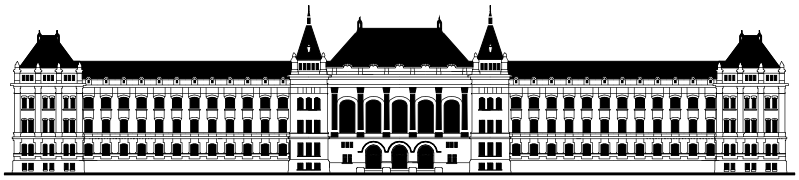


Akusztikus visszacsatolás kompenzálása hangosító rendszerekben

Czene Gábor IV. Vill.
Szilágyi László IV. Vill.

Konzulens: dr. Sujbert László

Budapesti Műszaki és Gazdaságtudományi Egyetem, Méréstechnika és Információs Rendszerek Tanszék

Tartalomjegyzék

1. Az akusztikai visszacsatolás problémája

1.1. Bevezető

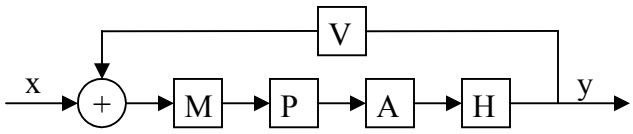
Kisebb-nagyobb rendezvényeken gyakran találkozunk hangosító rendszerekkel, úgynevezett PA (public address) rendszerekkel. Ezek a rendszerek egy vagy több előadó, hangszer vagy más hangforrás hangját erősítik fel a közönség számára a műsor élvezhetősége érdekében. Ha a hangforrás jele nem direkt módon, dedikált audiokimenet révén kerül a keverőpultba (pl. cd-lejátszó, szintetizátor), hanem mikrofon segítségével, és a forrás egy légtérben található hangsugárzóval, a rendszer instabillá válhat. A mikrofon ugyanis nem csak a beszélő, énekes, hangszer stb. hangját fogja fel, hanem csatolásba kerül a saját felerősített jelével meghajtott hangszórókkal is. Ezen hurok átviteli függvényében, ha egy frekvenciakomponens eléri az egységnyi erősítést, valamint fázisa 360° egész számú többszöröse, gerjedés lép fel, a jellegzetes, éles, sivító hang a műsort élvezhetetlenné teszi. Ebből következik, hogy a hangosító rendszer leadható maximális teljesítményét nemcsak a rendszer kivezérelhetősége korlátozza, hanem az akusztikai visszacsatolás is, melynek mértéke egyenesen arányos a hangosító rendszer által produkált hangnyomással. Mivel adott szituációban törekszünk a lehető legnagyobb teljesítmény leadására, a visszacsatolás kezelése ebből a szempontból kritikus lehet.

Dolgozatunkban a visszacsatolás hatásainak mérséklési lehetőségeit vizsgáljuk, a maximális gerjedésmentes erősítés fokozása érdekében. Kitérünk a visszacsatolást mintegy leutánzó, azt negatívan visszacsatoló adaptív FIR, illetve IIR szűrőket alkalmazó rendszerek megvalósítási lehetőségeire, majd pedig a hurokerősítés átviteli karakterisztikáját lyukszűrőkkel megváltoztató rendszereket tárgyaljuk.

1.2. A probléma megközelítése

Olyan hangosítási rendszerekben, melyekben a mikrofon és az annak felerősített jelét kiadó hangsugárzó egy légtérben van, a jel útja a következő: A mikrofon az általa vett hangnyomást elektromos feszültséggé alakítja, az kábelen keresztül a keverőpultba kerül. Itt történik meg a jel előerősítése, hangszínszabályzása, a különböző források hangerőarányainak beállítása az adott műsornak megfelelően. Az esetleges további hangprocesszáló egységek is itt kerülnek a jelútba, úgymint kompresszor, vagy további effektek. A keverő kimenete egy végerősítőre kerül, mely a hangsugárzók meghajtásához szükséges teljesítményre erősíti fel annak kimenetét, jelen esetben a mikrofon jelét. A hangsugárzó a leadott teljesítményt azonban nemcsak a megcélzott közönség felé továbbítja, hanem a mikrofonnal is csatolásba kerül.

A rendszer a következő ábrával modellezhető:



1. ábra: az akusztikai visszacsatolás rendszermodellje.

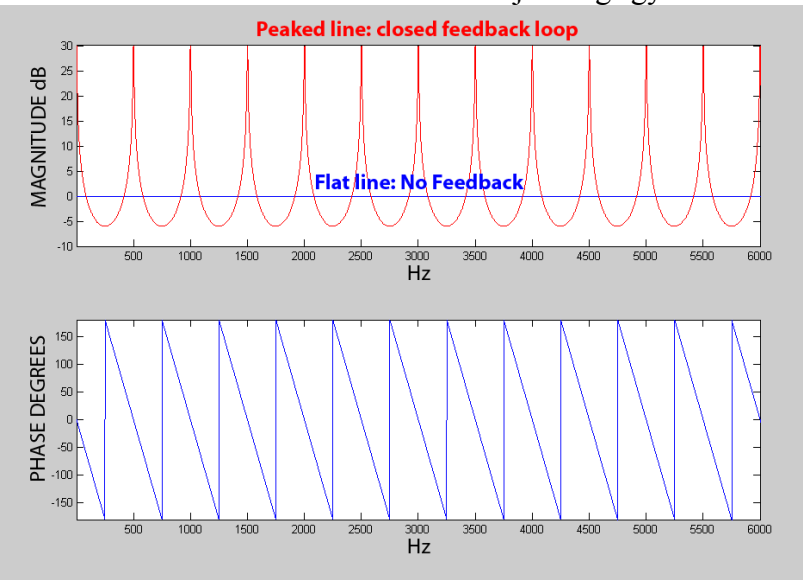
A hangosítandó jelforrást jelét az M mikrofon fogja fel, melynek jelét a P mikrofon-előerősítő vonalszintre erősíti. Az A teljesítményerősítő hajtja meg a H hangszórót. Ha a hangszóró és a mikrofon egymástól különválasztott akusztikai térben helyezkedne el, az átviteli karakterisztika az $M \cdot P \cdot A \cdot H$ szorzat lenne. Viszont a hangszóró és a mikrofon egyazon tér része, a hangszóróból a mikrofonba érkező V visszacsatolást is figyelembe kell vennünk, mely a rendszer átvitelét

$$\frac{Y}{X} = \frac{M \cdot P \cdot A \cdot H}{1 - M \cdot P \cdot A \cdot H \cdot V}$$

átvitelre módosítja. Az akusztikai visszacsatolás leginkább mint késleltetés jelenik meg a mikrofon és hangszóró között. Az így módosult átvitel megértéséhez induljunk ki a legegyszerűbb esetből!

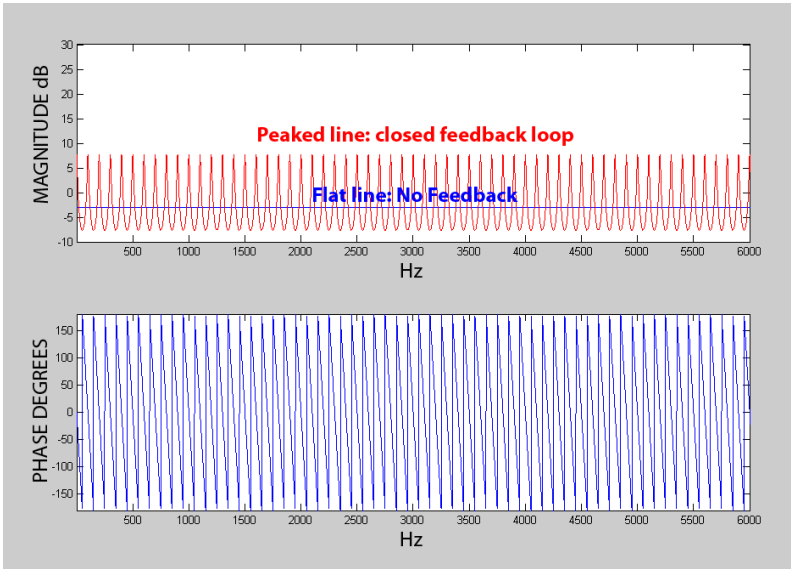
A legegyszerűbb rendszer egy konstans amplitúdókarakterisztikájú rendszer, melyet egy hangerő-szabályozóval és egy késleltetővel lehet modellezni [6]. A késleltetést a hang véges terjedési sebessége okozza hangszóró és mikrofon közötti visszacsatolásban. Ezen egyszerű modell elemzésével, az amplitúdó, illetve a késleltetés változtatásával megérthetjük a gerjedés természetét. A rendszer késleltetésének következménye az lesz,

hogy a különböző frekvenciakomponensek különböző fázishelyzettel kerülnek visszacsatolásra. A zárthurkú átvitelben ennek hatása jól megfigyelhető.



2. ábra. A rendszer nyílt (kék vonal), illetve zárthurkú (piros vonal) átvitele (fent) a fázis feltüntetésével (lent). [6]

A 2. ábra a rendszer nyílt-, illetve zárthurkú átvitelét szemlélteti. A kék görbe a nyílthurkú ($M \cdot P \cdot A \cdot H \cdot V$) rendszer egységnyi erősítésű 2 ms késleltetésű rendszer átvitele, a piros görbe ugyanezen rendszer a hurok zárása után, azaz $\frac{M \cdot P \cdot A \cdot H}{1 - M \cdot P \cdot A \cdot H \cdot V}$ átvitellel. A zárthurkú átvitel csúcsai a 0 fázishelyzetnek megfelelő helyeken találhatók, völgyei a 180 foknak megfelelő helyeken. A következő ábrán az előzőhöz képest 3 dB-lel kisebb erősítésű, 10 ms-ra növelt késleltetésű rendszer nyílt és zárthurkú átvitelét mutatja.



3. ábra. A rendszer nyílt (kék vonal), illetve zárthurkú (piros vonal) átvitelének módosulása az erősítés 3 dB-es csökkentésével, illetve a késleltetés 10 ms-ra növelésével (fent) valamint a fázis (lent). [6]

A változás hatására a zárthurkú erősítés jól láthatóan lecsökkent, a késleltetés növelése miatt a csúcsok sokkal közelebb kerültek egymáshoz. Az átvitel ezen csúcsai a potenciális gerjedési frekvenciák. Mivel jelen esetben lineárfázisú rendszerről van szó, a csúcsok frekvenciahelyzete a rendszer késleltetésének a függvénye.

$$t_p = \frac{\Delta\varphi}{\Delta f \cdot 360^\circ}$$

Ahol t_p a késleltetési idő, $\Delta\varphi$ az a fázisköz, ami a csúcsokat elválasztja, Δf pedig a spektrum csúcsai között lévő frekvenciatávolság. Jelen esetben $\Delta\varphi = 360^\circ$ -onként van csúcs. Ezért a keresett frekvenciaköz a következő képlettel számolható:

$$\Delta f = \frac{1}{t_p}$$

Néhány példa a potenciális gerjedési frekvenciák sűrűségére:

2 ms késleltetés: $\frac{1}{0.002s} = 500Hz$ frekvenciaköz

10 ms késleltetés: $\frac{1}{0.01s} = 100Hz$ frekvenciaköz

100 ms késleltetés: $\frac{1}{0.1s} = 10Hz$ frekvenciaköz

Tehát minél nagyobb a késleltetés, annál több potenciális gerjedési frekvenciánk lesz adott frekvenciasávban.

A késleltetés nagysága a már létrejött gerjedés amplitúdójának növekedési sebességét is meghatározza [6]. Ha pl. 10 ms késleltetés van a mikrofon és a hangsugárzó között, az átvitel pedig 0.5 dB, az amplitúdó növekedési sebessége 0.5 dB / 10 ms, azaz 50 dB / s lesz, 100 ms ra növelve a késleltetést ez a növekedés 5 dB / s-ra csökken.

Az fenti esetben a spektrum csúcspontjai egyforma magasságúak, de ez a valóságban, konkrét akusztikai rendszerekben (pl. szoba, terem, utcai hangosítás stb.) természetesen nem így van. A rendszer átvitelébe például egy aluláteresztő szűrőt téve az átvitel csúcsai különböző magasságúak lesznek, ez teszi lehetővé számunkra, hogy a legmagasabb csúcsok frekvenciáján csillapítva a rendszer leadott hangteljesítményét növelhessük. Amíg a csúcsok nem érik el az egységnyit, gerjedés nem fog fellépni.

2. A visszacsatolás csökkentésének lehetőségei

Ha beleavatkozunk az akusztikai rendszer átviteli karakterisztikájába, megfelelően módosítjuk annak hurokerősítését, a gerjedékenységet is csökkenthetjük. Az átvitelt a rendszer minden komponense meghatározza, így a mikrofon, keverőpult, erősítő, hangsugárzó, valamint a mikrofon és a hangsugárzó egymáshoz viszonyított helyzete is. Dolgozatunkban egy olyan digitális jelfeldolgozó rendszer megvalósítási lehetőségeit vizsgáljuk, melyet a keverőpult és az erősítő közé vagy a keverőpult egy csatornájának effekthurkába (insert-jébe) helyezve igyekszik a gerjedést minimalizálni a maximális gerjedésmentes erősítés elérése érdekében.

2.1. A visszacsatolás varianciája

Mielőtt a visszacsatolást mérséklő lehetőségekre részletesen kitérnénk, meg kell különböztessük azok két csoportját [8]. A hangosítási szituációk egy részében feltételezhetjük, hogy a hangsugárzó és a mikrofon közötti csatolás időinvariáns. Ez az eset akkor áll fenn, ha a műsor alatt a hangsugárzó és a mikrofon egymáshoz viszonyított helyzete, illetve a közöttük lévő közeg változatlan, valamint a zárt hurok többi eleme sem változik. Nem változtatjuk jelentősen a közeg tereptárgyait, a hurok többi elemét nem állítjuk (pl. keverő, hangszínszabályozó), a levegő hőmérséklete, páratartalma is állandónak tekinthető. Ezen esetben elegendő a gerjedéscsökkentő rendszerünket a műsor előtt konfigurálni, paramétereit a műsor folyamán változatlanul hagyni. Az ilyen rendszereket off-line gerjedésgátló rendszereknek nevezzük.

Az esetek jelentős részében viszont nem biztosíthatjuk a fenti invarianciát. Gondoljunk csak egy, a kezében mikrofonnal fel-alá járkáló előadóra, énekesre, vagy egy olyan színpadra, ahol a tereptárgyak, előadók helyzete folyamatosan változik, így a visszacsatoló közeg, azaz a visszacsatolás is. Ilyen szituációban egy, a műsor előtti átvitel alapján elvégzett konfiguráció elégtelen. A műsor alatt változó körülményekhez folyamatosan alkalmazkodó rendszerre van szükség. Az ilyen rendszereket off-line gerjedésgátló rendszereknek nevezzük.

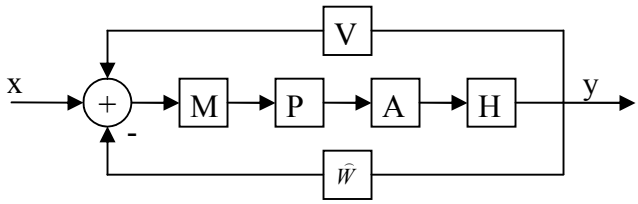
A gyakorlatban feltételezhetjük, hogy egy adott akusztikai rendszer gerjedési frekvenciáinak egy része változatlan, a mikrofon és hangsugárzó relatív pozíciójától független, az adott akusztikai térre jellemző, mindemellett műsor közben a visszacsatolás dinamikusan változik. Ez indokolja az off-line és on-line rendszerek kombinálását.

2.2. Off-line gerjedésgátló rendszerek

A következőkben olyan rendszer megvalósítási lehetőségeit tárgyaljuk, melyek invariáns visszacsatolást feltételeznek. A gerjedés megszüntetésére, illetve kialakulásának megelőzésére több módszer is lehetőséget nyújt, melyek működésének elméleti hátterét ebben a fejezetben részletesen tárgyaljuk.

2.2.1. Az akusztikai visszacsatolás kioltása ellenfázisú visszacsatolás megvalósításával

A jelterjedési modellt figyelembe véve magától értetődő az a törekvés, hogy a V visszacsatolást azon a helyen próbáljuk meg kioltani, ahol az a hatását kifejti, azaz a mikrofonnál. Ezt a kioltást szemlélteti a 4. ábra.



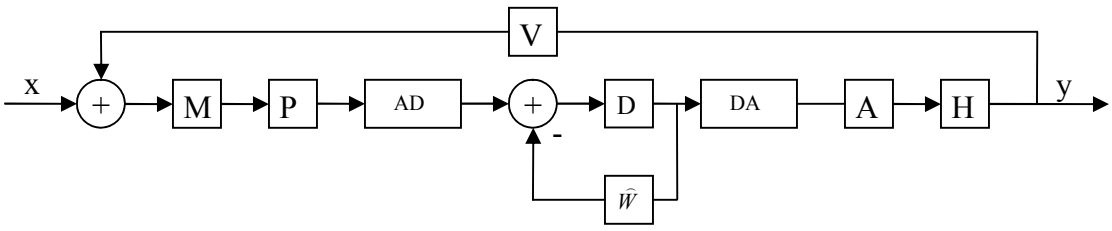
4. ábra. Az akusztikai visszacsatolás kioltásának rendszermodellje.

Világos, hogy ha tökéletes, visszacsatolásmentes rendszert szeretnénk, akkor a $\hat{W} = V$ feltétel kell hogy teljesüljön. Ehhez $\hat{W}$ ismeretén túl az lenne szükséges, hogy a kioltást az akusztikai térben valósítsuk meg, azaz egy a mikrofon helyére koncentrált hangsugárzóval kellene a visszacsatoláshoz képest ellenfázisú jelet kiadni. Ha a H átvitelű hangsugárzó által kibocsátott hang nem periodikus zaj, akkor gyakorlatilag lehetetlen ezt megoldani,mert nem becsülhető meg előre, hogy a mikrofon mögött álló beszélő mit fog mondani, vagy hogy a zenekar mit fog előadni.

A kioltás megvalósítására azonban nem csak az akusztikus térben van lehetőség, hanem a jelútba a mikrofon és a hangszóró között beavatkozva is. Ennek megvalósítására analóg és digitális módon is lehetőségünk van. Az analóg rendszer a bonyolultsága miatt nem szerencsés választás, igen nehéz implementációs feladat lenne.

Ha azonban egy DSP segítségével a mikrofon és a hangszóró között beavatkozunk a jelútba, akkor a digitális tartományban már igen jó lehetőségeink vannak mind a V akusztikai visszacsatolás becslésére, mind a kioltás megvalósítására. A mikrofonból érkező jelet digitalizáljuk, majd a kívánt műveletek elvégzése után (identifikáció, kioltás) a már processzált jelet D/A átalakítva vezéreljük a hangsugárzót.

A digitális eszközökkel megvalósított kioltás elvi blokkvázlata:

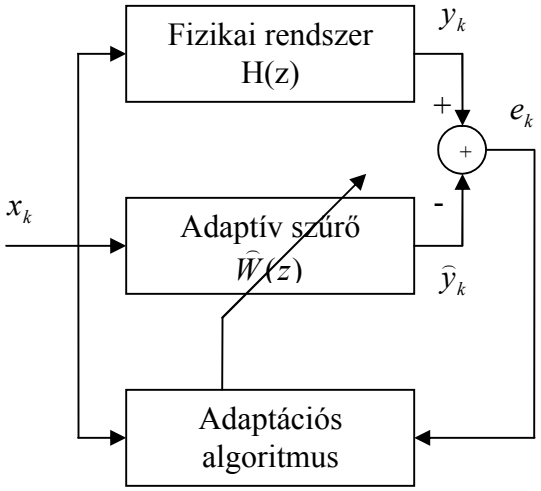


5. ábra. Az akusztikai visszacsatolás kioltása digitális tartományban.

Első lépésben arra van szükség, hogy a hangsugárzó és a mikrofon közötti átvitelt identifikáljuk, hogy aztán annak alapján a kioltást meg lehessen valósítani. $\hat{W}$-nek tartalmaznia kell az A/D és D/A átalakítók átvitelét, valamint a V akusztikai visszacsatolást. Az identifikáció megvalósítására sok lehetőség kínálkozik. Az adaptív szűrőkkel történő approximálás kedvező tulajdonsága, hogy anélkül vagyunk képesek modellt illeszteni egy fizikai rendszerhez, hogy előzetes ismeretekkel rendelkeznénk róla. A következő fejezetek az adaptív szűrők elméleti hátterét tárgyalják, egyes adaptív algoritmusok megismerésén keresztül.

2.2.1.1. Az LMS-algoritmus

Tegyük fel, hogy van egy H (z), általunk nem ismert átvitellel rendelkező fizikai rendszerünk és szeretnénk a rendszer átvitelét meghatározni, azaz a rendszert identifikálni spektrumanalizátor nélkül vagy anélkül, hogy adott frekvenciafelbontással a gerjesztés frekvenciáját növelve mérnénk a gerjesztés és a válasz hányadosát és ábrázolnánk. Erre egy alkalmas módszer a már említett, adaptív szűrőkkel történő approximáció. Ennek során egy approximáló szűrő (jelen esetben FIR) együtthatóit változtatjuk valamilyen algoritmus szerint. A 6. ábrán az LMS algoritmussal történő identifikáció modellje látható:



6. ábra. Az LMS-algoritmussal történő identifikáció modellje.

A W(z) digitális FIR szűrő együtthatóit minden lépésben (t_s mintavételi időpontonként) egy konvergenciaparaméterrel súlyozva frissítjük, annak függvényében, hogy mennyi az eltérés a fizikai rendszer y_k kimenete és az azt approximáló $\hat{y}_k$ jel között. Ez az LMS (Least Mean Squares), azaz a legkisebb négyzetes hibát biztosító algoritmus, mely a pillanatnyi hibát igyekszik minimalizálni a pillanatnyi hibagradiens alapján. A hiba négyzetes értéke ennek megfelelően [1]:

$$\hat{\varepsilon}_k = [y_k - \hat{y}_k]^2 = [y_k - w_k{}^T x_k]^2$$

a pillanatnyi gradiens:

$$\hat{\nabla}(n) = \frac{\partial \hat{\varepsilon}_k}{\partial w_k} = -2 \cdot [y_k - w^T{}_k \cdot x_k] \cdot x_k = -2 \cdot e_k \cdot x_k$$

$$e_k = [y_k - w_k{}^T x_k]$$

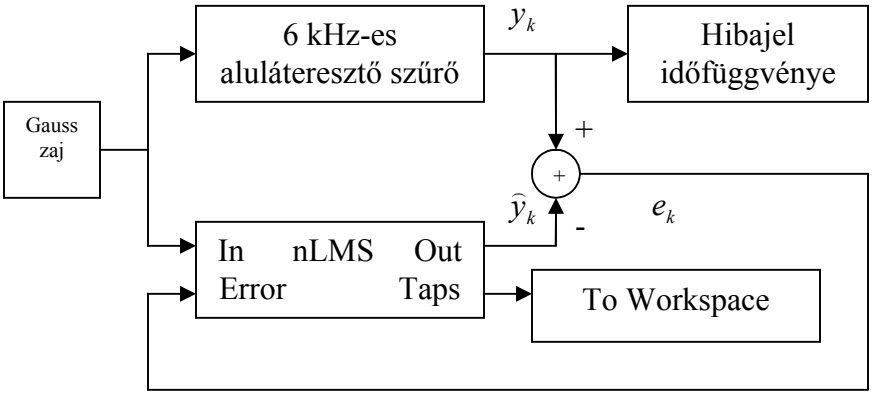
A W(z) digitális FIR szűrő együtthatóinak frissítése a következő egyenlet szerint történik:

$$w_k = w_k - \mu \cdot \hat{\nabla}_k = w_k + 2 \cdot \mu \cdot e_k \cdot x_k$$

A fenti egyenletben egyetlen változó szerepel, amelyről eddig nem esett szó, ez a μ konvergenciaparaméter (step size). A gyakorlatban ezzel az érékkel súlyozzuk, hogy az aktuális szűrőfrissítés mennyire változtassa meg az eddigi w_k együtthatókat. Az x_k gerjesztőjel fehérzaj, melynek sávszélességét a számunkra érdekes frekvenciatartomány határozza meg.

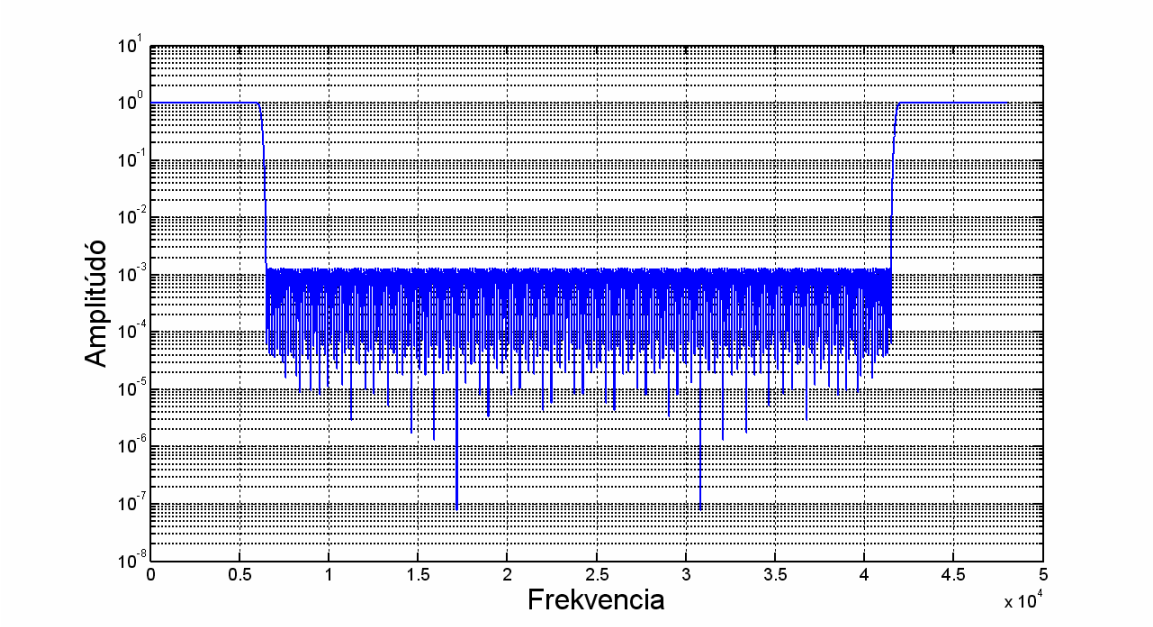
2.2.1.2. Az LMS-algoritmus szimulációja Matlab segítségével

Tegyük fel, hogy az identifikálandó fizikai rendszer egy 300 fokszámú aluláteresztő szűrő. A Matlabban több lehetőségünk van szűrőt tervezni, mi az fdatool nevű, grafikus felhasználói felülettel rendelkező programot használtuk. A megfelelő paraméterek megadása után a létrehozott szűrőt exportálva lehet azt az identifikálandó rendszer helyére beilleszteni.
A szűrő beillesztése után a Simulink elemeiből (azon belül is a DSP blocksetet felhasználva) állítottuk össze a szimulációs modellt, melyet 7. ábra mutat.



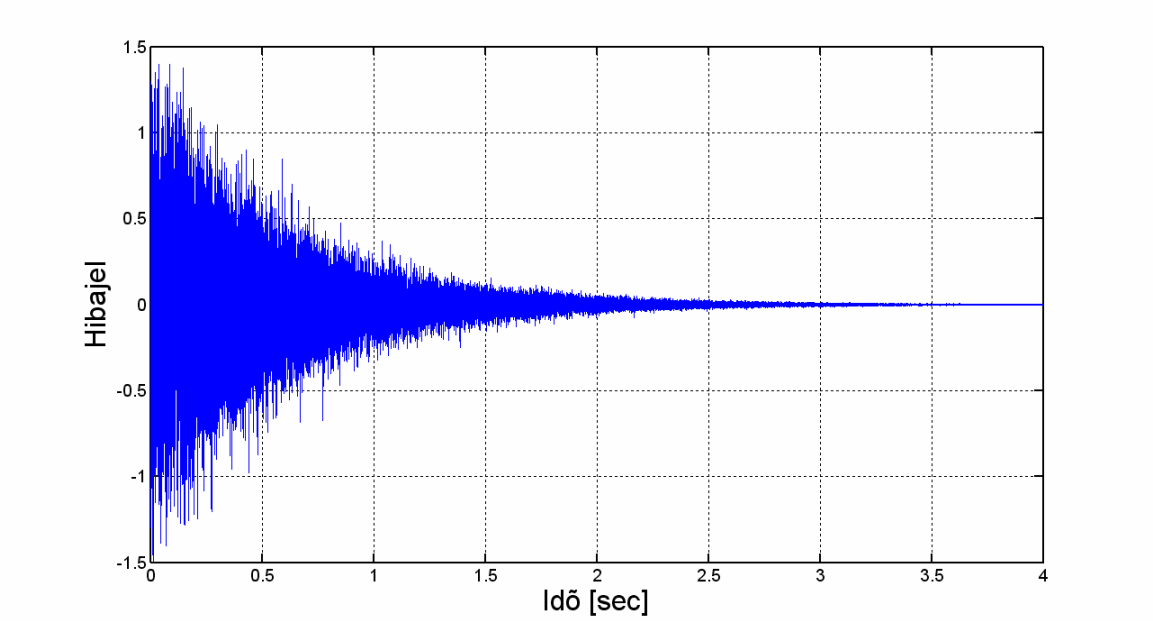
7. ábra. A Matlabban szimulált modell.

A tervezett identifikálandó szűrő átviteli karakterisztikája:



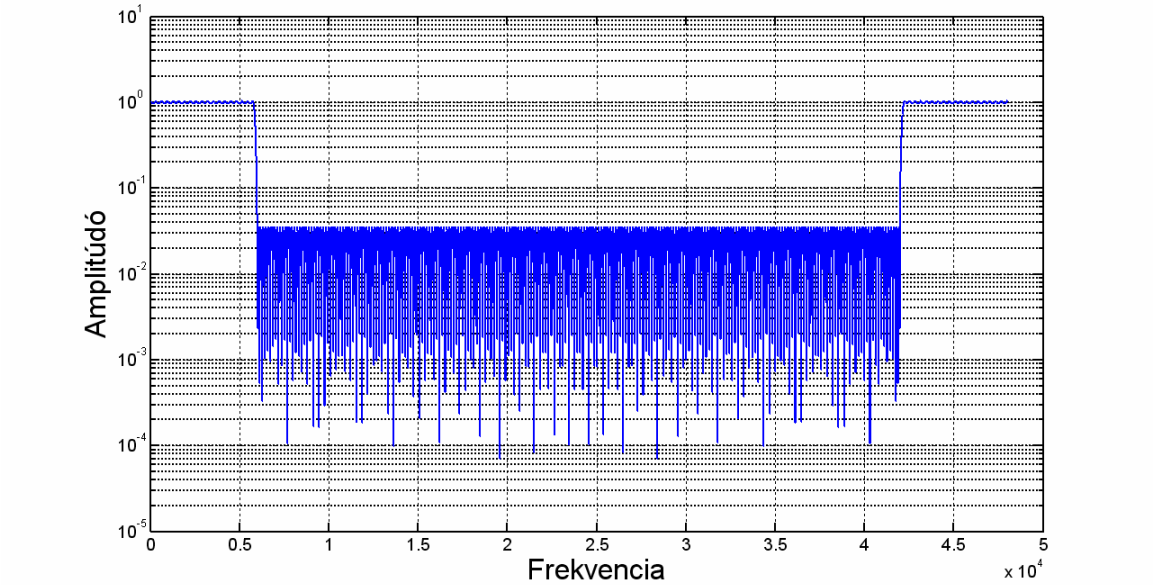
8. ábra. Az fdatool-lal tervezett szűrő átviteli karakterisztikája.

A szimulációt lefuttatva, a hibajelet egy kijelzőre (Time Scope) vezetve jól követhető a hibajel alakulása. Három és fél másodperc (azaz 48 kHz-es mintavételi frekvencia mellett 170.000 minta után) a hibajel jól láthatóan nem csökken tovább, 0.01-es konvergencia paraméter mellett. Így a szűrő együtthatóinak adaptálása a lehető legfinomabb mértékben történik, mert ─ mint ahogy azt az előbb láttuk ─ az új értékek nagysága nemcsak a konvergencia paramétertől függ, hanem a hiba nagyságától is. A hiba nullához konvergál, ezért a szűrőegyütthatók a szimuláció végén már nem változnak, illetve elhanyagolható mértékben ingadoznak egy „holtpont" körül (lásd 9. ábra). Ekkor az identifikációt befejezettnek tekinthetjük, mert nyilvánvaló, hogy kisebb hibát nem tudunk elérni. Matlab esetén az egyetlen korlátozó tényező a számábrázolási pontosság, ami 64 bit, azaz 10^{-16} az elérhető pontosság. Látni fogjuk, hogy a digitális jelfeldolgozó processzoron megvalósított algoritmus 16, illetve 32 bites számábrázolásra támaszkodhat és hogy ez milyen hártányokkal jár.



9. ábra. A hibajel időbeli lefutása a szimuláció során.

Ezek után a szűrő paramétereit a munkalapra kivezetve és azok FFT-jét képezve adódik az adaptált szűrő átviteli karakterisztikája a 10. ábrán:



10. ábra. Az identifikáció eredménye.

Az eremény meggyőző, a vágási frekvenciák Hertzre pontosan egybeesnek, bár az áteresztőtartományon kívüli elnyomás kisebb, mint az eredeti 60 decibel.

2.2.1.3. Adaptív IIR szűrők

Láttuk, hogy a FIR szűrők esetében az impulzusválasz hossza egyenesen arányos a fokszámmal, ami megszabja a rendszer korlátait a mikrofon – hangsugárzó távolság szempontjából. Kecsegtető gondolat rekurzív (IIR, Infinite Impule Response), azaz végtelen impulzusválaszú szűrők alkalmazása az identifikáció során. Azonos fokszám mellett a végtelen impulzusválaszú szűrők hatékonyabbak, illetve azonos teljesítményű szűrők közül az IIR szűrők kevesebb számítási kapacitást igényelnek, mert alacsonyabb a fokszámuk. Kedvező tulajdonságaik mellett azonban veszélyeket hordoz magával a módszer alkalmazása:

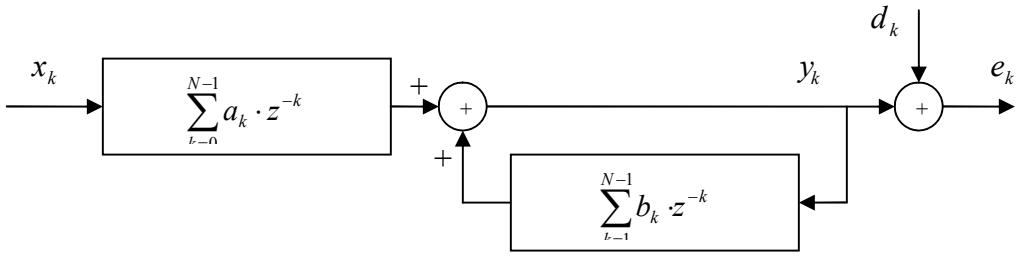
1. A visszacsatolt, pólussal és zérussal egyaránt rendelkező struktúra miatt stabilitási problémák léphetnek fel, ha az adaptáció során a szűrőegyütthatók olyan irányba változnak, hogy a pólusok kívül kerülnek az egységkörön.

2. A konvergencia szempontjából nem biztosított a globális minimum megtalálása, mert az átlagos négyzetes hiba nem négyzetes függvénye a szűrőegyütthatóknak (azaz nem feltétlenül szigorúan monoton). Elképzelhető olyan szituáció, amikor a hiba még nem elég kicsi ahhoz, hogy az identifikációt sikeresnek nyilvánítsuk, ugyanakkor nem is csökken tovább, mert a hibafüggvény lokális minimumából nem tud kikerülni az algoritmus. A konvergenciát erősen befolyásolják a szűrőegyütthatók kezdeti értékei, valamint a konvergenciaparaméter nagysága.

Az adaptáció megvalósítására rengeteg algoritmust dolgoztak ki, melyek elsősorban az imént említett két problémára nyújtanak megoldást. A 2.2.1.4. alfejezetben bemutatjuk a legegyszerűbb adaptív IIR szűrő struktúrát, majd a további fejezetekben ennek kettő, kedvezőbb konvergenciatulajdonságokkal rendelkező változatát.

2.2.1.4. Az IIR LMS algoritmus bemutatása

Az algoritmus nem más, mint az LMS-algoritmus kiterjesztése oly módon, hogy az már nem csupán a bejövő értékekből számítja az aktuális kimenetet, hanem a korábbi kimeneti értékek súlyozott összegét is figyelembe veszi [2]. A rekurzív struktúra a 11. ábrán figyelhető meg.



11. ábra. Rekurzív adaptív szűrő.

A 11. ábrán látható modell átvitele:

$$y_k = \sum_{n=0}^{N-1} a_n \cdot x_{k-n} + \sum_{n=1}^{N-1} b_n \cdot y_{k-n}$$

A z - tartományban ez a következőképpen írható le:

$$H(z) = \frac{A(z)}{B(z)} = \frac{\sum_{k=0}^{N-1} a_k \cdot z^{-k}}{1 - \sum_{k=1}^{N-1} b_k \cdot z^{-k}}$$

Vezessük be a következő jelöléseket:

$$W_k = [a_{0k}, a_{1k}, \ldots . a_{(N-1)k}, b_{1k}, \ldots \ldots b_{(N-1)k},]$$

$$U_k = [x_k, x_{k-1}, \ldots . x_{k-N+1}, y_{k-1}, \ldots \ldots . y_{k-N+1}]$$

A kívánt jel legyen d_k. Így a hiba a következőképpen írható fel:

$$\hat{\varepsilon}_k = d_k - y_k = d_k - W_k^T \cdot U_k$$

Mindeddig az algoritmus nem különbözik a nem rekurzív esettől, leszámítva azt, hogy az U_k és a W_k vektorok nemcsak bemeneti, hanem kimeneti változókat, illetve azok együtthatóit is tartalmazzák. A gradiensképpzés itt is ugyanúgy megtalálható:

$$\hat{\nabla}_k = \frac{\partial \varepsilon_k^{\,2}}{\partial W_k} = 2 \cdot \varepsilon \cdot \frac{\partial \varepsilon_k}{\partial W_k}$$

$$= 2 \cdot \varepsilon_k \cdot \left[\frac{\partial \varepsilon_k}{\partial a_{0k}} \ldots \frac{\partial \varepsilon_k}{\partial a_{(N-1)k}}, \frac{\partial \varepsilon_k}{\partial b_{1k}} \ldots \frac{\partial \varepsilon_k}{\partial b_{(N-1)k}} \right]^T$$

$$= -2 \cdot \varepsilon_k \cdot \left[\frac{\partial y_k}{\partial a_{0k}} \ldots \frac{\partial y_k}{\partial a_{(N-1)k}}, \frac{\partial y_k}{\partial b_{1k}} \ldots \frac{\partial y_k}{\partial b_{(N-1)k}} \right]^T$$

Az előző egyenlet alapján rekurzív módon definiálható α_{nk}, valamint β_{nk}:

$$\alpha_{nk} \hat{=} \frac{\partial y_k}{\partial a_n} = x_{k-n} + \sum_{l=1}^{N-1} b_l \cdot \frac{\partial y_{k-l}}{\partial a_n}$$
$$= x_{k-n} + \sum_{l=1}^{N-1} b_l \cdot \alpha_{n,k-l}$$
$$\beta_{nk} \hat{=} \frac{\partial y_k}{\partial b_n} = y_{k-n} + \sum_{l=1}^{N-1} b_l \cdot \frac{\partial y_{k-l}}{\partial b_n}$$
$$= y_{k-n} + \sum_{l=1}^{N-1} b_l \cdot \beta_{n,k-l}$$

Ezeket felhasználva a gradiens felírása a következőképpen alakul:

$$\hat{\nabla}_k = -2 \cdot \varepsilon_k \left[\alpha_{0k} \ldots \alpha_{(N-1)k}, \beta_{1k} \ldots \beta_{(N-1)k}\right]^T$$

Hasonlóan az LMS-algoritmushoz, a szűrőegyütthatók adaptációja:

$$W_{k+1} = W_k - M \cdot \hat{\nabla}_k$$

azzal a különbséggel, hogy ott az egyetlen konvergenciaparamétert itt felváltotta egy konvergenciaparaméter-mátrix, mely értelemszerűen egy diagonálmátrix, a művelet elvégezhetőségének érdekében. Ennek a mátrixnak az átlója tartalmazza a szükséges együtthatókat.

Meg kell jegyeznünk, hogy egyes szakirodalmak ezt az algoritmust RLS-algoritmusként említik [3].

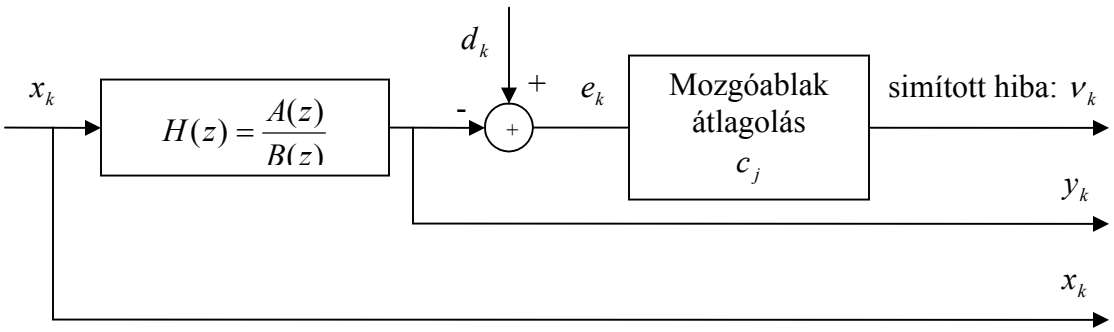
2.2.1.5. A SHARF algoritmus bemutatása

Az előző fejezetben bemutatott algoritmus nem biztosítja számunkra a globális konvergenciát. A HARF (Hyperstable Adaptive Recursuve Filter [3]) „algoritmuscsalád” volt az első, adaptív IIR szűréshez használt algoritmus, mely bizonyíthatóan konvergens. Legegyszerűbb képviselője a SHARF (Simplified Hyperstable Adaptive Recursuve Filter [3]). Tekintsük az előző algoritmusban szereplő α_{nk} valamint β_{nk} tagokat és közelítsük őket a kifejezések első részével:

$$\alpha_{nk} \hat{=} x_{k-n}$$

$$\beta_{nk} = y_{k-n}$$

A másik lényeges különbség, hogy a $\hat{\nabla}$ gradiens képzésekor a nem az eddigi $\hat{\varepsilon}_k = d_k - y_k$ hibajellel dolgozunk, hanem annak egy simított értékével, $v - vel$. A SHARF algoritmus struktúrája és lépései:



12. ábra. A SHARF algoritmus rendszermodellje.

$$y_k = W^T \cdot U_k$$

$$\varepsilon_k = d_k - y_k$$

$$\nu_k = \varepsilon_k + \sum_{n=0}^{L} c_n \cdot \varepsilon_{k-n}$$

$$\widehat{\nabla}_k = -2 \cdot \left[x_k, x_{k-1}, \ldots . x_{k-N+1}, y_{k-1}, \ldots \ldots . y_{k-N+1} \right]^T$$

$$W_{k+1} = W_k - M \cdot \widehat{\nabla}_k$$

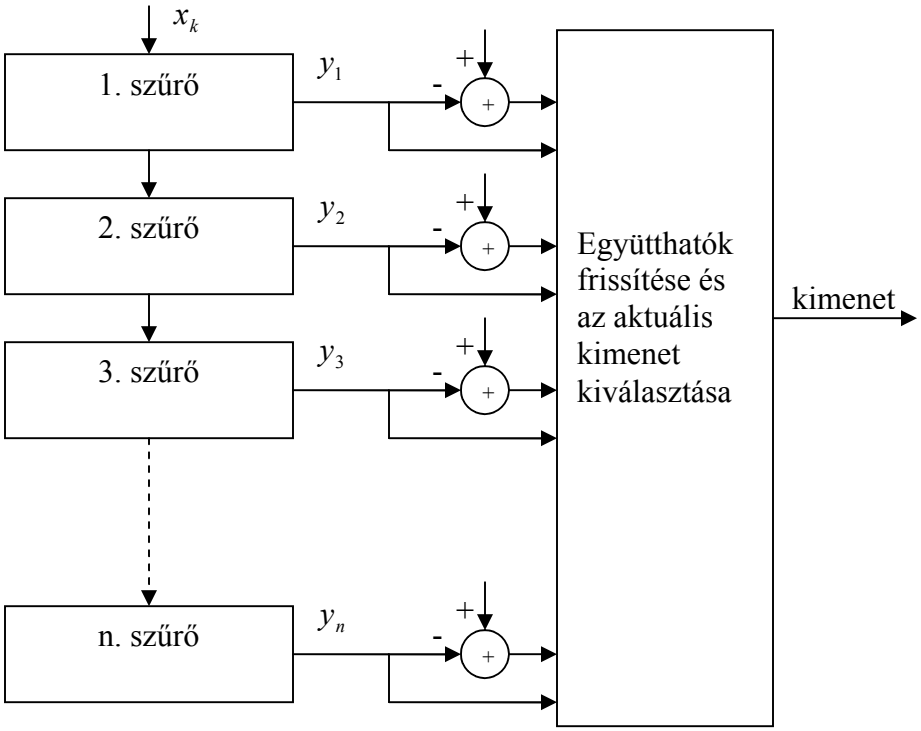
Ahhoz, hogy az algoritmus feltételezései teljesüljenek,a konvergenciaparamétereknek kellően kicsinek kell lenniük, továbbá a c simítóvektorra is vannak kikötések. Definiáljuk a $G(z)$ átvitelt a következő módon:

$$G(z) = \frac{C(z)}{B(z)} = \frac{\left(1 + \sum_{n=0}^{L} c_n \cdot z^{-n}\right)}{\left(1 - \sum_{k=1}^{N-1} b_k \cdot z^{-k}\right)}$$

Ahol a c_n együtthatók a hiba simítóvektorával egyeznek meg, a b_k együtthatók pedig a visszacsatoló ág együtthatói. Ennek a törtnek kell teljesítenie az SPR (Strictly Positive Real) feltételt, azaz $\mathrm{Re}\{G(z)\} > 0$ minden |z|=1 esetre, így a gyökhelygörbe egészen biztosan nem tartalmazhatja a (-1,0) koordinátájú pontot, azaz stabil. A c_n együtthatókat ennek a feltételnek megfelelően kell megválasztani, melyre több lehetőségünk is adódik [3].

2.2.1.6. Az IIR LMS algoritmus egy továbbfejlesztett verziója

A [4] irodalomban a szerzők a fent tárgyalt „hagyományos” IIR LMS algoritmust egészítik ki a globális konvergencia gyorsabb és megbízhatóbb elérésének érdekében. Az alapelgondolás az, hogy az adaptív IIR szűrők konvergenciáját és stabilitását alapvetően befolyásolják a szűrőegyütthatók kezdeti értékei. A kiegészítés abból áll, hogy egyszerre több szűrőt ($\Theta_1....\Theta_n$) valósítanak meg, melyeknek mind különböző a kezdeti együtthatóvektoruk, amit iterációnként frissítenek. Az IIR LMS algoritmust mindegyikükre külön-külön alkalmazzák, azonos bemeneti jellel és approximálni kívánt jellel, ahogy azt a 13. ábrán is láthatjuk:



13. ábra. A párhuzamos IIR LMS struktúra.

Minden egyes megvalósított szűrőre külön számolja a négyzetes hibát és amelyikre aktuálisan a legkisebb, azt továbbítja a rendszer kimenete felé. Mielőtt a szűrőegyütthatókat frissítenék, stabilitásvizsgálatot végeznek, és ha valamelyik szűrő instabillá válik, vagy a konvergencia sebessége egy kritikus érték alá csökken, akkor egy evolúciós számítási elv segítségével véletlenszerűen perturbálják az előző stabil állapot együtthatóit, majd ezután ismét stabilitásvizsgálat következik. Ha ennek a során is instabilnak bizonyul a rendszer, akkor végleg az előző stabil rendszer együtthatóit örököljük. Ezt akár többször is el lehet végezni iterációnként, ha van rá processzoridő. A párhuzamos konfiguráció mindig megbízható kimenetet eredményez, még akkor is, ha az egyik szűrő nem robusztus. Az algoritmusban az evolúciós számítási elvet alapvetően stabilitásvizsgálatra használják.

A módszer egyetlen hátránya, hogy a számításikapacitás-igény a szűrők számával lineárisan nő. Alacsony mintavételi frekvencia mellett egy ilyen algoritmus akár egyetlen processzoron is elfuthat, de akár multiprocesszoros rendszert is el lehet képzelni, vagy több azonos hardvert, melyeken függetlenül fut az identifikáció, hálózatba vannak kötve és kommunikálnak egymással vagy egy „szerver” hardverrel.

2.2.2. Egy off-line, lyukszűrőket felhasználó rendszer

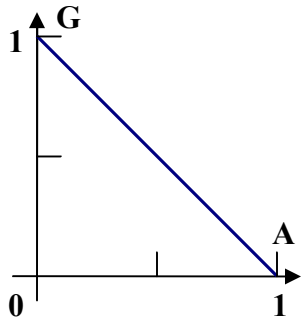
A rendszer működésének alapja a következő. Ahogy a fentiekben is tárgyaltuk, a visszacsatolt hurok átvitele a csúcsokkal tarkított. Ha a rendszer erősítését növeljük, az átvitel az amplitúdó tengely mentén pozitív irányba tolódik. Gerjedés akkor alakul ki, ha a visszacsatolt hurok átvitelének csúcsai közül valamelyik erősítése eléri az egységnyit, valamint fázisa 360° egész számú többszöröse lesz. Ha a legmagasabb csúcsok frekvenciáin nagyon keskeny szűrővel csillapítjuk az átvitelt, mivel a csúcsok nem egyforma magasak, távolabb kerülünk az egységnyi hurokerősítéstől, nagyobb hangteljesítményt adhatunk le gerjedés nélkül [8]. Keskeny szűrők (lyukszűrők) alkalmazásával elérhető, hogy a hangminőség se romoljon jelentősen. Tapasztalatunk szerint 10 darab a jelútba beillesztett lyukszűrő nem rontja jelentősen a hangminőséget. A 10 legmagasabb csúcs frekvenciájának meghatározása, majd az ezen frekvenciákon alkalmazott lyukszűrők erősítésnövelést tesznek lehetővé.

A megfelelő hatás eléréséhez minél pontosabban kell meghatároznunk ezen csúcsok frekvenciáit. A hagyományos spektrumanalizátort használó megoldások itt nem lennének célravezetők egy ilyen rendszer nagy időállandója miatt, a mérés túl hosszadalmas lenne. Érdemesebb magát a jelenséget előidézni, azaz gerjedést létrehozni és mérni annak frekvenciáját.

Egy ilyen, a jelútba helyezett rendszer konfigurációja a következőképpen történhet. A hangosítás hangerejét addig növeljük, míg a hurokerősítés a fentieknek megfelelően úgy nem módosul, hogy a rendszer gerjedni kezd. Ekkor a hurok karakterisztikájának egy olyan csúcsát találtuk meg, a gerjedés frekvenciája egy olyan frekvencia, melynek fázisa 360° egész számú többszöröse. Megmérve a gerjedés frekvenciáját, a jelútban már elhelyezett, de egyelőre konfigurálatlan lyukszűrők közül egyet a mért frekvenciára hangolunk. Így a rendszer hurokerősítésének átviteléből kiiktatva egy csúcsot az erősítés tovább növelhető a következő gerjedésig, mely frekvenciáját megmérve egy további lyukszűrő paraméterezhető fel. Ezt az eljárást addig folytatjuk, míg az összes rendelkezésre álló lyukszűrőnket fel nem konfiguráltuk. A hurokerősítés átvitelének ilyen módosítása a maximális erősítés növekedéséhez vezet. Az így felparaméterezett lyukszűrőbankunk a műsor alatt végig üzemel.

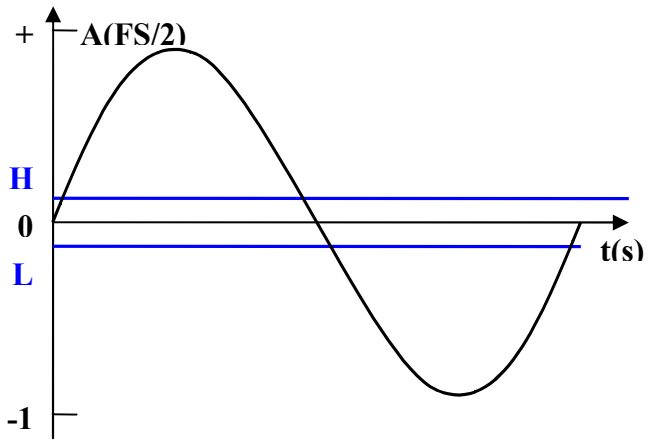
A frekvenciamérésnek azonban van egy nagy nehézsége. A gerjedés egy exponenciálisan növekvő amplitúdójú szinuszjel, mely rövid idő alatt telítésbe viheti a rendszert, ami a pontos frekvenciamérést lehetetlenné teszi, mindemellett a berendezést is tönkreteheti, a gerjedést stabilizálnunk kell.

A visszacsatolt rendszerünket egy oszcillátorként is felfoghatjuk. Az előrecsatoló ág a mikrofon – előerősítő – végfok – hangsugárzó, a visszacsatolás a csatolás a hangsugárzó és mikrofon között, a csatoló közeg a levegő. Állandó amplitúdójú oszcillálás az adott frekvencián egységnyi erősítésű hurok esetén áll fenn. Ezt úgy oldhatjuk meg, ha a rendszer figyeli a gerjedés amplitúdóját, és ha az túl nagy, csökkentjük a hurok erősítését, ha túl kicsi, növeljük azt. Így elérhető egy stabil, állandó amplitúdójú oszcilláció a zárt hurok átvitelének éppen legmagasabb amplitúdójú frekvenciáján. Ezen stabilizált szinuszjel frekvenciamérése megoldható. Az előfok és végfok közé beiktatott gerjedésgátló rendszer képes megvalósítani egy ilyen nemlineáris karakterisztikát. Több ilyen karakterisztika is létezik, pl. $\frac{1}{x}$, de mi a lenti, legegyszerűbben megvalósítható karakterisztika hatékonyságát is elégségesnek találtuk. A megvalósított karakterisztikát a 14. ábra mutatja:



14. ábra. A a mért amplitúdó, G a kimeneti erősítés

A frekvenciamérés a nullátmenetek számolásával és egységnyi időközönkénti osztással oldható meg. Érdemes 2 nullszinthez nem túl közeli komparálási szintet alkalmazni. Ez azért ajánlott, mert egy zajos jelnél csak egy komparálási szintet alkalmazva nullátmenet a kelleténél többször is történhet, emiatt a ténylegesnél nagyobb frekvenciát mérnénk. 2 komparálási szint esetén, ha az egyik szintet keresztezte a jel, de a másikat még nem, nem érdekes, hogy a zaj miatt esetleg többször tette azt, csak a keresztezés ténye fontos. Nullátmenet csak akkor regisztrálódik, ha a másik komparálási szinttel is kereszteződik a jel.



15. ábra. Nullátmenet mérése két komparálási szinttel. A az amplitúdó, FS/2 (Full Scale) dimenzióban, t az idő, H illetve L a két komparálási szint.

A frekvenciamérés így nagy pontossággal megoldható.

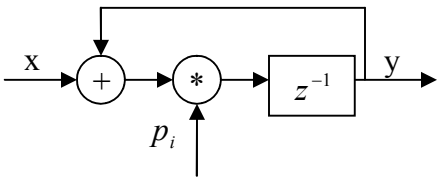
A lyukszűrőket felhasználó rendszer korlátja a lyukszűrők okozta hangminőség-romlás, mely zenekari hangosításnál kritikus lehet, bár egy élő koncert esetén nem lehet stúdióminőség a cél. Meg kell kötni a megfelelő kompromisszumot az elérhető maximális gerjedésmentes erősítés és a hangminőség között.

2.2.2.1. A lyukszűrőkről

A lyukszűrő bankunkat egy úgynevezett rezonátoros struktúrával valósítottuk meg [9]. Ennek alapeleme a rezonátor melynek diszkrét idejű átvitele:

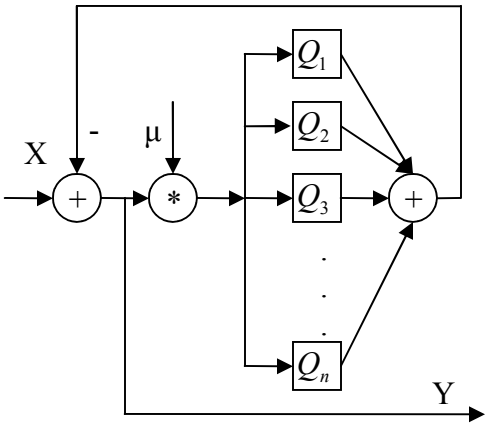
$$Q_i(z) = \frac{p_i}{z - p_i}$$

A rezonátor egy olyan lineáris rendszer, melynek egy pólusa van, mégpedig az egységkörön. Diszkrét idejű megvalósítása lehet a következő:



16. ábra. Egy rezonátor

A rezonátorok a stabilitás határhelyzetében működnek, de a rendszer a visszacsatolásnak köszönhetően stabil. A 17. ábrán P egy-egy rezonátort jelöl.

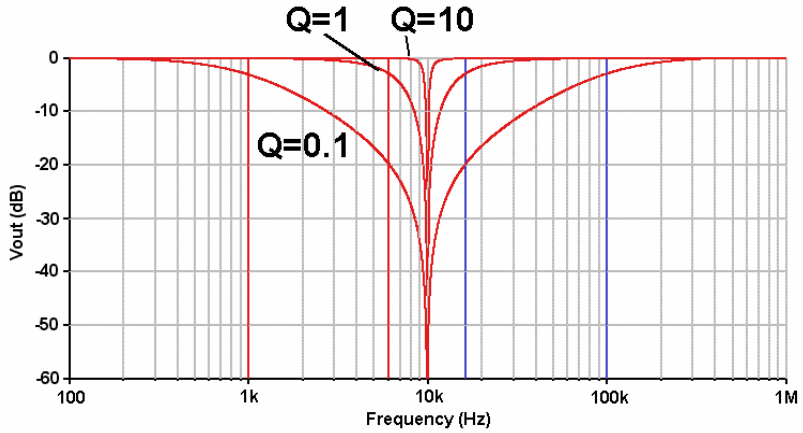


17. ábra. A rezonátorstruktúrával megvalósított szűrőbank.

A rendszer eredő átvitele:

$$E(z) = \frac{1}{1 + \alpha \Sigma_{i=1}^{N} Q_i(z)}$$

Az eredő átvitelnek a rezonátorpozícióban vannak zérusai. A rezonátorstruktúrát felépítő rezonátorok pólusai meghatározzák a lyukszűrők vágási frekvenciáit. Tehát ha lyukszűrőbankunk vágási frekvenciáit akarjuk állítani, az egyes rezonátorok pólusait kell pozícionálni. A pólusok fázisa a mintavételi frekvencia függvényében meghatározzák a vágási frekvenciákat. A 2π fázis felel meg a mintavételi frekvenciának, tehát ha a mintavételi frekvencia negyedénél szeretnénk lyukszűrést, az egyik rezonátor pólusát $\pi/2$ fázisúra kell állítanunk.
A fázistól független működés érdekében a rezonátorstruktúra átvitelében konjugált komplex zérus párokra van szükség. Mivel a kimenet képzésekor a képzetes rész kiesik és a valós rész kétszereződik, elegendő csak a konjugált komplex pár egyik tagját megvalósítani, venni az egész rész kétszeresét és azt visszacsatolni.
A hibaképzés utáni μ szorzótényező exponenciális átlagolást visz a körbe, mely a lyukszűrő frekvenciaszelektivitását nagymértékben megnöveli.



18. ábra. Egy lyukszűrő Q függvényében. A Q szűrő minőségi faktora, mely a középfrekvencia és a sávszélesség hányadosaként van definiálva. [7]

Egy lukszűrő tulajdonságait tehát 3 változó határozza meg. A pólusok meghatározzák a szűrés frekvenciáját. A lyukszűrő mélységét a pólus egységkörtől való távolsága határozza meg. Az egységkörön elhelyezkedő pólus -∞ dB erősítést jelent az adott frekvencián, távolodva egységkörtől a vágás mérséklődik. A μ szorzótényezővel a lyukszűrő szélességét állíthatjuk, mivel azonban a szűrő szélességét valamelyest a pólus helyzete is befolyásolja, a szűrő megtervezésekor a két változó kölcsönhatását figyelembe kell venni.

2.3. On-line gerjedésgátló rendszerek

A következőkben olyan gerjedésgátló rendszerek megvalósítási lehetőségeit tárgyaljuk, melyek variáns visszacsatolást feltételeznek.

2. 3. 1. Egy on-line, lyukszűrőket felhasználó rendszer

A off-line rendszerek legnagyobb hátránya, hogy feltételezik az akusztikai rendszer statikus voltát. Egy dinamikusan változó visszacsatolású rendszer hurokerősítésének átvitele dinamikusan változik, bármikor megjelenhet egy olyan frekvenciájú csúcs, amely erősítése meghaladja az egységnyit, a rendszer ekkor gerjedni kezd. A következőkben tárgyalt on-line rendszer az off-line lyukszűrőket felhasználó rendszerhez hasonlóan szintén nem a hangosító rendszer átviteli karakterisztikájából indul ki, hanem a gerjedés tényét igyekszik detektálni [8].

A gerjedés létét és annak frekvenciáját egy ilyen dinamikus rendszerhez alkalmazkodó, a hangképet nem befolyásoló módon kell megállapítani, digitális jelfeldolgozó processzoron implementálható módon. A visszacsatolási hurkon és így a jelfeldolgozó kártyán is áthaladó jel diszkrét Fourirer transzformáltját bizonyos időközönként képezve megfigyelhetjük a jel spektrumának alakulását. A véges rendelkezésre álló számítási kapacitás és az alkalmazás időkritikus volta indokolja a gyors Fourier transzformáció (Fast Fourier Transformation: FFT) alkalmazását. Ez annyi megkötést jelent számunkra, hogy a transzformáló függvény bemeneti vektora és a kapott – a bemenetivel megegyező méretű - kimenti vektor mérete 2 hatványa kell legyen. A transzformált vektor megadja a bemeneti vektor által reprezentált diszkrét jel diszkrét spektrumát. A kapott (diszkrét) spektrum frekvenciafelbontását az FFT pontszáma (így a bemeneti vektor elemszáma) határozza meg. A diszkrét spektrum elemeit tartalmazó vektor tagjai egy-egy frekvenciaintervallum energiáját reprezentálják.

Az algoritmus azon a feltevésen alapul, hogy egy műsor spektruma dinamikusan változik, a gerjedés viszont egy exponenciálisan növekvő amplitúdójú szinuszjel. Ezek szerint azon frekvenciasáv teljesítménye, mely a gerjedést tartalmazza, folyamatosan növekszik, ellenben a hangosított műsor diszkrét spektrumának egy adott összetevője nem monoton növekvő. Az egymás után következő transzformációk megegyező indexű, azaz ugyanazon frekvenciasávjait vizsgáljuk, és ha a tömb egy tagja egy bizonyos időkorláton

túl is monoton növekvő, az adott frekvenciasávban gerjedés kell legyen. Tapasztalataink szerint 0,25 – 0,5 s időkorlát a detektáláshoz elégséges.

A gerjedőnek nyilvánított frekvenciasáv tömbindexéből annak középfrekvenciája kiszámolható, ezzel a szűrőbank egyik rezonátorának pólusa is. Mivel a szűrők helye az átvitel dinamikus volta miatt folyamatosan változik, a pólusokat egy cirkuláris pufferben tároljuk, új gerjedés detektálásakor mindig a legrégebbi pólust írjuk felül. A rendszer alapja tehát a folyamatos FFT képzés, és az így kapott egymás után következő diszkrét spektrumok vizsgálata.

A fenti rendszer működőképes, viszont a hangminőséggel adódhatnak problémák. Ugyanis ha 1024 pontos FFT használunk, spektrumunk felbontása körülbelül $\frac{48000Hz}{1024} = 48Hz$ lesz, a gerjedés detektálásának ez a pontossága. Mivel a frekvenciatartomány bármely frekvenciáján lehet a gerjedés, az ide elhelyezendő lyukszűrő az egész tartományban elégséges vágást kell biztosítson. Ennélfogva a kioltáshoz használandó lyukszűrő szélessége legalább 48 Hz kell legyen, valamint érdemes olyan szűrőt tervezni, mely nem végtelen vágású egy bizonyos frekvencián, hanem kisebb csillapítású, de ezen csillapítás közel azonos az egész frekvenciatartományban. A pólust nem kell az egységkörön elhelyezni, a μ állandó segítségével pedig a frekvenciatartományt lefedő szélességű szűrőt kell létrehozni.

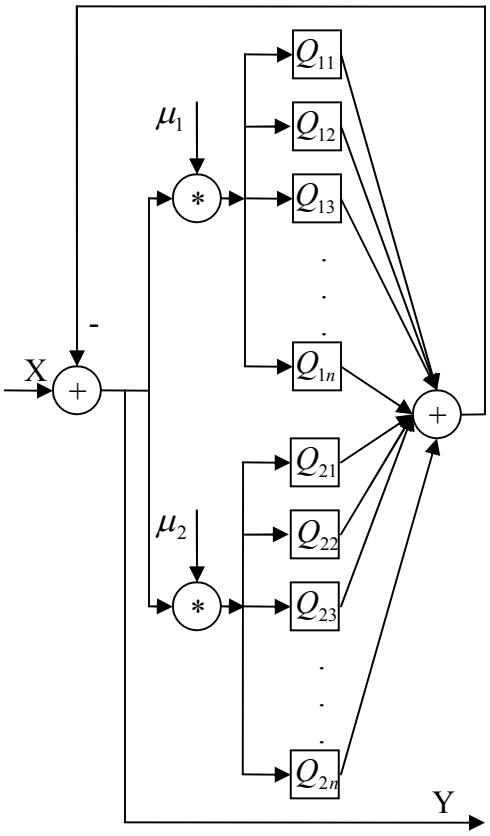
Mivel az emberi hallás egy hanganyag frekvenciatartományaira a frekvenciával logaritmikusan érzékeny, szubjektív módon egy alacsony frekvenciás tartományban elhelyezett például 50 Hz szélességű lyukszűrő sokkal jobban befolyásolja a hangképet, mint magas frekvencián. Tehát ezen lyukszűrő 200 Hz környékére paraméterezve egy a műsort hallgató személy számára sokkal nagyobb minőségromlást jelent, mint ugyanezen szűrő 10 kHz környékén. Tapasztalatunk szerint viszont egy 5 Hz széles lyukszűrő már nem okoz túlzott minőségromlást az alsó frekvenciatartományokban, elérendő célnak ezt a felbontást tűztük ki.

A felbontás növelésének egyik módja az FFT pontszámának növelése, viszont a jelfeldolgozó processzor véges mennyiségű rendelkezésre álló memóriája korlátozó tényező. A felbontás növelésének lehetőségeit kutatva kiindulhatunk abból is, hogy mindezt az alacsony frekvenciás összetevők miatt tesszük. A szűk keresztmetszet tehát a műsor spektrumának alacsony frekvenciás része, a lyukszűrők szélessége itt a legkritikusabb. Konstruálhatnánk egy olyan rendszert, mely frekvenciafelbontása alacsony frekvencián nagy, magasabb frekvenciákon alacsony. Alacsony frekvencián a nagyobb felbontásnak köszönhetően keskeny lyukszűrőket alkalmazhatunk, ezzel a hangminőséget nem rontjuk le jelentősen.

Ennek megoldási módja a következő lehet. Ha az alacsony frekvenciás tartományban nagyobb felbontást kívánunk elérni, pl. 4,8 kHz-et, alkalmazhatunk egy decimálást, azaz csak minden tizedik mintát veszünk figyelembe a bejövő minták közül, így mivel a bemeneti minták mintavételi frekvenciája 48 kHz, a decimált jel mintavételi frekvenciája 4,8 kHz lesz. Ezen jel sávszélessége tehát 2,4 kHz lesz. Itt megjegyezzük, hogy

decimálás a mintavételi frekvencia utólagos csökkentése, és természetesen nem csak tizedelés lehetséges. A spektrumátlapolódás elkerülése érdekében a decimálást megelőzően egy aluláteresztő szűrést is alkalmazni kell 2,4 kHz vágási frekvenciával. A szűrő-decimáló együttes a decimáló szűrő. Ezen mintákból egy 1024 pontos FFT képezve a frekvenciafelbontás 4.8 Hz körüli lesz, tehát 2,4 kHz frekvenciáig ezzel a pontossággal számolhatunk. Az alacsony frekvenciás felbontás ezen elgondolás alapján tehát növelhető. Mindeközben képezve a decimálatlan jel transzformáltját, a magasabb frekvenciás összetevők felbontása 48 Hz marad.

A gondolatmenet persze finomítható, további megfontolások tehetőek. Például nem muszáj pont tizedére csökkenteni a frekvenciát, a két frekvenciatartományban különböző lehet az FFT pontszáma is, több különböző felbontású frekvenciatartományt is használhatunk. A megvalósítás a szűrőbank módosítását is megköveteli. A különböző felbontású frekvenciatartományok különböző szélességű szűrők használatát teszik szükségessé. (Végső soron pont ez volt a célunk.) Ennek érdekében a fentebb tárgyalt rezonátoros struktúrát a következőképp módosítottuk.



19. ábra. Két, különböző szelektivitású lyukszűrő-tömböt tartalmazó rezonátorstruktúra

A hibaképzés utáni μ_1 és μ_2 szorzótényezőket különbözőre választva a rezonátorstruktúra két különböző szelektivitású lyukszűrőket tartalmazó részre bontható, a különböző szélességű szűrők problémája ezzel megoldott. A lyukszűrők számát itt is a

hangminőség korlátozza. A két frekvenciatartományban érdemes a tartományok átlagos energiája szerint elosztani a lyukszűrőket, mivel ez a gerjedések frekvenciáinak eloszlására is hatással van.

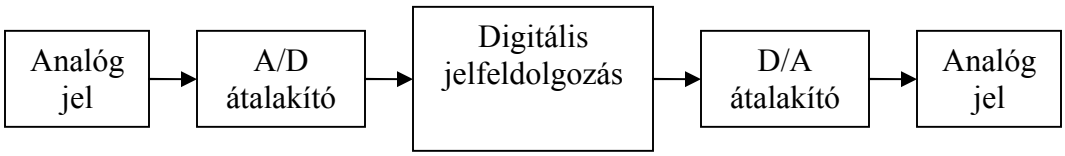
3. Az implementáció eszközei

3.1. Bevezetés

A feladat megvalósítása során felhasznált eszközök bemutatása elengedhetetlen, mert az implementáció során rengeteg olyan problémával találkoztunk, melyek feltárása csak úgy lehetséges, ha előbb tisztázzuk a mögöttük álló hardveres és szoftveres komponensek felépítését, működését.

3.2. A jelfeldolgozó processzorokról

Az analóg világból érkező jeleket A/D átalakítókkal mintavételezve, majd kvantálva a mért és feldolgozandó jelek számokkal reprezentálhatóak lesznek, így mikroprocesszorral és egy rajta futható szoftverrel feldolgozhatóak. A kívánt kimenet D/A konverterrel kerül a digitális környezetből vissza az analóg világba. Ezt a műveletsort szemlélteti a 20. ábra.



20. ábra: a digitális jelfeldolgozás folyamatábrája.

A digitális jelfeldolgozó processzor (digital signal processor, DSP) egy speciális mikroprocesszor, melyet digitális jelfeldolgozásra terveztek, főleg valós idejű számításokhoz. Egy DSP sok tekintetben megegyezik egy hagyományos mikroprocesszorral, viszont utasításkészlete ki van egészítve a diszkrét jeleken végezhető műveletek meggyorsítása érdekében, valamint több műveletvégző egységet is tartalmazhat, melyek egymással párhuzamosan is működhetnek. Mindemellett az adat- és programmemória is szét van választva (Harvard-architektúra), így azok párhuzamosan is elérhetőek, ami további teljesítménynövekedést eredményez. Egyes processzorokban az adatmemória is több részre van osztva, így a különböző részek párhuzamosan érhetőek el. Egy órajelciklus alatt kiolvashatunk két operandust, összeszorozhatjuk őket és már ugyanabban a ciklusban hozzáadhatjuk (akkumulálhatjuk) őket egy harmadik számhoz. Ezen párhuzamos működés sok szituációban, például a FIR szűrők esetében, ahol szintén diszkrét konvolúciót kell megvalósítani, nagy segítséget jelent. Jelfeldolgozást természetesen hagyományos processzorokon is végezhetünk, de ott kisebb hatásfokkal.

Az DSP-k aritmetikája az összeadás, léptetés, műveleteken felül a szorzást is elvégzik egy órajelciklus alatt, egyes esetekben korlátozott pontosságú osztás is támogatott. A memória direkt és indirekt címzése is támogatott. A processzorok az indirekt címzés

számára külön regisztereket biztosítanak, melyeket a címaritmetika használ fel az egyszerűbb és legfőképp gyorsabb programok létrehozása érdekében. Lehetséges a címek inkrementálása, dekrementálása, hozzárendelt ofszetregiszter segítségével 1-nél nagyobb címmel is. Hozzárendelt regiszterek segítségével cirkuláris buffer is létrehozható. A cirkuláris buffer egy olyan memóriaterület, amelynek elemeit ciklikusan olvassuk ki vagy ciklikusan írunk bele. Például sok esetben egy bemeneti tömböt cirkulárisan kell kezelni, azaz, ha elértünk az utolsó (N-1.) elemhez, akkor az első (0.) elemmel kell folytatni a számítást.

A DSP-k egyik legfontosabb jellemzője a számábrázolás. E feltétel szerint alapvetően két csoportba lehet sorolni a processzorokat: fixpontos, illetve lebegőpontos. A fixpontos számábrázolás azt jelenti, hogy a memóriában tártolt számok helyi értéke állandó, ám emellett a tény mellett is többféle számábrázolási mód létezhet:

- előjeles egész
- előjel nélküli egész
- előjel nélküli tört
- előjeles tört

Az eltérő számábrázolási módok külön függvényeket igényelnek, hogy az adott ábrázolási módban elvégzett műveletek eredménye megfelelő legyen.

A lebegőpontos számábrázolás során a számot a következő formában ábrázoljuk:

$$x = m \cdot 2^e$$

ahol m a mantissza, e az exponens $0.5 \le m \le 1$ és mindkettő előjeles szám. Ily módon egész és tört számok is ábrázolhatók. Az IEEE duplapontos számábrázolási szabványban ez a következőképpen jelenik meg:

- 1 bit előjel
- 11 bit exponens
- 52 bit mantissza

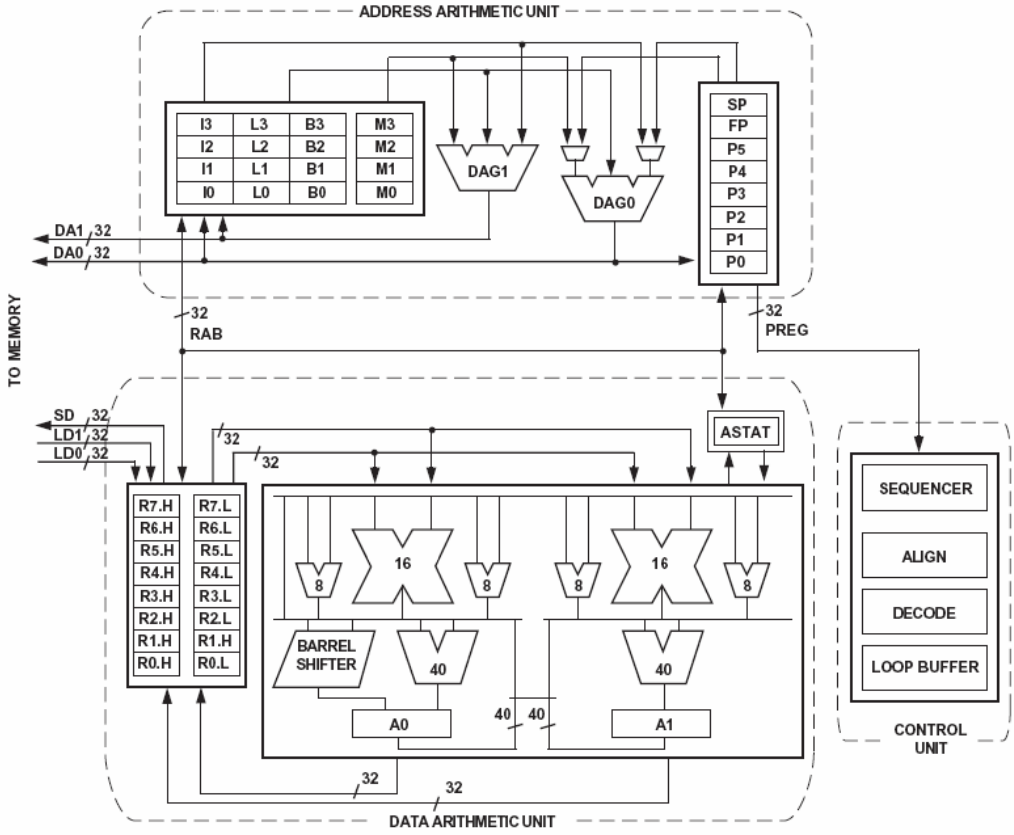
3.2.1. A fejlesztőkörnyezet

A jelfeldolgozó algoritmusaink szoftveres implementálását a gyakorlatban nagyban megkönnyítik az úgynevezett fejlesztőkártyák és az azokat támogató szoftveres fejlesztőkörnyezetek. Ha a fejlesztés során kiválasztottuk felhasználni kívánt processzort, a szoftverfejlesztési fázisban nem kell bajlódjunk azzal, hogy egy kis példányszámban esetleg amúgy is nehezen beszerezhető processzorral prototípust építsünk, nyákot tervezzünk, alkatrészeket ültessünk be, hanem az adott processzor egy fejlesztőkártyáját használhatjuk. Egy ilyen kártya úgy van kialakítva, hogy a processzor lehetőleg összes funkciójához, kivezetéséhez, perifériájához hozzáférhessünk. Mindemellett egy támogatott szoftverfejlesztő környezetben a debugger és az emulátor funkciókat is kihasználhatjuk. Egy így elkészült szoftver kifejlesztése után a hardver megtervezése is könnyebb feladat. A szoftverfejlesztés során általában nemcsak az adott processzor gépi

kódján (assembly nyelven) támogatott a fejlesztés, használhatunk magasabb szintű nyelveket is, mint például C, C++. Az ilyen módon elkészült szoftvereket, ahhoz, hogy a processzoron futtathatók legyenek, le kell fordítani egy processzorra feltölthető kódra. Az úgynevezett fordítók minősége, illetve paraméterezhetősége ezen művelet során kritikus. Egy rosszul fordított kód ugyanis lassabb. A fordítók nagyon fontos szolgáltatása az optimalizáció, mely funkció segítségével az adott processzor speciális funkciói (cirkuláris buffer, párhuzamosan elérhető memóriablokkok, MAC, stb.) úgy kerülnek kihasználásra, hogy a magas szintű programkódban ezen funkciók explicit módon nincsenek kifejtve, azok el vannak rejtve előlünk. Egy jó fordító észreveszi az adódó optimalizációs lehetőségeket, és ennek megfelelően fordít. A fordítás előtt az optimalizációt be kell kapcsolni, valamint fontos megjegyezni, hogy a sikeres optimalizálást elősegíti, ha az adott fordító szállítójának ajánlásait betartjuk programozási stílusunk kialakítása közben.

A fejlesztés során az Analog Devices cég BF537 processzorát tartalmazó BF537 EZ-KIT Lite fejlesztőkártyát használtuk, a szoftverfejlesztés az ezt a kártyát támogató VisualDSP++ integrált fejlesztőkörnyezetben történt.

3.2.2. Az ADSP BF537 jelfeldolgozó processzor főbb jellemzői [10]



21. ábra. Az ADSP BF-537 processzor magjának blokkvázlata.

3.2.2.1. Struktúra, memória

A processzor egy program- és egy adatmemóriát címez meg. Minden memóriaterület 16 bites. A processzor 48 kByte belső programmemóriát és 64 kByte adatmemóriát tartalmaz. Az utasítások 16 és 32 bitesek lehetnek, az adatok pedig 8, 16 vagy 32 bitesek. Mivel az implementáció során külső memóriát nem használunk, az adatmemória méretét mindig szem előtt kell tartanunk változóink tárolásakor.

3.2.2.2. Aritmetika

Az aritmetikai műveleteket 3 különböző egység (ALU - aritmetikai-logikai egység, MAC -szorzó-összeadó egység, Shifter - léptető) végzi. Az aritmetikai egység blokkvázlata a 21. ábrán található.

3.2.2.3. Címzés

A címzésre használható regisztereket csak assembly nyelvű programozás esetén kell közvetlenül kezelnünk, C programozás esetén a fordító gondoskodik erről. Az indirekt címzésre a P0..5 regiszterek használhatók. Cirkuláris bufferek megvalósítására a B0..3, I0..3, L0..3 és az M0..3 regisztereket kell használni. A buffer kezdőcímét a B (Base) regiszterben, hosszát az L (Length) regiszterben, a léptetés értékét az M (Modify) regiszterben kell megadni. Például M0-ba 1-et kell írni ha 1-gyel szeretnénk léptetni a címet. Az I (Index) regiszterben található az aktuális memóriacím, tehát léptetéskor ez a cím változik az M módosító regiszter értékével.

3.2.3. A BF537 EZ-KIT Lite fejlesztőkártya és a VisualDSP++ integrált fejlesztőkörnyezet

A VisualDSP++ integrált fejlesztőkörnyezet az Analog Devices cég szoftverfejlesztő környezete jelfeldolgozó processzoraihoz. A környezet nagy előnye, hogy támogatja a cég processzoraihoz készült fejlesztőkártyákat [10], [11].

A projekt editor segítségével szerkeszthetjük forrásfájljainkat, ezek alkotnak egy projektet. Ezt a VisualDSP++ - szal lefordítva és linkelve egy DSP-re letölthető fájlt kapunk. A forrásfájlokat elkészíthetjük assembly, C vagy C++ nyelven, egy projekten belül a különböző nyelveken megírt szoftverkomponenseket keverhetjük. A debugger funkcióval nyomon követhetjük a program működését, szoftveres környezetben szimulátor vagy hardveres környezetben, emulátor segítségével a processzor regisztereinek illetve memóriaterületeinek tartalma monitorozható. A memória tartalmi tetszőleges címtartományból fájlra kimenthető, később más programok (pl. Matlab) segítségével feldolgozható.

A BF537 EZ-KIT Lite fejlesztőkártya USB porton csatlakozik a számítógéphez. A kártyán található főbb alkotórészek:

Analog Devices ADSP-BF537 processzor
- 600 MHz órajel
- 182-pin mini-BGA tokozás
- 25 MHz-es kristályoszcillátor

Szinkron, dinamikus, random elérérésű memória (SDRAM)
- 64 MB (8M x8-bis x 4 bank) x 2 chip

Flash memória
- 4MB (2M x 16-bits)

Analóg audio interfész
- AD1871 96 kHz-es mintavételi frekvenciájú A/D átalakító
- AD1854 96 kHz-es mintavételi frekvenciájú D/A átalakító
- 1 bemeneti sztereó jack
- 1 kimeneti sztereó jack

Ethernet interfész
- 10-BaseT (10 Mbits/sec) és 100-BaseT (100 Mbits/sec) Ethernet Medium Access Controller (MAC)
- SMSC LAN83C185 eszköz

Controller Area Network (CAN) interfész
- Philips TJA1041 nagysebességű CAN interfész
- National Instruments Educational Laboratory Virtual Instrumentation Suite Interface (ELVIS interfész)
- LabVIEW™-alapú virtuális eszközök

Univerzális aszinkron vevő/adó (UART)
- ADM3202 RS-232 vonalmeghajtó/vevő

LED-ek
- 9 LED: 1 power, 1 kártya reset, 6 általános célú, és 1 USB figyelő

Nyomógombok
- 5 nyomógomb: 1 reset, 3 programozható flag pergésmentesítéssel
- 1 programmable flag without debounce logic

Bővítő interfészek
- a processzor minden jele kivezethető

Egyéb tulajdonságok
- JTAG ICE 14 pin port

A mi feladatunk csak az audió interfész, valamint bizonyos esetekben a nyomógombok és ledek használatát teszi szükségessé.

3.3. A jelfeldolgozó szoftverekről

A digitális jelfeldolgozás flexibilitását az adja, hogy az A/D konverzió után a mintavett és kvatált jelünket reprezentáló szám egy megszakítási rutint követően egy memóriacímre kerül. Innentől kezdve a jel csak egy változó, melyet a jelfeldolgozó szoftverünkben tetszőlegesen felhasználhatunk. Valós idejű rendszerekben azonban fellép a következő korlát: az A/D konverter minden mintavételkor generál egy megszakítást, ebben a megszakításban vehetjük át a keletkezett mintát és annak feldolgozását, esetleg kimeneti jelek D/A konverterre küldését a következő megszakítás előtt el kell végeznünk.

A program működésének lépései:

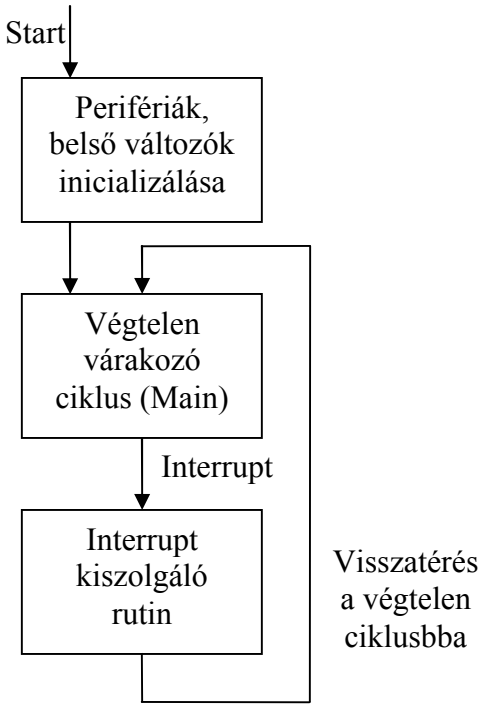
- **inicializálás**: ez vonatkozik a perifériák felkonfigurálására, a programban használt interruptok engedélyezésére.

- az egyszer elvégzendő műveletek elvégzése: nyilvánvaló, hogy processzoridőt takarítunk meg**,** ha az egyszeri műveleteket nem a *main()* végtelen ciklusban hajtuk végre

- végtelen ciklus: a processzor alapértelmezett státusza az *idle*, azaz elfoglaltság nélküli. Akkor történik adatfeldolgozás, amikor azt valamelyik periféria igényli egy megszakítással.

- kiszolgáló interrupt rutin: ebben történik meg a perifériákról érkezett adatok feldolgozása, majd visszatérünk a végtelen ciklusban való várakozáshoz.

Egy digitális jelfeldolgozó processzoron futó program általános felépítése a 22. ábrán látható.



22. ábra: egy jelfeldolgozó program általános felépítése.

A további programleírásokban csak magát az interruptban lefutó jelfeldolgozó függvényt tárgyaljuk, a programok keretrésze nem különbözik.

A jelfeldolgozó szoftver lényegében az ADC interrupt rutinja köré van szervezve. A konverter másodpercenként 48000 mintát vesz a bemeneti audiojelből és azt 24 biten kvantálja, azaz másodpercenként 48000 hardveres interruptot generál az ADC, azaz minden konverzió megtörténte után. Az interrupt rutinban a dedikált memóriaterületre, bufferre került mintát mint változót felhasználhatjuk. A DAC számára fenntartott kimeneti bufferre írhatunk a kimeneti audiojel előállítása érdekében.

Ahhoz, hogy minden kimeneti mintát időben elő tudjunk állítani, illetve a bemeneti minták közül se veszítsünk, az ADC megszakítását követően $\frac{1}{48000Hz} \approx 20$ mikroszekundum áll rendelkezésre a megszakítási rutin lefutására. Egy szoftver komplexitását ez a tény, valamint a belső memória korlátozza.

3.4. További felhasznált eszközök

3.4.1. Az ALTO AMX 140 keverőpult

23. ábra. ALTO AMX 140 keverőpult.

A keverőpultot a 29. ábrának megfelelően alkalmazzuk. Segítségével vizsgálhatjuk a rendszer viselkedését,ha változik az alacsony-, közép- vagy magasfrekvenciás átvitel. A pult lefgontosabb tulajdonságai [5]:

- 5 mikrofon bemeneti csatorna, aranyozott XLR csatlakozókkal és kiegyensúlyozott LINE bemenettel
- 4 sztereó bemeneti csatorna kiegyensúlyozott TRS jack-dugókkal.
- Ultra alacsony zajú mikrofon előerősítők
- Háromsávos equalizer minden csatornán
- 24 bites digitális effektprocesszor

3.4.2. A Krohn – Hite 3323 változtatható szűrő

24. ábra. Krohn – Hite 3323 változtatható szűrő.

Beállítási módjai:

- sáváteresztő
- sávzáró
- felüláteresztő
- aluláteresztő

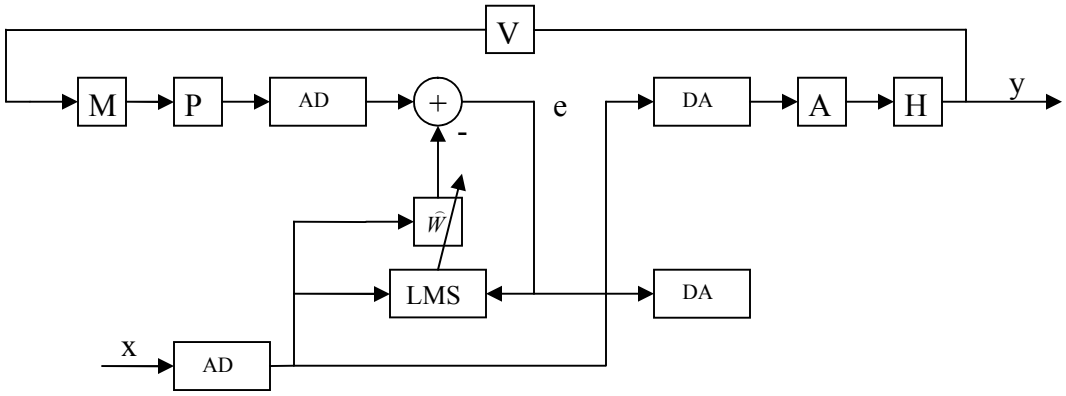
Specifikáció	
Frekvenciatartomány	.01 Hz – 99.9 kHz
Csillapítás dB/Oktáv	24/48
Max csillapítás (dB)	80
Kimenet Volt/Amper (RMS)	5 V/ 50 mA
3 dB pont	DC - 1 MHz

4. Az algoritmusok implementációja

4.1. Az LMS-algoritmus implementálása

4.1.1. Az LMS-algoritmussal megvalósított off-line rendszeridentifikáció

Az algoritmust az Analog Devices már korábban említett Blackfin BF537-es fejlesztőkártyájára implementáltuk. Az identifikáció elvét mutatja 25. ábra.



25. ábra. Az identifikáció elvének rendszermodellje.

A jelfeldolgozó processzoron elvégzendő műveletsor az LMS-algoritmusnak megfelelően a következő lépésekből áll:

1. A gerjesztőjel (Right In) aktuális érékének A/D-átalakítása, beolvasása, majd ismét kiküldése (Right Out). A kiküldött jel nem lesz más, mint a fizikai rendszer gerjesztőjele. Ezúton modellezzük a fejlesztőkártya A/D, illetve D/A átalakítójának késleltetését. Erre azért van szükség, mert amikor a mikrofon és a hangszóró között beavatkozunk a jelútba, azzal a céllal, hogy meggátoljuk a gerjedés kialakulását akkor beviszünk a rendszerbe egy plusz késleltetést, melyről nem szabad megfeledkezni. Ha nem így dolgoznánk, akkor a kioltás során nem egymáshoz tartozó bemenet – kimenet kombinációk kerülnének egymás mellé, mely ellehetetlenítené a rendszer működését.

2. azonos jellel (Right Out) gerjesztett fizikai rendszer aktuális kimeneti értékének A/D átalakítása és beolvasása (y_k).

3. az elmúlt N (a szűrő fokszáma) bemeneti minta alapján $\hat{W}$ kimenetének kiszámítása, mely a következő egyenlet szerint történik:

$$\hat{y}_k = \sum_{i=0}^{N-1} x[i] \cdot W[i]$$

ahol x a bemeneti mintákat tartalmazó tömb.

4. a fizikai rendszer kimenete és az aktuális $\hat{y}_k$ közötti különbségképzés, ami nem más, mint a hibajel

$$e_k = y_k - \hat{y}_k$$

5. a hibajel alapján $\hat{W}$ együtthatóinak változtatása az LMS-algoritmusnak megfelelően. Ennek a lépésnek az elvégzésekor vigyázni kell a konvergenciaparaméter a megadásával, mert ha túl kicsire választjuk, akkor a számábrázolási pontosság miatt előfordulhat, hogy egyáltalán nem adaptálódik a szűrőnk, mert a bejövő adatból és a konvergenciaparaméterből képzett szorzat egyszerűen nulla lesz. Tapasztalati alapon a $0.0001 \le \mu \le 0.125$ választás bizonyult megfelelőnek. Ezen értékeken belül gyakorlatilag csak a konvergencia sebességét határozza meg.

6. a hibajel kiküldése a nem használt kimeneti csatornára, hogy követni tudjuk a rendszer működés közbeni állapotát.

Ennek a műveletsornak az elvégzésére tehát 20 mikroszekundum áll rendelkezésünkre, ez a legfontosabb korlátozó tényező a rendszerben a fix mintavételi frekvencia miatt. Ha nem sikerül elvégezni a fent említett műveletsort, azaz a következő minta hamarabb megérkezik, mint hogy a jelfeldolgozó processzor befejezte volna az aktuális értékek számítását, akkor az nem megfelelő működést eredményez. Alapvetően a szorzások és az összeadások elvégzéséről (lásd 3. műveleti pont) van szó, melyek számát a $\hat{W}$ együtthatóinak száma (N) egyértelműen meghatározza, hiszen az $\hat{y}_k$ kiszámításához N darab összeadás és N darab szorzás szükséges, mely a MAC műveletnek köszönhetően együtthatónként egyetlen műveletet igényel. $\hat{W}$ együtthatóinak módosítása során egy egyszerűsített struktúrának köszönhetően ismét N szorzást és N összeadást (ill. kivonást) eredményez, ám itt már nem tudjuk használni a MAC műveletet, ezért ez együtthatónként kettő műveletet igényel. Az egyszerűsített struktúra azt jelenti, hogy a hibát és a konvergenciaparamétert egyszer szorozzuk csak össze, nem pedig minden egyes szűrőegyüttható frissítésekor, ily módon (N-1) szorzást megtakarítunk. Ennek köszönhetően N maximális értéke emelkedik. További javulást lehet elérni, ha használjuk a jelfeldolgozó processzor már említett speciális szolgáltatásait:

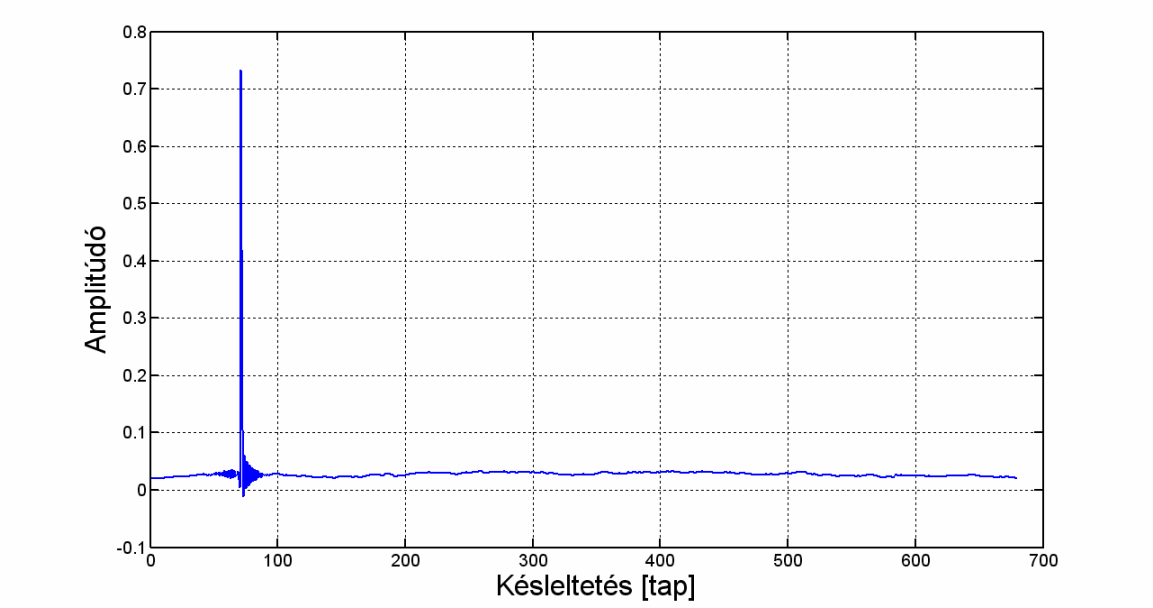
- optimalizáció
- cirkuláris buffer
- speciális számábrázolási módok
- szeparált memóriablokkok

4.1.2. A számábrázolás és a késleltetés szerepe az algoritmus működésében

A műveletek elvégzésének legkényelmesebb módja ilyen és hasonló esetekben az, ha lebegőpontos számokkal dolgozunk, viszont a fejlesztőkártyán található ADSP – BF537 processzor 16 bites fixpontos számábrázolású, ezért a lebegőpontos műveletek elvégzése nagyon megterheli a processzort, rengeteg plusz művelet elvégzése szükséges. Az érdekesség kedvéért kipróbáltuk, hogy milyen hosszú szűrőt tudunk így megvalósítani. Az eredmény az volt, hogy egy ötödfokú szűrő is alig fért bele a 20 mikroszekundumos időkorlátba.

Van lehetőségünk választani a fract16 illetve a fract32 számábrázolási módot. Ezek fixpontos törtszámok, melyek értéke a [-1, 1) intervallumban van, a processzorban pedig 16, illetve 32 biten vannak ábrázolva. A fract16 ill. fract32 számokkal végzendő műveleteket nem lehet a szokványos operátorokkal elvégezni, mindenre külön fract függvény van definiálva, melyek a matematikai műveleteket a számábrázolásnak megfelelően végzik el.

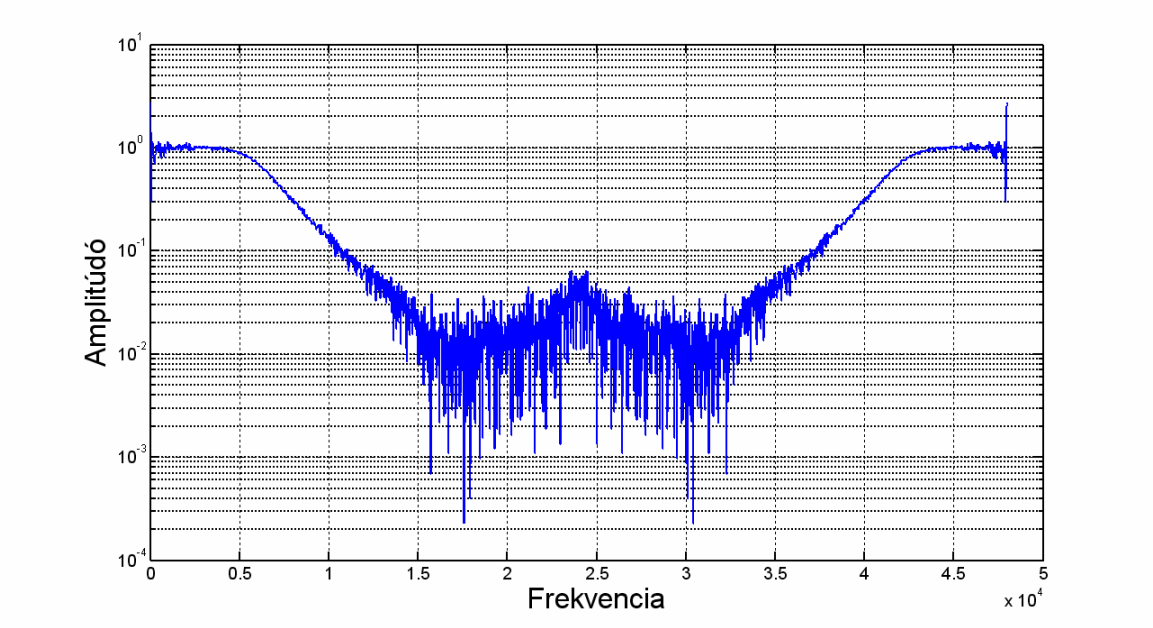
Mindennek tudatában, a rendszert maximálisan kihasználva a fract32-es esetben a szűrő legmagasabb fokszáma $N = 185$, míg a fract16 esetben $N = 700$. Az adott rendszer identifikációjához kell egy minimális fokszám, mert az A/D és D/A átalakítók önmagukban 60 foknak megfelelő késleltetést (azaz kb. $60 \cdot 20\ \mu s$) visznek be a rendszerbe (lásd 26. ábra). Ez jól látható a következő ábrán, ahol a fizikai rendszer egy kábel volt. Az együtthatókat Matlabba átmásolva és kiplottolva kapjuk a szűrő impulzusválaszát, melyből a késleltetés is leolvasható. Az impulzusválasz megfelel a kábel átvitelének, az A/D és D/A átalakítók késleltetése után gyakorlatilag egyetlen együttható átviszi a jelet. Ez igazolja azt, hogy egy alacsony, például 40 fokszámú szűrő egyszerűen nem lenne képes identifikálni a kábel átvitelét a késleltetés miatt.



26. ábra. Kábel identifikációjának eredménye a késleltetés felmérésére.

Mindezek után a fizikai rendszer helyére egy KROHN – HITE típusú szűrőt helyeztünk, 6 kHz sávszélességű aluláteresztő szűrőt megvalósítva. A gerjesztőjel 50 kHz sávszélességű Gauss-zaj volt, melyet egy HP zajgenerátor segítségével állítottunk elő. Azért szükséges a szélessávú gerjesztés, hogy a teljes frekvenciatartományban identifikáljuk a rendszert. Egyetlen szinuszjellel gerjesztve csak a szinuszjel f_0 frekvenciáján és környékén nyernénk értékelhető információt. Mindkét esetben a lehető legnagyobb fokszámmal dolgozva és ugyanazt a fizikai rendszert identifikálva, majd az eredményeket Matlabban ábrázolva adódnak a frekvenciatartománybeli átvitelek

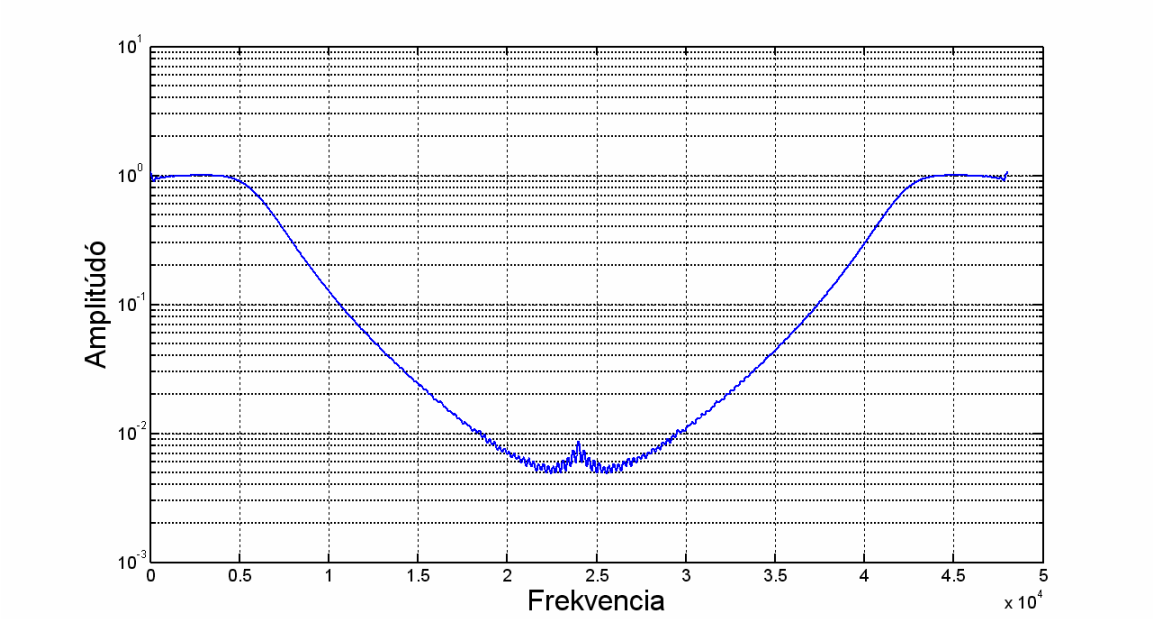
1. A fract16 számábrázolást használva:



27. ábra. Aluláteresztő szűrő identifikációja fract16 számábrázolással.

A vágási frekvencia jól láthatóan 6 kHz-nél van, ám a számábrázolási pontatlanságok miatt a $\widehat{W}$ átvitele az ideális karakterisztikát „körüllengi”.

2. A fract32 számábrázolást használva a 28. ábra mutatja az identifikáció eredményét:



28. ábra: Aluláteresztő szűrő identifikációja fract32 számábrázolással.

A különbség szemmel is igen jól látható. A fract32-es számábrázolással véghezvitt identifikáció eredménye igen jól megközelíti a fizikai rendszer átvitelét.

4.1.3. Az LMS-algoritmuson alapuló off-line identifikáció korlátai

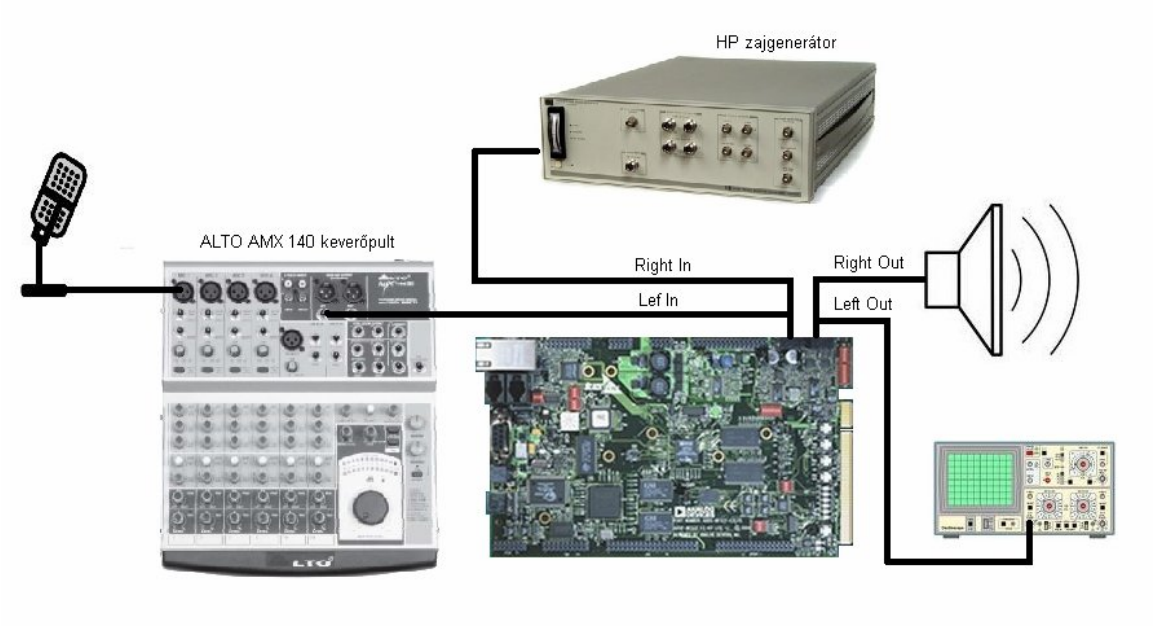
Jelen esetben $\widehat{W}$ egy FIR szűrőt valósít meg, ami azt jelenti hogy az impulzusválaszának a hossza véges, $N \cdot \frac{1}{f_s}$ értékű. Mindez fract16 ábrázolású változók esetén, ahol N maximális értéke 700, azt jelenti, hogy olyan rendszert tudunk identifikálni, melynek impulzusválasza nem haladja meg a 0.0145 másodpercet. Ez az érték a hang sebességével számolva azt jelenti, hogy a mikrofon és a hangszóró legnagyobb távolsága 4 méter körül lehet. Ezek alapján a 32 bites fractional változók esetében még nehezebb dolgunk van, hiszen ott a maximális fokszám 180, azaz átszámolva kevesebb, mint 1 méter a maximális távolság. Minél távolabb van a mikrofon a hangszórótól, annál hosszabb az impulzusválasz, amit approximálni szeretnénk. Kültéri hangosításoknál ez sokkal több is lehet, oda extrém hosszú szűrők lennének megfelelők.

Az akusztikai átvitel off-line identifikálása LMS algoritmussal és FIR szűrővel tehát olyan elrendezésekben lehet hasznos, ahol a távolság a mikrofon és a hangszóró között nem nagy, és nem is változik, azaz invariáns. Ilyenek pl: notebook-ba beépített mikrofon, telefon kihangosító, hallókészülék.

Azokban az esetekben, amikor a felhasználási terület engedi, lehetséges megoldás a korlátok bővítésére a mintavételi frekvencia csökkentése. Ekkor több idő jut egy-egy műveletsor elvégzésére, magasabb fokszámot lehet elérni, ugyanakkor azonos hosszúságú impulzusválasz approximálásához alacsonyabb fokszámú szűrő is elégséges.

4.1.4. Az akusztikus visszacsatolás kioltása az LMS-algoritmuson alapuló rendszerientifikáció segítségével

A mikrofon és a hangszóró közötti akusztikai átvitel identifikációja a következő ábra szerinti elrendezésben valósult meg:



29. ábra. Az identifikáció elrendezése.

Az elrendezés alapelve hasonlít a 6. ábrán láthatóhoz, annyi különbséggel, hogy az identifikálandó rendszer itt a hangszóró – levegő – mikrofon együttese. Egy keverőpultot is elhelyeztünk az összeállításba, hogy az átvitelt módosítani tudjuk.

A kioltás megvalósításakor egy olyan program megvalósítására törekedtünk, amely egyben a rendszer működési módjait is megvalósítja. Az egyes működési módok között a kártyán található nyomógombbal lehet mozogni. A programok keretrendszere azonos a 3.3 alfejezetben leírtakkal, csupán a jelfeldolgozó rutin műveleteiben térnek el egymástól. A megvalósított három funkció:

1. Bypass üzemmód: ez a működési üzemmód arra szolgál, hogy a jelet minden változtatás nélkül átküldjük a fejlesztőkártyán.

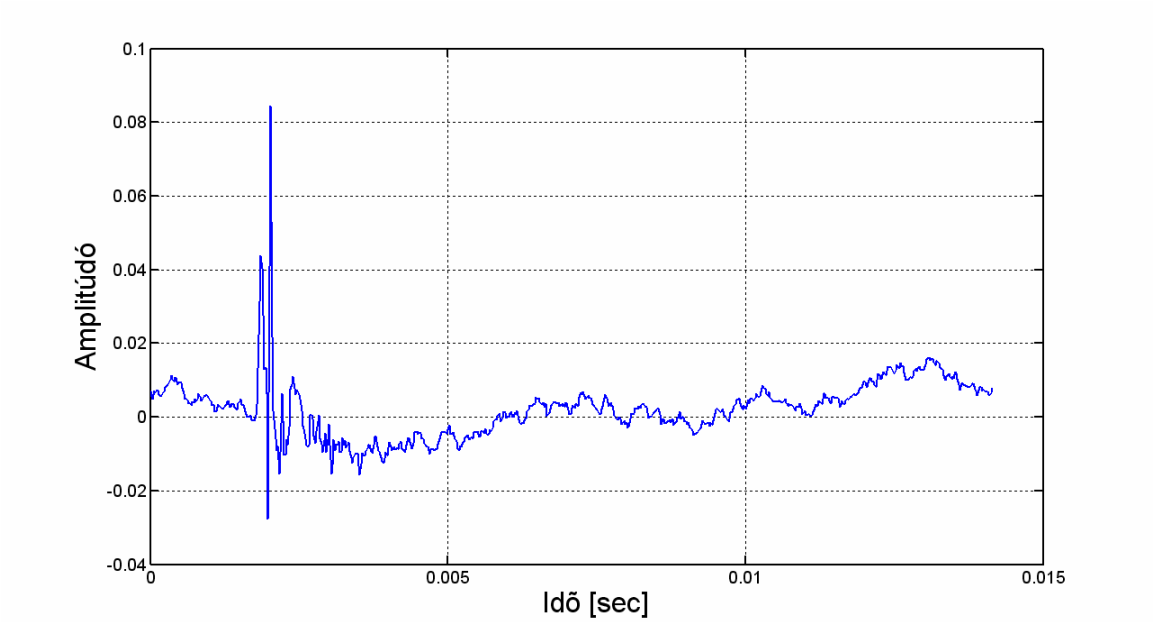
2. Identifikáció: ebben a részben történik a rendszer fehérzaj-gerjesztéssel történő approximálása. A konvergenciaparaméter függvényében ez pár másodperc alatt is megtörténhet, de eltarthat percekig is. Amikor a hiba értéke állandósul, akkor továbbléphetünk a kioltáshoz.

3. Kioltás: ha az akusztikus visszacsatolás identifikációja során a hibajel elérte a minimális értékét, akkor az algoritmus értelmében a fizikai rendszer aktuális

kimenete y_k és az azt becslő $\hat{y}_k$ között az eltérés minimális. A mikrofon és a hangszóró között beavatkozva, a becsült átvitel értékét levonva érhető el a visszacsatolás kioltása. A kioltás lépései :

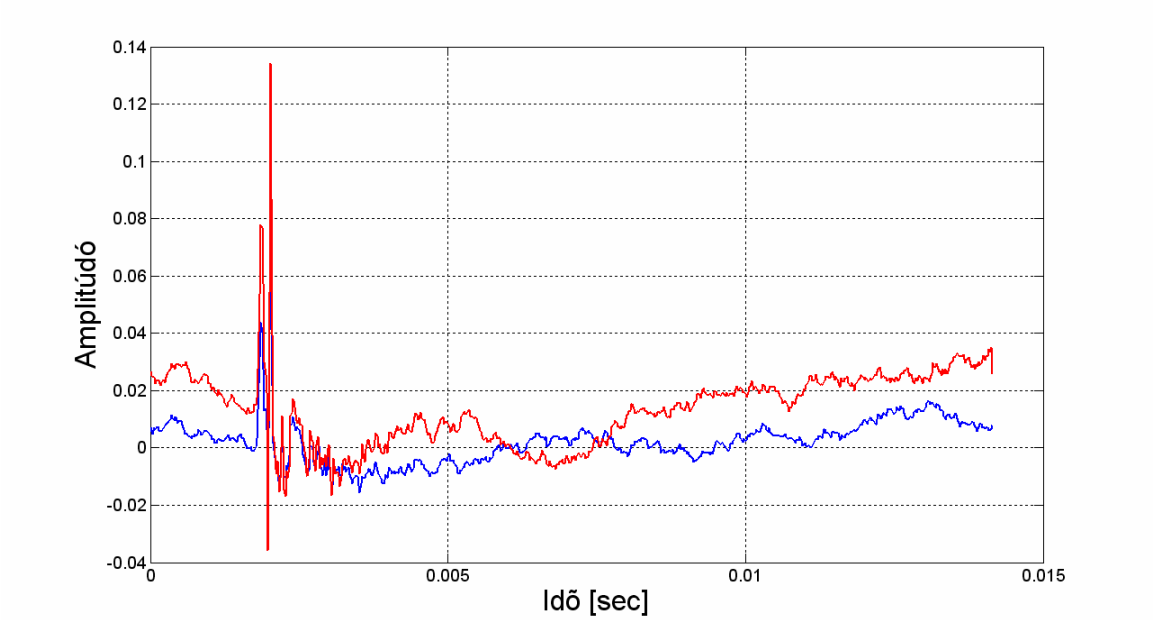
- A mikrofon kimenetének A/D átalakítása és a bemeneti tömbbe másolása.
- A bemeneti tömb és $\hat{W}$ együtthatói alapján az átvitel becsült értékének kiszámítása.
- A becsült érték levonása az aktuális bemeneti értékből
- A különbség kiküldése a hangszóró felé.

A megvalósított program a fract16 és a fract32 számábrázolási módban megegyezik, csupán a megfelelő függvények vannak átírva, illetve a fokszám különbözik. Ha azonos jelutat szeretnénk identifikálni, akkor ügyelni kell arra, hogy fract32 esetben a szűrő fokszáma legfeljebb 185 lehet, tehát a mikrofont elég közel kell helyezni a hangsugárzóhoz, hogy az identifikáció megtörténhessen. A következő két ábra szemlélteti a fract16 ill. fract32 számábrázolással identifikált akusztikai átvitel impulzusválaszát.



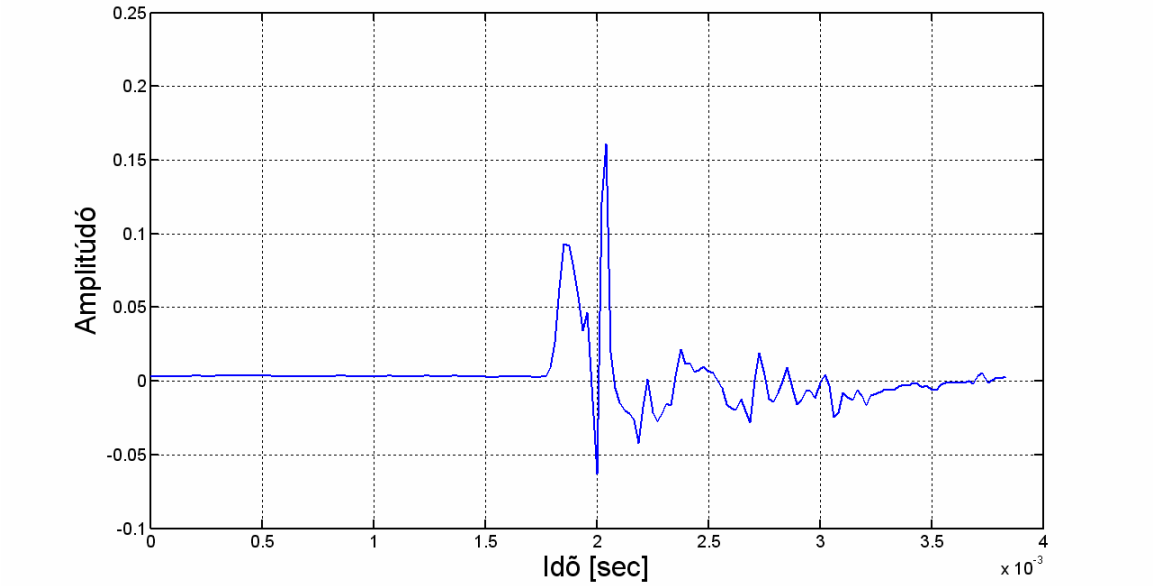
30. ábra. Akusztikai visszacsatolás identifikációjának eredménye fract16 számábrázolással.

A következő ábrán látható a gerjesztő jel amplitúdója növelésének hatása. A 30. ábra mellé beillesztettem egy olyan identifikáció eredményét, amely során az amplitúdó nagyobb volt.



31. ábra. A gerjesztőjel hatása az identifikáció eredményére.

A fract32-es identifikáció eredménye:



32. ábra. Akusztikai visszacsatolás identifikációjának eredménye fract32 számábrázolással.

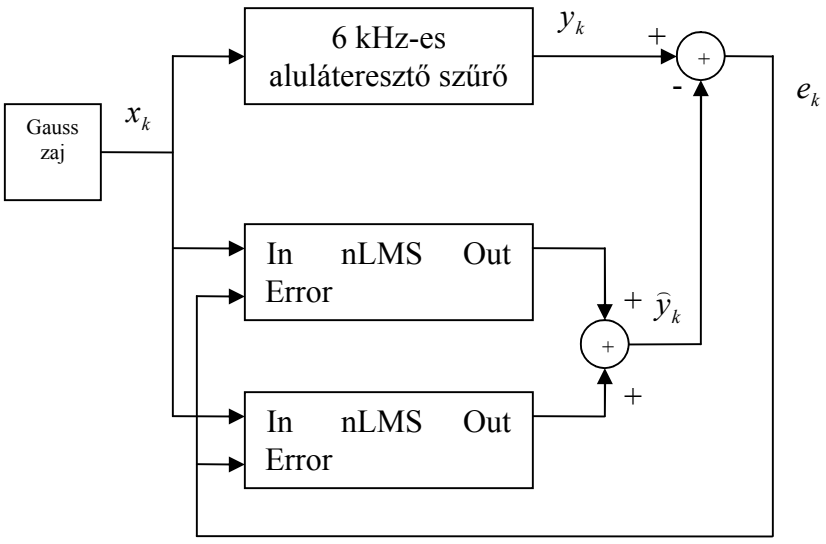
Feltűnő különbség, hogy a késleletetési szakaszban, továbbá az impulzusválasz lezajlása után is sokkal kisebb a nullától való eltérése az impulzusválasznak. Véleményünk szerint, ha a fract32-es approximáló FIR szűrő hossza kétszerese lenne, akkor az identifikáció tökéletes lehetne.

Itt is kiemelkedő szerepű a számábrázolási pontosság. Már az identifikáció során is feltűnő, hogy egy olyan elrendezésben, melynek átvitelét mindkét módon identifikálni lehet (az impulzusválasz „belefér” az approximáló szűrők fokszámába) a hibajel minimális értéke eltérő. Fract32-es számábrázolás esetén a hibajel jelentősen csökken, ám ugyanez nem mondható el a fract16 számokkal való műveletvégzés eredményéről. Ebben az esetben is alacsony értékű a maradó hibajel, ám gyakran ugrások találhatók benne, nem olyan „sima”, mint az előző esetben. Az átvitelt identifikálva fontos szerepe van még a kivezérlésnek. Kisebb amplitúdójú gerjesztőjel esetén a hiba is kisebb marad, ám a kioltásnál ez hátrányokkal járhat, mint majd látni fogjuk.

4.2. Az IIR LMS algoritmus implementálása

4.2.1. Az IIR LMS algoritmussal megvalósított off-line rendszeridentifikáció

Az identifikáció elvi működése abban tér el a 2.2.2 alfejezetben bemutatottól, hogy itt már visszacsatolás is szerepel a rendszerben, valamint természetesen az algoritmus felépítése eltérő az új elemeknek megfelelően. Az identifikáció elvi blokkvázlatát a 33. ábra mutatja.



33. ábra. Az IIR LMS algoritmus rendszermodellje.

Az algoritmus részletes leírásában, a 2.2.5 fejezetben egyetlen vektorban vannak a kimeneti és a bemeneti értékek, ennek megfelelően a konvergenciaparaméter, továbbá a gradiensképzés is olyan vektorok alapján történik, amelyek tartalmazzák mind a bemenethez, mind a kimenethez kapcsolódó értékeket. Az egyszerűség és a jólláthatóság kedvéért a megvalósításkor külön kezeltük ezeket a vektorokat a bemeneti értékekre és a kimeneti értékekre.

A program működésének lépései (process függvény):

1. A gerjesztőjel (Right In) aktuális érékének A/D átalakítása, beolvasása, majd ismét kiküldése (Right Out). Ennek ebben az esetben is az a célja, hogy a rendszerbe bevitt késleltetéseket modellezzük, és ezt figyelembe véve történjen meg az identifikáció.

2. Azonos jellel (Right Out) gerjesztett fizikai rendszer aktuális kimeneti értékének A/D átalakítása és beolvasása (y_k).

3. Az elmúlt N (a bemeneti szűrő fokszáma) bemeneti minta alapján az approximáló szűrő kimenete első tagjának kiszámítása, mely a következő egyenlet szerint történik:

$$\hat{y}_{1k} = \sum_{i=0}^{N-1} x[i] \cdot a[i]$$

ahol x a bemeneti mintákat tartalmazó tömb.

4. Az utóbbi M (a kimeneti szűrő fokszáma) kimeneti minta alapján az approximáló szűrő kimenete második tagjának kiszámítása, mely a következő egyenlet szerint történik:

$$\hat{y}_{2k}(n) = \sum_{i=0}^{N-1} u[i] \cdot b[i]$$

ahol u a kimeneti értékeket tároló tömb.

5. A két kiszámított érték alapján a kimenet: $y_k = \hat{y}_{1k} + \hat{y}_{2k}$

6. A hibajel kiszámítása a fizikai rendszer kimenete és az aktuális $\hat{y}(n)$ közötti különbségképzéssel:

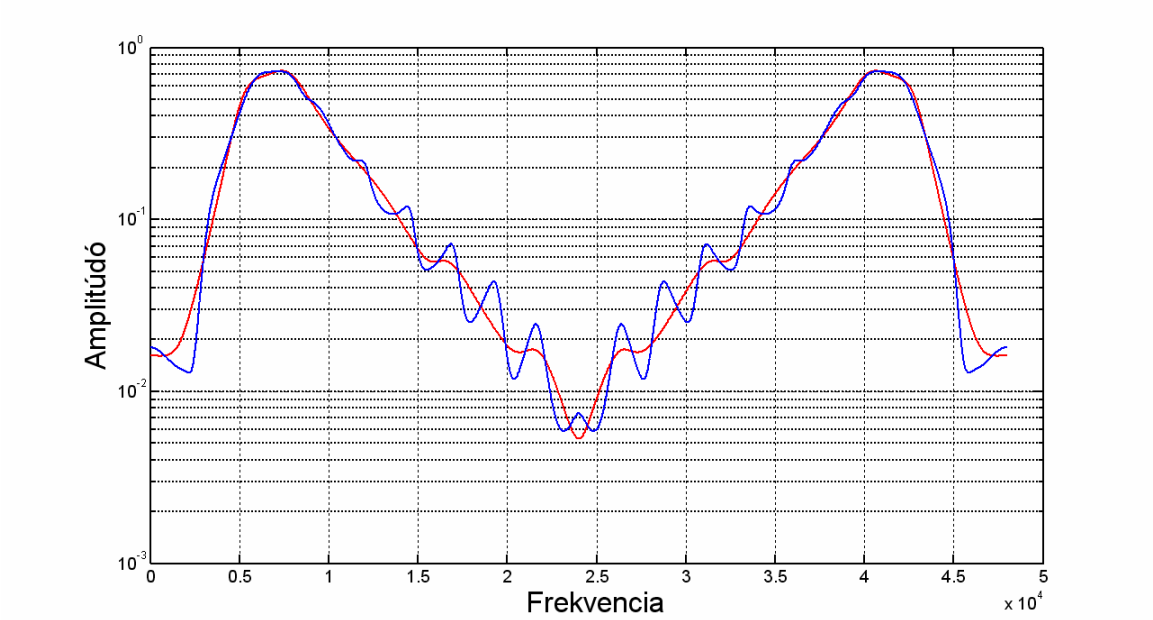
$$e_k = y_k - \hat{y}_k$$

7. A hibajel alapján az a_i és b_i együtthatók változtatása az LMS algoritmusnak megfelelően minden i-re.

8. A hibajel kiküldése a nem használt kimeneti csatornára, hogy követni tudjuk a rendszer működés közbeni állapotát egy oszcilloszkóp segítségével.

Az időkorlát ebben az esetben is 20 mikroszekundum, és mivel gyakorlatilag nem végzünk több műveletet, mint a FIR szűrő estében, ezért a két szűrő maximális fokszámának összege fract32 esetben nem haladhatja meg a 185-öt. Első lépésként ebben ez esetben is egy a KROHN – HITE szűrőn beállított rendszert identifikáltunk. Sávszűrőt állítottunk össze egy alul-, valamint egy felüláteresztő szűrő segítségével. Az

aluláteresztő sávkorlátja 8 kHz volt, a felüláteresztőé 6 kHz. A identifikálandó szűrőt közvetlenül gerjesztve először alacsony fokszámú szűrőt illesztettünk a fizikai rendszerre, majd ezt fokozatosan növeltük. A 34. ábrán látható az eltérő fokszámú szűrők illesztésének eredménye:



34. ábra. Az adaptív IIR szűrővel approximált sávszűrők.

Az implementáció során el kellett dönteni, hogy hogyan osszuk el azt a 185 szűrőegyütthatót, melyekkel dolgozhattunk. A hibajel alapján a legmegfelelőbb választásnak az bizonyult, amikor az *a* vektor hossza 140, a *b* vektor hossza 45.

4.2.2. A késleltetés szerepe az algoritmus működésében

Kezdetben alacsony fokszámú szűrőkkel dolgoztunk, ám ezek nem bizonyultak megfelelőnek, mert nem voltak képesek modellezni a késleltetéseket, így a hiba értéke jelentős maradt. Ha az identifikálandó rendszert (első lépésben egy sáváteresztő szűrőt) közvetlenül gerjesztettük a zajjal, akkor nyilvánvalóan nem lépett fel késleltetés és a hiba értéke megfelelően lecsökkent. A FIR szűrőnél már láttuk, hogy a DSP kártya kb. 60 szűrőfokozatnyi késleltetést okoz, ezt egy alacsony fokszámú IIR szűrővel nem lehet modellezni. Az IIR szűrő fokszámát növelve csökkent a hiba értéke.

A késleltetés okozta hiba csökkenthető, ha valamilyen módon figyelembe vesszük.

- Legegyszerűbb megoldás az, amikor növeljük a szűrő fokszámát, hogy elég információval rendelkezzen az identifikációhoz.
- A késleltetést szoftveresen kompenzálhatjuk, ami nagyobb tömbök alkalmazásában nyilvánul meg, melyeket eltolva címzünk.

- Lehetséges egy olyan megoldás, amikor egy $C(z)$ digitális FIR szűrővel identifikáljuk a késleltetést, majd a gerjesztőjelet a DSP-n kívül vezetve, a rendszert közvetlenül gerjesztve identifikáljuk a fizikai rendszert. Ekkor az átvitel:

$$H(z) = C(z) \cdot \frac{A(z)}{B(z)}$$

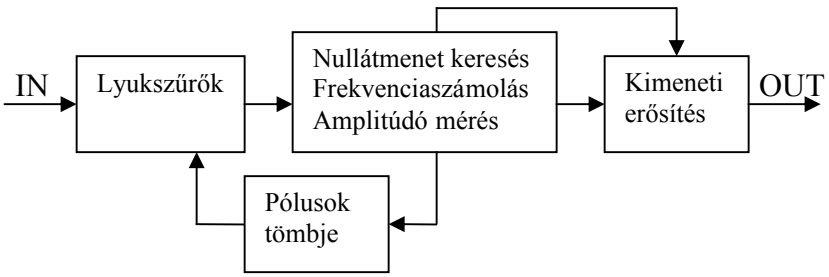
4.2.3. Az akusztikus visszacsatolás kioltása az IIR LMS algoritmuson alapuló rendszeridentifikáció segítségével

Az akusztikai visszacsatolás identifikációja során használt elrendezés azonos a XXX. Ábrán láthatóval, egyedül a fejlesztőkártyán futó program különbözik. Az üzemmódok megnevezése és a program keretrésze is azonos. Bypass módban itt is feldolgozás nélkül kerül továbbításra a jel. Az identifikációs üzemmód az IIR LMS algoritmus alapján fu,t és az a, illetve a b vektor értékeit adja át a kioltást megvalósító függvények. A kioltás itt is a 4.1.4 szerint működik.

4.3. Egy off-line, lyukszűrőket felhasználó rendszer implementációja

Az off-line, a zárt visszacsatolási hurkot lyukszűrőkkel módosító rendszerünk elvi működését fentebb tárgyaltuk. Most vizsgáljuk a kívánt működés eléréséhez szükséges szoftver felépítését.

A jelútba helyezett jelfeldolgozó rendszer A//D átalakítója 48 kHz-es mintavételi frekvenciával mintavételez az audiojelből. Minden mintavételkor meghívódik a megszakítási rutin. A megszakítási rutinban van elhelyezve a program összes funkciója, az időkorlátokra emiatt ügyelni kell. A megszakítási rutin három feladatot lát el. Először a mintavett jelet szűri a fentebb ismertetett rezonátoros struktúrával, majd az esetleges nullátmenetet vizsgálja és számolja belőlük a frekvenciát, végül a kimenet erősítését állítja be a fent ismertetett nemlineáris karakterisztika alapján az oszcilláció egyensúlya érdekében.



35. ábra. A megszakítási rutin felépítése.

A szoftverben egy 10 rezonátoros struktúrát valósítottunk meg. A lyukszűrő-bankot alkotó rezonátorok állapotát minden megszakítási rutinban számolni kell, az előző megszakításban eltárolt állapotot is figyelembe véve. A rezonátorok z^{-1} eleme így egy 10 elemű tömbbel vehető figyelembe. Egy 10 elemű for ciklussal minden tag aktuális kimenete kiértékelődik. Mivel a rezonátor pólusát megadó p_i komplex szám, a számítás folyamán komplex adattípusok felhasználása szükséges. Ezen adattípusok a complex.h által tartalmazott cmac_fr16 művelettel különösebb gond nélkül szorozhatók. A kiértékelés során a 10 (komplex) pólust egy 10 elemű tömb tárolja. A rezonátoros struktúra kimenete minden megszakítási rutinban az épp aktuális rezonátorkimenetek és bemenet alapján számolható. A megszakítási rutin további része ezen értéket kezeli bemenetként, sőt az akusztikai rendszer visszacsatolási hurka, illetve az egész átvitele így került megszűrésre.

A szűrők szelektivitását beállító μ-vel való szorzás valós típusokkal (fract16) van megoldva. A 48 Hz széles frekvenciasávokhoz kb. 60 Hz széles szűrőt terveztünk 50 dB elnyomással. Ehhez μ_1-nek 0,05 értéket adtunk. Az ezen lyukszűrőkhöz tartozó pólusok abszolútértékét 0.99-nek választottuk. A nagyfelbontású tartományban, a 4,8 Hz széles frekvenciasávokhoz kb. 6 Hz széles szűrőt terveztünk 30 dB elnyomással, ennek érdekében μ_2 értéke 0,00166 lett. A pólus abszolútértéke itt is 0,99.

A rutin következő része a frekvenciamérést valósítja meg. Ahogy azt már korább kifejtettük, a frekvenciamérést a nullátmenetek számolásán keresztül valósítottuk meg. Ha a jel nullátmenetét detektáljuk, inkrementálunk egy számlálót, egységnyi időnként (mondjuk 1s) lenullázunk, de ezt megelőzően a számláló tartalmát normáljuk az egy másodpercre vonatkozó nullátmenetek felére. Ez lesz a frekvencia. Ha pontosabb mérést szeretnénk, több ideig számolunk, de tapasztalataink szerint 2 másodperc bőven elégséges.

A nullátmenetet detektáló szoftverkomponens minden megszakítási rutinban az előző megszakításkor érvényes (és eltárolt) bemenetet összehasonlítja az aktuális bemenettel. Ha a komparálási szintet az előző érték nem éri el, de az új már igen, vagy azt meghaladja, akkor a jelünk épp keresztezi azt. A nullátmenet vizsgálatának robusztussá tétele érdekében a korábban tett megfontolásoknak megfelelően két komparálási szintet használtunk. A frekvenciamérés így nagy pontossággal megoldható.

A rendszer harmadik funkcionális egysége a kimenet erősítését szabályozza. A szabályozás legegyszerűbb formáját választottuk, ezt is elégségesnek találtuk. Ez úgy történik, hogy mérjük a bejövő jel amplitúdóját, és azt kivonjuk 1-ből. A számábrázolás miatt a jel értéke -1 és +1 közötti. A különbség lesz a kimeneti jel erősítése, azaz a szűrőnkből érkező jelet ezzel megszorozzuk és a szorzatot írjuk a D/A átalakító bufferjébe.

Az amplitúdó mérését a nullátmenetet mérő szoftverkomponenst kiegészítve valósítottuk meg. Minden nullátmenetnél lenullázunk egy segédváltozót, ezután minden megszakításnál vizsgáljuk hogy a (szűrt) jel, vagy „minta” abszolút értéke nagyobb-e a segédváltozónál, ha igen, a segédváltozó új értéke a minta lesz. A következő nullátmenet

detektálásakor a segédváltozó ─ értéke lenullázása előtt ─ a jel amplitúdója lesz. Ez az amplitúdó egy változóban van eltárolva és minden nullátmenet detektálásakor frissül. Ezen változó segítségével számoljuk ki a kimenet erősítését.

Az amplitúdót felhasználva a frekvenciamérést egy további finomítással még robosztusabbá tehetjük. A komparálási szinteket az amplitúdóhoz igazítva nem merülhet fel az a probléma, hogy egy kisebb amplitúdójú jel nem éri el a komparálási szintet, nagy amplitúdó esetén pedig a zaj okozta bizonytalanság maximálisan kiküszöbölhető. A nullátmenet detektálását végző egység feltételében a komparálási szintek csak változók, melyek az amplitúdó változó frissülésekor szintén frissülnek. Értéküknek mi az amplitúdó 256-od részét választottuk. Ez az osztás 8 bit lefelé shiftelésnek felel meg.

A rendszer konfigurálása, azaz a lyukszűrők felparaméterezése a mi szoftverimplementációnkban a következőképp zajlik. A jelfeldolgozó processzort és az A/D - D/A átalakítókat tartalmazó fejlesztőkártyát a jelfolyamba kötjük, a mikrofonelőfok és a végfok közé. Gyakorlati alkalmazásokban ezt egy keverőpult insert bemenete tökéletesen megvalósítja, így a jelfeldolgozó rendszer a visszacsatolási hurok egyik eleme lesz. A fentieknek megfelelően a mintavételezett jelet a lyukszűrő megszűri. A szűrők alaphelyzetben a mintavételi frekvencia felénél vágnak, tehát a rendszer elindításakor gyakorlatilag nem befolyásolják a hangképet. A rendszer erősítését növelve előbb-utóbb megjelenik az első gerjedés, mely stabilizálásra is kerül. A stabilizálást követően egy állandó amplitúdójú szinuszjelet hallunk a hangsugárzóból. A rendszer tartalmaz egy időzítőt, mely a frekvenciamérés átlagolására szolgál, 4 másodpercnyi nullátmenet-detektálást követően (4*48000 megszakítás után) a sorra következő megszakítási rutin a mért frekvenciából kiszámolja az annak megfelelő pólust és a pólusok tömbjének első elemét erre módosítja. A szűrőbankunk ezt követően már tartalmazni fog egy ily módon konfigurált szűrőt, a hurok átvitelének ezen csúcsát tehát megszüntettük. A hangerőt tovább növelhetjük a következő gerjedésig, a rendszer azt is stabilizálja, méri, újabb lyukszűrőt paraméterez föl. Ezt az eljárást addig folytatjuk, amíg mind a 10 lyukszűrő be nem lesz állítva. A végeredmény egy 10 db komplex számot tartalmazó tömb lesz, mellyel a szűrőbankot paraméterezve a kritikus frekvenciákon elnyomást érünk el a hurokerősítés átvitelében, ezzel a hangosító rendszer erősítése növelhető.

4.4. Egy on-line, lyukszűrőket felhasználó rendszer implementációja

Az on-line, a zárt visszacsatolási hurkot lyukszűrőkkel módosító rendszerünk elvi működését fentebb tárgyaltuk. Most vizsgáljuk a kívánt működés eléréséhez szükséges szoftver felépítését.

A fentieknek megfelelően a rendszer egy dinamikusan változó átvitelű zárt visszacsatolási hurokban van elhelyezve. Ezt az átvitelt kell folyamatosan monitorozni és módosítani, úgy, hogy a gerjedési jelenségeket mérsékelve a hangosító rendszer erősítése növelhető legyen. Az átvitel módosítása az előző off-line rendszerhez hasonlóan

lyukszűrőkkel történik, viszont a gerjedési frekvenciákat más koncepció szerint kell meghatározni, azt is figyelembe véve, hogy ezen frekvenciák folyamatosan változnak.

A megvalósított algoritmus a már létrejött gerjedést detektálja és orvosolja lyukszűrők segítségével. A lyukszűrő-bankunk felparaméterezéséhez itt is meg kell határoznunk a gerjedési frekvenciákat, és ez alapján feltölteni a rezonátorok pólusait tároló tömböt

Képeznünk kell tehát a bejövő jelből egy bemeneti vektort, azaz egy FFT pontszám hosszúságú tömböt kell feltöltsünk a bejövő mintákkal. Ezt kell átadni az FFT-t számoló függvénynek. A képzett transzformációs vektor értelemszerűen szintén egy tömb. Egy for ciklussal végigmehetünk a tömb elemein, és egy erre a célra létrehozott „state" tömbben inkrementáljuk a megfelelő sorszámú változó értékét, ha az adott frekvenciasáv növekedett az előző FFT képzés megfelelő vektoreleméhez képest. Ellenkező esetben nullázzuk az adott sorszámú változót. Definiálhatunk egy „maxstate" konstanst, melyet ha egy „state" tömbelem elér, akkor a neki megfelelő frekvenciasávot gerjedő sávnak nyilvánítjuk. A sáv tömbindexéből annak középfrekvenciája kiszámolható, ezzel a szűrőbank egyik rezonátorának pólusa is. Mivel a szűrők helye az átvitel dinamikus volta miatt folyamatosan változik, a pólusokat egy cirkuláris pufferben tároljuk, új gerjedés detektálásakor mindig a legrégebbi pólust írjuk felül. A rendszer alapja tehát a folyamatos FFT képzés, és az így kapott egymás után következő diszkrét spektrumok vizsgálata. A megvalósított funkciót ezzel vizsgáltuk, a szoftver struktúráját viszont nem. Most ez következik.

A jelútba helyezett jelfeldolgozó rendszer A/D átalakítója 48kHz-es mintavételi frekvenciával mintavételez az audiojelből. Minden mintavételkor meghívódik a megszakítási rutin. Az off-line rendszerrel ellentétben az on-line rendszer nem csak a megszakítási rutinban fut, az FFT képzése bizonyos pontszám felett ugyanis nem oldható meg egy megszakítási rutin alatt, nem áll rendelkezésre elég idő a számítások elvégzésére. Egy 1024 pontos FFT kiszámolása tapasztalataink szerint kb. 6 mintavételnyi, azaz 6 megszakításnyi időt vesz igénybe. A kapott frekvenciaintervallumok szélessége az FFT pontszámából adódik:

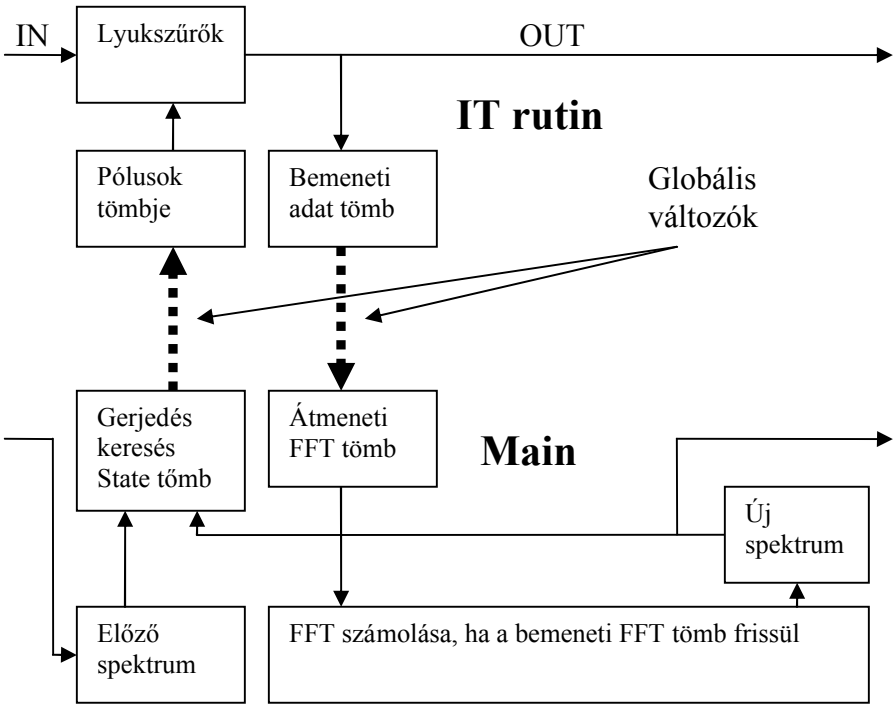
$$\frac{48000Hz}{1024} \approx 48Hz$$

Ennek megfelelően az alkalmazott lyukszűrők vágási szélessége is 48 Hz körüli.

Ennél szélesebb lyukszűrők alkalmazása már túlságosan a hangminőség rovására menne, tehát az FFT pontszámát nem csökkenthetjük a számítások lerövidítésének érdekében. A transzformációt a megszakítási rutinon kívül kell elvégezni.

A megszakítási rutinban fut a szoftver időkritikus része. Itt fut a lyukszűrő algoritmusa, mely a bemenetet szűrve állítja elő a kimenetet, valamint a bemeneti mintákból itt töltődik az FFT méretű tömb, melyből később a transzformált képződik. Ezen műveletek nem töltik ki teljesen a két mintavétel közötti időtartamot, miután lefutottak, a program a programtörzsben fut tovább. Az FFT számolása itt történik. Ha az FFT pontszámának

megfelelő mennyiségű bemeneti minta összegyűlt a bemeneti tömbben, akkor annak tartalmát átmásoljuk az FFT erre fenntartott átmeneti tömbjébe és engedélyt adunk a transzformáció elvégzésére. A transzformáció a program törzsében fog futni, de ez alatt, mintegy párhuzamosan az A/D átalakító megszakítása hatására a megszakítási rutinban a bemeneti tömb frissítése új mintákkal, valamint a kimenet előállítása a lyukszűréssel zavartalanul folytatódik. Ha a megszakítási rutin lefutott, a kezdetekor megszakított transzformáció a programtörzsben visszakapja a vezérlést és fut tovább. Ilyen módon folyamatosan képezhetjük a bemenet diszkrét Fourier transzformáltját, melyeket a korábban elmondottak szerint összehasonlítva, a gerjedési frekvenciákat megtalálva, paraméterezhetjük a megszakítási rutinban helyet foglaló lyukszűrőinket.

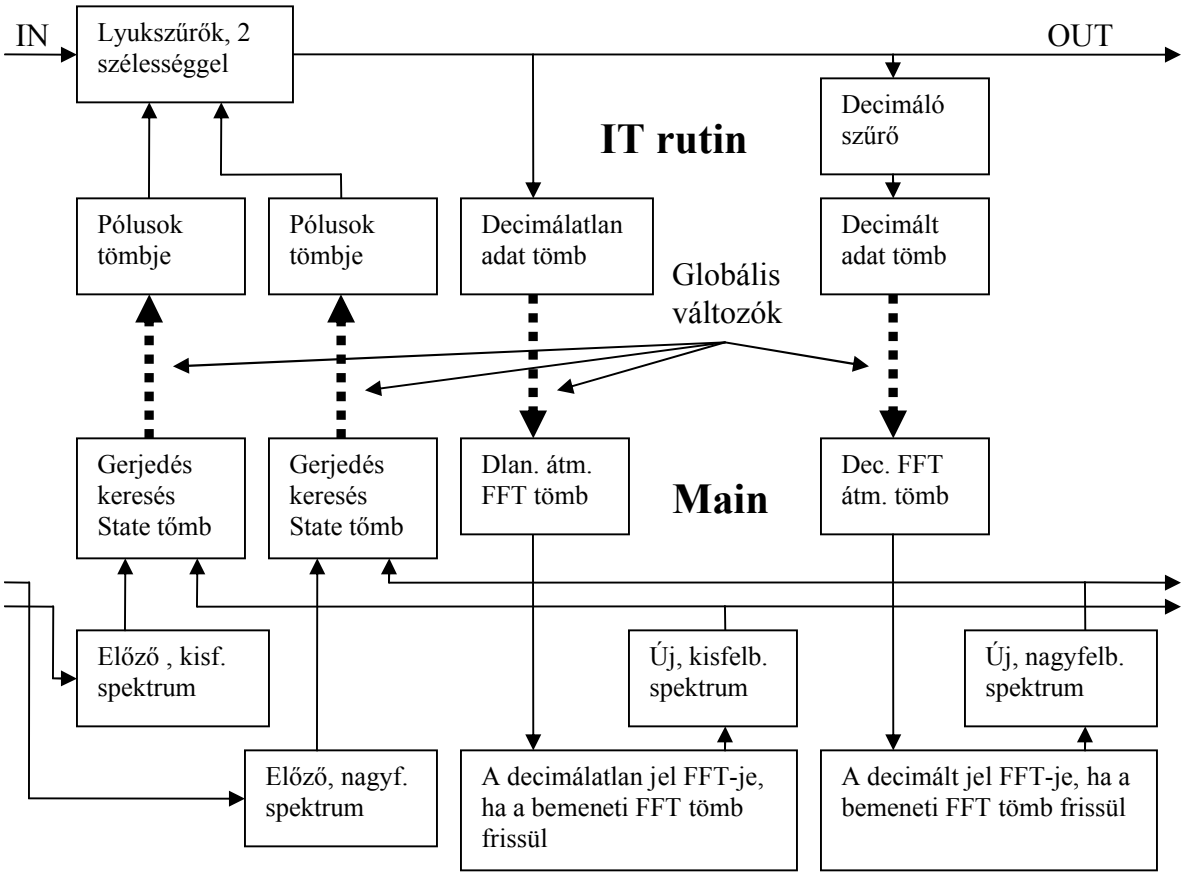


36. ábra. A program szerkezete. Az IT rutin és a programtörzs (Main) globális változók segítségével cserélnek adatokat.

A fenti rendszer működőképes, viszont a hangminőséggel adódhatnak problémák. A problémát orvosolandó megpróbálhatjuk az FFT pontszámát növelni. Viszont egy bizonyos pontszám felett elérjük a jelfeldolgozó processzor adatmemóriájának felső határát. A maximális FFT pontszám, amit meg tudunk valósítani 4096, mely 12 Hz körüli frekvenciafelbontást tesz lehetővé. Ezt mi nem találtuk elégségesnek.

A korábban tárgyalt kétlépcsős FFT módszerrel a kritikus, alacsonyfrekvenciás tartományban elérhető a 4.8 Hz-es felbontás, miközben a memória közel sem telik meg. A szoftver struktúráját azonban módosítani kell. A megszakítási rutinban ezentúl egy decimáló szűrőt megvalósító szoftverkomponens is elhelyezésre kerül. Mivel mi a mintavételi frekvenciát tizedeltük, ez a komponens minden tizedik megszakítás alkalmával lefut és előállít egy újabb mintát, melyet a decimálatlan mintához hasonlóan

egy FFT pontszámának megfelelő tömbbe tesz. Amint a tömbünk megtelt, transzformáltat képezhetünk, mely felbontása a fentieknek megfelelően a decimálatlan jel transzformáltjának tízszerese lesz a 0-2,4 kHz-es tartományban. Itt is azonban egy további probléma is felmerül. Mivel a mi megvalósításunkban az FFT pontszáma 1024 volt, a decimálás pedig tizedelő, a decimált jel transzformáltjához szükséges bemeneti vektorhoz 10240 minta szükséges, ami másodpercenként csupán kb. 5 spektrumszámítást tenne lehetővé az alsó frekvenciatartományban és ez a gerjedésdetektálás sebességét, hatásfokát túlzottan lerontaná. Viszont nem muszáj mindig „teljesen" friss mintákból FFT képezni. Megoldás a mozgóablakos FFT számolás lett. Minden 100. decimált mintát követően - ami körülbelül minden 1024. decimálatlan mintának felel meg – képezzük mind a decimált, mind a decimálatlan jel transzformáltját. Mivel mindkét jel bemenetét eleve cirkuláris bufferben tároljuk, az FFT átadandó bemeneti tömb előállítása nem okoz gondot. A decimált jel FFT-nek átadott bemeneti tömbje csak 100 új változót tárol, a többi az azt megelőző 904 minta. Ilyen módon elérhető a másodpercenként 40 transzformáció mindkét frekvenciatartományban. A fent említett „maxstate" változót 10-re választva a gerjedésdetektálás, illetve megszüntetés 0.25 s.



37. ábra. A módosított program szerkezete.

5. Mérési eredmények

A méréseket a 29. ábra szerinti elrendezésben valósítottuk meg, annyi különbséggel, hogy a jelútba helyeztünk egy Brüel & Kjaer 2638 típusú erősítőt, melyen 1 dB felbontással lehetett az erősítést állítani. A mérés elve mindegyik esetben ugyanaz volt, azaz először a gerjedésgátló rendszer működése nélkül a gerjedés határára vezéreltük a rendszert, majd megmértük, hogy a gerjedésgátló rendszer működése mellett mennyi többleterősítést lehet elérni. A méréseket az I.E.317-es teremben végeztük.

5.1. Az LMS-algoritmussal megvalósított off-line módszer

Amikor ennek, valamint az IIR LMS a rendszernek a hatékonyságáról beszélünk, akkor alapvetően az identifikáció és a kioltás sikeressége fontos. Már az algoritmus implementációjánál említettük a számábrázolás kiemelt szerepét a rendszer működésében. Az LMS-algoritmussal megvalósított off-line rendszert mindkettő számábrázolási módszerrel implementáltuk. Ahogy azt már tárgyaltuk a 4.1.2 alfejezetben a 32 bites felbontás segítségével sikeresebb volt az identifikáció. Igaz, hogy a legfeljebb 185 fokszámú szűrő nem képes csak korlátozott hosszúságú impulzusválaszokat identifikálni, ugyanakkor a kioltásban nagyobb szerepet játszik a számábrázolás. Ezt az eredmények is igazolják, mivel a fract16-os szűrő véleményünk szerint a számábrázolás pontatlansága miatt nem képes a kioltás üzemmódban hatásosan működni. A 32 bites számokkal dolgozó kioltással azonban sikerült 3 dB erősítéstöbbletet elérni, melyet a bypass üzemmód segítségével demonstrálni is tudunk, hiszen ha a kioltás módról átváltunk, akkor a rendszer ismét gerjedni kezd.

5.2. Az IIR LMS algoritmussal megvalósított off-line módszer

Az IIR LMS algoritmussal megvalósított off-line módszer azonos keretek között működik, csupán az identifikáció módja tér el. Ebben az esetben is 32 bites számokkal dolgoztunk. Az 5.1. alfejezetben leírtak szerint itt is az identifikáció a kritikus. A hibajel értéke alacsonyabb a sima LMS algoritmusos módszernél és ez az erősítéstöbbletben is jelentkezik, hiszen itt 4 dB-t tudtunk elérni. Fontos továbbá megjegyezni, hogy ha a rendszer elegendő fokszámú ahhoz, hogy az A/D és a D/A átalakító késleltetését leküzdje, akkor megmutatkozik az IIR szűrő legnagyobb előnye: a mikrofon távolságának változtatására érzéketlenebb, mint az egyszerű FIR szűrő. Az erősítéstöbbletet nem korlátozza olyan mértékben a megvalósított szűrő fokszáma, mint véges impulzusválaszú szűrő esetén.

5.3. Az off-line, lyukszűrőket felhasználó módszer

Ha frekvenciamérés pontos, keskeny sávú lyukszűrőket alkalmazhatunk, ami a hangképet kevésbé befolyásolja, de a gerjedés frekvenciáján vágva megszünteti a gerjedést. A mérés során a frekvenciamérés időtartalmát olyan hosszúra kellett megválasztani, mely kellően pontos mérést tesz lehetővé, viszont a mérés sem válik túl hosszadalmassá. Ezt a feltételt a minden frekvenciát 4 másodpercig mérő megoldás elégítette ki. A pontos mérés teszi lehetővé a pontos szűrőparaméterezést, ami a hatékony gerjedéselnyomáshoz nélkülözhetetlen. 6 Hz széles lyukszűrőket használtunk, melyek elnyomását 30 dB-re állítottuk. A szűrőbank összesen 10 db ilyen szűrőt tartalmazott. A korábbiakban tárgyalt módon konfiguráltuk a rendszert, azaz a gerjedési frekvenciákat mérve felparamétereztük a 10 szűrőt. Az általunk mért erősítéstöbblet 12 dB.

5.4. Az on-line, lyukszűrőket felhasználó módszer

A rendszer működését többféle módon vizsgáltuk. Először is nagyon fontos szempont a rendszer szelektivitása, mert ha a hasznos hanganyagot (emberi beszédet, zenét) is gerjedésnek ítéli, akkor az ellehetetleníti a műsort. Ha a gerjedés állapotát figyelő változó értékét túl kicsire választjuk, akkor a beszédet is kiszűrik a lyukszűrők. A rendszer erősítését tehát olyan paraméterek mellett kellett megmérni, melyek mellett a rendszer megfelelő szelektivitással tud működni.
A mérés során olyan mikrofon-hangsugárzó összeállításra törekedtünk, mely megfelelően modellez valamilyen életszerű helyzetet, például azt, amikor egy beszélő a felé fordított mikrofonba beszél, és a terem túloldalán vele szemben helyezkednek el a hangszórók. Ebben a konfigurációban, 10 - 10 vándorló, alacsony- és magasfrekvenciás lyukszűrőt megvalósítva az elért erősítéstöbblet az igen meggyőző 11 dB-t is átlépte.

LMS FIR	3 dB
LMS IIR	4 dB
Off-line lyukszűrő	12 dB
On-line lyukszűrő	11 dB

1. táblázat. A mérési eredmények összefoglalása.

6. Összefoglalás

Dolgozatunkban egy hangosító rendszerekben gyakran fellépő probléma, az akusztikai visszacsatolás következtében fellépő gerjedés kiküszöbölésének digitális jelfeldolgozó eszközöket felhasználó megoldásait vizsgáltuk. Tisztáztuk a gerjedés kialakulásának feltételeit, majd a kínálkozó megoldásokat vizsgáltuk részletesen. Ezen megoldások off-line, illetve on-line csoportokra oszthatók. Off-line módszerek azok, melyek a műsor előtt megmért, statikus akusztikai út esetén próbálják a kiadható hangteljesítményt maximalizálni. A megvalósítás adaptív algoritmusokon alapuló, valamint lyukszűrőket felhasználó lehetőségeit részletesen elemeztük. Az adaptív transzverzális szűrők (FIR-LMS) megvalósításának számítási igényéből adódó korlátok miatt adaptív rekurzív (IIR-LMS) szűrők megvalósítási lehetőségeit is vizsgáltuk. Ez utóbbi témakört ítéljük a dolgozat egyik legperspektivikusabb részének, mert rekurzív adaptív algoritmusok implementációja nem triviális jelfeldolgozási feladat.

Az elméleti eredmények bemutatása után ismertettük a tárgyalt megoldások egy részének implementációját ADSP-BF537 16 bites fixpontos processzort tartalmazó fejlesztőkártyán. Az implementációkat teszteltük, hatékonyságukat elemeztük. A téma még számos, kutatási és implementációs feladatot ad számunkra. Tudományos szempontból, az adaptív rekurzív szűrők mellett a legkecsegtetőbbek az on-line adaptív rendszerek, további kutatásaink során ezek felé fordulunk.

7. Irodalomjegyzék

[1] BME MIT Tanszéki Munkaközösség: *Digitális jelfeldolgozás*, Budapesti Műszaki Egyetem, segédlet, 2004.

[2] B. Widrow, S.D.Stearns: *Adaptive Signal Processing*, Prentice-Hall, Inc 1985.

[3] Thaier N. Mohammad: IIR *adaptive filtering methods and algorithms*, Budapesti Műszaki Egyetem, 1992.

[4] Deependra Talla, Sathyanarayan S. Rao,Lizy K. John: *An Evolutionary Computation Embedded IIR LMS Algorithm*, University of Texas.

[5] *Alto AMX – 140 User's Manual*, www.altoaudio.com

[6] RaneNote Tutorials & Reprinted Published Articles: *Understanding Acoustic Feedback & Suppressors*, http://www.rane.com/library.html

[7] Texas Instruments: Design Support, What is a Notch Filter?, http://www-k.ext.ti.com/SRVS/Data/ti/KnowledgeBases/analog/document/faqs/notch.htm

[8] Sabine FBX Feedback Exterminators, *www.sabine.com*

[9] Péceli Gábor: A common structure for recursive discrete transforms, 1986

[10] ADSP-BF53x/BF56x Blackfin® Processor Programming Reference

[11] VisualDSP++ 4.5 C/C++ Compiler and Library Manual for Blackfin Processors