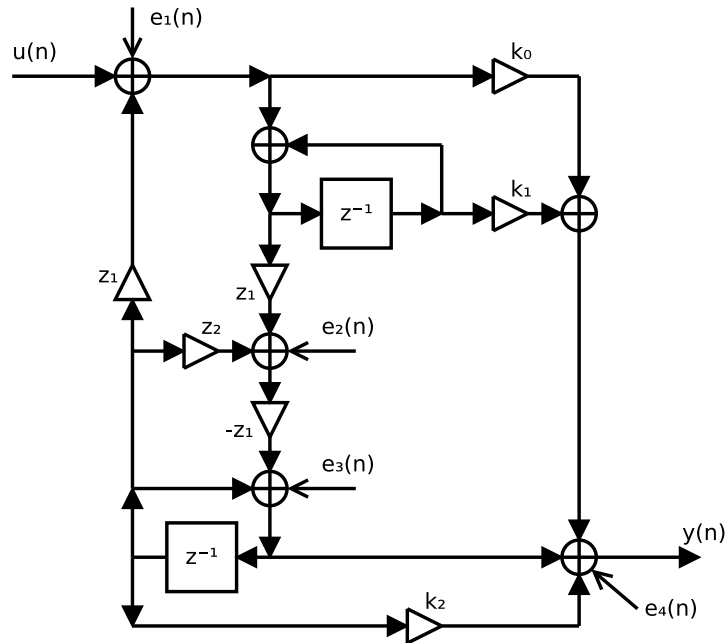
4.11. ábra. Zaj becsatolási pontok Chamberlin másodfokú tagban

Chamberlin	
Töltések száma:	9
Tárolások száma:	3
MAC utasítások:	3
Szorzások:	3
Összeadások:	4
Akkumulátorból kivétel:	1
Akkumulátor törlés:	1
Számításigény:	24

4.7. táblázat. Chamberlin tag erőforrásigénye

4.7.4. Zölzer tag

Optimális implementáció (ld. F.9. algoritmus) esetén 4 helyen szükséges kerekíteni. MAC műveleteket itt a kimenet előállításán kívül még egy helyen, a z_1 és z_2 szorzótényezőknél lehet alkalmazni.



4.12. ábra. Zaj becsatolási pontok Zölzer másodfokú tagban

Zölzer	
Töltések száma:	9
Tárolások száma:	3
MAC utasítások:	5
Szorзások:	2
Összeadások:	4
Akkumulátorból kivétel:	2
Akkumulátor törlés:	2
Számításigény:	27

4.8. táblázat. Zölzer tag erőforrásigénye

5. fejezet

Másodfokú tagok összehasonlítása

Az előző fejezetben számszerűsítettem a számításigényeket, adtam egy módszert az együttható-kerekítés vizsgálatára és megmutattam a kvantálási pontokat. Ebben a fejezetben konkrét pólus-zérus elrendezéseken hajtom végre a vizsgálati módszereket.

Az 5.1. táblázatból látható, hogy a másodfokú warpolt IIR szűrőtag lényegesen nagyobb számításigényű, mint a többi vizsgált struktúra. A transzponált struktúrákkal kapcsolatban elmondható, hogy nem célszerű az implementálásuk, már csak azért sem, mert együtthatóik ugyanazok, mint a sima direkt formának, de több helyen történik bennük kerekítés és nagyobb számításigényűek is. Emiatt a transzponált tagokat a vizsgálatoknál kihagyom.

Struktúra	DF1	DF2	DF1T	DF2T	WIIR
Utastíásszám	19	19	26	22	36
Struktúra	Gold & Rader	Kingsbury	Chamberlin	Zölzer	
Utastíásszám	25	25	24	27	

5.1. táblázat. Másodfokú tagok számításigényének összehasonlítása

5.1. Másodfokú tagok érzékenysége az együttható-kerekítésre

Az együttható-kerekítés hatását a 4.1.2. pontban tárgyaltak szerint vizsgáltam. A szűrőstruktúráknál csak a különböző paraméterrel rendelkezőket vettem figyelembe, így a direkt form tagok közül csak DF2-vel dolgoztam. Minden szűrőstruktúrával terveztem egy-egy másodfokú tagot 64 bites lebegőponton, majd a 16 bites fixpontos számábrázolás küszöbszintjének felét adtam hozzá/vontam le az együtthatókból. Az adott pólus-zérus elrendezés mellett azt a struktúrát választottam ki, amelyik átvitelének worst-case megváltozásának maximuma a 20 Hz–20kHz tartományban a legkisebb volt ($f_s = 48$ kHz).

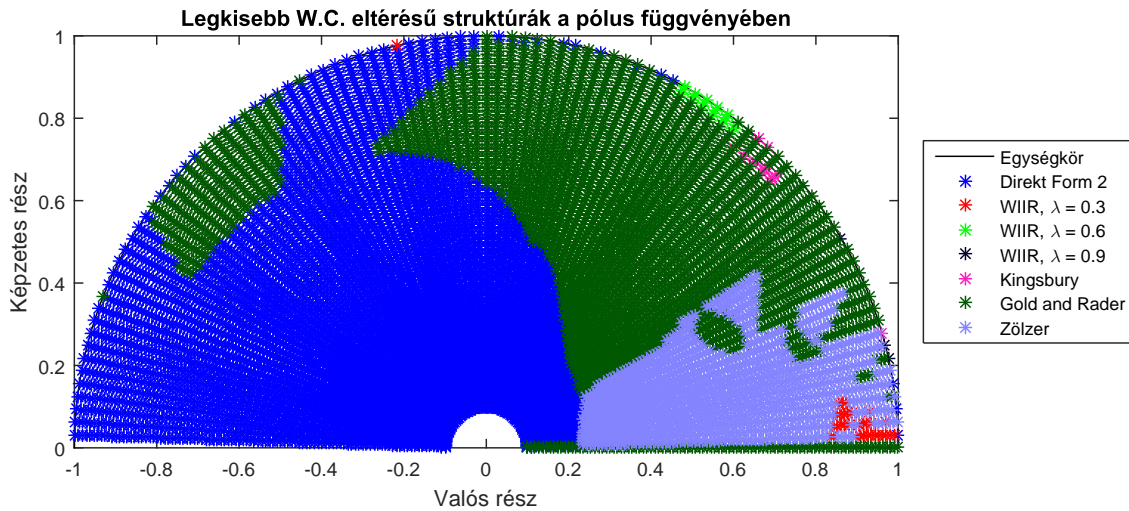
Az eredményeket úgy is kiértékeltem, hogy maximum 1 dB megváltozást engedtem meg az átviteleknek. Az 1 dB választásának oka, hogy nagyjából ezt lehet hallás útján érzékelni. Amelyik pólus-zérus elrendezésnél nem volt tag, ami teljesítette volna a feltételt, azt nem jelöltem az ábrákba.

A vizsgálatokat 10000 pólus-zérus elrendezésre végeztem el ábránként. Ez egy igen szá-

mításigényes feladat: egy adott átvitel esetén 10 db struktúrát kell tervezni, mindegyiknél el kell végezni a 2^K db átvitel kiértékelését, majd meg kell keresni közülük a legjobbat. A MATLAB Parallel Computing Toolbox-szának felhasználásával így egy ábra előállítására 4 órába telik egy mai, csúcskategóriás számítógépen.

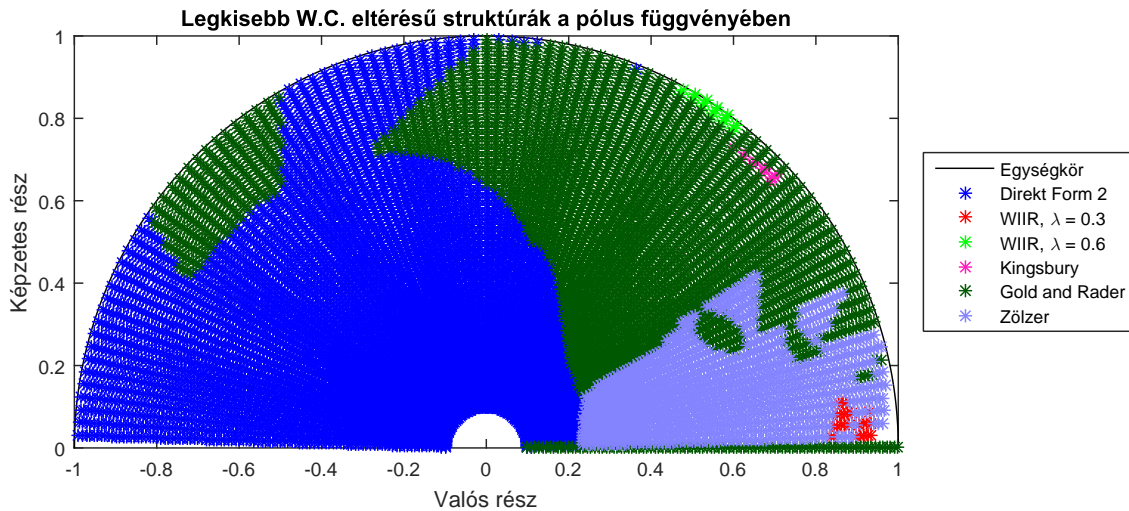
5.1.1. Csak pólusos tagok

Elsőként all-pole másodfokú tagokra vizsgáltam, azonban ezeknek is vannak zérusaik: a komplex számsík origójában, kettes multiplicitással. Az eredmények az 5.1. ábrán láthatók.



5.1. ábra. Együttható-kerekítésre legkevésbé érzékeny másodfokú tag, $z_{zérus} = 0$

Látható, hogy all-pole szűrőknél a direkt form tagok mellett a Gold & Rader tagok szerepeltek a legjobban. Az ábrán látható, hogy a Gold & Rader tagok inkább a mintavételi frekvencia negyede alatt robusztusabbak. Ennek oka az zérusok megvalósításában keresendő: a struktúrában nem függetlenek az előrcsatoló ág együtthatói a pólustól.

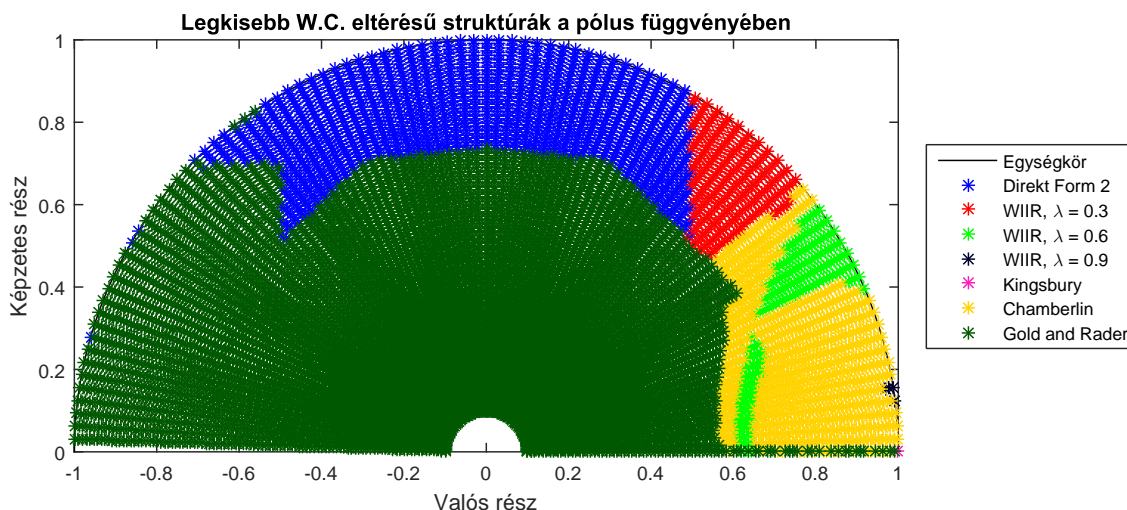


5.2. ábra. Együttható-kerekítésre legkevésbé érzékeny másodfokú tag, $z_{zérus} = 0$, maximum 1 dB eltérés

Az 5.2. ábrán látható az az eset, ahol maximum 1 dB megváltozást engedtem a kerekítés hatásaként. Annak ellenére, hogy 16 bites számábrázolásnál kerestem a worst-case hibát, kevés esetben lett a hiba 1 dB-nél nagyobb. Ennél nagyobb bitszámú számábrázolásnál tehát nem fog gondot okozni az együtthetők kerekítése.

5.1.2. Mindentáteresztő tagok

Azt a speciális esetet is megvizsgáltam, amikor a zérusok egybeesnek a pólusokkal. Az eredményeket az 5.3. ábra tartalmazza.



5.3. ábra. Együtthető-kerekítésre legkevésbé érzékeny másodfokú mindentáteresztő tag

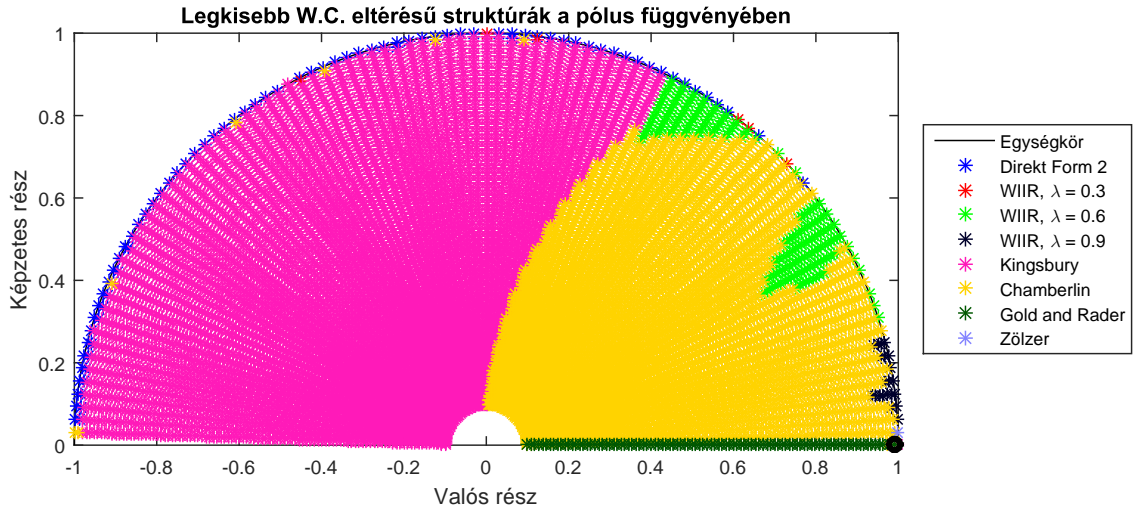
Látható, hogy a pólusok és a zérusok egybeesésekor a legtöbb helyen a Gold & Rader tagok a legrobosztusabbak. Említésre méltó még, hogy a 16 biten legkevésbé érzékeny tagok megváltozása minden esetben 1 dB alatti volt.

5.1.3. Valós zérusok

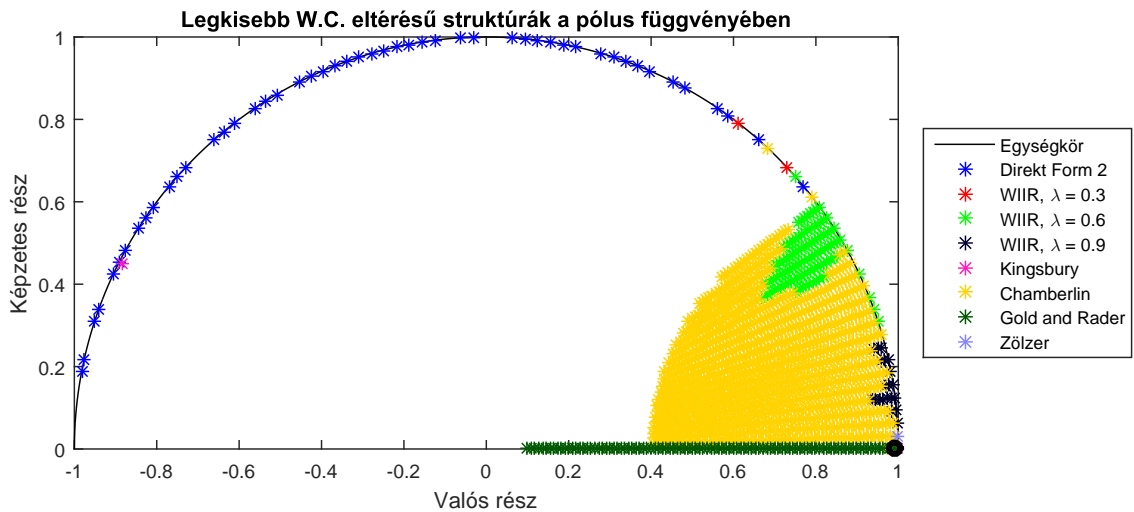
Valós zérusoknál azt az esetet vizsgáltam, ahol a másodfokú zérus 0,99-ben helyezkedik el. A választást elsősorban az indokolta, hogy $z = 1$ -be helyezve a zérust, már nagyon kicsi elmozdulások is drasztikus átvitel módosítással járnak.

Az eredményeket az 5.4–5.5. ábrák tartalmazzák. Látható, hogy egy kiélezett helyzetben 16 biten már jócskán lesznek olyan pólus-zérus elrendezések, ahol a worst-case hiba meghaladja az 1 dB-t. Érdekes lehet összevetni az ábrákat a korábbiakkal: közös vonásként észrevehetjük, hogy valós pólusoknál mindig a Gold & Rader tagok a legkevésbé érzékenyek az együtthetők kerekítésére.

Láthattuk, hogy az együtthetők kerekítése 16 biten sok esetben elhanyagolható mértékben változtatja meg a másodfokú tagok átvitelét. Ahol viszont a worst-case számítás eredménye szerint túl nagy lesz az átvitel megváltozása, ott ténylegesen el kell végezni a kerekítést, majd utána kiértékelni az átvitelt. Amennyiben a kerekítés utáni átvitel nem kielégítő, megpróbálkozhatunk az együtthetők más irányba történő kerekítésével.



5.4. ábra. Együttható-kerekítésre legkevésbé érzékeny másodfokú tag, $z_{zérus} = 0,99$ (fekete körrel jelölve)



5.5. ábra. Együttható-kerekítésre legkevésbé érzékeny másodfokú tag, $z_{zérus} = 0,99$ (fekete körrel jelölve), maximum 1 dB eltérés

5.2. Kvantálási zaj másodfokú tagokban

A továbbiakban a másodfokú tagok numerikus zaját vizsgálom, és azt keresem, hogy különböző pólus-zérus elrendezésekben melyik struktúra lesz a legkisebb zajú. Ehhez először egy vizsgálati módszert ismertetek, majd kiértékelem a zajteljesítményeket.

5.2.1. A struktúra kiválasztásának módszere

A kvantálási zaj minimumának megkeresésére adhatunk egy általánosan használható módszert, amely alapján a másodfokú tagokból felépített szűrőnk zaját optimalizálhatjuk. A gyakorlati példánál is ezt az algoritmust fogom használni.

Szűrőtervezés után a másodfokú tagok pólusai és zérusai kiadódnak. Az egyes tagoknál a legkisebb zajú realizációt az alábbi algoritmussal kereshetjük meg:

1. Szűrőtagok tervezése a kívánt átvitel alapján,
2. Az együttható-kerekítés hatására túlságosan megváltozó átvitelű struktúrák elvetése,
3. Gerjesztés rózsazajjal, skálázási tényező meghatározása,
4. Zajspektrumok kiszámítása,
5. Spektrumok összegzése zajteljesítményé,
6. Legkisebb zajteljesítményű struktúra kiválasztása.

A módszer a gyakorlatban jól használható, párhuzamos szűrőknél a legkisebb zajú megvalósítást találja meg, soros realizációnál pedig az adott tagsorrend mellett ad optimumot.

5.2.2. Optimális zajú másodfokú tag keresése

A vizsgálatokban az 5.2.1. bekezdésben tárgyalt módszer szerint keresem meg az adott pólusnál legjobb tagot és azt jelölöm be a diagramba. A pólus-zérus ábránál csak a pozitív képzetes félsíkot ábrázolom.

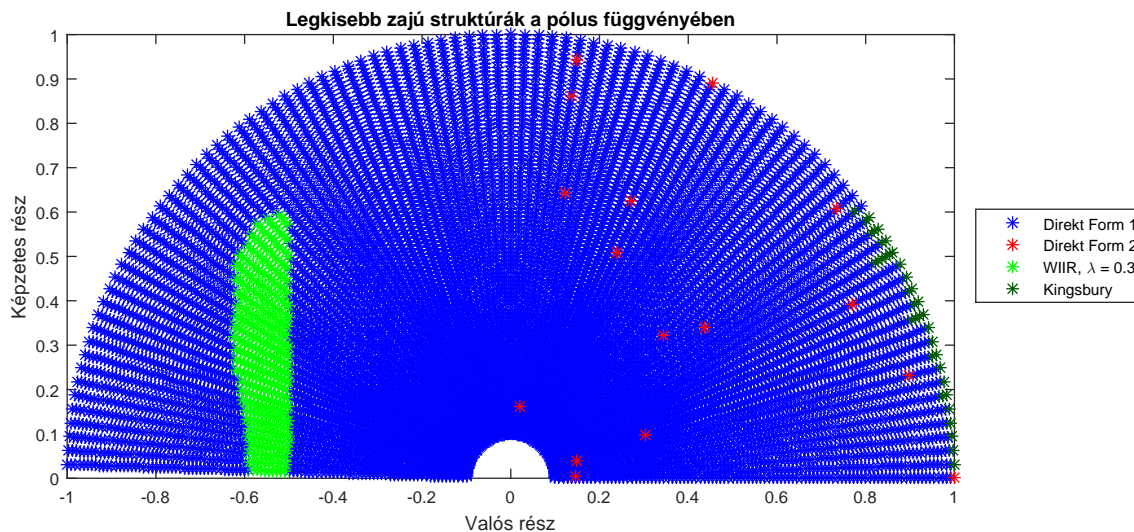
A módszer kifejezetten számításigényes, mivel ábránként 10000 pontban kell kiszámolni minden vizsgált szűrőtagnak a spektrumát, majd abból a zajteljesítményét. A MATLAB Parallel Computing Toolbox-szát is felhasználtam az implementálásnál, így egy mai csúcskategóriás számítógépen egy-egy ábra kiszámítása nem egészen 3 órába telik.

A fejezetben a következő másodfokú struktúrákkal végzem el a vizsgálatokat: DF1, DF2, Gold & Rader, Kingsbury, Chamberlin, Zölzer és WIIR. Ez utóbbinál négy különböző warpolási tényezővel is vizsgálódok: $\lambda = \{0,3; 0,5; 0,7; 0,9\}$. Az ábrákon csak azokat a struktúrákat jelölöm, amelyek legalább egy pólus-zérus elrendezésben minimális zajúak, így az adott ábrán szerepelnek.

5.2.3. Csak pólusos tagok

A kvantálási zajjal foglalkozó szakirodalom sokszor csak pólusos (all-pole) szűrőstruktúrákat vizsgál részletesen [3, 4, 5]. Valójában ezeknek is vannak zérusaik: a komplex számsík origójában, a foksámnak megfelelő multiplicitással. Ebből kiindulva először megvizsgáltam azt az esetet, hogy az origóba helyezek két darab zérust, majd kiértékelem a zajteljesítményeket különböző pólusokra. A legjobb teljesítménnyel rendelkező szűrőtagokat mutatja az 5.6. ábra.

Az 5.6. ábrán látható, hogy origóban elhelyezett zérusoknál van egy tartomány, ahol a WIIR másodfokú tagok zaja a legkisebb. További vizsgálatokból kiderült, hogy függőleges éles határt a fixpont elhelyezése okozza: az ábrán a vonaltól balra a direkt form a_1 együtthatója -1 -nél kisebb értéket vesz fel, így el kell tolni a fixpontot, ami miatt a zaj is nagyobb lesz. Az eredmények részletes elemzéséből kiderült továbbá az is, hogy ebben a tartományban a warpolt szűrő csak tized decibellel jobb a direkt formnál. Alacsony frekvencián, az egységkörhöz igen közel elhelyezett pólusnál van realitása a Kingsbury struktúra alkalmazásának, de ez eléggé speciális eset. Általános javaslatként tehát azt

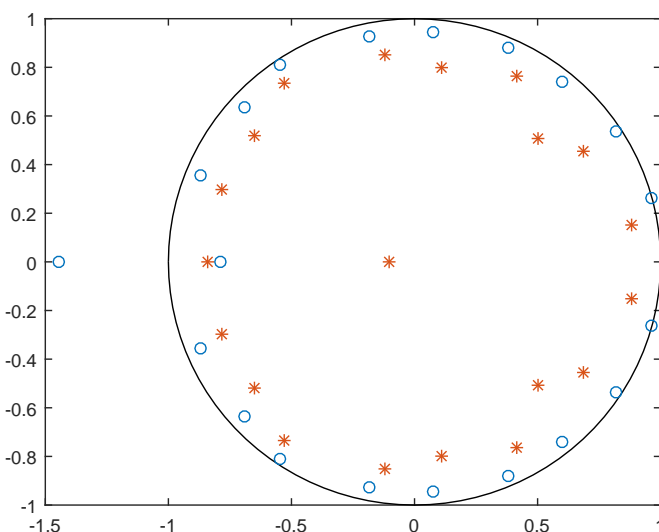


5.6. ábra. Legkisebb zajteljesítménnyel rendelkező másodfokú tag, $z_{z\acute{e}rus} = 0$

mondhatjuk, hogy amennyiben a zérusaink az origóban helyezkednek el, direkt form tagot célszerű alkalmazni.

5.2.4. Konjugált komplex zéruspárok

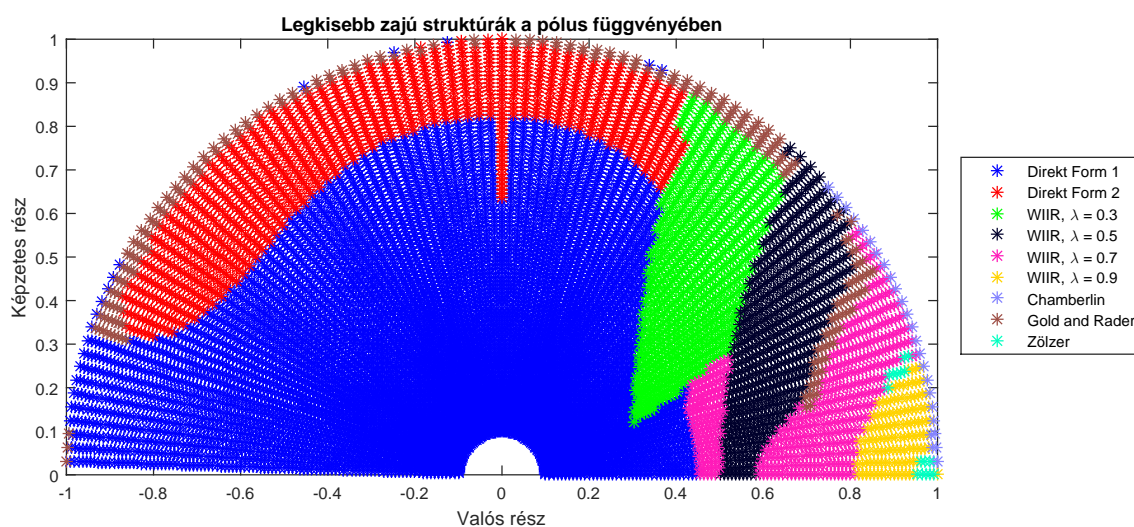
Belátható, hogy a csak pólusokkal rendelkező digitális szűrő igencsak speciális eset, a gyakorlatban ritkán fordul elő. Egy átvitel modellezésénél sokkal realisabb, hogy a szűrőnek legyenek zérusai is. Példaként egy szoba átvitelére terveztettem Prony módszerrel [24] egy 20-ad fokú szűrőt. A pólusok és zérusok darabszámát azonosra állítottam, majd a tervezett szűrő pólusait és zérusait kirajzoltattam (5.7. ábra).



5.7. ábra. Prony-val terveztetett, 20-ad fokú IIR szűrő pólusai és zérusai

Megfigyelhető, hogy nagyobb fokszámú szűrők pólusai és zérusai viszonylag közel he-

lyezkednek el egymáshoz. Másodfokú tagokra bontásnál érdemes az egymáshoz közel álló pólusokat és zérusokat összerendelni. Az ilyen módon felépített tagoknál többségében elmondható, hogy a zérus közelebb van az egységkörhöz, mint a pólus. Ezt modellezendő megvizsgáltam olyan eseteket, ahol a zérus ugyanakkora frekvenciájú, mint a pólus, de az egységkörtől helyezkedik el. Az eredményeket az 5.8. ábra mutatja. Jól látszik, hogy nagyobb frekvenciáknál, valamint rövid pólusoknál a direkt form tagok előnyösek: kb. $\frac{f_s}{6}$ és $0,45f_s$ között 0,75-ös pólushossz alatt legkisebb zajú a DF1, ennél hosszabb pólusoknál pedig a DF2 tag. Amennyiben a pólus az egységkörhöz nagyon közel helyezkedik el, a Gold & Rader tag lesz az optimális. Ennél nagyobb frekvenciákon a DF1 tagot célszerű implementálni.

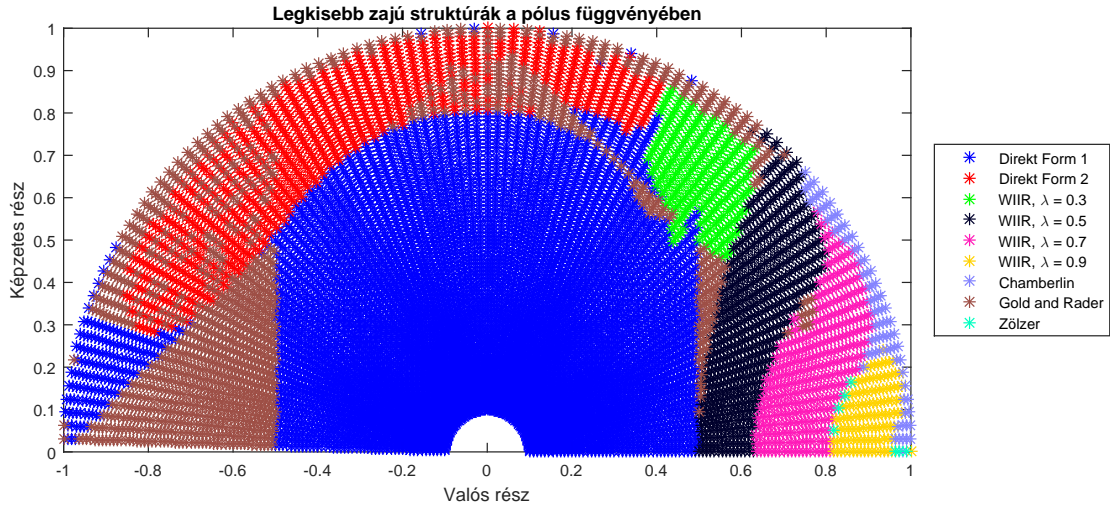


5.8. ábra. Legkisebb zajú tag, ha a zérus az egységkörtől helyezkedik el.

Alacsonyabb frekvenciáknál a WIIR tagok dominálnak, de a részletesebb vizsgálatból kiderült, hogy az ábrán látható sávokon kívül gyakran több decibelyivel nagyobb a zajuk. Ennek oka az, hogy többségében csak az ábrán látható optimális sávokban esnek a szűrőegyütthatók a $(-1; +1]$ tartományba. A WIIR tagokkal ellentétben a Chamberlin tag zaja csak kis mértékben változik, ráadásul a $z = 1$ -hez közeledve sem emelkedik meg jelentősen. Ezt figyelembe véve a $z = 1$ -hez közel mindig el kell végezni az 5.2.1. bekezdésben leírt módszer szerinti keresést. Amennyiben ez nem lehetséges, akkor érdemes lehet inkább Chamberlin struktúrában megvalósítani az adott tagot.

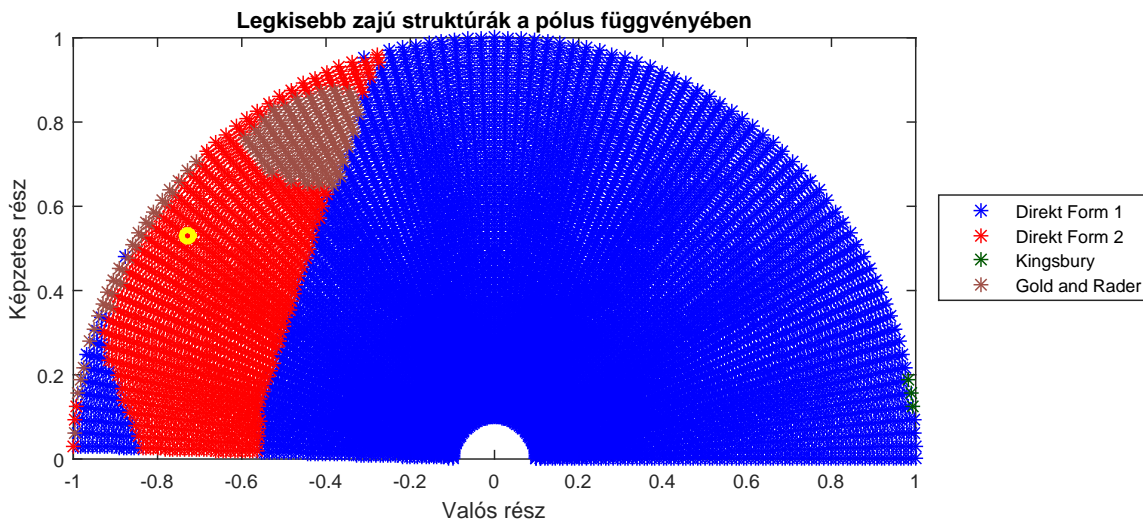
Warpolt szűrőknél előfordul, hogy a szűrő pólusai és zérusai olyan közel kerülnek egymáshoz, hogy szinte egybeesnek. Ezt szem előtt tartva megvizsgáltam minden olyan esetet is, ahol a pólus és a zérus ugyanott helyezkedik el. Az eredményeket az 5.9. ábra mutatja. Látható, hogy az eredmények hasonlóak ahhoz az esethez, mikor az egységkörtől tettem a zérusokat. Jelentős különbséget a mintavételi frekvencia feléhez közel találunk, ott ugyanis jól láthatóan körvonalazódik egy háromszög, amin belül a Gold & Rader tagok a legkisebb zajúak. Részletes vizsgálatokból kiderül, hogy a $-0,5$ -nél jelentkező függőleges vonalat a direkt formában megjelenő túl nagy együttható okozza (odébb kell tolni a fixpontot), a háromszög átfogójánál pedig a Gold & Rader tagban történik ugyanez. Az

ábrán látható továbbá, hogy a mintavételi frekvencia negyede és harmada környékén foltookban a Gold & Rader tagok lesznek a legkisebb zajúak. Valójában itt a DF2 tagok zaja is ugyanakkora, de a lebegőpontos számábrázolás miatt a zajteljesítmény-számító algoritmus kicsi különbséget talál. Ebben az esetben azt javaslom, hogy DF2 struktúrában legyen megvalósítva az adott tag, mivel az alacsonyabb számításigényű, mint a Gold & Rader.

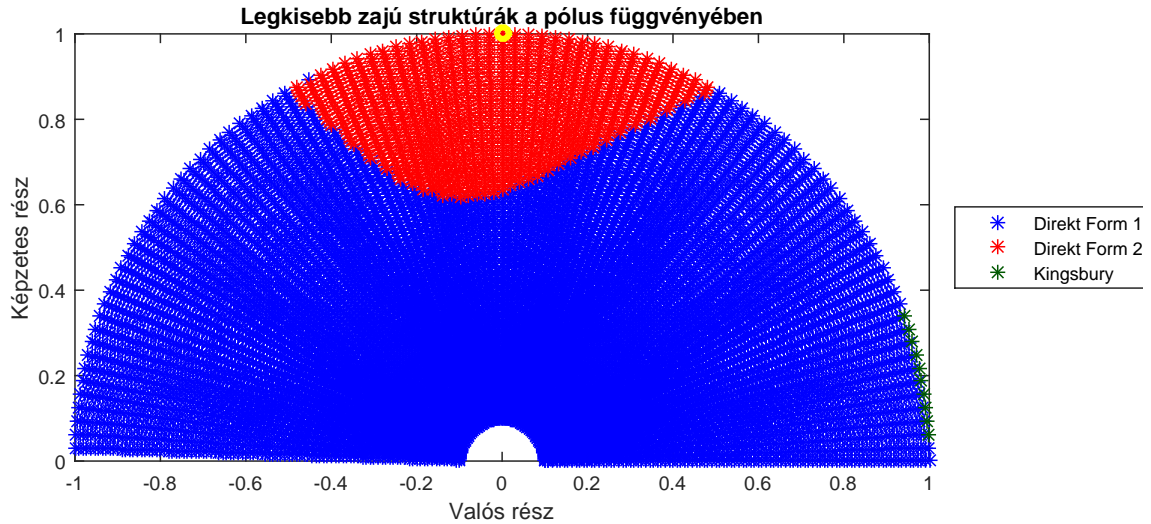


5.9. ábra. Legkisebb zajú tag, ha a pólus és a zérus egybeesik.

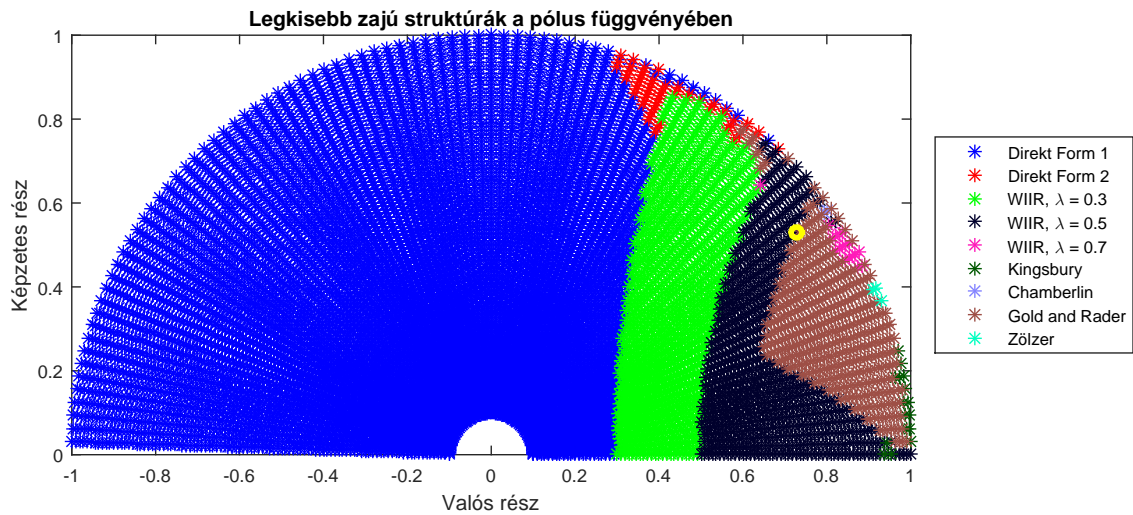
Természetesen előfordulhatnak olyan esetek is, amikor a pólus távol esik a zérustól. Erre mutatnak példát az 5.10–5.12. ábrák. A diagramok elemzéséből arra a következtetésre juthatunk, hogy a mintavételi frekvencia negyede körül a DF2 tagok a legkisebb zajúak (5.11. ábra). Hasonló a helyzet magasabb frekvenciákon is (5.10. ábra). Konklúzióként azt fogalmazhatjuk meg, hogy ha a zérus frekvenciája nagyobb, mint a mintavételi frekvencia hatoda, akkor DF2 struktúrát abban az esetben célszerű megvalósítani, ha a pólus és a zérus távolsága kisebb, mint 0,4.



5.10. ábra. Legkisebb zajteljesítménnyel rendelkező másodfokú tag, $z_{zérus} = -0,72 + 0,53j$ (sárga körrel jelölve)



5.11. ábra. Legkisebb zajteljesítménnyel rendelkező másodfokú tag, $z_{z\acute{e}rus} = j$ (sárga körrel jelölve)



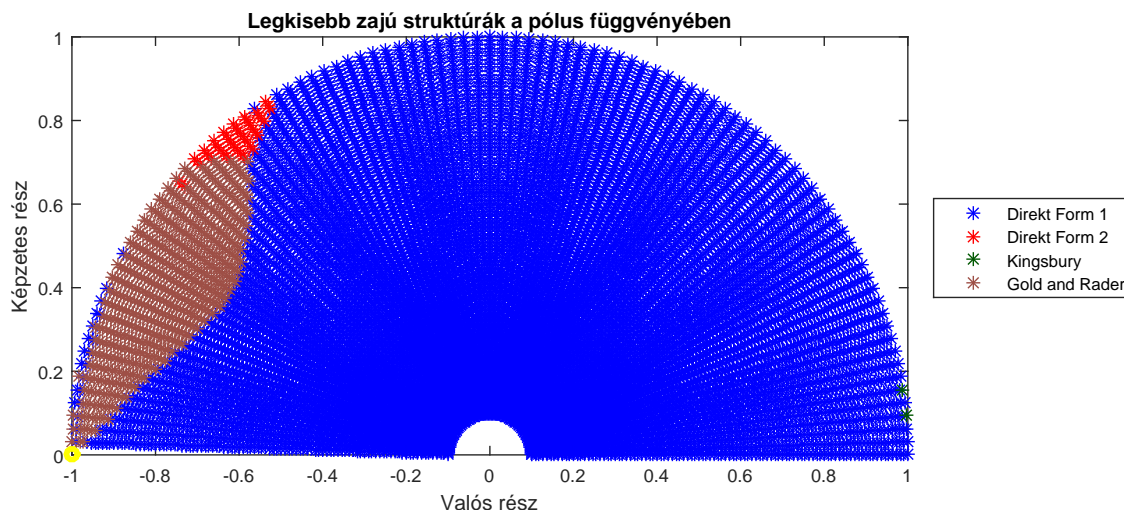
5.12. ábra. Legkisebb zajteljesítménnyel rendelkező másodfokú tag, $z_{z\acute{e}rus} = 0,72 + 0,53j$ (sárga körrel jelölve)

Alacsony frekvenciáknál árnyaltabb a kép. Az 5.12. ábrán látható, hogy a nagyobb warpolási tényezővel rendelkező IIR tagok már nem lesznek optimálisak akkor, ha a pólus és a zérus távolabb van egymástól, így itt a Gold & Rader tagot célszerű használni. Megjegyzendő viszont, hogy valós pólusoknál nem használható a Gold & Rader tag, így a valós tengelyhez közel a $\lambda = 0,5$ -ös WIIR tag lesz optimális.

Látható, hogy nehéz általános javaslatokat adni. Összességében viszont megállapíthatunk pár dolgot: ha egy tagon belül a pólusok és zérusok egymástól távol helyezkednek el, valamint ha a pólus hossza kisebb, mint 0,5, akkor DF1 struktúráként célszerű implementálni. Egyébként, ha a pólusok és zérusok egymáshoz közeli (távolságuk kisebb 0,4-nél), de frekvenciájuk az $[\frac{f_s}{6}; \frac{f_s}{3}]$ tartományba esik, akkor DF2 tagként célszerű implementálni. A többi esetben az 5.9. ábra szerinti implementációt érdemes használni.

5.2.5. Valós zérus

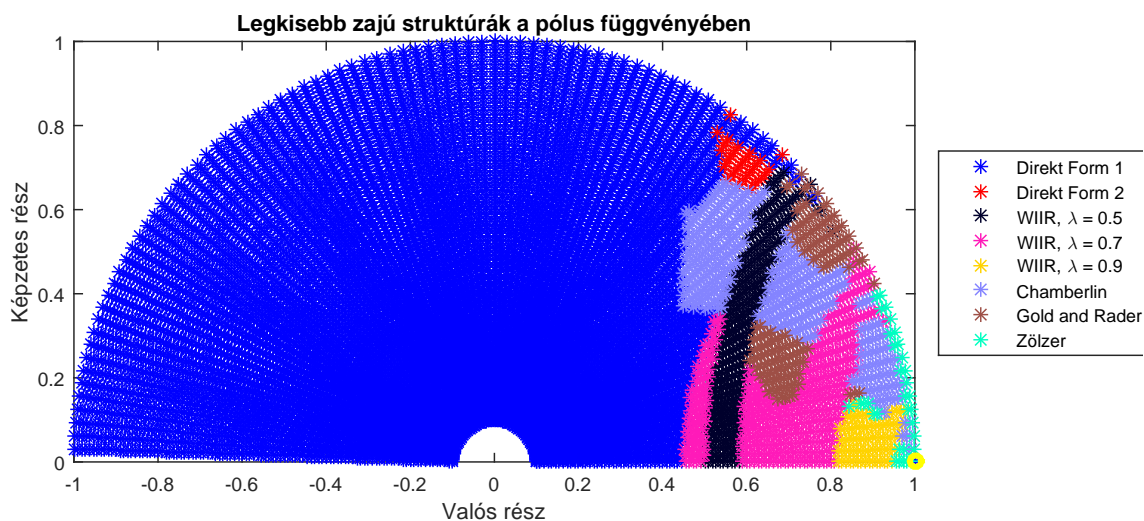
Az eddig bemutatott eredmények soros másodfokú tagok implementálásánál használhatóak. Párhuzamos tagoknak ugyanis valós zérusaik vannak, emiatt ilyen esetekre is el kell végezni a vizsgálatokat. Ezeket mutatják be az 5.13–5.16. ábrák.



5.13. ábra. Legkisebb zajteljesítménnyel rendelkező másodfokú tag, $z_{z\acute{e}rus} = -1$ (sárga körrel jelölve)

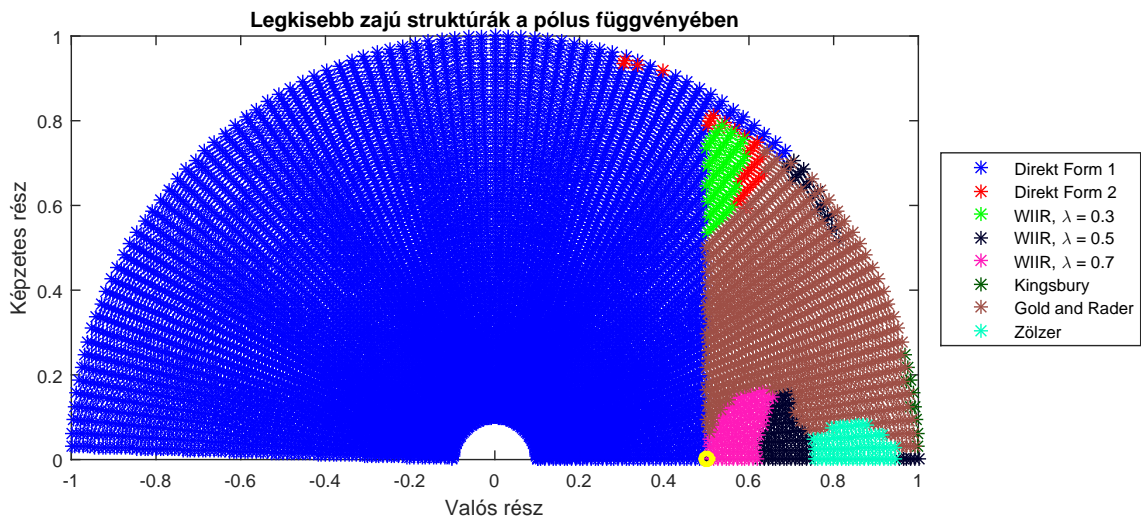
Az 5.13. ábránál megfigyelhetjük, hogy ha a pólus frekvenciája az $[\frac{f_s}{6}; \frac{f_s}{2})$ tartományba esik és hossza meghaladja a 0,7-et, akkor célszerű Gold & Rader struktúrában implementálni.

Az 5.14–5.16. ábrákon látható, hogy alacsony frekvenciáknál bonyolultabb a helyzet. Valós pólusoknál nem használható a Gold & Rader tag, de 0,6-es pólushossz fölött, a mintavételi frekvencia negyede alatt zajteljesítménye kisebb, vagy csak elhanyagolható mértékben nagyobb a többi alternatívánál.

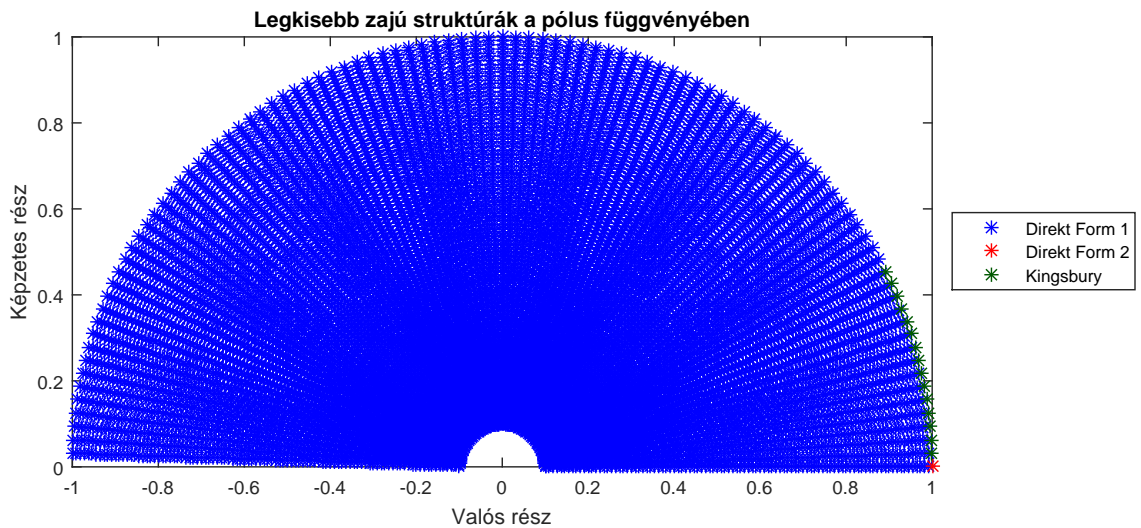


5.14. ábra. Legkisebb zajteljesítménnyel rendelkező másodfokú tag, $z_{z\acute{e}rus} = 1$ (sárga körrel jelölve)

Valós zérusoknál tehát az alábbi megállapításokat tehetjük: ha egy tagon belül a pólus és a zérus egymástól távol ($> 0,5$) helyezkednek el, akkor DF1 tagként érdemes megvalósítani. Amennyiben a pólushossz meghaladja a $0,7$ -et, a pólus frekvenciája pedig a $(\frac{f_s}{3}; \frac{f_s}{2})$ tartományba esik, célszerű a másodfokú tagot Gold & Rader struktúrában implementálni. Az $\frac{f_s}{6}$ alatti frekvenciájú, $0,5$ -nél hosszabb pólusokról viszont nehéz általános konklúziót megfogalmazni, így ott minden esetben ki kell értékelni a másodfokú tagok zajteljesítményét.



5.15. ábra. Legkisebb zajteljesítménnyel rendelkező másodfokú tag, $z_{z\u00e9rus} = 0,5$ (s\u00e1rga k\u00f6rrel jel\u00f6lve)



5.16. \u00e1bra. Legkisebb zajteljes\u00edtm\u00e9nyel rendelke\u00f6 m\u00e1sodfok\u00fa tag, $z_{z\u00e9rus} = 2$

6. fejezet

Nagyfokszámú szűrők megvalósítása

Az előző fejezetben tárgyalt optimalizálás eredményességét különböző soros és párhuzamos szűrőkön is kipróbáltam. A specifikáció minden esetben egy szoba impulzusválasza volt, mivel azt állítom, hogy tetszőleges átvitelre működik a módszer. Az összehasonlításához a másodfokú tagokat először DF1 tagokkal valósítottam meg, mert ez tulajdonképpen az implementáció leggyakrabban használt módja. Később az 5.2.1. bekezdésben közölt módszer szerint megválasztott tagokkal is megvalósítottam a szűrőket, majd összehasonlítottam az implementációk zajteljesítményét és számításigényét. Minden vizsgálatot elvégeztem 16, 24 és 32 bites fixpontos számábrázolás mellett is.

A zajt a fixponton és a dupla pontosságú lebegőponton megvalósított szűrő kimenetének a különbségéből számítom, gerjesztésként rózsazajt alkalmazva. A periodogramot Bartlett módszer szerint számolom 1024 mintán, 25-szörös átlagolással.

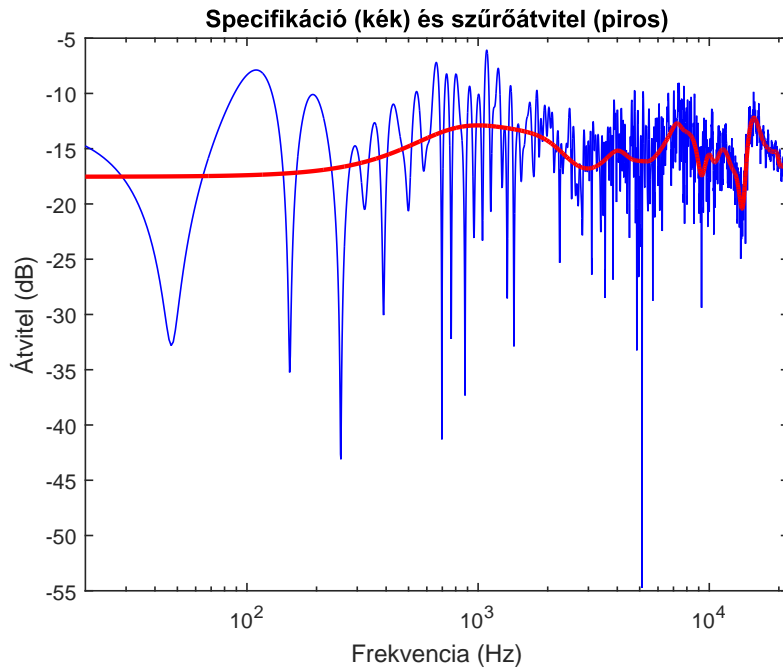
Kaszád realizációjú szűrőnél a tagokat a pólushosszuk szerinti növekvő sorrendben helyeztem el, hogy a sorban később következő tagok ne okozzanak túlcsoordulást [24].

Az egyes tagok előtt és után is skálázom a jelet a túlcsoordulás ellen. Ez az implementációban két utasítást (skálázási tényező betöltése és szorzás) és egy-egy kvantálási pontot is jelent a tagok előtt és után. Ennek hatása van mind a számításigényre, mind pedig a numerikus zaj teljesítményére is, így előfordulhat, hogy a soros/párhuzamos szűrő nagyobb zajú és számításigényű lesz.

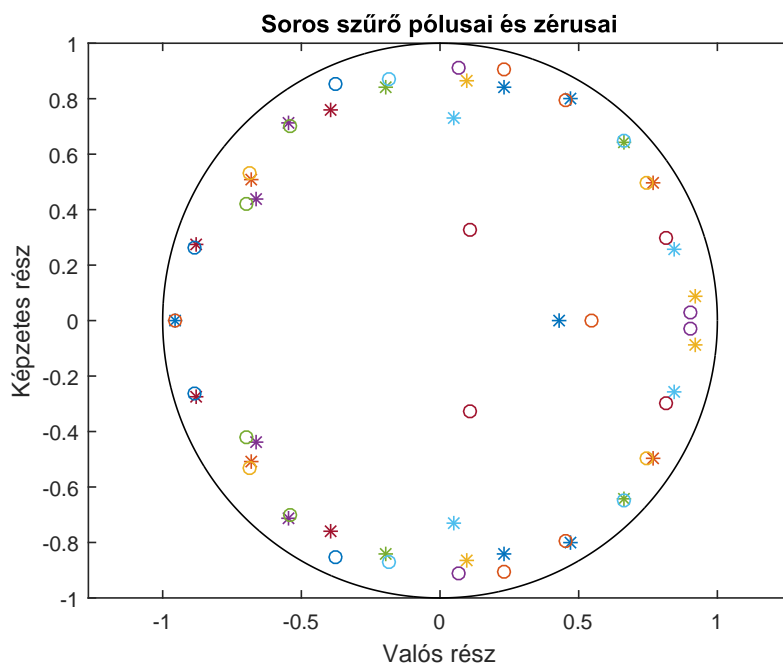
Soros struktúránál a kimeneti skálázó tag összevonható a soron következő bemeneti szorzójával, így tagonként megspórolható 2 db utasítás és 1 db kerekítés.

6.1. IIR

A példaként megvizsgált 30-ad fokú IIR szűrőt Prony algoritmussal terveztem, átvitelét a 6.1. ábra, pólusait és zérusait a 6.2. ábra mutatja. Soros megvalósításnál a másodfokú tagok ugyanezekből a pólusokból és zérusokból állnak össze. Párhuzamos realizációnál a pólusok maradnak ugyanazok, viszont a zérusok mind a valós tengelyen helyezkednek el (6.3. ábra).

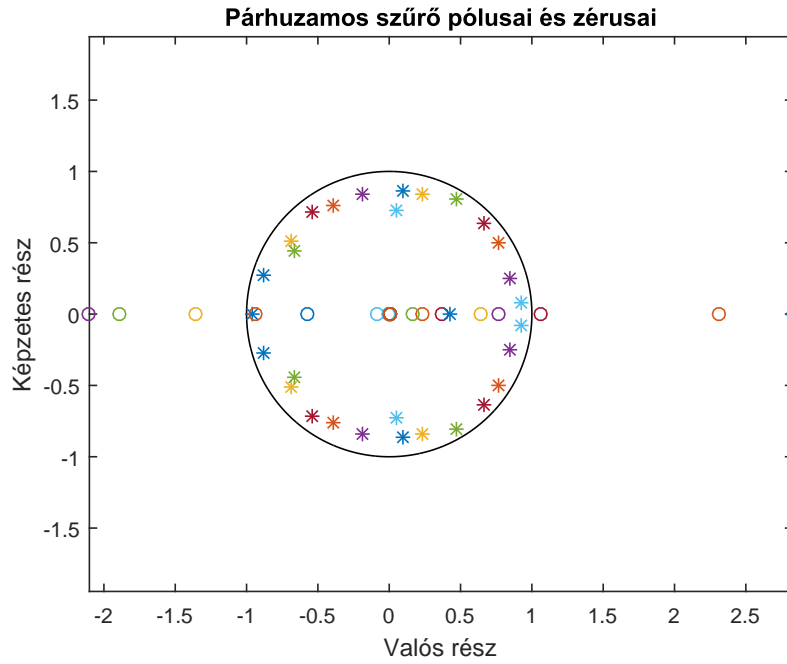


6.1. ábra. 30-ad fokú IIR szűrő átvitele



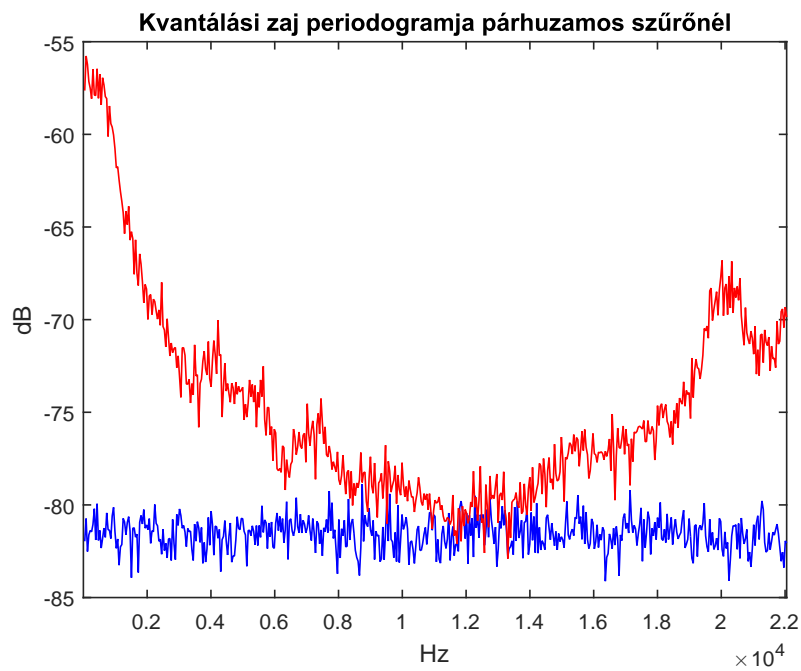
6.2. ábra. 30-ad fokú IIR szűrő pólusai és zérusai

Optimalizálás után a kaszkád szűrő 8 db DF2, 1 db DF1, 1 db Gold & Rader és 5 db WIIR másodfokú tagból áll. A párhuzamos szűrő 6 db Gold & Rader, 4 db DF2 és 5 db WIIR tagból áll. A kvantálási zaj teljesítménye a 6.1. táblázatban látható. Párhuzamos szűrőnél a javaslataim több, mint 10 dB zajcsökkenést eredményeztek. Soros megvalósításnál nem sikerült lényeges javulást elérni. A számításigényeket megvizsgálva látható, hogy amennyiben nem indokolja az együtthatók kerekítése, IIR szűrőt direkt form alakban

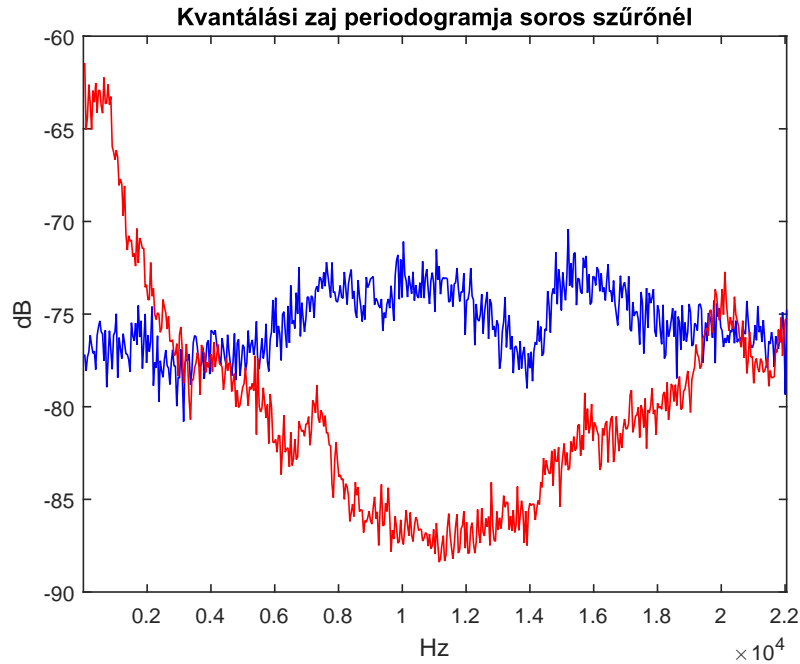


6.3. ábra. 30-ad fokú IIR szűrő párhuzamos tagjainak pólusai és zérusai

érdeemes megvalósítani.



6.4. ábra. 30-ad fokú IIR szűrő párhuzamos realizációjának zaja 16 biten
Piros: megvalósítás DF1 tagokkal, kék: optimalizált struktúra zaja



6.5. ábra. 30-ad fokú IIR szűrő soros realizációjának zaja 16 biten
Piros: megvalósítás DF1 tagokkal, kék: optimalizált struktúra zaja

Struktúra	DF2	DF1 soros	DF1 - párhuzamos	Opt. soros	Opt. - párhuzamos
Zajtjeljesítmény 16 biten	-54,5 dB	-44,9 dB	-39,3 dB	-45,3 dB	-51,7 dB
Zajtjeljesítmény 24 biten	-101,9 dB	-92,4 dB	-87,6 dB	-93 dB	-98,1 dB
Zajtjeljesítmény 32 biten	-149,2 dB	-141,3 dB	-135,9 dB	-141,7 dB	-148,1 dB
Utastításszám	187	317	345	408	466

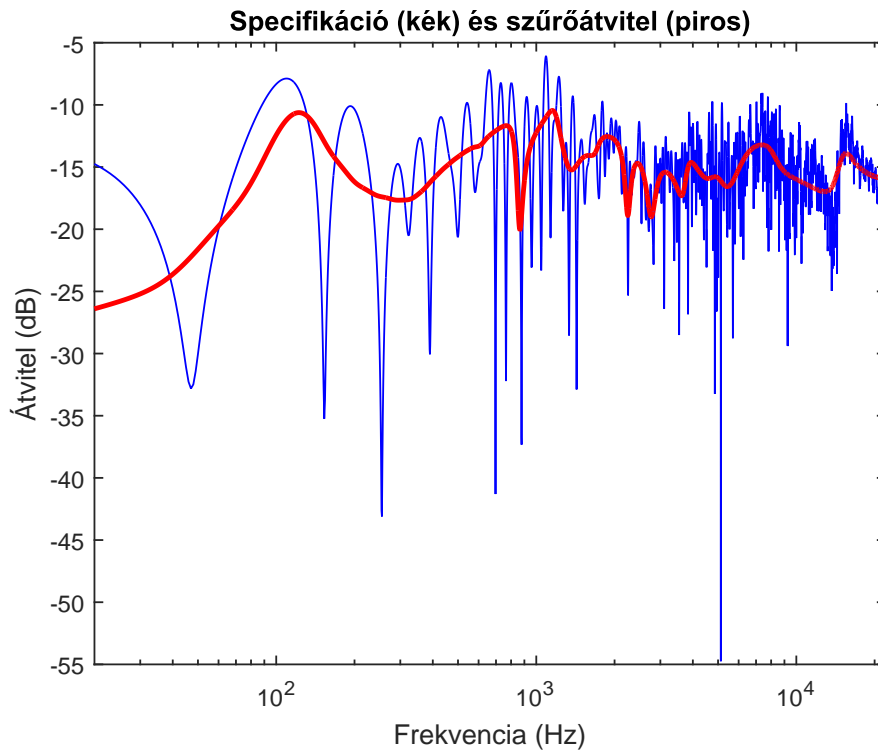
6.1. táblázat. 30-ad fokú IIR realizációinak zaja és számításigénye

6.2. WIIR, $\lambda = 0,75$

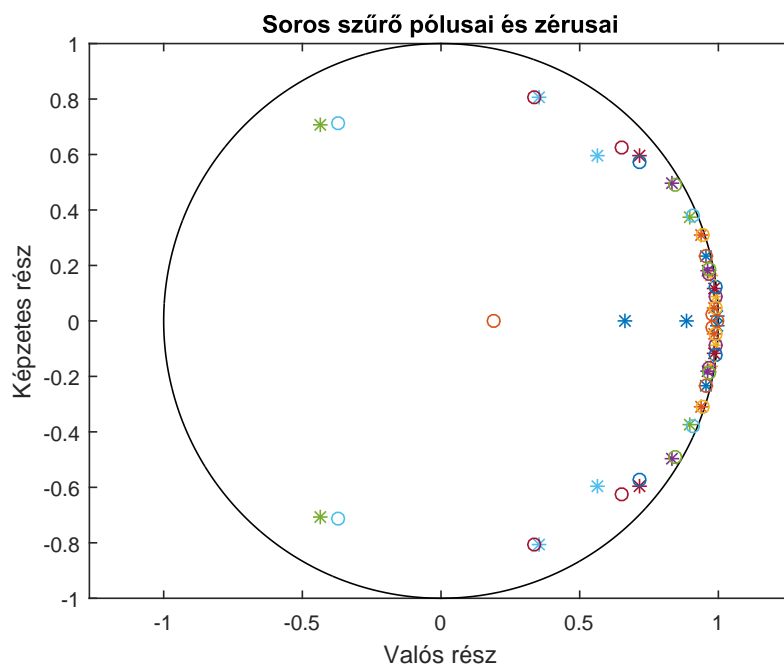
Warpolt szűrőknél fontos paraméter a warpolási tényező. A $\lambda = 0,75$ választással az emberi halláshoz közel álló Bark frekvenciaskála mellett optimalizálja a tervezési algoritmus a specifikációtól való eltérést [25].

Példaként egy 30-ad fokú WIIR szűrőt terveztettem, melynek átvitele a 6.6. ábrán, pólusai és zérusai pedig a 6.7. ábrán található. Jól látható, hogy a pólusok alacsony frekvenciára koncentrálnak, míg a zérusok soros realizációnál a pólusokhoz közel, párhuzamosnál pedig a valós tengelyen helyezkednek el.

A 6.9–6.10. ábrákon található a zaj spektrumának becslője. Látható, hogy az optimalizálás nem csak a zaj teljesítményét csökkenti, hanem a spektrumát is egyenletesebbé teszi, ami a szubjektív zajérzetre is jótékony hatással van.



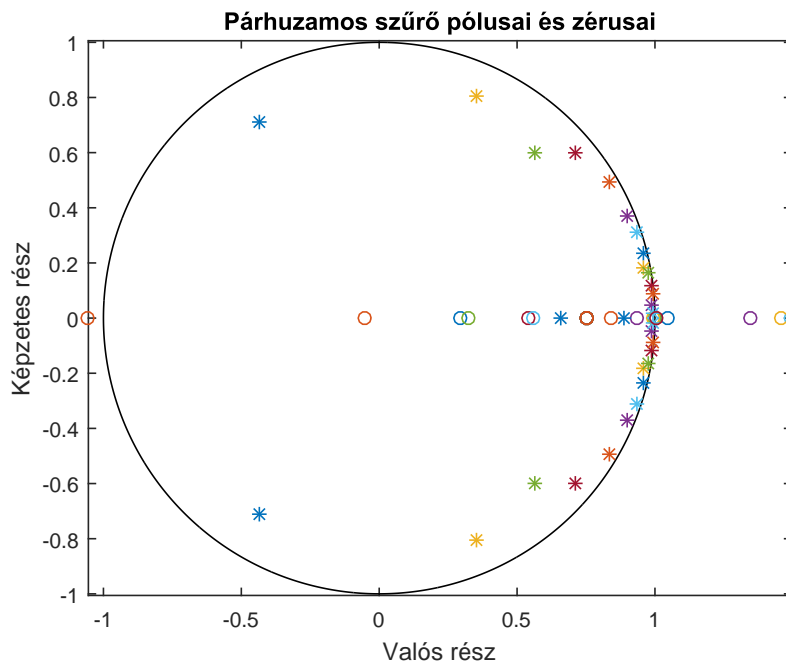
6.6. ábra. 30-ad fokú WIIR szűrő átvitele, $\lambda = 0,75$



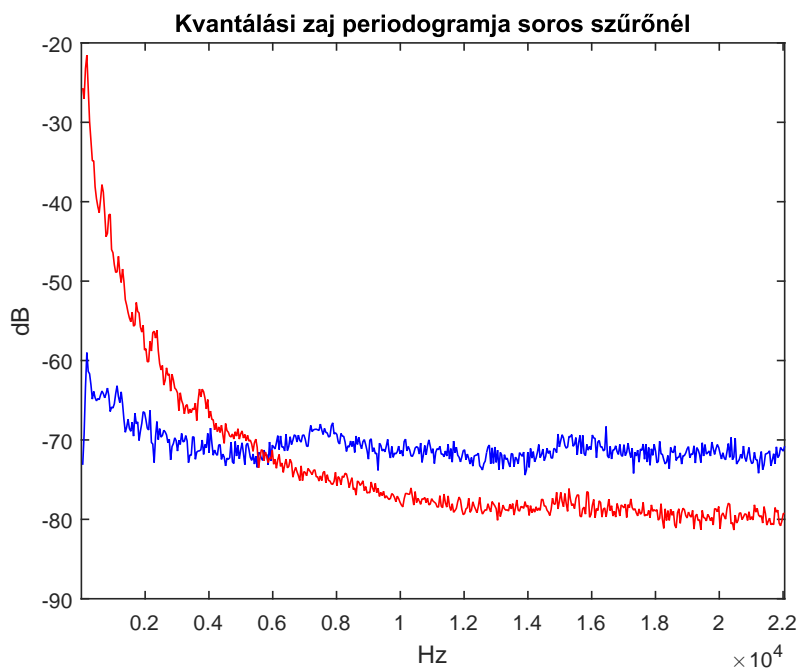
6.7. ábra. 30-ad fokú WIIR szűrő pólusai és zérusai, $\lambda = 0,75$

Optimalizálás után a soros struktúra 6 db Chamberlin, 7 db WIIR és 2 db DF2 tagot tartalmaz. A zaja ennek hatására kb. 30 dB-lel lett kisebb, míg számításigénye 43%-kal lett nagyobb.

A párhuzamos realizáció optimalizálás után 1 db DF2, 1 db Gold & Rader, 5 db

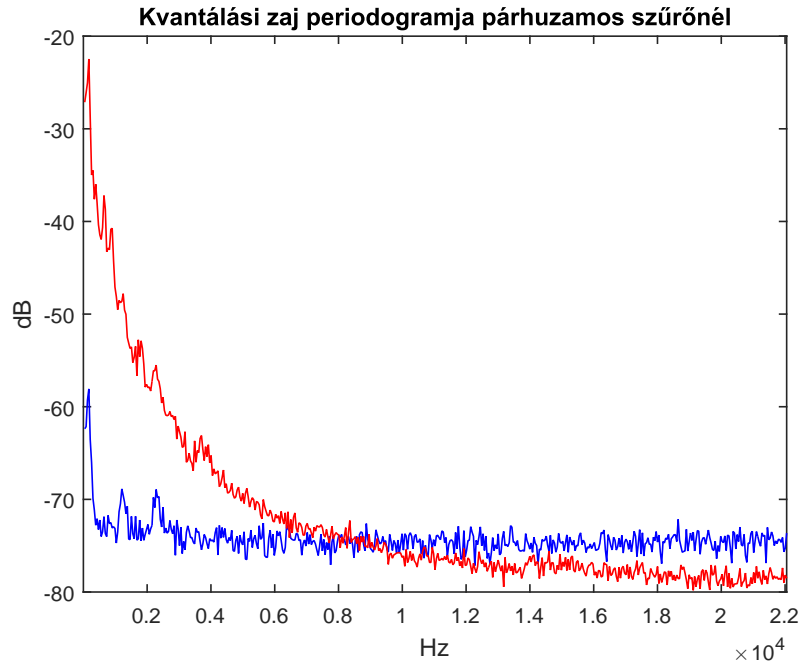


6.8. ábra. 30-ad fokú WIIR szűrő párhuzamos tagjainak pólusai és zérusai, $\lambda = 0,75$



6.9. ábra. WIIR szűrő zaja soros megvalósításban, 16 biten, $\lambda = 0,75$
 Piros: megvalósítás DF1 tagokkal, kék: optimalizált struktúra zaja

Kingsbury, 3 db Chamberlin, 4 db WIIR és 1 db Zölzer tagból áll. A zaja kb. 28 dB-lel kisebb, mint az optimalizálatlané, számításigénye pedig 37%-kal nagyobb.



6.10. ábra. WIIR szűrő zaja párhuzamos megvalósításban, 16 biten, $\lambda = 0,75$
 Piros: megvalósítás DF1 tagokkal, kék: optimalizált struktúra zaja

Struktúra	WIIR	DF1 soros	DF1 párhuzamos	Opt. soros	Opt. párhuzamos
Zajtelsítmény 16 biten	-52,7 dB	-13,4 dB	-14,7 dB	-40,2 dB	-43,5 dB
Zajtelsítmény 24 biten	-101,2 dB	-59,9 dB	-61,8 dB	-88 dB	-89,9 dB
Zajtelsítmény 32 biten	-152 dB	-108,2 dB	-110,3 dB	-136,6 dB	-137,9 dB
Utastítség	344	317	345	466	472

6.2. táblázat. 30-ad fokú WIIR realizációinak zaja és számításigénye, $\lambda = 0,75$

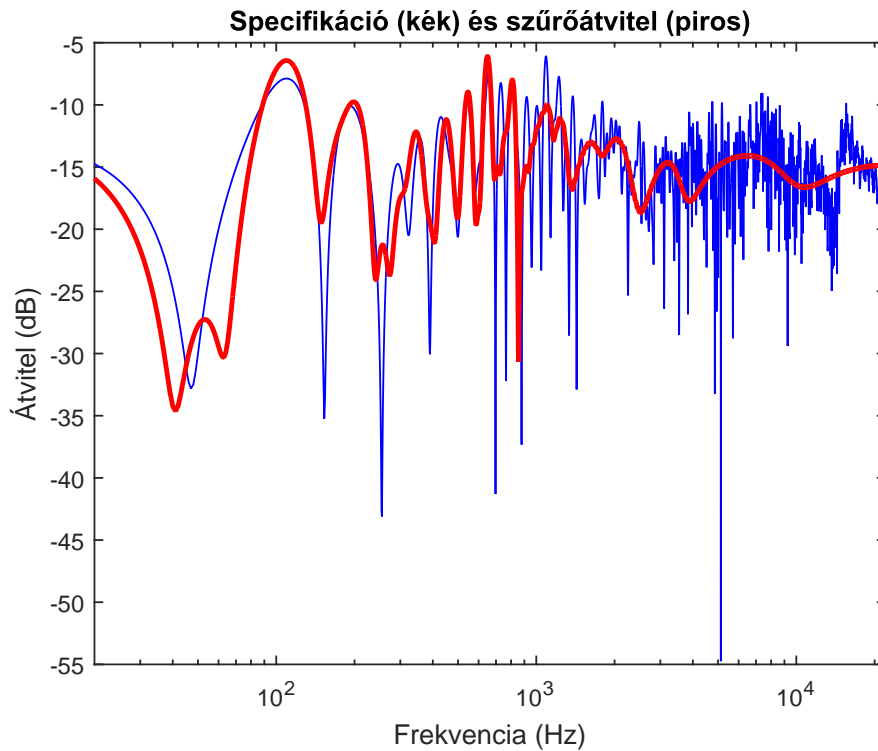
6.3. WIIR, $\lambda = 0,9375$

Átvitelek modellezésénél gyakoriak az olyan warpolt szűrők, melyek nagy fokszámuk mellett 0,9 fölötti warpolási tényezővel bírnak [7]. Itt is egy ilyen szűrőt fogok bemutatni: a λ értékét úgy választottam meg, hogy kettes számrendszerben könnyen lehessen ábrázolni, így a 40-es fokszámú szűrő warpolási tényezője $\lambda = 0,9375$ lett, ezáltal a pólusai és zérusai nagyon közel kerülnek a $z = 0$ -hoz.

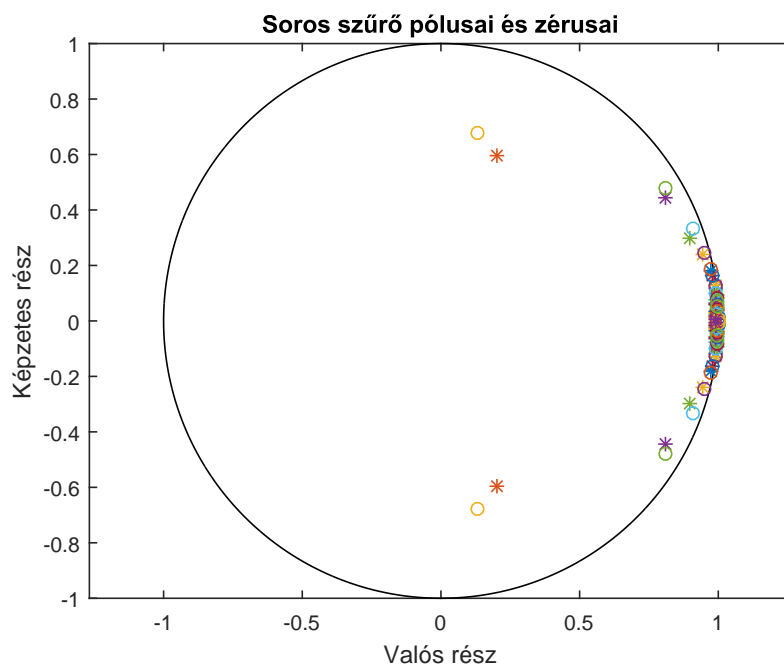
A tervezést itt is egy szoba impulzusválaszára végeztem el. A szűrő átvitele a 6.11. ábrán, a soros és párhuzamos realizáció pólus-zérus elrendezése pedig a 6.12–6.13. ábrákon látható.

A vizsgálatokat ennél a szűrőnél csak 32 biten végeztem el, mert a nagy fokszám és az 1-hez közeli warpolási tényező együttesen azt eredményezte, hogy már 24 biten is instabillá vált a szűrő. Az eredmények a 6.3. táblázatban találhatóak.

Optimalizálás után a soros megvalósítás 12 db Chamberlin, 6 db WIIR, 1 db

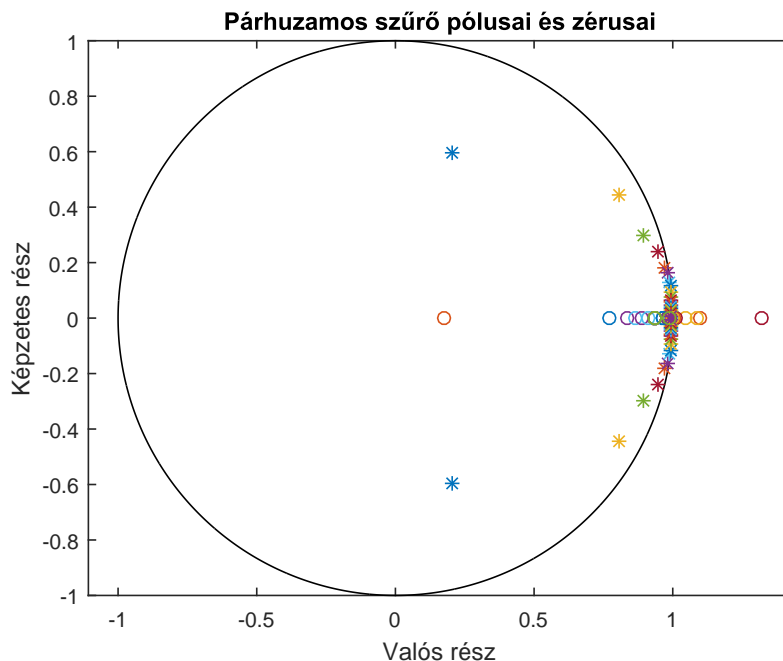


6.11. ábra. 40-ed fokú WIIR szűrő átvitele, $\lambda = 0,9375$



6.12. ábra. 40-ed fokú WIIR szűrő pólusai és zérusai, $\lambda = 0,9375$

Gold & Rader és 1 db Zölzer másodfokú tagból áll. A DF1 tagokból álló struktúrához képest kb. 11 dB-lel kisebb a zaj, de a közvetlen WIIR realizációhoz viszonyítva a javulás kb. 37 dB. A számításigény az optimalizálatlan soros implementációnál 38%-kal, a közvetlen WIIR-nél 40%-kal nagyobb.



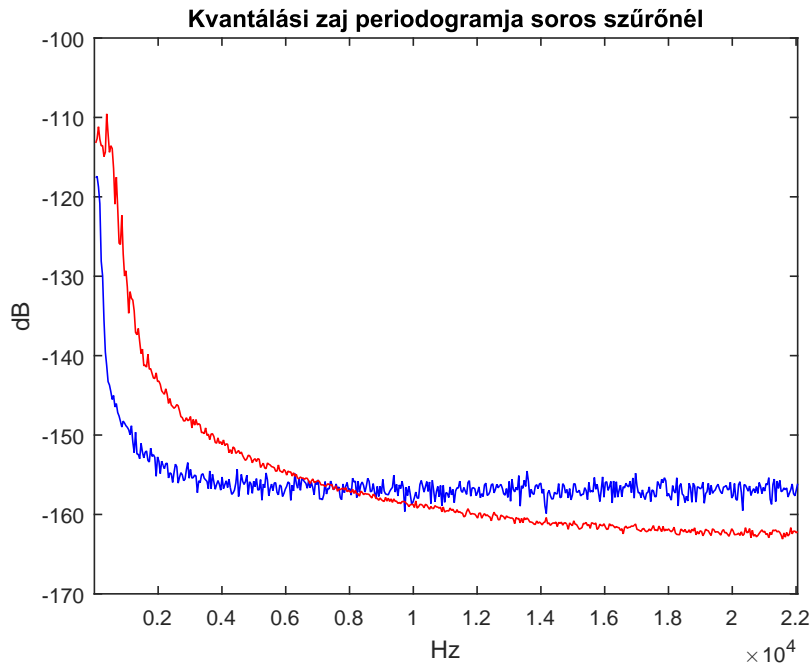
6.13. ábra. 40-ed fokú WIIR szűrő párhuzamos tagjainak pólusai és zérusai, $\lambda = 0,9375$

A párhuzamos struktúra optimalizálása után 8 db Chamberlin, 5 db Zölzer, 3 db Kingsbury, 3 db WIIR és 1 db Gold & Rader másodfokú tagból áll. A DF1 tagokból álló struktúrához viszonyítva kb. 36 dB a javulás, a közvetlen WIIR realizációhoz képest viszont kb. 53 dB-lel kisebb a zaj. A számításigény optimalizálás után 34%-kal nőtt a DF1 tagokból felépített struktúrához viszonyítva, míg a közvetlen WIIR realizációhoz képest 35%-kal nagyobb.

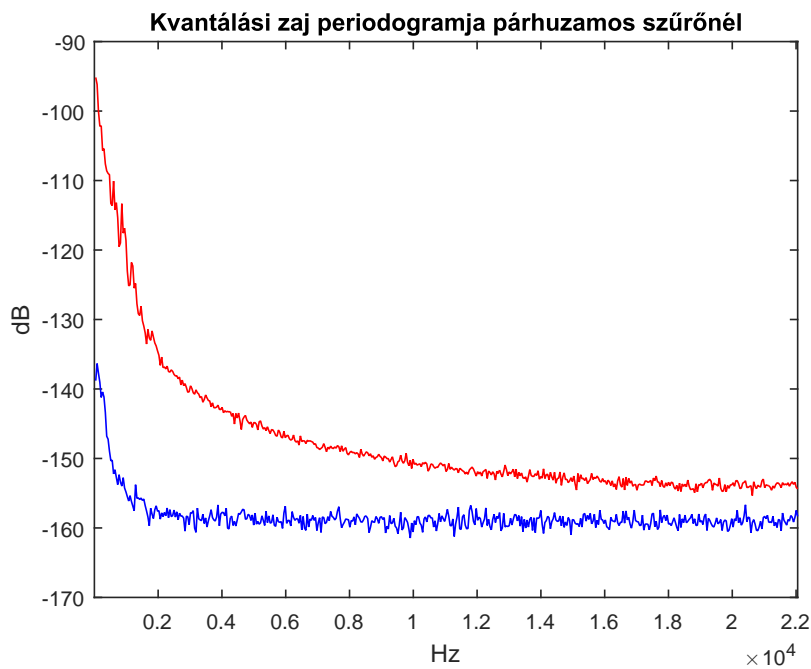
A soros és párhuzamos struktúrák zajspektrumai a 6.14–6.15. ábrákon található. Látható, hogy az optimalizált szűrők kb. 200 Hz fölött egyenletes zajspektrummal rendelkeznek, alatta viszont meredek kiemelés tapasztalható.

Struktúra	WIIR	DF1 soros	DF1 párhuzamos	Opt. soros	Opt. párhuzamos
Zajtjeljesítmény 32 biten	-72,2 dB	-98,2 dB	-87,3 dB	-109,1 dB	-124,8 dB
Utasításszám	454	422	460	598	615

6.3. táblázat. 40-ed fokú WIIR realizációinak zaja és számításigénye, $\lambda = 0,9375$



6.14. ábra. WIIR szűrő zaja soros megvalósításban, 32 biten, $\lambda = 0,9375$
Piros: megvalósítás DF1 tagokkal, kék: optimalizált struktúra zaja

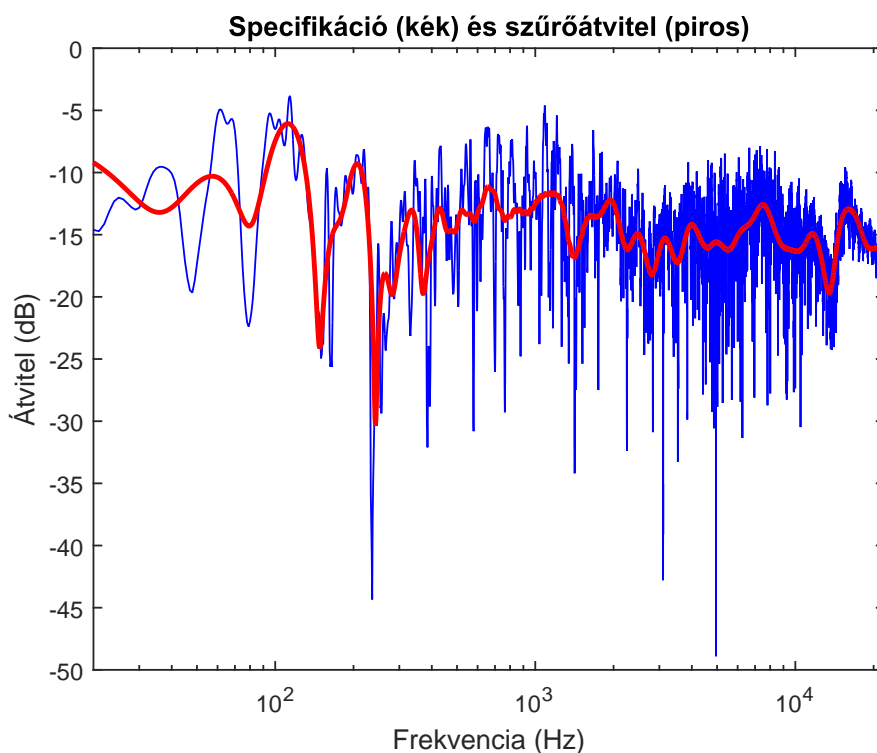


6.15. ábra. WIIR szűrő zaja párhuzamos megvalósításban, 32 biten, $\lambda = 0,9375$
Piros: megvalósítás DF1 tagokkal, kék: optimalizált struktúra zaja

6.4. Fixpólusú párhuzamos szűrő

A példában az előre rögzített pólusokat logaritmikus skála mentén helyeztem el 30 Hz és 18 kHz között. A pólusok jósági tényezőjét úgy állítottam be, hogy az egymás melletti pólusokból képzett tagok zérusok nélküli átviteli a -3 dB-es pontban metsszék egymást.

A vizsgálatokat itt is csak 32 biten végeztem el, mert az alacsony frekvenciákra kerülő nagy jóságú pólusok igen nagy kvantálási zajt eredményeznek. A zajteljesítmények és a számításigények a 6.4. táblázatban találhatóak.



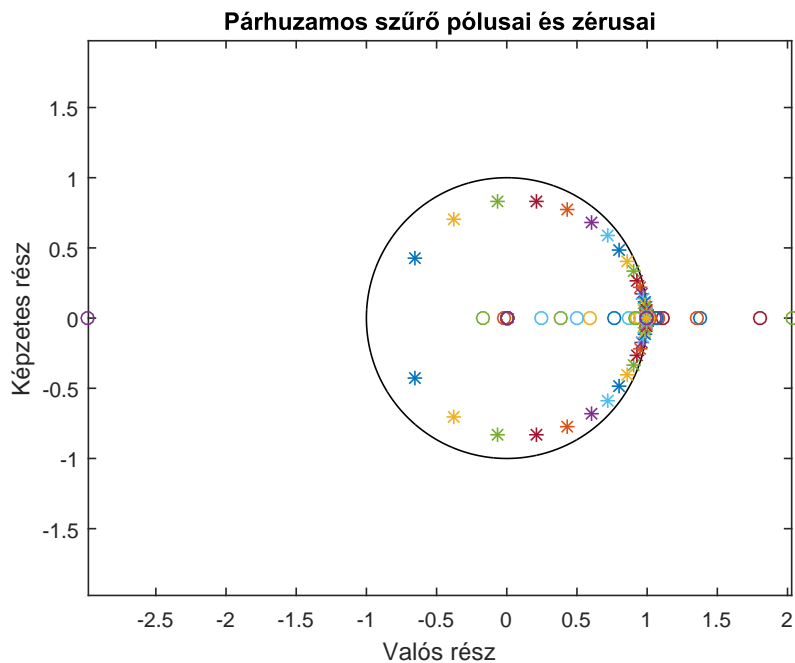
6.16. ábra. 30-ad fokú fixpólusú párhuzamos szűrő átvitele

Az optimalizált párhuzamos realizáció 1 db Gold & Rader, 1 db DF2, 4 db WIIR, 3 db Kinsbury és 6 db Chamberlin tagból áll. A DF1 tagokból felépített struktúrához képest kb. 32 dB-lel kisebb az optimalizált megvalósítás zaja, míg számításigénye kb. 35%-kal nagyobb.

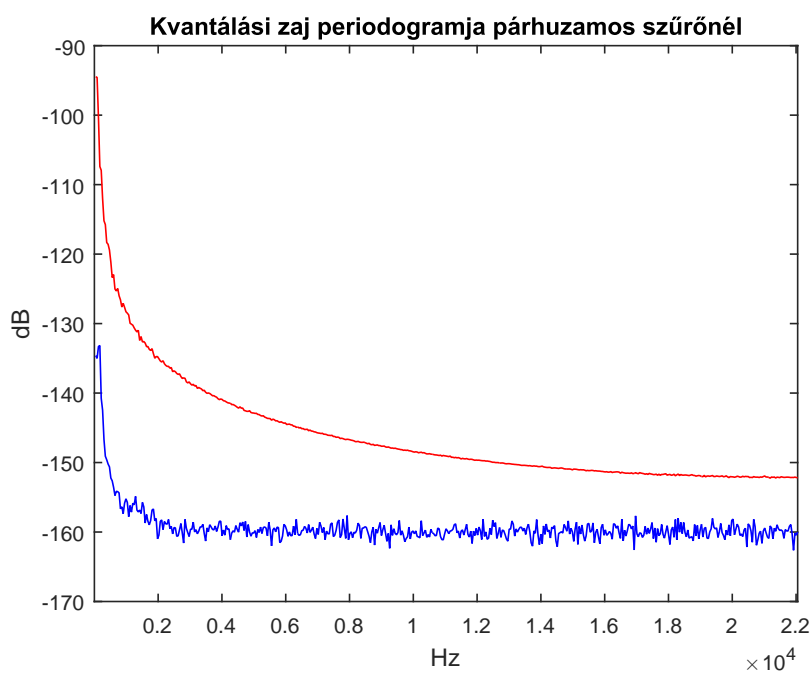
A zajspektrumok a 6.18. ábrán találhatóak. Látható, hogy az optimalizált struktúra zaja kb. 200 Hz fölött egyenletes, alatta meredek kiemelés tapasztalható.

Struktúra	DF1 párhuzamos	Opt. párhuzamos
Zajteljesítmény 32 biten	-90,1 dB	-122,8 dB
Utasításszám	345	467

6.4. táblázat. 30-ad fokú fixpólusú párhuzamos szűrő zaja és számításigénye



6.17. ábra. 30-ad fokú fixpólusú párhuzamos szűrő pólusai és zérusai



6.18. ábra. 30-ad fokú fixpólusú párhuzamos szűrő zaja 32 biten
 Piros: megvalósítás DF1 tagokkal, kék: optimalizált struktúra zaja

6.5. Értékelés

A fejezetben bemutatott gyakorlati példák alapján elmondható, hogy a javasolt optimalizálás jelentősen csökkenti a soros és párhuzamos szűrők zaját. Általánosan audio területen a -90 dB alatti, stúdió eszközöknél a -110 dB alatti zajteljesítményt szokás megkövetelni. Sajnos ez alapján 16 bites számábrázolásnál a zaj még optimalizáltan is túl nagy

marad az audio célú felhasználásra.

A 24 bites implementáció ezzel szemben már megfelelő lehet. Itt nagyon fontos feladat a szűrő optimalizációja, az elérhető javulás hallhatóan jobb szűrőt eredményez. Nagy fokszámú szűrőknél ráadásul előfordul, hogy a közvetlen realizáció nem valósítható meg az együtttható-kerekítés miatt, így mindenképpen másodfokú tagokból kell felépíteni a szűrő-átvitelt.

Láthattuk, hogy nagy fokszámnál és warpolási tényezőnél, valamint fixpólusú párhuzamos szűrőknél már 32 biten is fontos feladat a másodfokú tagokra bontás és az optimalizálás. Bizonyos esetekben ugyanis a közvetlen realizációnak túl nagy lesz a zaja, a gyakran használt DF1 másodfokú tagokkal felépített szűrő viszont szintén nem felel meg az elvárásainknak.

Az optimalizálás hatására jelentősen csökken a soros és párhuzamos szűrők zaja, a számításigényük viszont tipikusan 40%-kal lesz nagyobb. Elmondhatjuk tehát, hogy sok esetben érdemes lehet elvégezni az optimalizációt.

7. fejezet

Összefoglalás

Dolgozatomban az audio területen használatos logaritmikus frekvenciafelbontású warped és fixpólusú párhuzamos szűrőket mutattam be, részletezve a tervezésüket és implementációjukat.

Megmutattam, hogy fixpontos számábrázolás mellett milyen nemkívánatos jelenségekkel számolhatunk, így az együttható-kerekítéssel, a kvantálási zajjal és a túlcordulással részletesen foglalkoztam.

Bemutattam a különböző szűrőstruktúrákat, és ahol szükséges volt, zérusokkal is kiegészítettem őket. Az egyes szűrőstruktúrák kerekítési pontjait és számításigényét egy processzormodellen vizsgáltam meg. Az így meghatározott erőforrásigény pontosabb becslést ad egy adott processzoron megvalósított szűrő futásidejéről, mint az elvégzett matematikai műveletek darabszámának összege.

Az együttható-kerekítés vizsgálatára egy érzékenységszámításon alapuló analitikus módszert ismertettem. Az átvitel megváltozásának worst-case esetére egy gyakorlati módszert is adtam, amit később minden másodfokú struktúrára alkalmaztam, különböző pólus-zérus elrendezések mellett. Az eredmények alapján arra a következtetésre jutottam, hogy másodfokú tagok esetében ritkán jelent problémát az együtthatók kerekítése 16 biten, nagyobb pontosságú számábrázolás mellett pedig elhanyagolható a hatása.

A kvantálási zaj vizsgálatára egy olyan módszert mutattam be, amivel az adott szűrőstruktúra zajspektruma előállítható. A zajteljesítmény-számításához abból indultam ki, hogy a kerekítés művelete additív fehérzajjal modellezhető, így a szűrő zaja a kerekítési pontok és a kimenet közötti átvitelből számolható. A tapasztalatok alapján a zaj periodogramja az ismertetett módon számolt zajspektrumhoz konvergál.

A bemeneti skálázási tényező meghatározására az a javaslatom, hogy a szűrőt lebegőponton implementálva gerjesszük meg egy kellően hosszú rózsazajjal. A skálázási tényezőt ekkor a struktúrában ilyen feltétel mellett fellépő legnagyobb számértéknek megfelelően válasszuk. Ezzel az eljárással elérhető, hogy a különböző struktúrák hasonló nagyságú bemenőjelekre csorduljanak túl, így a zajteljesítményük összehasonlíthatóvá váljon.

A vizsgálatok elvégzésével bebizonyosodott, hogy a zérusok helyzete is erősen befolyásolja a kvantálási zaj teljesítményét, így csak korlátozottan használhatóak azon korábbi irodalmak eredményei, ahol csak a pólusokat vizsgálták. Néhány esetet megvizsgálva

javaslatokat adtam arra vonatkozóan, hogy a pólusok és zérusok elhelyezkedésének függvényében milyen másodfokú struktúrával célszerű megvalósítani az adott tagot, hogy annak minimális zaja legyen.

Ezen vizsgálati módszerre alapozva olyan eljárást javasoltam, amivel csökkenthető a másodfokú tagokból felépített szűrők numerikus zaja. Javaslataim jól használhatóak párhuzamos szűrők esetén, ahol az eredő zaj az egyes tagok zajának összegeként adódik. Az összegzés soros kapcsolásnál más, ott ugyanis a sorban elől lévők zaja végighalad a többi tagon is, így azok átvitele formálja a spektrumot. Nagy általánosságban elmondható viszont, hogy ha az átvitelben nincsenek túl nagy kiugrások, akkor soros szűrőnél nem véltünk nagy hibát a dolgozatban megfogalmazott javaslatok alkalmazásával. Ezt támasztják alá a gyakorlati példák, ahol sikerült jelentősen csökkenteni a soros és párhuzamos szűrők zaját.

Jövőbeli tervek

Dolgozatomban a numerikus zaj teljesítményét minimalizáltam, de audio alkalmazásokról lévén szó, a zaj érzékelt teljesítményét is célszerű lenne vizsgálni. Ehhez a kvantálási zaj spektrumát érdemes lehet valamilyen érzeti modell szerinti átvittel súlyozni, így gyakorlatilag az érzékelt zajteljesítményt kaphatjuk meg. A legkisebb szubjektív zajú tagot ilyen módon kiválasztva is elvégezhetjük az optimalizációt.

A soros szűrőnél további kérdések is felmerülnek, ugyanis a kimenet zajspektrumát az egyes tagok sorrendje is befolyásolja. Ugyan létezik a megoldásra „ökölszabály” (pólushossz szerinti sorba állítás), de ez nem biztos, hogy speciális tagok esetén is működik. E téren további vizsgálatok szükségesek.

Érdemes lehet még a tagok kiválasztásánál a számításigényt is figyelembe venni. Ha ugyanis a másodfokú tagokat az elérhető zajteljesítmény-csökkenés alapján állítjuk sorba, akkor csak az első N darabot optimalizálva kompromisszum köthető a zajteljesítmény és a számításigény között. Így elérhető az, hogy csak pár tag esetén változtatjuk meg a struktúrát, a zajteljesítményben mégis jelentős javulást érünk el.

A bemutatott másodfokú tagokon kívül lehetséges még más struktúrák alkalmazása is, akár negyedfokú, vagy nagyobb fokszámú tagok formájában is, amennyiben a kvantálási zaj csökkenthető velük. Az ilyen hibrid struktúrák vizsgálatára a jövőben kerül sor.

Irodalomjegyzék

- [1] G. Mitra, B. Johnston, A. P. Rendell, E. McCreath, and J. Zhou, „Use of SIMD vector operations to accelerate application code performance on low-powered ARM and Intel platforms,” in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International*, pp. 1107–1116, IEEE, 2013.
- [2] R. Keil, „Digital signal processing with Cortex-M microcontrollers,” tech. rep., ARM Germany GmbH., 2011.
- [3] W. Chen, „Performance of cascade and parallel iir filters,” *Journal of the Audio Engineering Society*, vol. 44, no. 3, pp. 148–158, 1996.
- [4] B. Gold and C. M. Rader, „Effects of quantization noise in digital filters,” in *Proceedings of the April 26-28, 1966, Spring joint computer conference*, pp. 213–219, ACM, 1966.
- [5] U. Zölzer, *Digital audio signal processing*. John Wiley & Sons, 2008.
- [6] A. Härmä, M. Karjalainen, L. Savioja, V. Välimäki, U. K. Laine, and J. Huopaniemi, „Frequency-warped signal processing for audio applications,” *Journal of the Audio Engineering Society*, vol. 48, no. 11, pp. 1011–1031, 2000.
- [7] B. Bank, „Direct design of parallel second-order filters for instrument body modeling,” in *Proc. International Computer Music Conference (ICMC 2007), Copenhagen, Denmark*, pp. 458–465, 2007.
- [8] B. Bank, „Loudspeaker and room response equalization using parallel filters: comparison of pole positioning strategies,” in *Audio Engineering Society Conference: 51st International Conference: Loudspeakers and Headphones*, Audio Engineering Society, 2013.
- [9] S. Ahuja and S. D. Roy, „Variable digital filters,” *IEEE Transactions on Circuits and Systems*, vol. 27, no. 9, pp. 836–838, 1980.
- [10] C. Asavathiratham, *Digital audio filter design using frequency transformations*. PhD thesis, Massachusetts Institute of Technology, 1996.
- [11] M. Tyril, J. A. Pedersen, and P. Rubak, „Digital filters for low-frequency equalization,” *Journal of the Audio Engineering Society*, vol. 49, no. 1/2, pp. 36–43, 2001.

- [12] B. Bank and G. Ramos, „Improved pole positioning for parallel filters based on spectral smoothing and multiband warping,” *Signal Processing Letters, IEEE*, vol. 18, no. 5, pp. 299–302, 2011.
- [13] B. Bank, „Audio equalization with fixed-pole parallel filters: An efficient alternative to complex smoothing,” in *Audio Engineering Society Convention 128*, Audio Engineering Society, 2010.
- [14] S. W. Smith, „The scientist and engineer’s guide to digital signal processing,” 1997.
- [15] N. Kingsbury, „Second-order recursive digital-filter element for poles near the unit circle and the real z axis,” *Electronics Letters*, vol. 8, no. 6, pp. 155–156, 1972.
- [16] C. M. Rader and B. Gold, „Effects of parameter quantization on the poles of a digital filter,” *Proceedings of the IEEE*, pp. 688–689, 1967.
- [17] U. Zölzer, *Entwurf digitaler Filter für die Anwendung im Tonstudiobereich*. PhD thesis, Technischen Universität Hamburg-Harburg, 1989.
- [18] A. Eletri, E. S. Salem, A. R. Zerek, and S. Elgandus, „Effect of coefficient quantization on the frequency response of an IIR digital filter by using software,” in *2010 7th International Multi-Conference on Systems Signals and Devices (SSD)*, pp. 1–6, IEEE, 2010.
- [19] W. R. Bennett, „Spectra of quantized signals,” *Bell System Technical Journal*, vol. 27, no. 3, pp. 446–472, 1948.
- [20] K. Steiglitz, „A note on variable recursive digital filters,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28, no. 1, pp. 111–112, 1980.
- [21] M. Karjalainen, A. Harma, and U. K. Laine, „Realizable warped IIR filters and their properties,” in *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing, 1997. ICASSP-97.*, vol. 3, pp. 2205–2208, IEEE, 1997.
- [22] H. Chamberlin, *Musical Applications of Microprocessors*. Indianapolis, Indiana, USA: Sams, 1984.
- [23] U. Zölzer, „A low roundoff noise digital audio filter,” in *Proc. Fifth European Signal Processing Conference (5^a, 1990, Barcelona, Spain)*. *Eusipco*, vol. 90, pp. 529–532, 1990.
- [24] T. W. Parks and C. S. Burrus, *Digital filter design*. Wiley-Interscience, 1987.
- [25] M. Karjalainen, E. Piirilä, A. Järvinen, and J. Huopaniemi, „Comparison of loudspeaker equalization methods based on DSP techniques,” *Journal of the Audio Engineering Society*, vol. 47, no. 1/2, pp. 14–31, 1999.

Függelék

F.1. Szűrő implementációk a processzormodellen

A 4. fejezetben bemutatott szűrők számításigényét a processzor modellen implementálva számoltam. Az egyes implementációkat itt helyeztem el.

```
for n = 0...mintaszám do
  CLR Acc
  LOAD B, b[0]
  LOAD C, u[n] /* bemenet */
  MAC Acc, B, C
  for i = 1...N do
    LOAD B, b[i]
    LOAD C, x1[i] /* x1[] az első sor késleltető */
    LOAD D, a[i]
    LOAD E, x2[i] /* x2[] a második sor késleltető */
    MAC Acc, B, C
    MAC Acc, D, E
  end
  MOV A, Acc
  STORE A, y[n]
  STORE A, x2[0]
end
```

F.1. algoritmus: N -ed fokú DF1 szűrő implementációja a processzormodellen.

```
for n = 0...mintaszám do
  CLR Acc1
  CLR Acc2
  for i = 1...N do
    LOAD B, b[i]
    LOAD C, a[i]
    LOAD D, x[i] /* x[] a késleltető */
    MAC Acc1, C, D
    MAC Acc2, B, D
  end
  MOV A, Acc1
  ADD A, A, u[n] /* bemenet */
  STORE A, x[0]
  LOAD B, b[0]
  MAC Acc2, A, B
  MOV A, Acc2
  STORE A, y[n]
end
```

F.2. algoritmus: N -ed fokú DF2 szűrő implementációja a processzormodellen.


```

for  $n = 0 \dots \text{mintaszám}$  do
  LOAD B, x1[0] /* bal oldali késleltetősor */
  LOAD C, u[n] /* bemenet */
  ADD A, B, C
  for  $i = 2 \dots N$  do
    LOAD B, b[i]
    LOAD C, x1[i] /* x1[] a bal oldali késleltetősor */
    LOAD D, a[i]
    LOAD E, x2[i] /* x2[] a jobb oldali késleltetősor */
    MUL B, B, A
    MUL D, D, A
    ADD C, C, B
    ADD E, E, D
    STORE C, x1[i-1]
    STORE E, x2[i-1]
  end
  LOAD B, b[0]
  MUL A, A, B
  STORE A, y[n]
end

```

F.3. algoritmus: N -ed fokú DF1 transzponált szűrő implementációja.

```

for  $n = 0 \dots \text{mintaszám}$  do
  CLR Acc
  LOAD A, x[1] /* x[] a késleltető */
  LOAD B, r[1] /*  $r[1] = r \cdot \cos(\text{phi})$  */
  LOAD C, x[2]
  LOAD D, r[3] /*  $r[3] = -r \cdot \sin(\text{phi})$  */
  MAC Acc, A, B
  MAC Acc, C, D
  MOV E, Acc
  LOAD D, u[n] /* bemenet */
  ADD E, E, D
  CLR Acc
  LOAD D, r[2] /*  $r[2] = r \cdot \sin(\text{phi})$  */
  MAC Acc, A, D
  MAC Acc, B, C
  MOV B, Acc
  LOAD D, k[0]
  MAC Acc, E, D
  LOAD D, k[1]
  MAC Acc, A, D
  LOAD D, k[2]
  MAC Acc, C, D
  MOV A, Acc
  STORE E, x[1]
  STORE B, x[2]
  STORE A, y[n]
end

```

F.4. algoritmus: Gold & Rader másodfokú tag implementációja.

```

for  $n = 0 \dots \text{mintaszám}$  do
  LOAD A, u[n] /* bemenet */
  LOAD B, b[0]
  LOAD C, x[1]
  MUL B, A, B
  ADD B, B, C
  for  $i = 1 \dots N$  do
    LOAD C, b[i]
    LOAD D, a[i]
    LOAD E, x[i+1] /* x[] a késleltető */
    MUL C, A, C
    MUL D, B, D
    ADD C, C, D
    ADD C, C, E
    STORE C, x[i]
  end
  STORE B, y[n]
end

```

F.5. algoritmus: N -ed fokú DF2 transzponált szűrő implementációja.

```

for  $n = 0 \dots \text{mintaszám}$  do
  CLR Acc
  for  $i = 1 \dots N$  do
    LOAD C, c[i]
    LOAD D, x[i] /* x[] a késleltető */
    MAC Acc, C, D
  end
  MOV A, Acc
  LOAD C, u[n] /* bemenet */
  ADD A, A, C
  LOAD C, c[0]
  MUL A, A, C
  LOAD D, lambda
  CLR Acc
  for  $i = 0 \dots N$  do
    LOAD B, b[i]
    MAC Acc, A, B
    LOAD C, x[i+2]
    SUB C, C, A
    MUL C, C, D
    LOAD B, x[i+1]
    STORE A, x[i+1]
    ADD A, A, B
  end
  MOV A, Acc
  STORE A, y[n]
end

```

F.6. algoritmus: N -ed fokú warpolt IIR szűrő implementációja.

```

for  $n = 0 \dots \text{mintaszám}$  do
  CLR Acc
  LOAD A, x[1] /* x[] a késleltető */
  LOAD B, x[2]
  LOAD C, l[3]
  LOAD D, u[n] /* bemenet */
  MAC Acc, C, D
  LOAD C, k[2]
  MAC Acc, C, A
  MOV C, Acc
  SUB C, C, B
  LOAD D, k[1]
  MUL C, C, D
  ADD C, C, A
  MUL D, C, D
  ADD D, D, B
  CLR Acc
  LOAD E, l[1]
  MAC Acc, E, B
  LOAD E, l[2]
  MAC Acc, E, A
  MOV E, Acc
  ADD E, D
  STORE C, x[1]
  STORE D, x[2]
  STORE E, y[n]
end

```

F.7. algoritmus: Kingsbury másodfokú tag implementációja.

```

for  $n = 0 \dots \text{mintaszám}$  do
  LOAD A, x[1] /* x[] a késleltető */
  LOAD B, x[2]
  LOAD C, q
  LOAD D, u[n] /* bemenet */
  MUL C, C, A
  ADD C, C, B
  ADD C, C, D
  LOAD D, -f
  MUL D, C, D
  ADD D, D, A
  LOAD E, f
  MUL E, E, D
  CLR Acc
  LOAD F, k[0]
  MAC Acc, F, C
  LOAD F, k[1]
  MAC Acc, F, A
  LOAD F, k[2]
  MAC Acc, F, B
  MOV A, Acc
  ADD A, A, E
  STORE D, x[1]
  STORE E, x[2]
  STORE A, y[n]
end

```

F.8. algoritmus: Chamberlin másodfokú tag implementációja.

```

for  $n = 0 \dots \text{mintaszám}$  do
  LOAD A, x[1] /* x[] a késleltető */
  LOAD B, x[2]
  LOAD C, z[1] /* z1 együttható */
  LOAD D, u[n] /* bemenet */
  MUL E, C, B
  ADD D, D, E
  ADD E, D, A
  CLR Acc
  MAC Acc, C, E
  LOAD C, z[2] /* z2 együttható */
  MAC Acc, C, B
  MOV F, Acc
  LOAD C, z[3] /* -z1 együttható */
  MUL F, F, C
  ADD F, F, B
  CLR Acc
  LOAD C, k[0] /* k0 együttható */
  MAC Acc, C, D
  LOAD C, k[1] /* k1 együttható */
  MAC Acc, C, A
  LOAD C, k[2] /* k2 együttható */
  MAC Acc, C, B
  MOV A, Acc
  ADD A, A, F
  STORE E, x[1]
  STORE F, x[2]
  STORE A, y[n]
end

```

F.9. algoritmus: Zölzer másodfokú tag implementációja.