Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Adaptív FIR-szűrők hatékony megvalósítása aktív zajcsökkentő alkalmazásokhoz

Diplomaterv

Készítette
Szőke Kálmán Benjamin

Konzulens
Dr. Sujbert László

2016. május 29.

HALLGATÓI NYILATKOZAT

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

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

Budapest, 2016. május 29.

Szőke Kálmán Benjamin
hallgató

Köszönetnyilvánítás

Ezúton is szeretném megköszönni azoknak akik segítettek ennek a dolgozatnak a megvalósításában. Mindenek előtt szeretném megköszönni témavezetőmnek *Dr. Sujbert Lászlónak* a támogatását, útmutatását, és türelmét, mely végig kísért munkám során. Különös köszönet illeti *Orosz Györgyöt*, akitől rengeteget tanultam a fejlesztőkörnyezet programozási technikáiról, és mindig készségesen segített problémáimban.

Ezenkívül köszönöm mindazoknak, akik elolvasták a készülő dolgozatot, és megjegyzéseikkel segítették annak befejezését. Köszönet barátaimnak akik a nehezebb időszakokban is fenntartották lelkesedésemet, és kitartásomat.

Kiemelt köszönet illeti szüleimet, hogy az egyetem alatt végig támogattak tanulmányaimban, hogy megvalósulhasson ez a munka.

Tartalomjegyzék

Ábrák jegyzéke

Kivonat

Akusztikus zajok, zavarhatások elnyomására, az alacsony frekvenciás tartományban a destruktív interferencia elvén működő aktív zajcsökkentés egy nagyon ígéretes megoldási módszert tud biztosítani. Ezek nagy konvergenciasebességű, nagy fokszámú adaptív szűrők megvalósítását igénylik, amelyek megvalósítása nem triviális és igen nagy komplexitásúak lehetnek. A mai jelfeldolgozó processzorok számítási teljesítménye mellett néhány kilohertz mintavételi frekvencián valós időben néhány ezer együtthatós adaptív FIR-szűrő futtatható. Az aktív zajcsökkentési alkalmazások viszont megkívánják a nagyobb fokszámú szűrők alkalmazását, és akár a magasabb mintavételi frekvenciát is. Ekkor az alapegyenletek implementálása eredeti formájukban nem célravezető, a frekvenciatartományban kell a szükséges konvolúciót elvégezni. A módszer alapja, hogy a diszkrét Fourier-transzformáció az FFT algoritmussal gyorsan végrehajtható, és elegendően nagy pontszám esetén a konvolúció kevesebb műveletet igényel, mint az időtartományban. Ezt a lehetőséget használják ki az OLA (overlap and add) algoritmusok, ezek azonban a transzformáció adatgyűjtése és végrehajtása miatt jelentős késleltetéssel szolgáltatják a kimeneti adatokat.

A szakirodalom számos frekvenciatartománybeli számítási módszert kínál FIR-szűrőkhöz, amelyek a kimenet és a bemenet közti késleltetést hivatottak csökkenteni. Ezeket tanulmányozva egy olyan eljárást mutatok be, amely az idő- és a frekvenciatartománybeli számítás együttes futtatásával küszöböli ki a transzformáció adatgyűjtéséből származó késleltetést. Ez az eljárás már a frekvenciatartománybeli számításhoz használt OLA algoritmus adatgyűjtési ideje alatt is szolgáltat szűrési eredményt, az időtartományban végzett részleges konvolúció és az előző OLA blokkokból származó részeredmények segítségével. A módszernek köszönhetően adaptív FIR-szűrők alkalmazásakor is lehetséges az adatgyűjtésből származó késleltetések teljes mértékű megszüntetése, valamint továbbra is gyorsabb, mint számos más időtartománybeli eljárás, amiknél ilyen fajta késleltetés nem lép fel.

Ezt az idő- és frekvenciatartománybeli késleltetésmentesen működő adaptív FIR-szűrőt MATLAB-ban és ADSP-21364 jelprocesszoron valósítottam meg. Ehhez először implementáltam a feladathoz hatékonyan használható frekvenciatartománybeli adaptációt végző Fast Block LMS algoritmust, majd ezt és a konvolúcióval kiegészített OLA algoritmust egy jelprocesszoros fejlesztőkártyán kialakított, adaptív FIR-szűrőt működtető aktív zajcsökkentő struktúrába integráltam. Ezen a kísérleti alkalmazáson méréseket végeztem olyan impulzusszerű, metronómból származó csattogó zajok és fehérzajok elnyomására, amelyeken jól demonstrálható a nagy fokszámú és magas mintavételi frekvencián működő adaptív FIR-szűrők szükségszerűsége.

Abstract

The active noise control is a quite promisi method to reduce acoustic noise in the low frequency range. The implementation of adaptive filters is not trivial and very high complexity with high convergence rate, large number of taps and high sampling rate, required by active noise control applications. Several signal processors are able to run an adaptive FIR filter with few thousands of coefficients and few kilohertz sampling rate in real-time. Some tasks require the use of filters with more taps or higher sampling frequency, in this case the basic equations are not useful for implementation in their original form, therefore, we may need to do the necessary convolution in the frequency domain. The method is based on the fast Fourier transform (FFT) algorithm, as for large number of taps the convolution requires less operations than that in the time domain. This capability is used in the OLA (overlap and add) algorithms, however, they introduce a significant delay of the output, due to the data collection of the transformation. In general, this delay is not tolerated in many applications of adaptive filters, so in my work I have been looking for a solution to this problem.

There are a number of frequency-domain calculation methods for FIR filters published in the literature, which are designed to reduce the latency between the input and the output. Based on the known methods I introduce one that is able to eliminate this latency by a smart calculation of the convolution both in the time and the frequency. The method I have chosen provides filtering results even in the data collection period, thanks to the time domain convolution and partial results of the previous OLA's blocks. Besides the latency can be eliminated entirely, the calculation is still faster than other time domain filtering methods, where such delay does not occur.

I have implemented this algorithm in MATLAB environment and on an Analog Devices Sharc (ADSP-21364) DSP. After that I have embedded it to an adaptive FIR filter application. To this end first I have implemented an FBLMS (Fast Block LMS) algorithm, which can perform the adaptation in the frequency domain. Then I have integrated this module and the smart OLA filtering method into an active noise control procedure. Finally, I made noise control measurements for white and impulsive noise to introduce the necessity of adaptive FIR filters with large number of taps and high sampling rate.

Bevezető

Motiváció

Manapság egyre több alkalmazás igényli az adaptív szűrők használatát, mint például aktív zajcsökkentő, visszhangcsökkentő rendszerek vagy épp méréstechnikai és szabályzástechnikai alkalmazások. Egyes feladatok, legfőképpen az akusztikai aktív zajcsökkentő alkalmazások megkívánják a nagyobb fokszámú szűrők alkalmazását, vagy magasabb mintavételi frekvenciát, ezért a dolgozatomban bemutatom, hogy az adaptív FIR-szűrők használata során milyen problémák és korlátok lépnek fel ilyen követelmények teljesítése esetén, és ezek milyen jelfeldolgozási módszerekkel elégíthetők ki és valósíthatók meg hatékonyan.

Ezek az adaptív szűrőt valós időben működtetető alkalmazások ma már számos fajta hardveren és programozási környezetben implementálhatók. A legelterjedtebbek az FPGA-n, GPGPU-n vagy épp digitális jelprocesszoron történő megvalósítások. A mai jelfeldolgozó processzorok számítási teljesítménye mellett néhány kilohertz mintavételi frekvencián valós időben néhány ezer együtthatós adaptív FIR-szűrő futtatható. Ez azt jelenti, hogy az időtartományban végzett együtthatók adaptálása és az ezekkel való szűrés a konvolúció segítségével, két mintavételezés között csak néhány ezer együtthatóra végezhető el. Annak érdekében, hogy az ily módon adódó maximális együtthatószámot túllépjük, a frekvenciatartományban kell a szükséges konvolúciót elvégezni a hatékonyság érdekében.

Ehhez adott a diszkrét Fourier-transzformáció, amely az FFT algoritmussal gyorsan végrehajtható, és elegendően nagy pontszám esetén így a konvolúció kevesebb műveletet igényel, mint az időtartományban. Azonban az FFT algoritmussal történő transzformáció blokkonkénti adatgyűjtést és végrehajtást igényel, ami előnyökkel és hátrányokkal is jár egyaránt. Előnye, hogy az adaptív FIR-szűrő konvergenciasebességét nagyban megnöveli az együtthatók frekvenciatérben történő blokkonkénti adaptációja, hátránya pedig az, hogy a blokkonkénti adatgyűjtés és a transzformációs művelet az aktív zajcsökkentő alkalmazásokban nem megengedhető késleltetést okoz az adaptív FIR-szűrő kimenetén.

Megoldási módszerek

Az időtartományban implementált direkt konvolúció kedvező tulajdonsága, hogy nem jár késleltetéssel, viszont a kedvezőtlen az, hogy igen számításigényes a szorzás/összeadás műveletei miatt. Nem mellesleg ez a számítási igény mintánként lineárisan növekedik a FIR-szűrő működtetése során, amíg a rendszer el nem éri az állandósult állapotot, amikor is a FIR-szűrő hosszával egyezik meg ezen műveletek száma [2]. Nagy fokszámú FIR-szűrők

esetén tehát a konvolúcióval megvalósított valós idejű szűrés nem célravezető megoldás.

Elegendően nagy pontszám esetén a frekvenciatartományban végzett konvolúció kevesebb műveletet igényel, mint az időtartományban. Ezt a lehetőséget használják ki az OLA (overlap-add) és az OLS (overlap-save) algoritmusok [3], melyek számítási igénye a direkt konvolúcióhoz képest sokkal kedvezőbb, mert ez logaritmikusan növekedik a blokkok méretének függvényében. Ennek a hatékony műveletvégzésnek azonban a kedvezőtlen tulajdonsága az, hogy a blokkonkénti adatgyűjtés késleltetést okoz, amely mértéke legalább akkora vagy nagyobb, mint a szűrés során alkalmazandó FIR-szűrő hossza [2]. E két módszer közös használatával a késleltetés megszüntethető/csökkenthető, miközben a számítási igény továbbra is kedvező szinten tartható [4].

William G. Gardner [2] bemutatott egy hatékony, nem egyenletesen particionált eljárást is, ami azon alapszik, hogy az impulzusválasz növekvő méretű blokkokra van felosztva. Ez azt használja ki, hogy a rövidebb blokk méretű impulzusválaszokkal való szűrések alacsony késleltetést okoznak, míg a hosszabbak, a frekvenciatartományban szűrve az FFT algoritmus által kevesebb műveletet igényelnek. Ezt a két kedvező tulajdonságot kihasználva a nem egyenletesen particionált impulzusválasszal való szűrés hatékonyabban alkalmazható a késleltetés megszüntetésére/csökkentésére.

Jelen diplomatervem a TDK dolgozatom folytatása amelyben egy olyan alapvető eljárást mutatok be, amely az idő- és a frekvenciatartománybeli számítás együttes futtatásával küszöböli ki a transzformáció adatgyűjtéséből származó késleltetést. Alapja, hogy a frekvenciatartománybeli számításhoz használt OLA algoritmus adatgyűjtési ideje alatt párhuzamosan végzett időtartománybeli konvolúció, és az előző OLA blokkokból származó részeredmények képesek késleltetés nélkül szolgáltatni a szűrési eredményt. A bemutatott módszeremet eleinte szakmai publikációkban nem találtam meg, csak olyan másfajta eljárásokat találtam, amik a késleltetést csak csökkenteni képesek, megszüntetni nem. A részletes irodalomkutatás során feltűnt, hogy vélhetőleg a Lake DSP vállalat korai szabadalma [5] miatt, ez a szakmai terület részletesebben tárgyalt a szabadalmi kivonatokban, így ezek között folytattam az irodalomkutatásomat. Ennek köszönhetően végül egy 2012-ben közzétett szabadalomban [6] találtam rá az általam bemutatott módszerhez igen közel álló eljárásra, amely szintén foglalkozik némi hatékonyság vizsgálattal, és ezek igen jó egyezést mutatnak az én diplomatervemben elvégzett és ezen túlmutató munkámmal.

A diplomatervem folytatásában a további irodalomkutatás mellett, továbbfejlesztési szempontból megvalósítottam egy aktív zajcsökkentő rendszert is. Az általános akusztikus zajok legfőképp csak alacsony frekvenciás komponensekkel rendelkeznek, ezért az aktív zajcsökkentést fehérzajra és olyan csattogó, metronómból származó impulzusszerű zajelnyomására alkalmaztam, amelyek gyakorlatban történő zajelnyomásával igen kevés szakirodalmi publikáció foglalkozik [7]. Ezen szélessávú zajokkal jól tudtam tesztelni a nagy fokszámú és nagyobb mintavételi frekvenciát igénylő zajelnyomás lehetőségeit. Az ilyen jellegű zajok elnyomására alkalmazott aktív zajcsökkentő algoritmusoknál egy érdekes kérdés lehet az, hogy a megfelelően nagy fokszám megválasztása mellett mekkora szerepe van a jelfeldolgozó rendszerben alkalmazott mintavételi frekvencia gondos megválasztásának is, hogy a zajelnyomás minél hatékonyabb legyen.

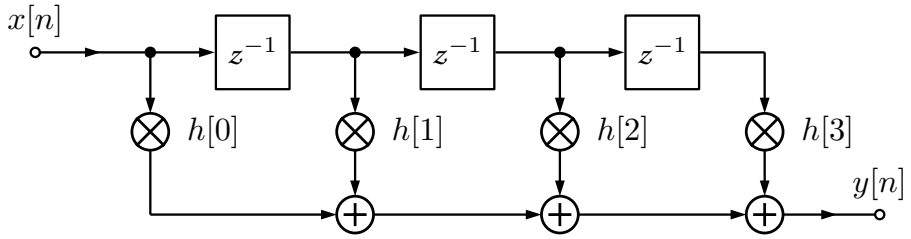
1. fejezet

Idő- és frekvenciatartománybeli FIR-szűrés

1.1. Időtartománybeli szűrés a direkt konvolúcióval

Egy FIR-szűrővel való időtartománybeli szűrésnél a bemeneti jel $x(n)$ előző értékeire és a szűrő impulzusválaszára h_i van szükségünk. A szűrési művelet matematikailag a konvolúcióval történik, vagyis az N hosszúságú impulzusválaszunkkal direkt konvolúciót hajtunk végre a bemenet aktuális és előző $N-1$ mintáin. Egy n-dik minta kiszámolásánál a műveletét elképzelhetjük úgy is, hogy a bemenetünk régebbi mintáit késleltetőkön keresztül szorozzuk meg az impulzusválasz értékeivel, majd ezeket összegezzük, amivel eredményül megkapjuk az n-edik ütemben számolt eredményét a szűrésnek.

$$y(n) = \sum_{i=0}^{N-1} h_i x(n-i) = (h * x)(n) \tag{1.1}$$



1.1. ábra. *A direkt konvolúció hatásvázlata, 4 együtthatós FIR-szűrő esetén.*

A direkt konvolúcióval végrehajtott időtartománybeli szűrésnek előnye, hogy a konvolúciós művelet minden egyes új minta beérkezése után azonnal elvégezhető, így adatgyűjtésből származó késleltetés nem áll fenn. Hátránya ennek a módszernek viszont az, hogy N nagyságú impulzusválasz esetén a direkt konvolúciós művelet szorzásai és összeadásai $\mathcal{O}(N^2)$ műveletet igényelnek. Ebből látható, hogy nagy fokszámú FIR-szűrők esetén

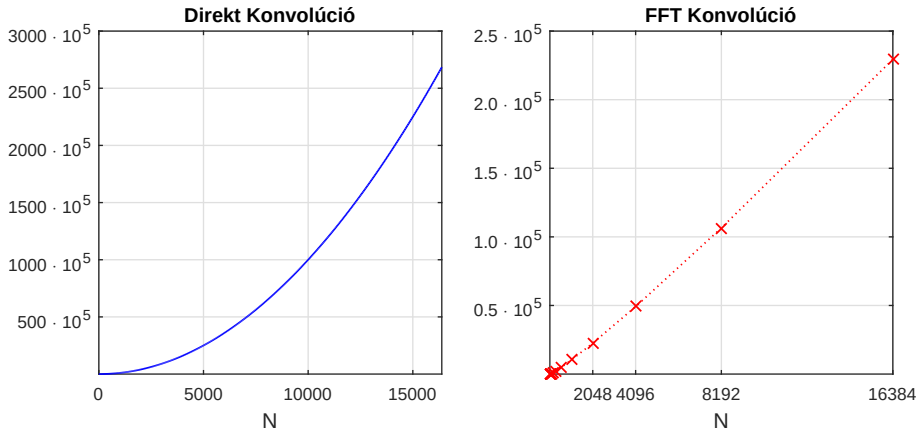
az időtartománybeli szűrés már gondot okozhat a nagy műveletigénye miatt, ezért célszerű megismerni frekvenciatartományban használható módszereket is.

1.2. Frekvenciatartománybeli szűrés az FFT algoritmussal

Lineáris rendszerek modellezése esetén az időtartománybeli konvolúció egyszerű szorzási műveletre egyszerűsíthető le a frekvenciatartományban. Ez azt jelenti, hogy a FIR-szűrő impulzusválaszának Fourier-transzformáltja és a bemeneti jelünk Fourier-transzformáltja szorzataként kiszámolhatjuk a szűrő kimenetének frekvenciatartománybeli eredményét, amely az időtartományba visszatranszformálva természetesen megegyezik az időtartománybeli szűrés eredményével.

$$\mathcal{F}\{\boldsymbol{b} * \boldsymbol{x}\} = \mathcal{F}\{\boldsymbol{b}\} \cdot \mathcal{F}\{\boldsymbol{x}\} \tag{1.2}$$

A diszkrét Fourier-transzformáció mátrixos koncepciójából következik, hogy N szám transzformációjához továbbra is $\mathcal{O}(N^2)$ szorzás szükséges, ami jól láthatóan megegyezik az előzőekben tárgyalt direkt konvolúciós eljárással. Azonban ezt a számítási igényt 1965-ben Cooley és Tukey matematikusok által kidolgozott gyors Fourier-transzformációs algoritmus [8] radikálisan lecsökkentette, amely forradalmat jelentett akkor a digitális jelfeldolgozásban. Az általuk kifejlesztett komplex szorzásokat végző lepkeműveletek lehetővé tették, hogy a teljes transzformáció elvégezhető $\mathcal{O}(Nlog_2(N))$ szorzási műveletből. Tipikusan jelprocesszorokon a Radix-2 FFT algoritmus érhető el, aminek megkötése mindössze annyi, hogy az algoritmus csak a kettő hatványaival megegyező nagyságú adatsorokat képes transzformálni a gyors Fourier-transzformáció során.



1.2. ábra. *Direkt konvolúció N^2 és az FFT algoritmussal számolt frekvenciatartománybeli konvolúció $N\log_2(N)$ műveletigénye.*

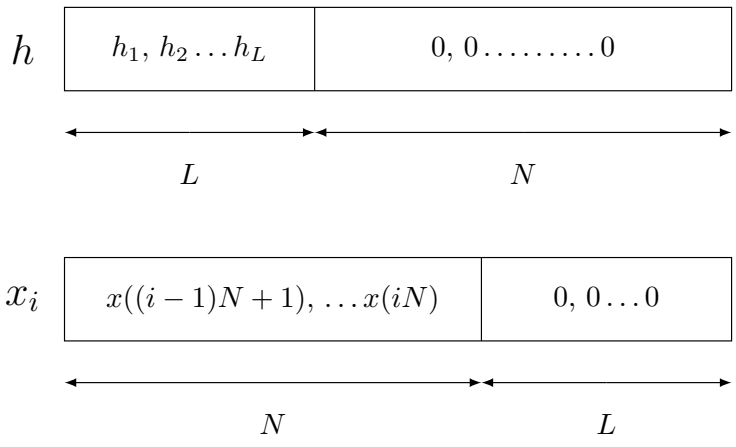
Tovább elemezve a gyors Fourier-transzformáció alkalmazási lehetőségeit, észrevehetjük, hogy ez a művelet nem alkalmas mintánkénti végrehajtásra. Vagyis valós idejű FIR-szűréshez csak úgy alkalmazható, ha a transzformációs számításokat megelőzően már elegendő adatot rögzítettünk ahhoz, hogy blokkonként végezhessük el a frekvenciatartománybeli szűrést. A valós idejű FIR-szűréshez számos ilyen blokkonkénti gyors Fourier-

transzformációs algoritmus lett kifejlesztve, ezek közül is a témához legfontosabbak az Overlap-Add és Overlap-Save algoritmusok, melyeket a továbbiakban ismertetek.

1.2.1. Overlap-Add módszer

Az Overlap-Add az egyike azon blokkonkénti gyors Fourier-transzformációt használó eljárásoknak, amik hatékonyan használhatók valós idejű FIR-szűréshez [9]. Ennél a módszernél a bemeneti adatokat $x_i(n)$ blokkokba rögzítjük, melyek méretei $N_{fft} = N+L$ nagyságúak. Az adatgyűjtés során egy OLA blokkba N mintát rögzítünk, majd a fennmaradó L nagyságú részt, amely megegyezik a szűrés során használt FIR szűrő impulzusválasz hosszával, nulla elemekkel töltjük fel. A módszer során a cél az, hogy az N hasznos adattal kitöltött blokkon a szűrést a frekvenciatartományban is elvégezhessük, és L nagyságú átlapolást és összegzést használva az y_i kimeneti eredményblokkok között, előállíthassuk a szűrés eredményét, hatékonyabban, mint azt a direkt konvolúcióval tehetnénk az időtartományban.

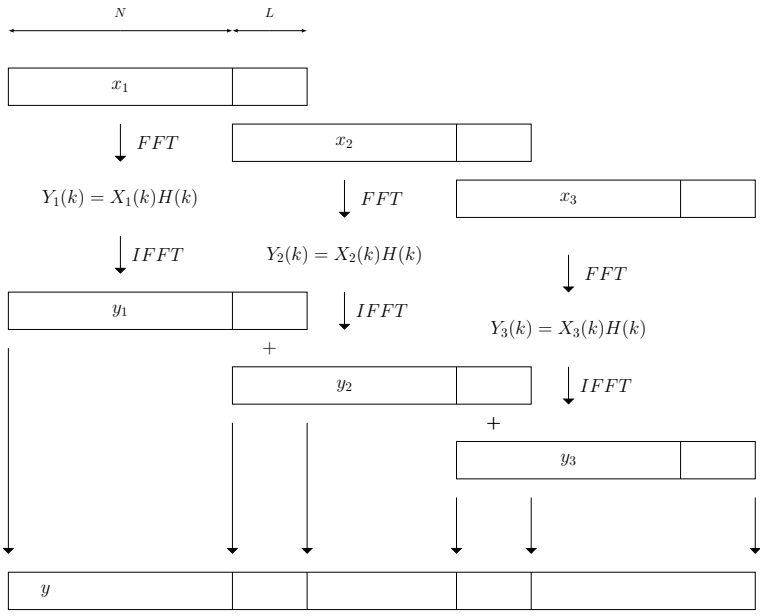
Ehhez az adatgyűjtések előtt a FIR-szűrő impulzusválaszának Fourier-transzformáltjára is szükségünk van, amelynek mérete az FFT algoritmus miatt egyaránt $N_{fft} = N + L$ kell hogy legyen. Ennek érdekében az impulzusválaszt az $x_i(n)$ bementi blokkokhoz hasonlóan tároljuk, csak épp az $N + L$ felosztást fordítva alkalmazzuk. Itt az L nagyságú $h(n)$ impulzusválasz végét egészítjük ki annyi nullával, hogy a Fourier-transzformációhoz megfelelően N_{fft} méretű tömb álljon a rendelkezésünkre, vagyis ekkor az L mintából álló impulzusválaszunk tömbjét N darab nulla elemmel egészítjük ki. Az ily módon történő felosztás után még a szűrés előtt az FFT algoritmus segítségével hatékonyan előállíthatjuk a FIR-szűrőnk frekvenciatartománybeli diszkrét átviteli függvényét $H(k)$-t, amelyre a szűrés során folyamatosan szükségünk lesz.



1.3. ábra. *Az $x_i(n)$ és $h(n)$ felosztása az Overlap-Add eljárás során*

Az $x_i(n)$ és $h(n)$ megfelelő $N_{fft} = N + L$ felosztása után az Overlap-Add módszerrel való valós idejű FIR-szűrés a következő lépések és ábrák szerint történik.

1. A FIR-szűrés megkezdése előtt létrehozzuk $h(n)$ impulzusválasz $N + L$ nagyságú



1.4. ábra. *Overlap-Add módszer blokkonkénti szűrése*

tömbjét, az előzőekben leírt felosztás szerint, majd az FFT algoritmussal kiszámoljuk ennek a Fourier-transzformáltját $H(k)$-t.

2. Az N minta rögzítése és az $x_i(n)$ blokkban használt $N + L$ felosztás helyes alkalmazása után, szintén az FFT-vel kiszámoljuk a bemeneti blokk Fourier-transzformáltját $X_i(k)$-t.

3. Az így kapott transzformáltak segítségével a i-edik blokk szűrését elvégezzük a frekvenciatartományban $Y_i(k) = X_i(k) \cdot H(k)$.

4. $Y_i(k)$-t inverz Fourier-transzformáljuk, amivel megkapjuk az i-edik bemeneti blokk szűrési eredményét az időtartományban, $y_i(n)$-t.

5. Az $y_i(n)$-nél használt L nagyságú átlapolás miatt az ábra szerint az előző $y_{i-1}(n)$ eredmény blokk utolsó L elemét hozzáadjuk $y_i(n)$ eredményblokk első L eleméhez. Ezzel az i-edik és az $i-1$-edik blokk átlapolási tartományában megkapjuk ugyanazt a szűrési eredményt, mint amit a direkt konvolúcióval kapnánk, vagyis ezek azok a minták a blokkos szűrés során amelyek a valós idejű szűrés alkalmazásakor a kimeneti mintákat szolgáltat a következő blokkos végrehajtás elkészültéig.

6. Az $y_i(n)$ átlapolási tartományába eső L értékeket elmentjük a következő $y_{i+1}(n)$ számításához, majd folytatjuk mintavételezést az $x_{i+1}(n)$ bementi blokkba, a 2. lépéstől folytatva.

A módszer bemutatásához olyan ábrákat használtam, ahol ez az L nagyságú átlapolás 50%-nál kisebb, de fontos megjegyezni, hogy ennek mértéke lehet ennél jóval nagyobb is. Ez akkor fordulhat elő, amikor használni kívánunk egy akkora impulzusválaszt, aminek L fokszáma nagyobb, mint az FFT algoritmushoz tervezett N_{fft} blokk méretének fele.

Ekkor fontos azt észben tartanunk, hogy ilyen esetben már nemcsak két $y_i(n)$ eredmény blokk lapolódik át, hanem a nagyobb L függvényében akár több is.

50%-nál nagyobb átlapolást használva, a kimenetnek szánt minták száma is változik az L növelésére. Ekkor az összegzések után az $y_i(n)$ blokknak csak az első N eleme az, ami megegyezik a direkt konvolúcióval történő szűrés eredményével, így ilyenkor a következő blokkos végrehajtás elkészültéig csak ez az N darab minta használható kimeneti adatként. Ebből még egy utolsó fontos dolog is adódik, amit fontos megemlíteni, mégpedig az, hogy ennek az N darab mintának a kiadása alatt, vagyis Nf_s valós idő alatt a következőleg elkezdett $y_{i+1}(n)$ kimeneti blokk eredményének mindenképpen el kell készülnie, hogy az eljárás folyamatosan és megszakítás nélkül tudja szolgáltatni az egymás után sorban következő N nagyságú kimeneti eredményeket.

Az FFT algoritmus előnyeit kihasználva az Overlap-Add módszer $\mathcal{O}(N_{fft}log_2(N_{fft}))$ műveletigénnyel bír, ami sokkal kedvezőbb annál, mintha direkt konvolúciót használnánk. Kedvezőtlen tulajdonsága viszont a blokkos végrehajtásból adódó N mintányi késleltetése, és a memóriaigényessége, ami abból adódik, hogy a módszer során tárolnunk a $H(k)$ komplex tömböt, és az előző $y_{i-1}(n)$ blokk eredményeket az átlapolásnál használt összegzés miatt.

1.2.2. Overlap-Save módszer

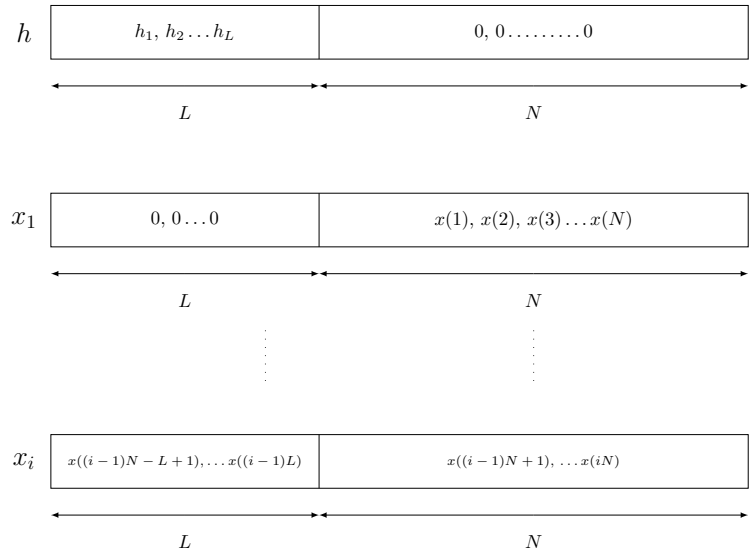
Az Overlap-Add módszernél mutatkozó memóriaigényt hivatott csökkenteni az Overlap-Save eljárás. Célja az, hogy a blokkos szűrés végrehajtás alatt ne legyen szükség az $y_i(n)$ kimeneti blokkok átlapolására, és az ebből adódó összegzésre. Ennek érdekében bemeneti adatokat $x_i(n)$ blokkokba egy másfajta $N_{fft} = L + N$ felosztás szerint rögzítjük, és az átlapolást most nem az $y_i(n)$ blokk eredményekre alkalmazzuk, hanem a bemeneti $x_i(n)$ blokkok között valósítjuk meg.

Ennél az eljárásnál $x_i(n)$ blokkot úgy osztjuk fel, hogy az új mintákat a blokk második N minta nagyságú része tartalmazza, mégpedig úgy, hogy ezt az új $x_i(n)$ blokkot a mintavételezés során az $N_{fft} - N$-edik helytől kezdjük el feltölteni. Az $x_i(n)$ blokk első L nagyságú része pedig az aktuális blokkok előtti mintavételezett bemeneti értékeket tartalmazza, vagyis itt az átlapolás olyan formában jelenik meg, hogy az $x_i(n)$ blokkban tárolt értékeket a következő $x_{i+1}(n)$ blokkban mindig N mintányit balra eltoljuk, és az így szabadon maradt N nagyságú helyre rögzítjük a bemenet új értékeit. A szűrés elején az $x_1(n)$ blokk első L elemét értelemszerűen nulla elemekkel tölthetjük fel.

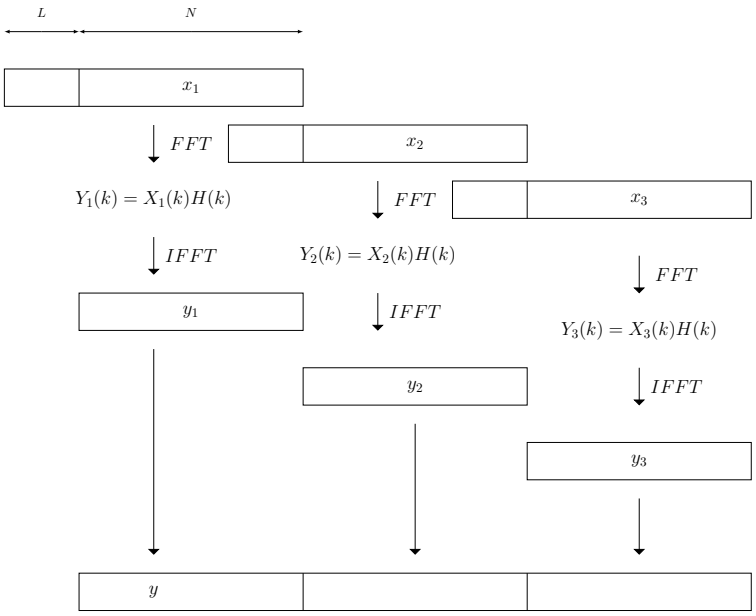
Ez az eltérő fajta blokkos szervezés az impulzusválasz tömbjének felosztásában viszont nem igényel különböző felosztást az Overlap-Add módszernél megismerthez képest. Továbbra is a $h(n)$ $N + L$ nagyságú tömbjét először az L darab FIR-szűrő együtthatóval töltjük fel, majd a maradék N mintányi helyre nulla értékeket tárolunk el.

Az $x_i(n)$ és $h(n)$ megfelelő felosztása után az Overlap-Save módszerrel való valósidejű FIR-szűrés a következő lépések és ábrák szerint történik.

1. A FIR-szűrés megkezdése előtt létrehozzuk $h(n)$ impulzusválasz $N + L$ nagyságú tömbjét, az előzőekben leírt felosztás szerint, majd az FFT algoritmussal kiszámoljuk



1.5. ábra. *Az $x_i(n)$ és $h(n)$ felosztása az Overlap-Save eljárás során*



1.6. ábra. *Overlap-Save módszer blokkonkénti szűrése*

ennek a Fourier-transzformáltját, $H(k)$-t.

2. Az N minta rögzítése és az $x_i(n)$ blokkban használt $N + L$ felosztás helyes alkalmazása után, szintén az FFT-vel kiszámoljuk a bemeneti blokk Fourier-transzformáltját, $X_i(k)$-t.

3. Az így kapott transzformáltak segítségével a i-edik blokk szűrését elvégezzük a frekvenciatartományban, $Y_i(k) = X_i(k) \cdot H(k)$.

4. $Y_i(k)$-t inverz Fourier-transzformáljuk, amivel megkapjuk az i-edik bemeneti blokk szűrési eredményét az időtartományban, $y_i(n)$-t.

5. A bemeneti blokkokon alkalmazott átlapolás hatására az $y_i(n)$ eredmény blokk utolsó N mintája egyezik meg a direkt konvolúcióval történő eljárással, így ez az utolsó N adat az, amit a valósidejű szűrő kimenetén használhatunk a következő $x_{i+1}(n)$ blokk szűrésének elkészültéig.

6. Az $x_i(n)$ bementi blokk első N elemét elhagyjuk, és a megmaradt $N_{fft} - N$ adatot a következő $x_{i+1}(n)$ blokkban használjuk fel az első L mintának, majd folytatjuk az eljárást az $x_{i+1}(n)$ blokkra is, a 2. lépéstől.

Az Overlap-Save módszerrel a blokkos végrehajtásból adódó N mintányi késleltetés továbbra is megmarad, de a hatékonyabb átlapolási módszernek köszönhetően a memóriaigénye kevesebb, mint az Overlap-Add módszernek.

1.3. Késleltetés nélküli szűrés idő- és a frekvenciatartománybeli eljárás együttes használatával

Az előző blokkonkénti végrehajtást használó Overlap-Save és Overlap-Add eljárásnál láttuk, hogy ezeket használva a valósidejű FIR-szűrés csak N minta késleltetése után szolgáltat eredményt. A szakirodalom számos frekvenciatartománybeli számítási módszert kínál FIR-szűrőkhöz, amelyek az adatgyűjtésből eredő kimenet és a bemenet közti késleltetést hivatottak csökkenteni. William Gardner cikkére [2] alapozva egy olyan módszert mutatok be, ami az idő- és a frekvenciatartománybeli számítás együttes futtatásával küszöböli ki a transzformáció adatgyűjtéséből származó késleltetést.

Az általam javasolt módszer Gardner módszeréhez képest abban tér el, hogy az impulzusválaszt particionálatlanul használom fel a szűrés során, hogy az eljárás minél kevésbé legyen bonyolult, vagyis nagy komplexitású, és ennek révén könnyebb legyen a használhatósága adaptív FIR-szűrőt használó rendszerekben is. Ez a késleltetés nélküli szűrés azon alapszik, hogy az Overlap-Add módszerrel való frekvenciatartománybeli konvolúciót kiegészítem egy parciális konvolúciót használó időtartománybeli szűréssel.

A módszer úgy működik, hogy az Overlap-Add eljárásnál használt $y_{i-1}(n)$ előző kimeneti blokk eredmények átlapolási részéhez nem a következő $y_i(n)$ blokk ezen részhez tartozó eredményeit adjuk hozzá (mivel ezek még nem állnak rendelkezésre), hanem ezeket előállítjuk még az adatgyűjtés ideje alatt egy parciális konvolúcióval $x_i(n)$ blokk első N éppen mintavételezet értékeiből. A parciális konvolúció alatt ebben az esetben azt kell érteni, hogy az $x_i(n)$ első N elemére a konvolúciós szummát mindig csak az adott n-edik indexszámmal megegyező FIR-szűrő együtthatóig végzzük el minden egyes új mintavételezési értékre. A tömbök indexelésének kezdetét nullától értelmezve ez a következőképen írható le matematikailag:

$$y_p(n) = \sum_{j=0}^{n} h_j x_i(n-j) \qquad n = 0, 1, 2 \ldots N-1 \tag{1.3}$$

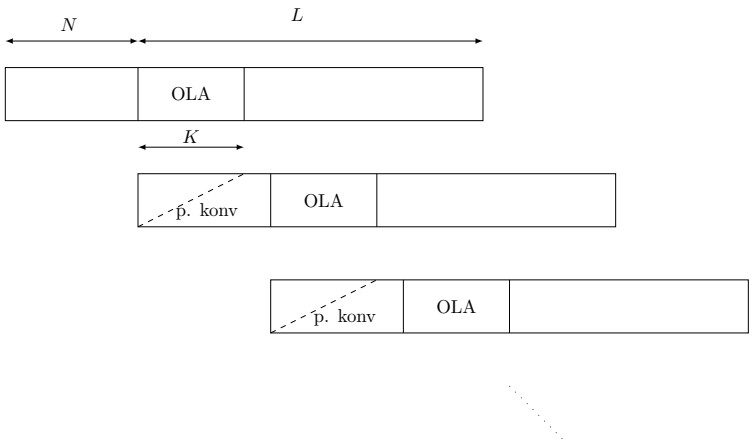
Amint az $x_i(n)$ n-edik mintájára elkészül ez az $y_p(n)$ konvolúciós eredmény, az előző $y_{i-1}(n)$ blokk átlapolási részének $y_{i-1}(N+n)$ elemének hozzáadásával megkaphatjuk a

helyes szűrési eredményt, már az adott n-edik minta beérkezése után azonnal. Ennek az eljárásnak a helyessége azzal igazolható, hogy az Overlap-Add eljárásnál az $y_i(n)$ átlapolási tartománya pont azt a részeredményt tartalmazza, amit a parciális konvolúció során $y_p(n)$ kiszámolásánál még nem végeztünk el.

Matematikailag ez az eredmény azon alapszik, hogy a konvolúciós művelet disztributív tulajdonságú, tehát $y(n)$ szűrési eredmény két külön végzett konvolúciós szumma segítségével is előállítható. (Ekkor $h(n)$ az OLA módszernek megfelelően N_{fft} nagyságú és $L+N$ felosztású, vagyis az utolsó N eleme nulla!)

$$y(n) = \sum_{j=0}^{n} h_j x_i(n-j) + \sum_{k=n+1}^{N_{fft}-1} h_k x_i(n-k) = y_p(n) + y_{i-1}(N+n) \tag{1.4}$$

$$n = 0, 1, 2 \ldots N-1$$



1.7. ábra. *Valós időjű késleltetés nélküli szűrés Overlap-Add módszerrel*

A valós idejű FIR-szűrés során persze azzal is számolnunk kell, hogy az $x_i(n)$ első N mintájának mintavételezése után az FFT algoritmussal végzett frekvenciatartománybeli szűrésnek is még időre van szüksége, ahhoz, hogy elkészüljön, és így szolgáltatni tudja a következő $x_{i+1}(n)$ blokknál végzett parciális konvolúció kiegészítéséhez szükséges $y_i(n)$ i-edik blokk szűrési eredményét. Tehát valós időben gondoskodnunk kell arról is, hogy míg az FFT algoritmus dolgozik, továbbra is legyen eredménye a FIR-szűrésnek. Ehhez az eddigieket csak kis mértékben kell kiegészíteni azzal, hogy az új $x_{i+1}(n)$ blokk N nagyságú részének első néhány eleménél még a konvolúciós szummázást folytatjuk $N+K$ FIR-szűrő együtthatóig, ahol K azon minták darabszáma, amennyi adat rögzítése után majd elvégződik az FFT algoritmussal való szűrés $x_i(n)$-re. A valós idejű késleltetésmentes FIR-szűrés érdekében használt parciális konvolúció ezzel a kiegészítéssel így módosul, az állandósult állapotban leírva:

$$y_p(n) = \sum_{j=K}^{n} h_j x_i(n-j) \tag{1.5}$$

$$n = K, K+1, K+2 \ldots N-1+K$$

A valós időjű működés ezen kiegészítése annyiban korlátozza az eljárást, hogy $y_i(n)$ és $y_{i-1}(n)$ közötti átlapolásnak nagyobbnak kell lennie, mint 50%. Ezt azért kell alkalmazni, hogy az N minta ideje alatt működő parciális konvolúciók fel tudják használni az előző OLA blokk szűrési eredményét. Kevesebb, mint 50%-os átlapolás esetén, az OLA szűrésének ideje tovább tartana, mint az átlapolási tartomány, és ekkor már a párhuzamosan működő parciális konvolúciók nem tudnák szolgáltatni a késleltetésmentes eredményt, helyettük teljes konvolúciót kellene végezni, ami a hatékony megvalósítás teljes kárára történhetne csak meg.

2. fejezet

Adaptív FIR-szűrők

Adaptív FIR-szűrőt egyaránt működtethetünk idő- vagy frekvenciatartományban, de ekkor a hatékonyság érdekében jól meg kell választanunk azt is, hogy milyen rendszeridentifikációs eljárást használjunk a működése során. A legkézenfekvőbb a legkisebb négyzetek módszerével történő adaptáció, mely időtartományban, de akár frekvenciatartományban is elvégezhető. Ebben a fejezetben bemutatom ezek működését, és kitérek arra is, hogy a dolgozatomban megvalósított valós idejű alkalmazásomhoz miért volt célszerűbb a frekvenciatartományban történő adaptációt választanom.

2.1. Rendszeridentifikációs módszerek

2.1.1. Időtartománybeli adaptáció (LMS)

A modellillesztés során történő paraméterbecslés alatt azt értjük, hogy a valóságos rendszer és a hozzá megválasztott modell kimeneteinek eltérését kívánjuk minimalizálni. A fizikai rendszer nem lesz ismert teljesen, ezért paramétereit nagy pontossággal nem lehet meghatározni, így meg kell, hogy elégedjünk egy olyan eredménnyel, amely a valóságos rendszert számunkra elfogadhatóan modellezi.

Ezen feladat során tehát a paraméterek becsléséhez a valóságos rendszer és az illesztett modell kimeneteinek különbségétől függő költségfüggvényt definiálunk, és a paraméterek változtatásával ennek minimalizálására törekedünk. Ezzel a modellből származó becsült kimenet a lehető legjobban közelít majd a valóságos rendszer kimenetét. [10]

$$\epsilon = E\left[(\boldsymbol{y} + \hat{\boldsymbol{y}})^T \cdot (\boldsymbol{y} + \hat{\boldsymbol{y}})\right] = E\left[e^2\right] \tag{2.1}$$

Cél a hibajel teljesítményének minimalizálása. Az illesztendő modell bemenete, kimenete és paraméterei közötti összefüggés a következő ($\boldsymbol{x} = f(\boldsymbol{u})$ a bemenőjel vektora, $\boldsymbol{w}$ a változó paraméterek vektora, $\hat{\boldsymbol{y}}$ a modell becsült kimenetének vektora):

$$\hat{\boldsymbol{y}} = \boldsymbol{w}^T f(\boldsymbol{u}) = \boldsymbol{w}^T \cdot \boldsymbol{x} \tag{2.2}$$

Egy bemenet és egy kimenet esetén az n-dik időpontban a becsült $\hat{y}(n)$ kimenet az n-edik időpontban rendelkezésre álló M minta hosszú $\boldsymbol{w}(n)$ paraméter és $\boldsymbol{x}(n)$ regressziós

vektorok skaláris szorzataként számolható.

$$\hat{y}(n) = \boldsymbol{w}(n)^T \cdot \boldsymbol{x}(n) = \sum_{i=1}^{M-1} w_i(n)\, x_i(n) \tag{2.3}$$

Ezt felhasználva az előzőekben definiált költségfüggvény az alábbi alakra hozható:

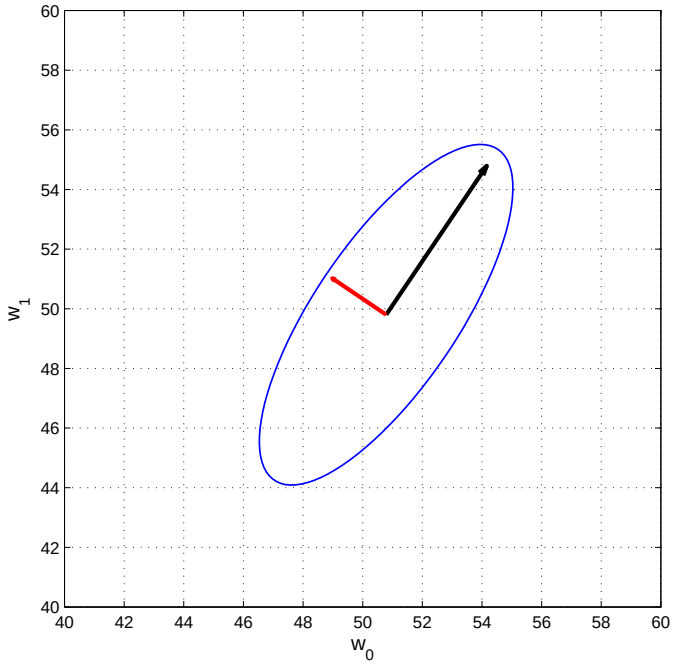
$$\epsilon = E\left[(y(n) + \hat{y}(n))^2\right] = E\left[\left(y(n) + \boldsymbol{w}(n)^T \boldsymbol{x}(n)\right)^2\right] \tag{2.4}$$

$$\epsilon = E\left[y(n)^2\right] - 2E\left[y(n)\,\boldsymbol{x}(n)^T\right]\boldsymbol{w}(n) + \boldsymbol{w}(n)^T E\left[\boldsymbol{x}(n)\,\boldsymbol{x}(n)^T\right]\boldsymbol{w}(n) \tag{2.5}$$

$$\epsilon = E\left[y(n)^2\right] - 2\boldsymbol{p}^T\boldsymbol{w}(n) + \boldsymbol{w}(n)^T\boldsymbol{R}\,\boldsymbol{w}(n) \tag{2.6}$$

Az eredményben azt kaptuk, hogy a költségfüggvényünkben megjelenik a kimenet négyzetes várható értéke $E\left[y(n)^2\right]$, a kimenet és a regressziós vektor közötti keresztkorrelációs vektor, $\boldsymbol{p}^T$, és a regressziós vektor autokorrelációs mátrixa, $\boldsymbol{R}$.

Ez a költségfüggvény egy M dimenziós ellipszis implicit egyenletét írja le, melyek változói a $\boldsymbol{w}$ pataméter vektor komponensei, és a főtengelyeinek irányát és nagyságát az autokorrelációs mátrix ($\boldsymbol{R}$) sajátvektorai és sajátértékei határozzák meg. Egy ilyen hibaellipszis számunkra legegyszerűbben abban az esetben vizualizálható a legkönnyebben, amikor a $\boldsymbol{w}$ paramétervektort úgy választjuk meg, hogy két komponenst tartalmaz, ugyanis ekkor egy 2 dimenzióban ábrázolható ellipszist ír le a költségfüggvény egyenlete [11].



2.1. ábra. *2 dimenziós hibaellipszis és a sajátvektorok ábrázolására egy példa*

Egy ilyen paraméterbecslési feladat során az a feladatunk, hogy megtaláljuk azt a $\boldsymbol{w}$

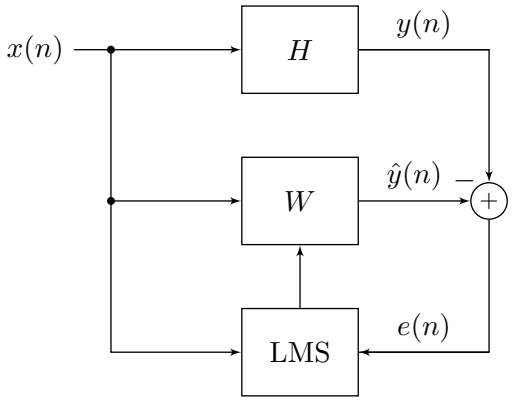
paramétervektort, amivel a valóságos rendszer a legjobban modellezhető matematikailag, más szóval a $\boldsymbol{w}$ paramétervektorban keressük azt az M darab szűrőegyüttható, amikkel a legnagyobb egyezést kaphatjuk a mérés során mért fizikai rendszerrel. Ezeket az optimális együtthatókat a költségfüggvény minimalizálásával kaphatjuk meg.

$$\frac{\partial \epsilon}{\partial \boldsymbol{w}} = -2\boldsymbol{p} + 2\boldsymbol{R}\,\boldsymbol{w} = 0 \tag{2.7}$$

$$\boldsymbol{w}_{opt} = \boldsymbol{R}^{-1}\boldsymbol{p} \tag{2.8}$$

Az előző fejezetben kapott keresztkorrelációs vektor és autokorrelációs mátrix inverzének számítása nagyon időigényes és nem célravezető megoldás. Ezért a költségfüggvényben az átlagos négyzetes hiba helyett csak a pillanatnyi hibát vesszük figyelembe, így a következő módosításokkal él a rekurzívan megoldható LMS-algoritmus (Least mean squares) [12]:

$$\hat{\boldsymbol{R}} = \boldsymbol{x}\,(n)\,\boldsymbol{x}\,(n)^T \qquad\qquad \hat{\boldsymbol{p}} = \boldsymbol{y}\,(n)\,\boldsymbol{x}\,(n)^T \tag{2.9}$$



2.2. ábra. *LMS algoritmus blokkvázlata*

$$e\,(n) = y\,(n) - \hat{\boldsymbol{w}}\,(n)^T\,\boldsymbol{x}\,(n) \tag{2.10}$$

$$\hat{\boldsymbol{w}}\,(n+1) = \hat{\boldsymbol{w}}\,(n) + \mu e\,(n)\,\boldsymbol{x}\,(n) \tag{2.11}$$

Az LMS-algoritmus által egy véges impulzusválaszú FIR-szűrőt kapunk a paraméterek becslése során. Minden iterációs lépésben az n-edik ütemben rendelkezésre álló szűrőegyüttható-készletet a $\hat{\boldsymbol{w}}\,(n)$ vektor definiálja. Az algoritmusban használt paramétertérben történő lépések mértékét μ (ez a lépésköz a költségfüggvény gradiens menti csökkentésének sebességét adja meg) az [11] hivatkozás alapján autokorrelációs mátrix legnagyobb sajátértékéből határozhatjuk meg, $0 < \mu < \frac{1}{\lambda_{max}}$. Viszont az előzőekben már kikötöttük, hogy ennek a mátrixnak a kiszámítása nem célravezető, így ezt kísérleti úton tudjuk csak meghatározni. A μ értékét célszerű a kívánt szűrőegyütthatók számából M megválasztanunk, vagyis az alábbi relációt betartva, tapasztalati úton megkeresni az optimális értékét annak érdekében, hogy az adaptív FIR-szűrő és a valóságos rendszer kimenetéből képzet

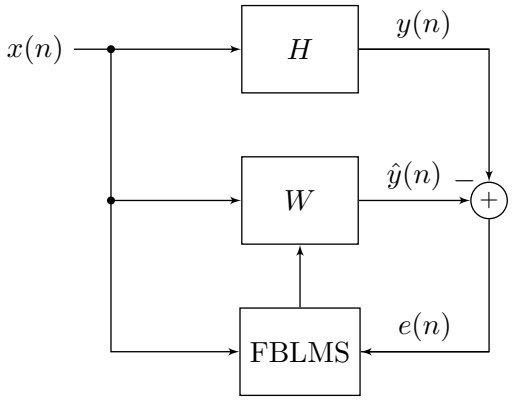
hiba, $e(n)$ a lehető legkisebb idő alatt csökkenjen le közel nulla értékre, azaz az algoritmus a lehető legjobb konvergenciasebességgel rendelkezzen.

$$0 < \mu \leq \frac{1}{M} \tag{2.12}$$

2.1.2. Frekvenciatartománybeli adaptáció (Fast Block LMS)

Számos adaptív szűrőt használó alkalmazásban, mint például az aktív zajcsökkentésben igen nagy fokszámú FIR-szűrők adaptálására is szükség lehet [13] [14]. Az időtartományban működő nagy fokszámú együtthatók adaptálása az LMS algoritmussal már igen lassú lehet, így célszerűbb az eljárást blokkonként végezni [15].

Ezt a blokkonkénti LMS eljárást nevezzük Fast Block LMS (FBLMS) algoritmusnak, mely során a gyors Fourier transzformáció segítségével az együtthatók adaptálása a frekvenciatartományban történhet. Ez a frekvenciatartománybeli blokkonkénti adaptálás számottevően csökkentheti a számítási igényt, valamint a konvergenciasebességet is növelheti.



2.3. ábra. *FBLMS algoritmus blokkvázlata*

A frekvenciatartománybeli blokkos adaptáció során az adaptált FIR-szűrő együtthatókkal való szűrést kézenfekvő szintén frekvenciatartományban végrehajtani, az FFT algoritmus segítségével. Ehhez a dolgozatomban már bemutatott módszerek közül az Overlap-Save a legmegfelelőbb választás [16]. A következő blokkdiagram mutatja be az FBLMS algoritmust, és a további felsorolásban részletezem a működést lépésről lépésre.

1. A bemenő jel régi is új blokkjának összefűzése $2N$ méretű blokká.
2. A $2N$ méretű bemenő jel blokkjának transzformálása a frekvenciatartományba FFT-vel.
3. Szűrés a frekvenciatartományban az adaptív FIR-szűrő együtthatóival $\hat{W}(k)$.
4. A frekvenciatartományban elvégzett szorzás eredményének inverz Fourier-transzformációja (IFFT) időtartományba.
5. Az utolsó blokk megtartása a kimenet eredményének, $\hat{y}(n)$.

6. Hibaképzés, a hibavektor, $e(n)$ kiszámítása a mért kimeneti vektor, $y(n)$ és az adaptív szűrés eredményvektora, $\hat{y}(n)$ segítségével.

7. Hibavektor, $e(n)$ elejének kiegészítése nullákkal, hogy biztosítva legyen a megegyező blokkméret a bemeneti blokkal.

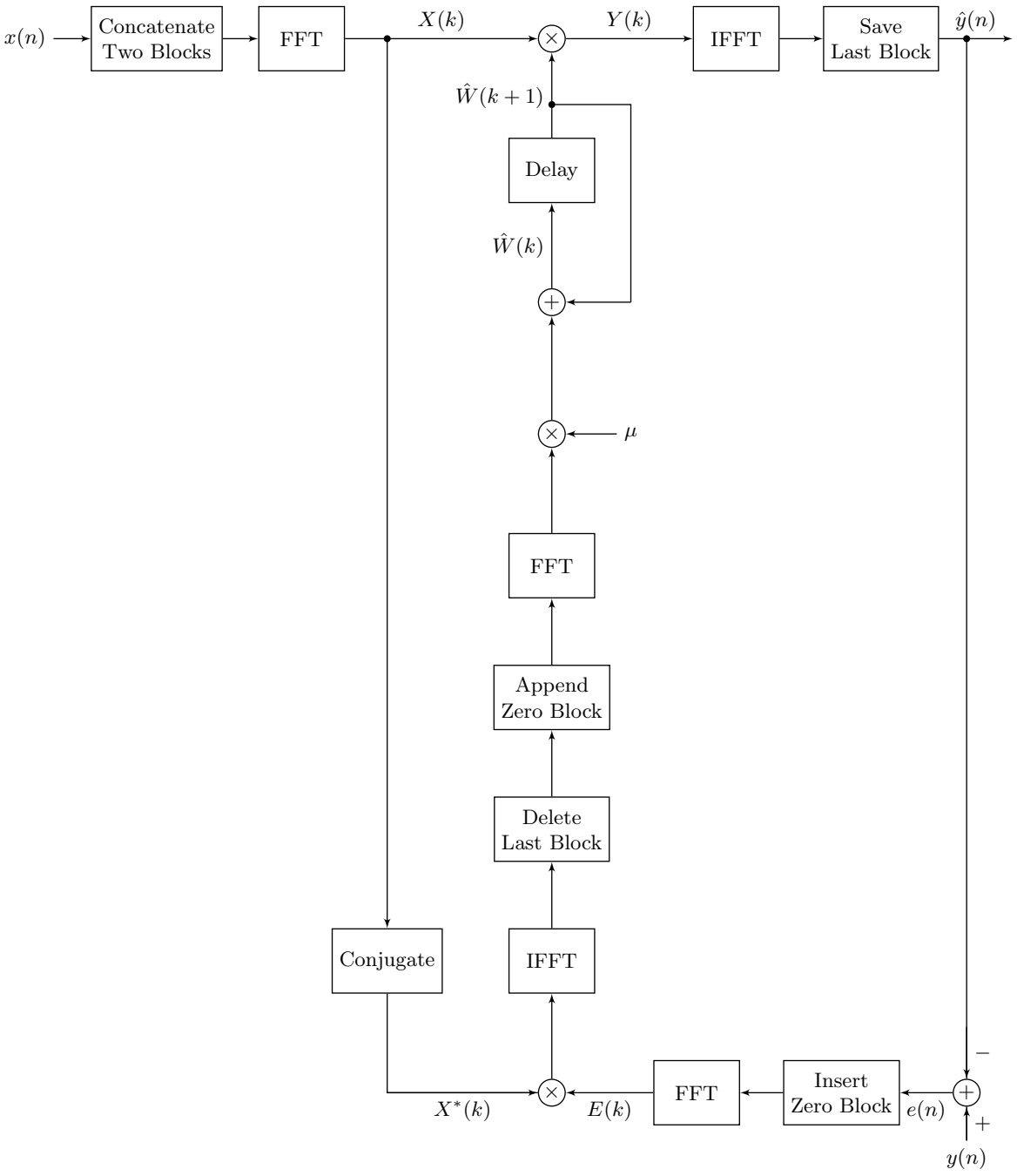
8. Hibavektor transzformálása frekvenciatartományba az FFT-vel.

9. Az eredmény összeszorzása a bemeneti blokk Fourier-transzformáltjának konjugáltjával a frekvencia térben.

10. Az szorzateredmény $X^*(k) \cdot E(k)$, vagyis a gradiens vektor inverz Fourier transzformációja.

11. Az időtartománybeli gradiens vektor utolsó blokkjának módosítása nulla elemekkel, majd ennek Fourier transzformálása újra a frekvenciatartományba.

12. A gradiensvektor szorzása a μ lépésközzel, majd hozzáadása az adaptív FIR-szűrőhöz $\hat{W}(k)$. Ez az utolsó lépés adaptálja az új együtthatókat, $\hat{W}(k+1)$.



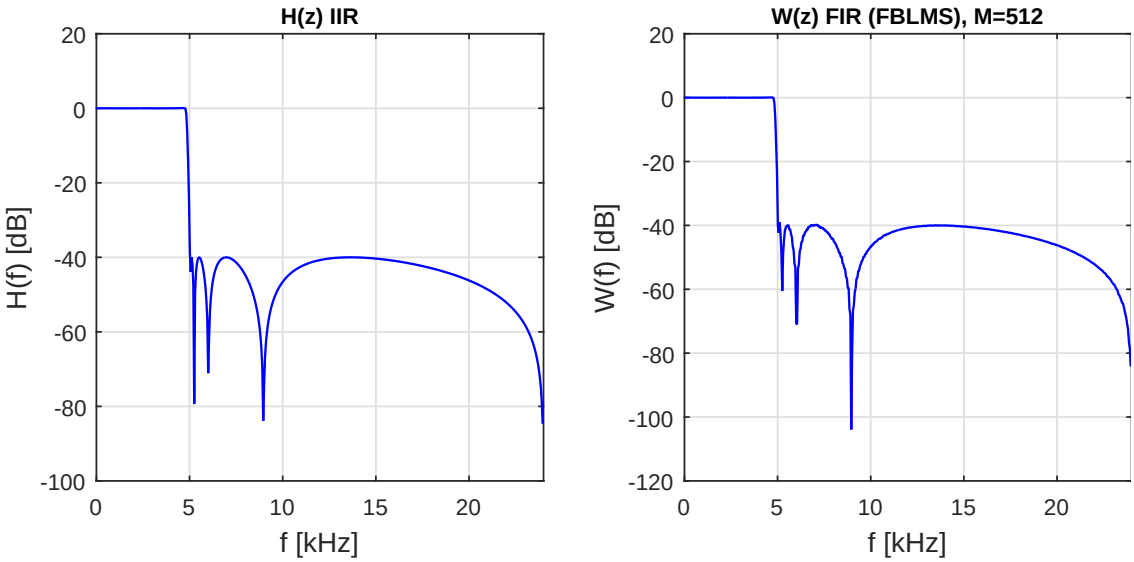
2.4. ábra. *Fast Block LMS algoritmus [1]*

2.2. LMS és Fast Block LMS algoritmusok összehasonlítása

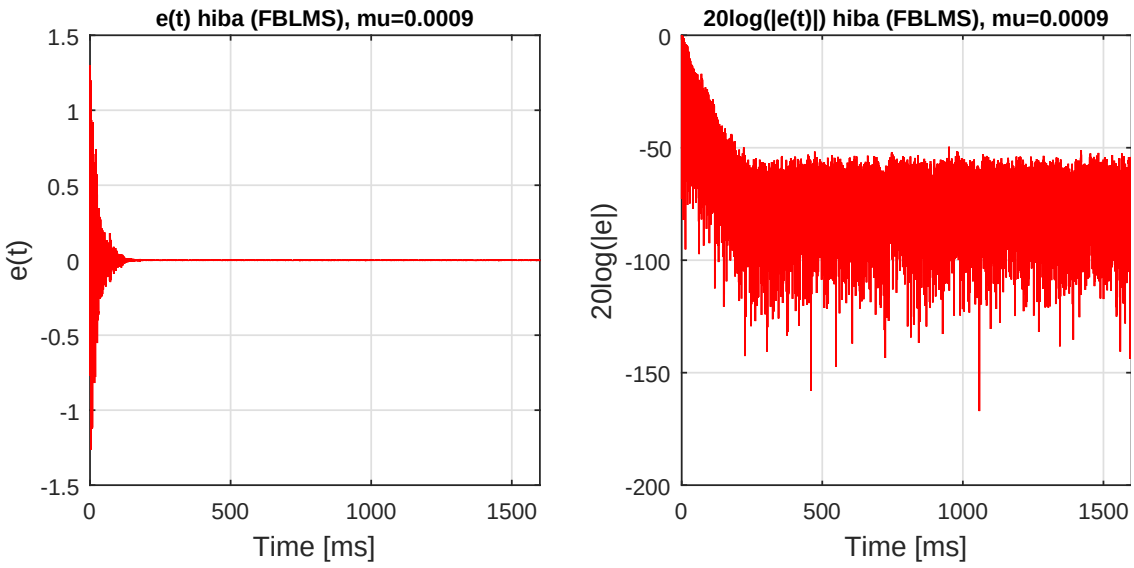
Az adaptív FIR-szűrők hatékony megvalósításához elengedhetetlen, hogy az adaptáció és a szűrés a frekvenciatartományban történjen. Ahhoz, hogy ezt a szükségességet jól demonstráljam, az LMS és FBLMS algoritmust megvalósítottam Matlab szimulációban is, és bemutatom a legfontosabb összehasonlítási eredményeket. Az LMS és az FBLMS közötti legfőbb eltéréseket pedig az alábbi felsorolásban részletezem.

- Az FBLMS algoritmus az együtthatókat a bemeneti jelből $x(n)$ vett blokkok segítségével végzi el (amik mérete megegyezik az adaptív szűrő hosszával), vagyis az együtthatók így blokkonként frissülnek. Ezzel szemben az időtartományban végzett LMS algoritmus esetén az együtthatók frissítése minden $x(n)$ mintavételezési pontban megtörténik.

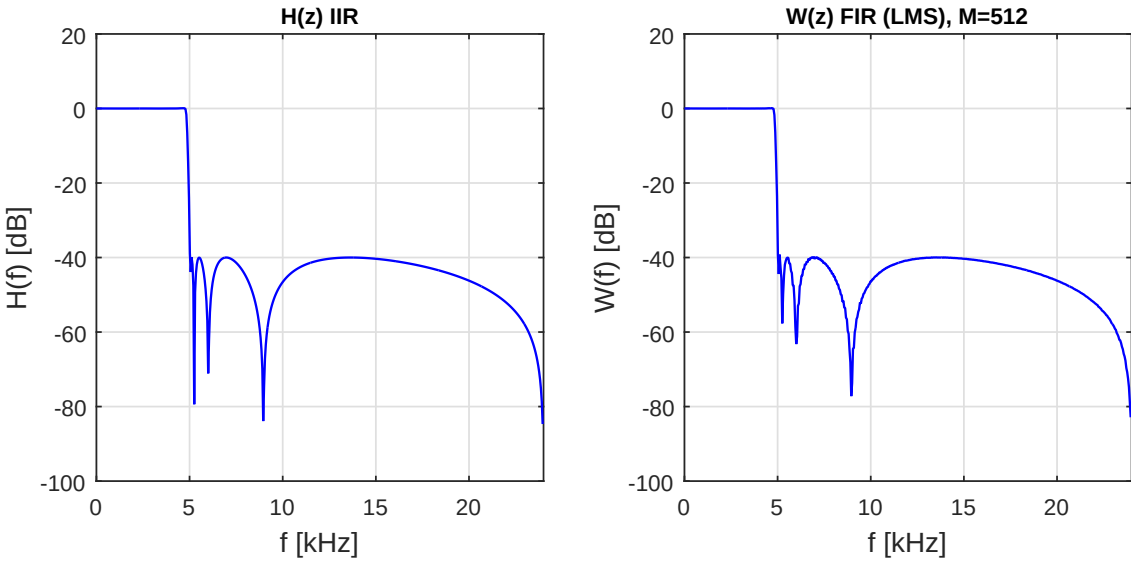
- Az FBLMS algoritmus kevesebb szorzási műveletet igényel, mint az időtartománybeli LMS. Abban az esetben, ha a blokkok mérete és az adaptív szűrő hossza, N megegyezik, akkor az LMS algoritmus $N(2N+1)$ db szorzási műveletet igényel, az FBLMS pedig csak $10N\log_2(2N)+26N$ szorzást. Konkrét értékkel bemutatva ez, azt jelenti hogy $N = 1024$ esetén az FBLMS algoritmus 16-szor gyorsabb, mint az LMS algoritmus. Ennek a hatékonyabb számítási eljárásnak köszönhetően adódik az, hogy a nagy fokszámú FIR-szűrők adaptálása esetén mindenképpen kifizetődőbb az FBLMS-sel a frekvenciatartományban elvégezni az adaptációt.



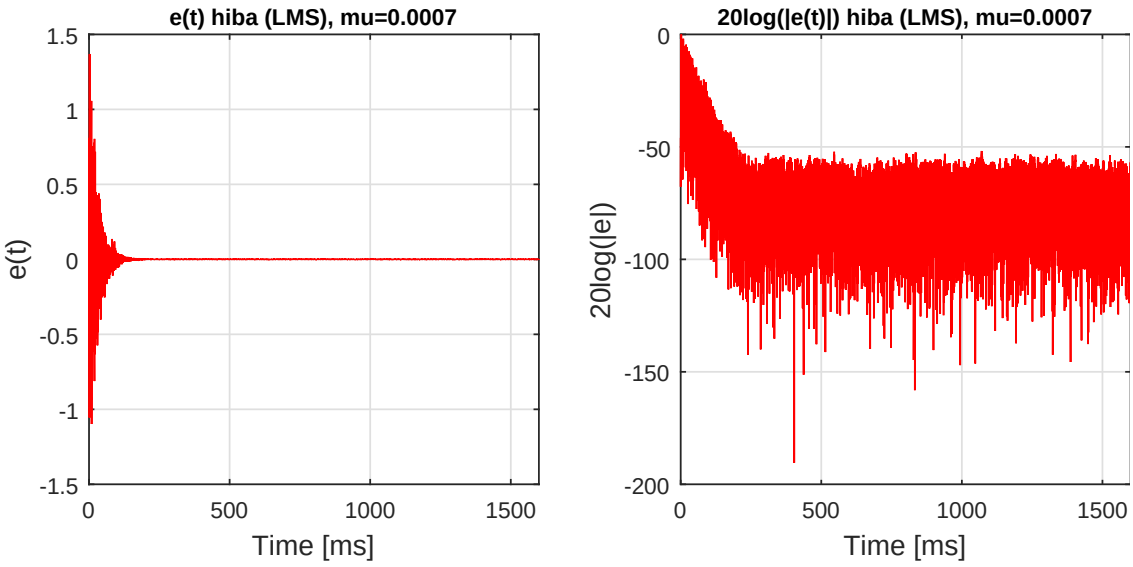
2.5. ábra. *Felhasznált IIR-szűrő és annak becsült adaptív FIR-szűrője az FBLMS algoritmussal,* $f_s = 48kHz$



2.6. ábra. *Hibalecsengés a Matlab szimulációban, FBLMS algoritmus esetén*



2.7. ábra. *Felhasznált IIR-szűrő és annak becsült adaptív FIR-szűrője az LMS algoritmussal,* $f_s = 48kHz$



2.8. ábra. *Hibalecsengés a Matlab szimulációban, LMS algoritmus esetén*

A Matlab szimulációs eredmények hibalecsengésein jól látszik, hogy a konvergenciasebesség az optimálisnak feltételezett μ lépésközöknél az FBLMS algoritmus esetén bizonyul valamivel jobbnak. A futási időket elemezve a különböző méretű FIR-szűrő együtthatók adaptálására, azt láthatjuk, hogy az FBLMS fölényesen győz az LMS-sel szemben. Ezekből az eredményekből látható, hogy az adaptív FIR-szűrők megvalósítása is csak úgy történhet hatékonyan, ha mind a szűrést mind pedig az adaptációt is a frekvenciatartományban végezzük.

N	N_{fft}	FBLMS futási idő [sec]	LMS futási idő [sec]
128	256	0.095	0.811
256	512	0.064	0.870
512	1024	0.049	0.992
1024	2048	0.043	1.239
2048	4096	0.041	1.910
4096	8192	0.041	4.607
8192	16384	0.044	7.118
16384	32768	0.043	11.049
32768	65536	0.041	16.600

2.1. táblázat. *FBLMS és LMS futási idő összehasonlítása* 204800 *minta nagyságú szűrt be- kimenetű adatpont adaptálására. A tesztelés egy átlagos PC-én Matlab környezetben készült.*

2.3. Filtered-X LMS algoritmus

Egyes adaptív szűrő alkalmazásoknál, mint például az aktív zajcsökkentés, az adaptív FIR-szűrő kimenetére egy másik átvitelű tag is kapcsolódik, és a hibajel képzés ennek kimenetével történik. Ezt a szűrőt a beavatkozó $\hat{y}(n)$ és a hibajel $e(n)$ között az aktív zajcsökkentéssel foglalkozó szakirodalom általában másodlagos útnak nevezi, a modellezendő rendszert pedig elsődlegesnek [17]. Ez a másodlagos út frekvenciatartománybeli szűrést, fázistolást, és késleltetést is okozhat. Ezeket a hatásokat a hagyományos LMS algoritmus önmagában nem képes kompenzálni, így az ilyen fajta alkalmazásoknál elkerülhetetlen a Filtered-X LMS algoritmus használata.

Az FxLMS algoritmus ezt úgy oldja meg, hogy az LMS algoritmus bemenetére érkező $x(n)$ referenciajelet a másodlagos átvitel becsült együtthatóival $\hat{S}(z)$ megszűri ($\hat{S}(z)$ a működés megkezdése előtt az LMS algoritmussal identifikálható), és ezt az $x_s(n)$ szűrési eredményt használja fel az adaptív FIR-szűrő együtthatóinak frissítésekor. Itt érdemes megjegyezni azt, hogy ez a szűrt referencia jel $x_s(n)$ csak az LMS algoritmus egyenletében van felhasználva, vagyis ennek az eljárásnak úgy kell működni, hogy az adaptálandó szűrő $W(z)$ referenciajel bemenetére ne legyen hatással ez a szűrés.

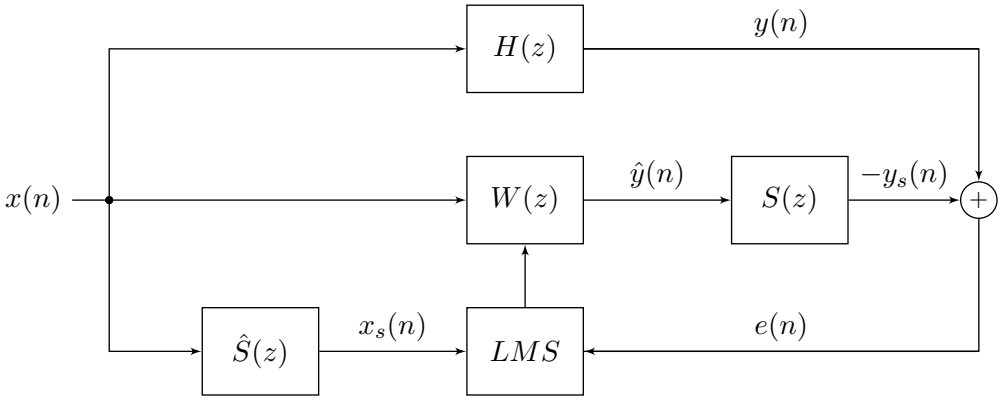
$$y_s(n) = \boldsymbol{s}(n)^T \hat{\boldsymbol{y}}(n) \tag{2.13}$$

$$e(n) = y(n) - y_s(n) \tag{2.14}$$

$$x_s(n) = \hat{\boldsymbol{s}}(n)^T \boldsymbol{x}(n) \tag{2.15}$$

$$\hat{\boldsymbol{w}}(n+1) = \hat{\boldsymbol{w}}(n) + \mu e(n) \boldsymbol{x_s}(n) \tag{2.16}$$

Az előzőekben már leírtak alapján az FxLMS algoritmus működését a következő ábrán mutatom be.



2.9. ábra. *FxLMS algoritmus blokkvázlata*

Az FxLMS blokkvázlatában a jelölések a következők:

- $x(n)$ az n-dik időpontban rögzített referenciajel minta
- $\hat{y}(n)$ az adaptív FIR-szűrő becsült kimenete
- $y(n)$ a modellezendő rendszer kimenete
- $S(z)$ a másodlagos út szűrője
- $y_s(n)$ a másodlagos út szűrési kimenete
- $e(n)$ a képzett hibajel $y(n)$ és $y_s(n)$-ből
- $\hat{S}(z)$ a másodlagos út becsült FIR-szűrője
- $x_s(n)$ a becsült másodlagos út szűrési eredménye

A valós alkalmazásokban, amelyekben másodlagos út is jelen van, általában a hibaképzéshez szükséges $y(n)$ és $y_s(n)$ külön-külön nem, vagy nehézkesen mérhető. Az esetek többségében ilyenkor a hibajel $e(n)$ közvetlen mérhető, és ennek teljesítményének minimalizálásával egyből működtethető az FxLMS algoritmus.

Az FxLMS alkalmazásakor az alábbi lényeges különbségek figyelhetők meg egy tipikus LMS alapú adaptív szűrőhöz képest.

- A jelen lévő másodlagos út impulzusválaszának becslése elengedhetetlen, még az FxLMS működtetése előtt.
- Egy tipikus adaptív szűrő esetén a hibaképzés a közvetlenül mért $y(n)$ és számolt $\hat{y}(n)$ jelekből számolható, míg a Filtered-X eljárással működtetett adaptív szűrő esetén a hibajel $e(n)$ közvetlen mérhető.
- Tehát a Filtered-X alapú adaptív szűrőknél az adaptáláshoz szükséges bemenetek az $x(n)$ és $e(n)$, míg egy sima LMS alapú adaptív szűrőnél $x(n)$ és $y(n)$.

Összefoglalva a Filtered-X LMS algoritmus az alábbi műveleteket végzi el lépésről lépésre.

1. Elvégzi az adaptív FIR-szűrő együtthatóival való szűrést, $\hat{y}(n)$-t.
2. Az $x(n)$ referenciajelen elvégzi a másodlagos út becsült FIR-szűrőjével történő szűrés eredményét, $x_s(n) - t$.
3. Frissíti az adaptív FIR-szűrő együtthatókészletét a (2.16) számú egyenlet alapján.

2.4. Normalized Filtered-X LMS algoritmus

Az FxLMS egy lehetséges módosított változata a normalized Filtered-X LMS algoritmus. Ez a módosítás a stabilitási tulajdonság és a beállási gyorsaság javítására szolgál, úgy, hogy a Filtered-X LMS eljárás az úgynevezett normalizált LMS eljárással van kombinálva. Az NLMS a lépésközt a bemeneti jel alapján normálja, így jobb stabilitási tulajdonságokat, és adott feltételek mellett gyorsabb beállást biztosíthat.

A Filtered-X LMS algoritmus rekurzív összefüggése (2.16) az alábbi módon módosul a normalizálás használata esetén:

$$\hat{\boldsymbol{w}}\left(n+1\right)=\hat{\boldsymbol{w}}\left(n\right)+\mu e\left(n\right)\frac{\boldsymbol{x_s}\left(n\right)}{\left|\left|\boldsymbol{x_s}\left(n\right)\right|\right|^2} \tag{2.17}$$

Az normalizált LMS eljárás gyakorlati alkalmazásakor ügyelnünk kell arra, hogy a (2.17) NLMS egyenletében használt szűrt referenciajel négyzetösszegével történő osztás ne történhessen nullával, vagy közel nulla értékű számmal. Ez abban az esetben léphet fel, ha a szűrt referenciajel teljesítménye, vagyis ebből adódóan négyzetösszege közel nulla. Ez a probléma egyszerűen úgy orvosolható, hogy a négyzetösszeg értékéhez egy a korrekciós tagot hozzáadunk, ami mindig biztosítani tudja a nullánál jóval nagyobb értékű osztást. Az optimális korrekciós érték a, és a μ lépésköz kísérleti úton állapítható meg a működés során.

$$\left|\left|\boldsymbol{x_s}\left(n\right)\right|\right|^2=\sum_{i=1}^{N_{fft}}x_{s,i}^2 \tag{2.18}$$

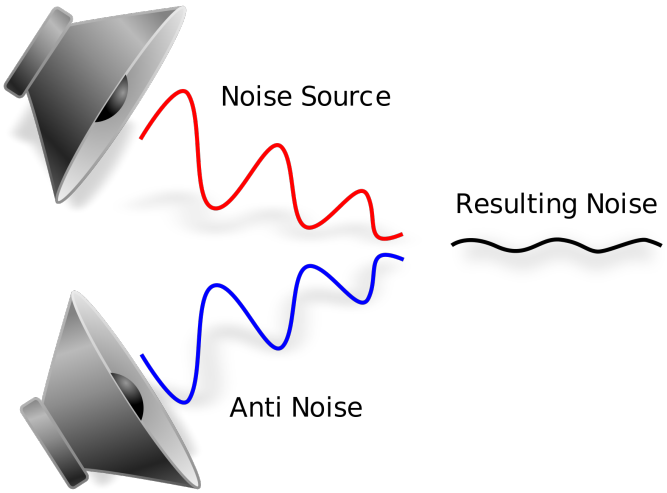
$$\hat{\boldsymbol{w}}\left(n+1\right)=\hat{\boldsymbol{w}}\left(n\right)+\mu e\left(n\right)\frac{\boldsymbol{x_s}\left(n\right)}{\left|\left|\boldsymbol{x_s}\left(n\right)\right|\right|^2+a} \tag{2.19}$$

3. fejezet

Aktív zajcsökkentés adaptív FIR-szűrővel

3.1. Aktív zajcsökkentés

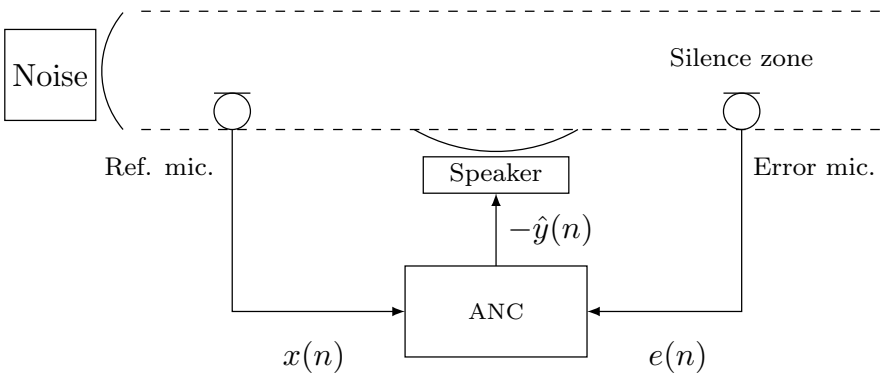
A nem kívánt zajok a csökkentésére két különböző megközelítésű eljárás használható. Ezek közül az egyik a passzív míg a másik az aktív zajcsökkentés. Passzív zajcsökkentés alatt azt értjük, amikor a zajforrás leárnyékolásával érjük el a zajcsökkentést. Ilyen esetekben használunk zajvédő falakat vagy bármilyen más hanggátló szerkezetet a zajforrás ellen. Aktív zajcsökkentés esetén viszont egy másodlagos forrást felhasználva az elnyomást a fizikai közegben egy ellenkező előjelű zaj hozzáadásával oldjuk meg [18]-[19], vagyis a zajt destruktív interferencia segítségével csökkentjük le. A destruktív interferencia feltétele, hogy a két hullám azonos frekvenciával rendelkezzen és a fáziskülönbségük 180° legyen, vagyis ellenfázisban legyenek. Ekkor az eredő hullám amplitúdója $A = |A_1 - A_2|$, és amennyiben $A_1 = A_2$, az eredő amplitúdó nullának adódik.



3.1. ábra. *Zajcsökkentés destruktív interferenciával*

A másodlagos zajt akusztikus zajcsökkentés esetén leggyakrabban hangsugárzókkal hozhatjuk létre, a kioltás helyén lévő hibamikrofonok révén pedig ez szabályozható, és végül a zajcsökkentés elvégezhető. Ez a fajta zajcsökkentés a digitális jelfeldolgozó processzorok megjelenésével és elterjedésével vált leginkább elérhetővé. A digitális jelfeldolgozó rendszernek mindössze azt a követelményt kell teljesítenie, hogy a megfelelő sebességgel és dinamikával képes legyen időben beavatkozni a generált másodlagos hangforrással, és ez által a hibamikrofonnál minimalizálódjon a hallható akusztikus zaj.

A legelterjedtebb aktív zajcsökkentő rendszerek adaptív szűrőket alkalmaznak, ezek többnyire LMS-alapú adaptációval működnek. A diplomamunkámban előrecsatolt rendszert alkalmazok melyek jellegzetessége, hogy a zajelnyomáshoz egy úgynevezett referenciajel-bemenetre is szükség van, aminek az elnyomandó zajjal jól korreláltnak kell lennie a sikeres zajelnyomás érdekében. Ez a fajta előrecsatolt rendszer a 2.3 fejezetben bemutatott Filtered-X LMS (FxLMS) algoritmussal valósítható meg [11]. Az eljárás, ahogy már a 2.3 fejezetben említve volt, igényli a másodlagos forrás és a hibamikrofon közötti úgynevezett másodlagos út modelljének megalkotását is, amelynek a pontossága meghatározza a rendszer működőképességét. Pontatlan modell esetén a hatékonyság csökken, súlyosabb esetben instabilitás lép fel [20].



3.2. ábra. *Előrecsatolt aktív zajcsökkentő rendszer, referenciajel-bemenettel*

Ezek az adaptív eljárások mind szélessávú, mind pedig harmonikus zajok elnyomására alkalmasak, viszont ez a passzív eljárásokra már mind nem mondható el. A passzív módszerek az alacsony frekvenciás zajokkal szemben hatástalanok, vagy épp igen költséges a kiépítésük. Számos zajcsökkentő rendszer egyszerre használja mind az aktív és passzív módszerű zajcsökkentést, azért, hogy az adaptív eljárással működő aktív zajcsökkentés működése csak az alacsony frekvenciatartományban legyen szükséges. Az ilyen kombinált működést megvalósító eszközökre egy jó példa az aktív zajcsökkentő funkcióval ellátott fejhallgatók, amelyek nevében igaz csak az aktív zajcsökkentésre van utalás, de valójában ezek passzív zajcsökkentést is biztosítanak az emberi fül megfelelő elszigetelésével.

A diplomamunkámban az aktív zajcsökkentéshez alkalmazott, adaptív FIR-szűrők frekvenciatartományban történő hatékony megvalósítása mellett azt is megvizsgálom, hogy az egyszerre alkalmazott aktív és passzív módszer mennyire nyújt jó zajelnyomást, és az egyes módok működése milyen frekvenciatartományokra koncentrálódnak.

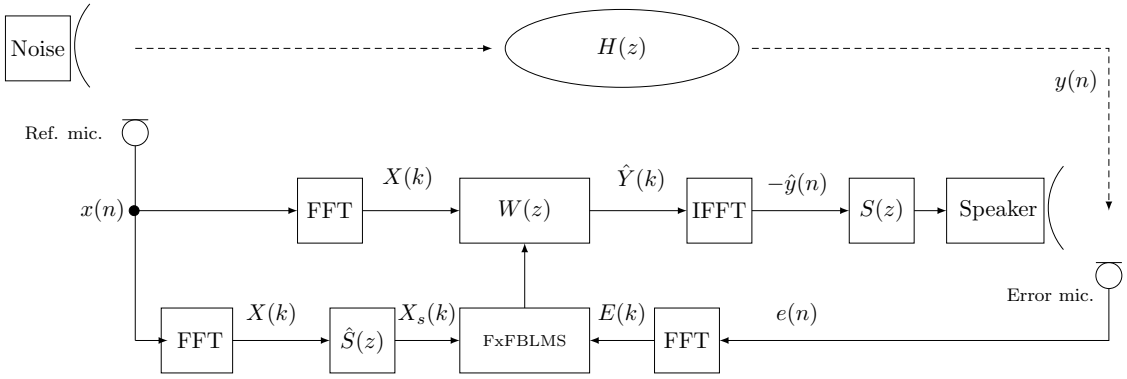
3.2. FxLMS alapú aktív zajcsökkentés a frekvenciatartományban

Az aktív zajcsökkentéshez elengedhetetlen FxLMS struktúra a Fast Block LMS algoritmussal valósítható meg a frekvenciatartományban. Lényegében az FBLMS algoritmust a Filtered-X eljárással kell csak kiegészítenünk, ami mindössze annyi módosítást igényel, hogy a referenciajel adatgyűjtésének befejeződése után, ezen blokkon el kell végeznünk a másodlagos út becsült FIR-szűrőjével való szűrést. A 2.4. ábra szerint az FBLMS algoritmus lépéseit követve a másodlagos úttal szűrt referenciajel $x_s(n)$ a frekvenciatartományban használódik fel a továbbiakban, így a minél kevesebb műveletszám érdekében ez a szűrés történhet egyből az OLS algoritmussal a frekvenciatartományban is.

$$\boldsymbol{X_s}(k) = \mathcal{FFT}\left\{\hat{\boldsymbol{s}}(n)\right\} \cdot \mathcal{FFT}\left\{\boldsymbol{x}(n)\right\} \tag{3.1}$$

Amennyiben a blokkos végrehajtások alatt számunkra fontosabb a memóriakapacitás takarékosabb beosztása, annak érdekében, hogy az elsődleges út adaptív FIR-szűrőjét minél nagyobb fokszám mellett működtethessük, akkor választhatjuk azt is, hogy a referenciajel szűrését az időtartományban végezzük. Ahogy az 1.2. ábrán latható is volt, ekkor több művelet szükséges a szűrés elvégzéséhez, de mivel az identifikált másodlagos út FIR-szűrője tipikusan már néhány száz együttható mellett is jó modellt szolgáltat, az ekkora blokkméretű szűrés az FFT algoritmussal elvégezve nem jelentene lényegesen sokkal kevesebb műveletet mint a konvolúcióval elvégezve ugyan ezt.

Az FBLMS főbb változtatásait a 3.3. ábrán szemléltetem. Aktív zajcsökkentő rendszerekhez alkalmazva ezt, a hibajel képzése nem a digitális jelfeldolgozó rendszerben képződik, ezáltal a módszer annyi egyszerűséget hordoz magában, hogy ezt és a referenciajelet csak blokkosan mintavételezni kell majd a gyors Fourier-transzformációkat és a másodlagos úttal való szűrést elvégezni.



3.3. ábra. *FxFBLMS algoritmus blokkvázlata*

A frekvenciatartományban megkapott $\boldsymbol{X_s}(k)$ és $\boldsymbol{E}(k)$ eredmények után az elkövetkezendő számítások már teljes mértékben megegyeznek az FBLMS további, frekvenciatartománybeli műveleteivel. Összegezve tehát, ezeknek a kiegészítéseknek köszönhetően egy Filtered-X Fast Block LMS algoritmust tudunk alkalmazni adaptív FIR-szűrőkhöz.

4. fejezet

Adaptív FIR-szűrő hatékony megvalósítása

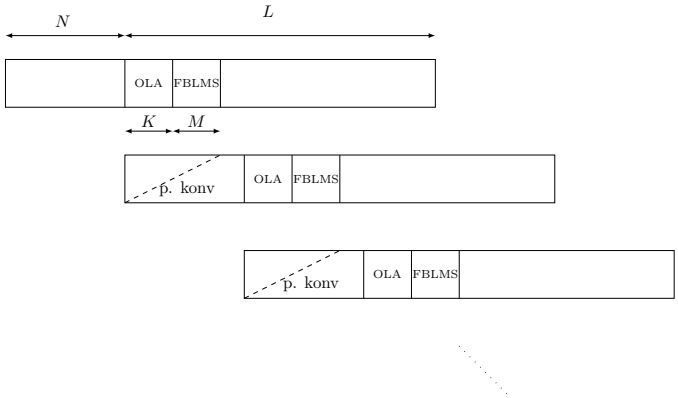
Az eddig bemutatott késleltetésmentes idő- és frekvenciatartományban működő FIR-szűréssel, valamint Fast Block LMS felhasználásával célom az volt, hogy megvalósítsak egy olyan adaptív FIR-szűrőt, amely nagy fokszámú együtthatók adaptálására is hatékonyan alkalmas, és emellett a tipikusan a frekvenciatartományban működő blokkosan végrehajtott konvolúciós eljárásoknál tapasztalható adatgyűjtés miatti késleltetéstől is mentes. Ebben a fejezetben részletezem ennek megvalósítási lehetőségét és a hatékonyságát.

4.1. Késleltetés nélküli szűrés alkalmazása adaptív FIR-szűrőhöz

Ahhoz, hogy az Overlap-Add eljárást és a parciális konvolúciót használó késleltetés nélküli FIR-szűrést adaptív FIR-szűrőkhöz használjuk, nem kell mást tennünk mint megfelelő programszervezéssel ezt egy Fast Block LMS algoritmussal kell párhuzamosan futtatnunk a valós idejű alkalmazás során.

Az 1. fejezetben szó volt arról, hogy gondoskodnunk kell arról is, hogy míg az Overlap-Add eljárás FFT algoritmusa dolgozik, továbbra is legyen eredménye a FIR-szűrésnek. Ez egyszerűen megoldható volt azzal, hogy a parciális konvolúciót az N mintagyűjtés után is folytattuk a következő blokk első néhány N minta adatgyűjtési ideje alatt, amíg az FFT algoritmus az előző bementi blokk szűrési eredményét el nem végezte, majd a parciális konvolúciót újra kezdtük az aktuális új blokkban.

Adaptív FIR-szűrő esetén ezt azzal tudjuk a legegyszerűbben kiegészíteni, hogy az Overlap-Add algoritmus után egyből, az újrakezdett parciális konvolúció alatt párhuzamosan a Fast Block LMS algoritmust folytatjuk, amely blokkonként végzi el az ismeretlen fizikai rendszer adaptációját. Ez a feladat egyébként a rendelkezésre álló rendszer és processzorarchitektúrától függően igen sokféleképpen valósítható meg. Ha több processzormagot is van módunkban használni a tervezendő rendszerünkben, akkor kézenfekvő megoldás lenne ezt a három algoritmust párhuzamosan futtatni. A munkám során egy Analog Devices SHARC ADSP-21364 jelprocesszor használatára volt lehetőségem, ami sajnálatos módon csak egy processzormaggal rendelkezik, így a legegyszerűbb nem többmagos meg-



4.1. ábra. *Késleltetés nélküli adaptív FIR-szűrő, OLA-t és FBLMS-t használva*

valósítás volt csak elérhető számomra a fejlesztés során.

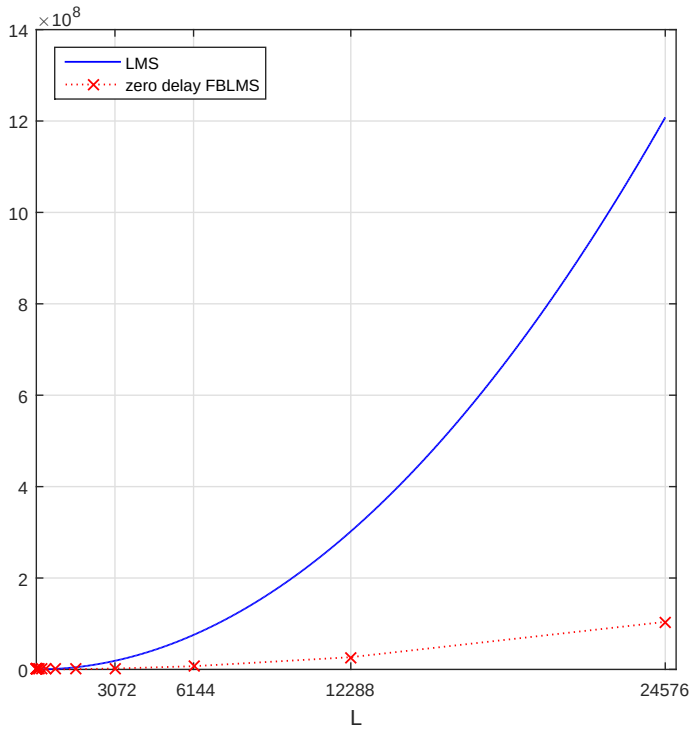
A fejlesztésem során Multitasking-gal rendelkező operációsrendszer sem állt rendelkezésemre az ADSP-21364 jelprocesszor fejlesztő környezetén, így ennek a három algoritmusnak az egymás melletti működését a lehető legegyszerűbben törekedtem megvalósítani. Ez a lehető legegyszerűbb megoldás így csak az lehetett, hogy az adaptálást végző Fast Block LMS algoritmusom minden Overlap-Add szűrési eljárás után kezdi meg működését. Vagyis ez után, az új mintáknál fellépő, megszakítások alatt elvégződő újrakezdett parciális konvolúciók után a következő minta érkezéséig a fennmaradó szabad processzoridőben az előző blokkra a Fast Block LMS elkezdheti a részszámításokat, majd egy bizonyos mintavételezési szám után elkészül ezzel, és megtörténik az adaptív FIR-szűrő együtthatóinak frissítése, amit már azonnal a következő OLA eljárásnál és parciális konvolúciónál fel is lehet használni.

Az adaptív FIR-szűrő hatékony megvalósításánál a cél az, hogy a blokkos végrehajtás alatt működő módszerek a lehető legnagyobb számú L együtthatóra tudjanak működni. Ezt az igényt elsősorban az $N_{fft} = N + L$ blokkméretet használó OLA és FBLMS eljárás limitálja, mert a véges memóriaterület és processzorkapacitás miatt tetszőlegesen nagy N_{fft} méretű blokkokra nem tudjuk elvégezni a műveleteket. Másodsorban az is limitálja a maximálisan elérhető L együtthatót, hogy az N blokkméret csak limitált méretig csökkenthető a módszer során, mivel ezen N minta beérkezési ideje alatt az előző x_{i-1} blokkon végzett szűrést az OLA algoritmusnak és a blokkos adaptációt az FBLMS-nek el kell tudnia végezni a megszakítások után fennmaradó szabad időben még az előtt, hogy az éppen aktuális új N mintával feltöltődő x_i-edik blokkon ezek meg nem kezdenék az új szűrést és adaptációt.

Ezt, hogy hogyan lehet a maximális L együtthatót megbecsülni az általam javasolt adaptív, késleltetés nélküli FIR-szűrőm használatakor egy adott memóriával és processzorteljesítménnyel rendelkező rendszerben, a következő alfejezetben fejtem ki bővebben.

4.2. Idő- és frekvenciatartománybeli, késleltetésmentes adaptív FIR-szűrés hatékony megvalósítása

Az időtartományban működő LMS algoritmussal való adaptálást az idő- és a frekvenciatartományban működő FBLMS-sel való adaptálással összehasonlítva kijelenthető az, hogy egy bizonyos L nagyságú FIR-szűrő adaptálása felett mindenképpen megéri az általam megvalósított idő- és a frekvenciatartománybeli, késleltetésmentes adaptív FIR-szűrőt használni, mert az FBLMS és az OLA eljárás még mindig elég hatékonyan képes csökkenteni a számítási igényt a parciális konvolúció mellett is. A két módszer számításigénye L függvényében az alábbi 4.2. ábrán látható (LMS-t használva a műveletigény N_{LMS}, FBLMS-t használva pedig N_{zerod}, 75%-os átlapolást használva, $N_{fft} = N + L$ méretű blokknál).



4.2. ábra. *A késleltetésmentesen működő adaptív FIR-szűrő műveletigényének összehasonlítása LMS és FBLMS algoritmust használva az L adaptált együtthatók függvényében (FBLMS-nél 75%-os átlapolással, $L = 0.75N_{fft}$).*

$$N_{LMS} = L(2L + 1)$$

$$N_{zerod} = 6N_{fft}\log_2(N_{fft}) + 13N_{fft} + \frac{3}{2}N^2 + N$$

Annak érdekében, hogy hatékonyan a lehető legnagyobb L együtthatót tudjuk adaptálni, meg kell becsülnünk az adaptív FIR-szűrő maximális együtthatószámát egy adott N_{fft} blokkméret és processzorteljesítmény mellett. Ehhez paraméteresen ábrázoltam a processzor kihasznált számítási teljesítményét a mintavételi idők függvényében, amin további számításokkal ez a becslés jó közelítéssel helyes eredményre vezethet. A modell során

használt paraméterek értelmezései a következők:

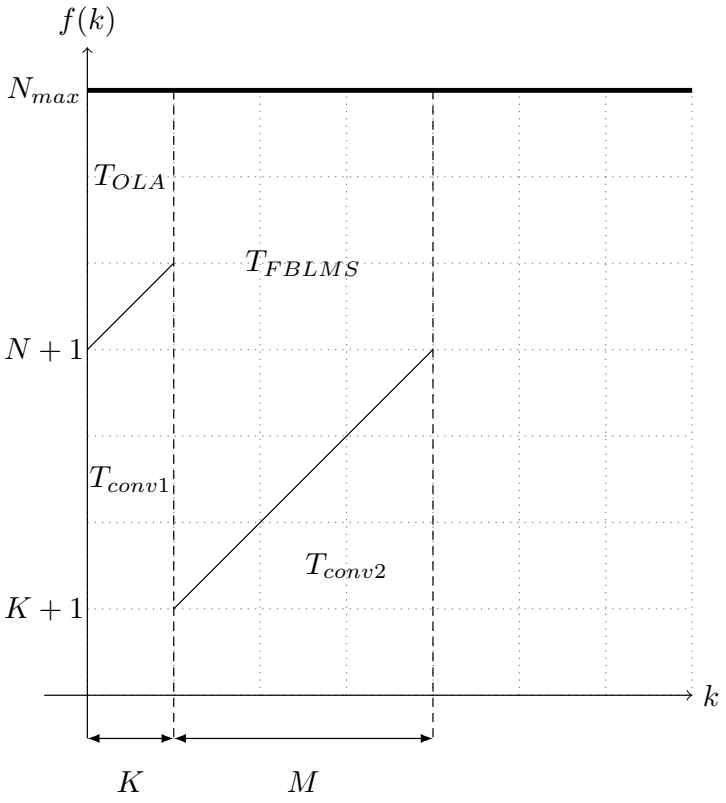
- k: A már mintavételezett minták száma az OLA blokk elejétől számolva.
- N_{max}: Egy adott k-adik mintavétel után a maximálisan elvégezhető műveletek száma a következő $k+1$ minta beérkezéséig. Ez a processzor teljesítményéből és a használt fs mintavételi frekvenciából származtatva egy állandó paraméter.
- N: Az OLA blokkos végrehajtásánál, egy blokkban mintavételezett bemeneti értékek száma.
- K: Az OLA algoritmus futási ideje alatt, a beérkező új minták száma.
- M: Az FBLMS algoritmus futási ideje alatt, a beérkező új minták száma.
- N_{OLA}: Az OLA algoritmus elvégzéséhez szükséges műveletek száma.
- N_{FBLMS}: Az FBLMS algoritmus elvégzéséhez szükséges műveletek száma.
- N_{fft}: Az OLA és FBLMS algoritmusok blokkméreteinek mintanagysága, amely a rendelkezésre álló maximális memóriaterület alapján szintén egy állandó paraméter.

A modell segítségével az a cél, hogy L együtthatószám maximális értéke megközelítőleg meghatározható legyen, vagyis az $N_{fft} = N + L$ összefüggés alapján N minimális értéke a kérdés, ami még épp elegendő lehet ahhoz, hogy a K és M mintaszámig tartó OLA és FBLMS algoritmus befejeződjön ezen minták beérkezése és parciális konvolúció számítása alatt. Tehát a modell során azzal a feltétellel élhetünk, hogy $K + M \leq N$.

A továbbiakban még azt tudjuk, hogy az OLA algoritmus kezdetekor a még folytatott parciális konvolúció szummájának felső határa megegyezik $N+1$-gyel, vagyis ekkor $N+1$ szorzást és összeadást végez el a processzor. Az is adódik, hogy a SIMD (single instruction, multiple data) módot kihasználva a jelprocesszorban, a parciális konvolúciós szumma felső határa minden egyes új minta beérkezése után eggyel növekszik, mivel eggyel több mintára végződik majd el a konvolúció. Ez a művelet igény egészen K minta beérkezéséig növekszik lineárisan 1 meredekséggel, amikor is befejeződik az OLA algoritmus, és elkezdhető a parciális konvolúció újbóli futtatása, immár $K+1$ felső szumma határral kezdve. Az ezután elkezdődő FBLMS algoritmus még M minta beérkezéséig fut a megszakításokban működő parciális konvolúciók mellett, aminek továbbra is 1-gyel nő a konvolúciós szumma felső határa minden új mintavételezés után.

Az előzőekben vázolt a priori információk alapján könnyen ábrázolhatjuk a 4.3. ábrán a beérkezett minták számában azt, hogy a két minta beérkezése között a processzornak mennyi további számítási kapacitása van a megszakításokban futó parciális konvolúciók mellett, az OLA és az FBLMS algoritmusok számára.

Az ábrán jól látszódik, hogy a parciális konvolúció kezdeti $N+1$ felső szumma határa, az OLA és az FBLMS algoritmushoz tartozó K és M mintányi futási időt befolyásolja, vagyis $K(N)$ és $M(N)$ értékei függnek N-től. Ahhoz hogy megkapjuk az összefüggést $K(N)$ és $M(N)$-re, egy egyszerű geometriai problémát kell megoldani az ábrán.



4.3. ábra. *A processzor kihasználtságának aránya a mintavételi időpontok függvényében.*

Láthatjuk hogy a $K+M$ minta beérkezése alatt a processzor $N_{max} \cdot (K+M)$ műveletet hajt végre. És azt is tudjuk, hogy az ez idő alatt a parciális konvolúciók során elvégzett műveletek összes száma (ami a lineáris növekedések alatti terület) plusz az N_{fft} blokkméretekre végzett OLA és FBLMS algoritmusok műveletszáma pontosan meg kell, hogy egyezzen az $N_{max} \cdot (K+M)$ összesen elvégzett művelet számával.

$$N_{max}(K+M) = T_{OLA} + T_{FBLMS} + T_{conv1} + T_{conv2} \tag{4.1}$$

$$T_{OLA} = N_{OLA} \qquad T_{FBLMS} = N_{FBLMS}$$

A területegyezésekből felírható egyenlet révén $K(N)$ és $M(N)$ kifejezhető egy adott N_{max} és N_{fft} állandó érték mellett. Ehhez nem kell mást tennünk, csak integrálás útján meg kell határoznunk paraméteresen T_{conv1} és T_{conv2}-őt, és ez után a következő két egyenlet segítségével $K(N)$ és $M(N)$ N-től valló függését ki kell fejeznünk.

$$T_{conv1} = \int_0^K k + (N+1)dk = \frac{K^2}{2} + K(N+1) \tag{4.2}$$

$$T_{conv2} = \int_{K}^{K+M} k + (K+1)dk = \frac{M^2}{2} + 2KM + M \tag{4.3}$$

$$N_{max}K = N_{OLA} + T_{conv1} = \frac{K^2}{2} + K(N+1) + N_{OLA} \tag{4.4}$$

$$N_{max}M = N_{FBLMS} + T_{conv2} = \frac{M^2}{2} + 2KM + M + N_{FBLMS} \tag{4.5}$$

Az utóbbi két egyenletből $K(N)$ megkapható a másodfokú polinom gyökének paraméteres kifejezésével, és ezután $K(N)$-et felhasználva ugyanígy megkapható az $M(N)$-re való összefüggés is.

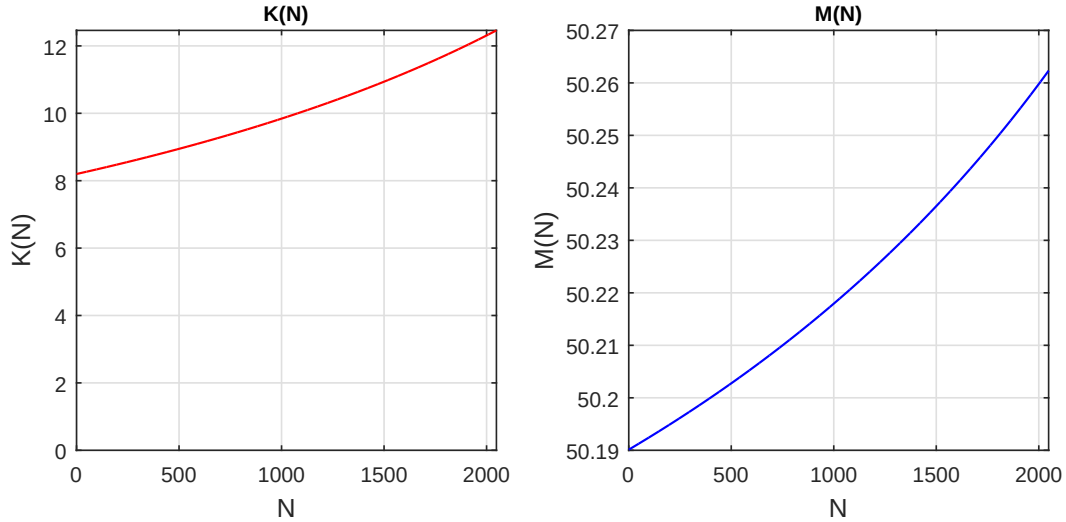
$$K(N) = (N_{max} - N - 1) \mp \sqrt{(N_{max} - N - 1)^2 - 2N_{OLA}} \tag{4.6}$$

$$M(N) = -(2K + 1 - N_{max}) \pm \sqrt{(2K + 1 - N_{max})^2 - 2N_{FBLMS}} \tag{4.7}$$

$$N_{OLA} = N_{fft} \log_2(N_{fft})$$

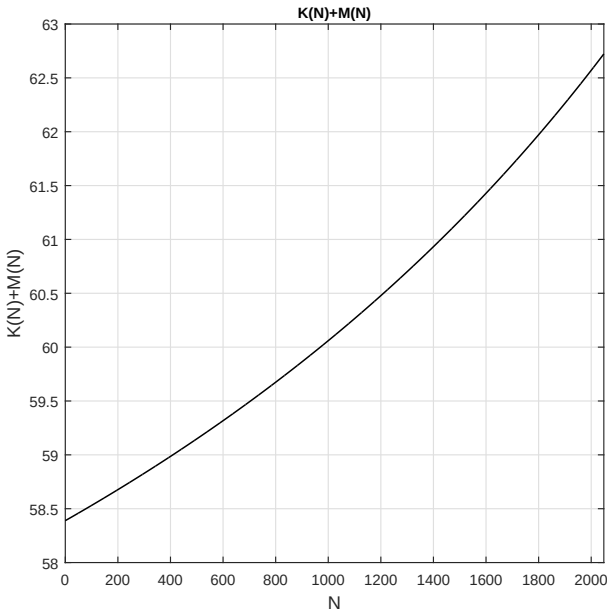
$$N_{FBLMS} = 5N_{fft} \log_2(N_{fft}) + 13N_{fft}$$

Ennek az eredménynek a hatására $K(N)$ és $M(N)$ függését N-től most már ismerjük egy adott processzorhoz és egy adott N_{fft} blokkméretre használva a módszert, tehát ha $f(N) = K(N) + M(N)$-t ábrázoljuk, akkor az adott fix paraméterekkel N_{fft} és N_{max}-xal, grafikusan megkaphatjuk azt az optimális N-et, ahol teljesül a $K(N)+M(N) \leq N$ reláció, és az adaptív FIR-szűrő L együtthatóinak számát a maximálisra választhatjuk.

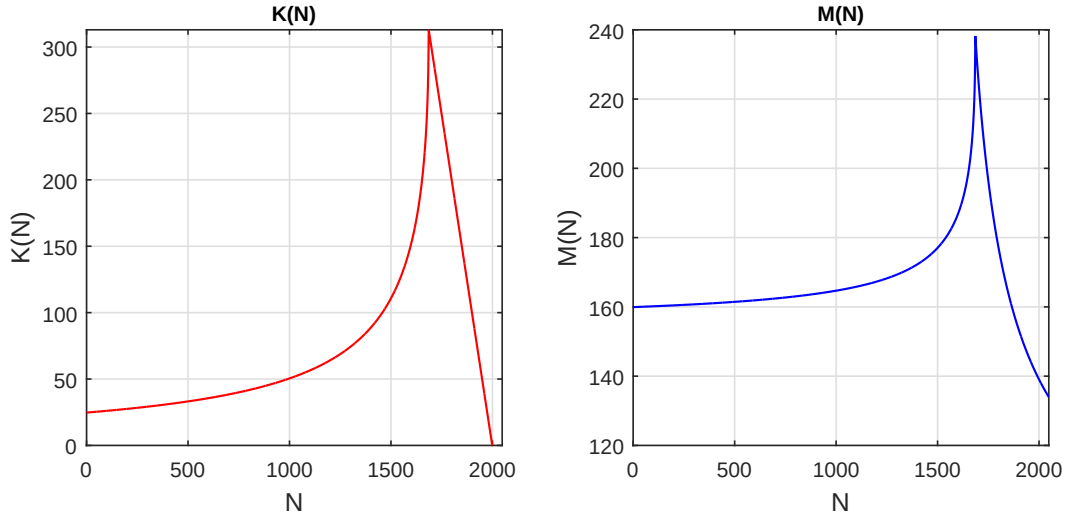


4.4. ábra. *$K(N)$ és $M(N)$ ábrázolása $N_{fft} = 4096$ és $N_{max} = 6000$ esetén.*

Konkrét példákkal ábrázolva az eredményt, $N_{max} = 2000$ és $N_{fft} = 4096$ rendszerpa-



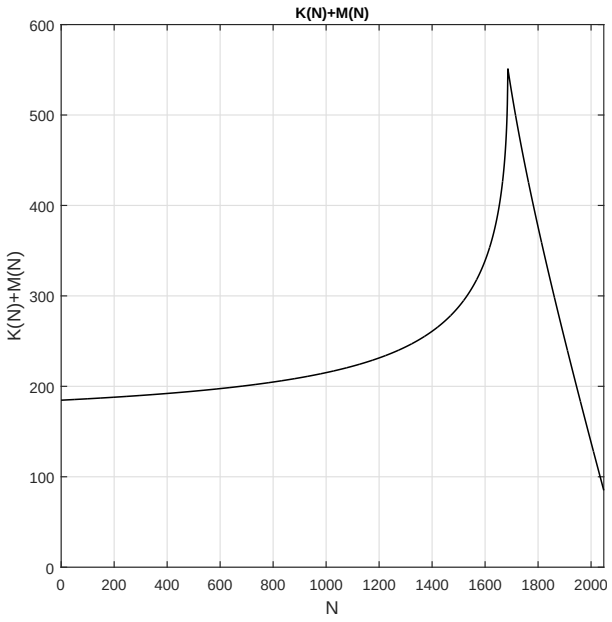
4.5. ábra. *$K(N) + M(N)$ ábrázolása $N_{fft} = 4096$ és $N_{max} = 6000$ esetén.*



4.6. ábra. *$K(N)$ és $M(N)$ ábrázolása $N_{fft} = 4096$ és $N_{max} = 2000$ esetén.*

raméterek esetén még az is jól megfigyelhető, hogy melyik az a maximálisan használható N méret, amikor még a megszakításokban futó parciális konvolúció szummájának felső határa az OLA algoritmus befejezésének pillanatában pont akkora, hogy a processzornak még éppen legyen rá kapacitása, hogy ezt elvégezze. Az ábrákon ez a $N + K(N) = N_{max}$ határesethez tartozó pont (csúcspont). Ha ennél a pontnál magasabbra választjuk meg N-et, a parciális konvolúció olyan nagy felső szumma határról indul, hogy ez már az OLA algoritmus befejezésének ideje előtt telítésbe viszi a processzorkihasználást, és egy mintavételi idő alatt már nem lesz rá elegendő kapacitása, hogy a processzor kiszámolja ezt. A matematikai modell esetén ez a határeset úgy jelentkezik, hogy ennél nagyobb N használatánál egyszerűen az eredmény komplexszé válik (csak a valós részt ábrázolva ez a

tartomány a csúcs utáni hirtelen csökkenő rész).



4.7. ábra. *$K(N) + M(N)$ ábrázolása $N_{fft} = 4096$ és $N_{max} = 2000$ esetén.*

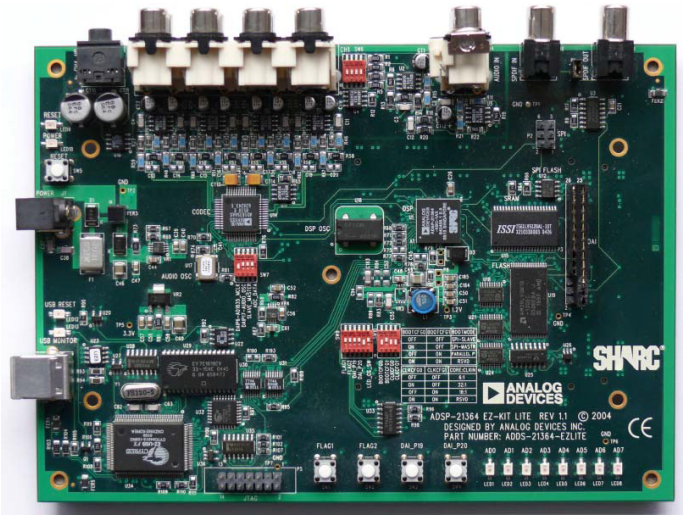
4.3. Késleltetésmentes adaptív FIR-szűrés implementálása ADSP-21364 jelprocesszoron

Az idő- és a frekvenciatartománybeli, késleltetésmentesen működő adaptív FIR-szűrést ADSP-21364 jelprocesszorra fejlesztettem ki, amelyhez a Méréstechnika és Információs Rendszerek Tanszék DSP laborjában rendelkezésre áll több ADSP-21364 EZ-KIT Lite fejlesztőkártya.

Ezen az ADSP-21364 EZ-KIT Lite fejlesztőkártyán az ADSP-21364 jelfeldolgozó processzorra történő fejlesztés hatékony és gyors. A fejlesztőkártyán kialakításra került a jelfeldolgozó processzor minden olyan szükséges perifériája és környezete, amelyre a leggyakoribb feladatok elvégzésekor szükség lehet. Az analóg és digitális hang ki- és bemenetek, LED-ek, kapcsolók, nyomógombok azonnal hozzáférhetőek, így a szoftverfejlesztés mindenféle hardvertervezési lépés nélkül azonnal elkezdhető, a fejlesztőkártya VisualDSP++ programozási környezetében. A kártya a következő hardverelemeket bocsátja a fejlesztők rendelkezésére. [12]

- Analog Devices ADSP-21364 processzor
- 4 Mbit SRAM
- 8 Mbit flash memória
- 2 Mbit SPI által kezelhető flash memória
- Analóg hanginterfész (AD1835A kodek, 1 db sztereó bemenet, 4 db sztereó kimenet)

- Digitális hanginterfész (1 db bemenet, 1 db kimenet)
- 11 db LED (ebből 1 db power, 1 db board reset, 1 db USB monitor, 8 db általános célú)
- 5 db nyomógomb (1 db reset, 2 db DAI lábra, 2 db FLAG lábra kötve)
- Bővítőinterfész (párhuzamos port, FLAG-ek, DAI, SPI)
- JTAG emulátor port
- USB port a PC csatlakoztatásához.

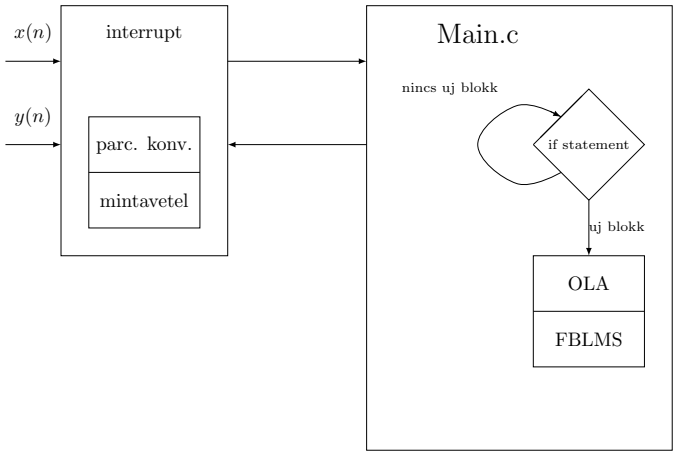


4.8. ábra. *ADSP-21364 EZ-KIT Lite fejlesztőkártya*

A Filtered-X és normalizált LMS eljárásokkal kiegészített FBLMS alapú késleltetésmentes adaptív FIR-szűrést egy igen egyszerű programszervezésben oldottam meg. Az adaptív szűrés alatt a referenciajel, $x(n)$ és az aktív zajcsökkentéshez szükséges hibajel, $e(n)$ mintavételezése megszakítások után történik, a mintavételi frekvenciáknak megfelelőn, valamint ekkor történik meg a parciális koncolúcióval való FIR-szűrés is, ami egy mintavételi idő alatt bőven elkészül, és így az előző OLA blokk szűrési eredményének segítségével a D/A kimeneteire szolgáltatja a szűrés eredményét, adatgyűjtési késleltetés nélkül.

Mialatt a mintavételezések és a parciális konvolúciók a megszakítási időpontokban megtörténnek, a main függvény végtelen ciklusában az OLA és FxFBLMS algoritmusok várják blokkjaik feltöltődését, és amikor egy új blokk N mintával feltöltődik, akkor ezek ebben a ciklusban egymás után futtatásra kerülnek egy feltételvizsgálat révén. A mintavételi idők között, amikor már megtörténtek a mintavételezések és konvolúciók a megszakításokban, ezek tovább folytatják futásukat, és $K + M$ minta alatt befejeződnek, még a következő bemeneti blokkjuk feltöltése előtt, amikor futtatásuk újból megtörténik.

A fejlesztés során hamar kiderült, hogy az OLA és FxFBLMS algoritmusok igen memóriaigényessé teszik a módszeremet a blokkos végrehajtások miatt. Ezért a programfej-



4.9. ábra. *Programszervezés*

lesztés során törekedtem minél jobban és hatékonyabban kihasználni a rendelkezésemre álló memóriakapacitást. Első lépésként arra törekedtem, hogy az OLA és FxFBLMS eljárások alatt működő FFT algoritmusok, átmeneti tömböket nem igénylő "in place" FFT-k legyenek. Később az is kiderült a fejlesztőkörnyezet alapos megismerése után, hogy a 4 Mbit SRAM-hoz képest, alapértelmezésben sokkal kevesebb memória használható a programfejlesztés során, így elengedhetetlen lett a fejlesztőkörnyezetben ennek átkonfigurálása is, amit végül a 3 Mbit nagyságig sikerült megnövelni. Ezeknek a módosításoknak köszönhetően maximálisan $N_{fft} = 4096$ méretű blokkokat tudtam használni az adaptív FIR-szűrő működtetésekor.

Az adaptív FIR-szűrő együtthatószáma, L, a blokkos végrehajtás miatt korlátozott az N_{fft} blokkméret által, mert ennél a méretnél csak kisebb alkalmazható a szűrés során. Az előző alfejezetben ábrázolt hatékonyságbeli különbségnél azt lehetett tapasztalni, hogy az $L = 4096$ együttható szám az, ami felett az elméleti számolások alapján a módszernek egyre jobban és hatékonyabban kellene működnie az időtartománybeli adaptív FIR-szűrőhöz képest. A rendelkezésemre álló ADSP-21364 processzoron ennek a hatékonyságnak az igazolása nem ígérkezett egyszerű feladatnak.

5. fejezet

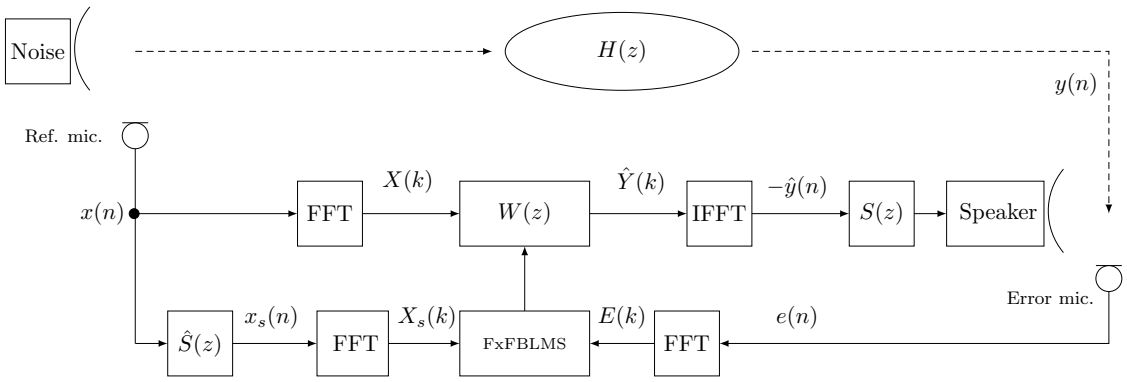
Adaptív FIR-szűrő alkalmazás

5.1. Aktív zajcsökkentés alkalmazása

A dolgozatomban bemutatott késleltetésmentesen, blokkos végrehajtások révén működő, frekvenciatartománybeli adaptív FIR-szűrőt egy aktív zajcsökkentési feladatba építettem be a 2.3 és 2.4 alfejezetekben vázolt FxLMS és NLMS kiegészítési eljárásokkal ellátva. Az FxLMS struktúra alkalmazása az előrecsatolt aktív zajcsökkentő rendszer megvalósítása miatt volt szükségszerű, a normalizált LMS eljárás beépítése pedig azért, hogy a zajcsökkentés alatt szabályzott hibamikrofon hibalecsengése gyorsabb, és a stabilitás pedig minél jobb legyen a működés során. A blokkosan működő adaptív FIR-szűrőhöz az alábbi paramétereket használtam. Ezek ismerete az eredmények értelmezésekor lényegesek lesznek.

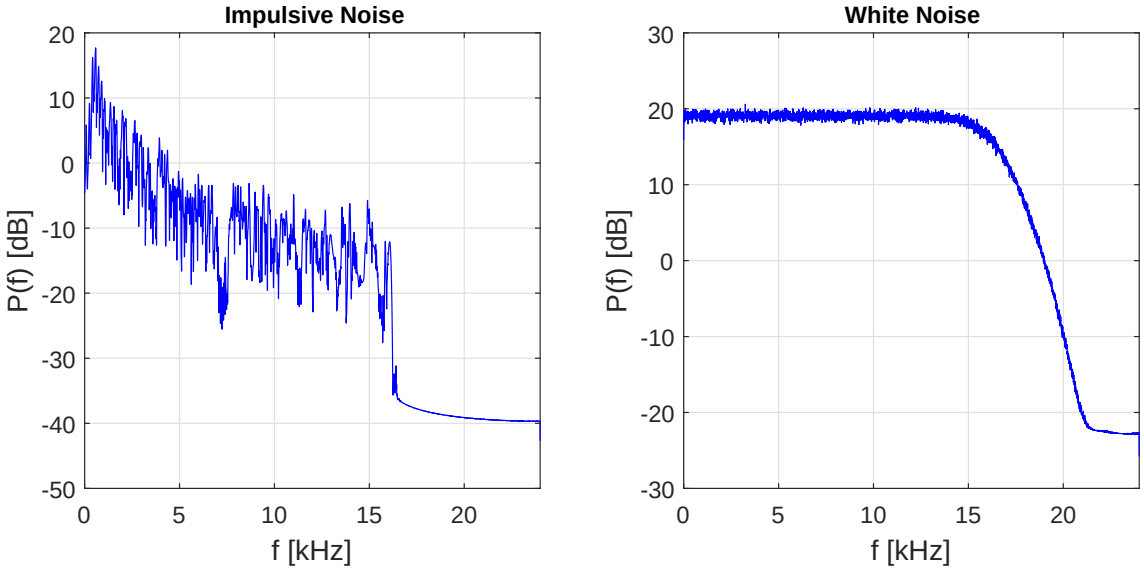
- N_{fft}, a blokkos végrehajtásokkor alkalmazott FFT blokk méret.
- L, az elsődleges út, $W(z)$ adaptív FIR-szűrőjének együtthatószáma. ($N_{fft} = L + N$)
- M, a másodlagos út, $\hat{S}(z)$ becsült FIR-szűrőjének együtthatószáma.
- μ, az aktív zajcsökkentés alatt működőd FBLMS algoritmus paramétertérben történő lépéseinek nagysága.
- a, a normalizálásnál használt korrekciós tag.

A méréseimet a Méréstechnika és Információs Rendszerek Tanszék DSP laborjában végeztem, a laborban rendelkezésre álló ADSP-21364 fejlesztőkártyával. A fejlesztőkártyán megvalósított aktív zajcsökkentő eljárásomat úgy fejlesztettem, hogy a 3.2 alfejezetben említett megfontolások miatt a másodlagos úttal történő szűrésem az időtartományban történjen. Az alábbi ábrán bemutatom ennek az eltérő megvalósításnak a blokkvázlatát.



5.1. ábra. *Aktív zajcsökkentés az FxFBLMS eljárással.*

A DSP laborban használt mérési elrendezésemben működtetett aktív zajcsökkentést fehérzajra és csattogó, impulzusszerű zaj elnyomására alkalmaztam [7]. A mindennapos, szokványos zajok sávszélessége általában 0-1 kHz közöttiek, de mindenképpen csak néhány kHz-es tartományt fednek le ezen az alacsony frekvenciatartományon [13]. A méréseim során azért ezeket a típusú zajforrásokat használtam, mert mind a fehérzaj mind pedig a csattogó zaj is 1 kHz-nél jóval magasabb frekvenciájú komponensekkel is rendelkezik, és így jól tesztelhető az, hogy nagyobb mintavételi frekvencián, a gyorsabb adaptív jelfeldolgozási algoritmus teljesítőképessége milyen, egy ilyen alkalmazási területen.



5.2. ábra. *Impulzusszerű csattogó zaj és fehérzaj teljesítményspektruma*

Csattogó zajok hagyományos előállítása metronómmal lehetséges. Sajnos a tesztelések során hamar kiderült, hogy a mechanikusan, metronómmal előállított csattogó zaj hangereje nem elég intenzív ahhoz, hogy ezzel az aktív zajcsökkentés működhessen, főleg abban az esetben, amikor még ez kombinálva is van passzív zajcsökkentési eljárással is. A probléma kiküszöbölése végett, ezért egy file-ból visszajátszott metronóm hangjellel kísérleteztem, amit egy állványra szerelt hangszórón keresztül működtettem. A fehérzaj

előállítását a DSP laborban található HP zajgenerátorral állítottam elő, majd ugyan ezen állványra szerelt hangszórón keresztül használtam fel mint elnyomandó zajforrás. A mérési összeállításban a hibamikrofont és a beavatkozó hangszórót ettől a zajforrástól kb. 3-4 méterre helyeztem el, és két mérési elrendezésben végeztem el a kísérleteket.

Az egyik alkalmazott zajcsökkentő elrendezés egy egyszerű szabadon elhelyezett beavatkozó hangszóró és hibamikrofonpár volt, egymástól 15 cm távolságra. Minden mérésben referenciamikrofonnak egy Behringer ECM8000 mikrofont használtam, ebben az elrendezésben pedig hibamikrofonnak egy Voltcraft SL-400 zajszintmérő műszert. Ebben a fajta kísérletben csak az aktív módszer volt használva zajelnyomásra.

5.3. ábra. *Szabadon lévő beavatkozó hangszóró és hibamikrofon*

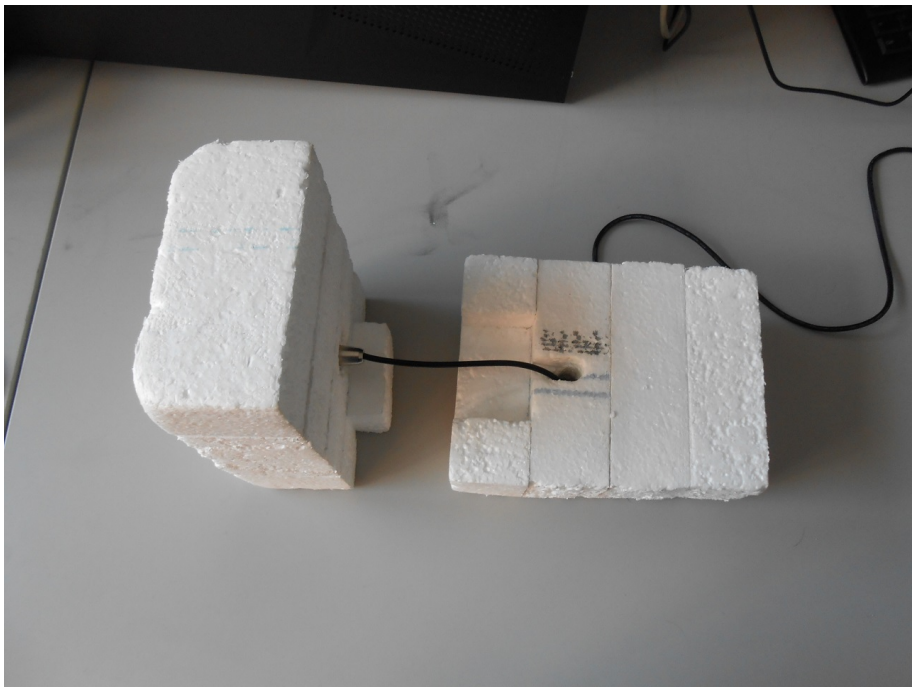
A mérések során használt másik elrendezésem már egy aktív zajcsökkentéssel ellátott fejhallgató használata volt, amiben a hibamikrofon a bal oldali fülhallgatórészben beépített volt. Ebben az elrendezésben egyszerre jelen volt mind az aktív és passzív zajelnyomás is a hungarocellből kialakított egyszerű fejmodell használata segítségével. Az önállóan működő passzív zajelnyomás és a kombinált, egyszerre működő aktív és passzív elnyomás teljesítőképességét külön-külön elemeztem a méréseim során.

5.4. ábra. *Aktív zajcsökkentő fejhallgató*

Mind a két fajta elrendezésnél az aktív zajcsökkentő, adaptív FIR-szűrőt alkalmazó eljárásomat két mintavételi frekvencián működtettem, $f_s = 24\,kHz$ és $f_s = 48\,kHz$-en, valamint a mérések során az adaptív FIR-szűrőt $L = 3700$, $L = 1900$ és $L = 960$ együtthatók mellett használtam. Ezen mérésekből arra kerestem a választ, hogy mely mintavételi frekvencia és együtthatószám mellett lehet a leghatékonyabban elnyomni az előzőekben már említett zajokat. A következő alfejezetekben a mérési eredményekből csak a legeredményesebbeket prezentálom, a további mérési eredmények a mellékletben tekinthetők meg.

5.2. Passzív zajcsökkentés teljesítőképessége fehérzajra és impulzusszerű csattogó zajra

A passzív zajcsökkentés eredményességét a fejhallgató és a hungarocellből kialakított fejmodell segítségével vizsgáltam $15\,kHz$ sávszelességű fehérzajjal és egy $75\,bpm$ ütésszámú, metronómból származó csattogó zajjal (forrás: metronomer.com). A mérés során a fejhallgatóba épített hibamikrofonról felvételt készítettem egy labor PC hangkártyájával, majd ábrázoltam ennek teljesítményspektrumát két esetre, amikor a fejhallgató szabadon volt elhelyezve, és amikor ez a hungarocell modellre volt rögzítve, a passzív elnyomás tesztelése érdekében. Mivel a hungarocell doboz kialakítása lehetővé teszi egy második külső hibamikrofon bevezetését is a bal oldali fülhallgatóhoz, így nem csak a fejhallgatóba épített mikrofonnal végeztem el a méréseket, hanem a Voltcraft SL-400 zajszintmérő műszer mikrofonjával is.



5.5. ábra. *Hungarocellbe bevezetett hibamikrofon*

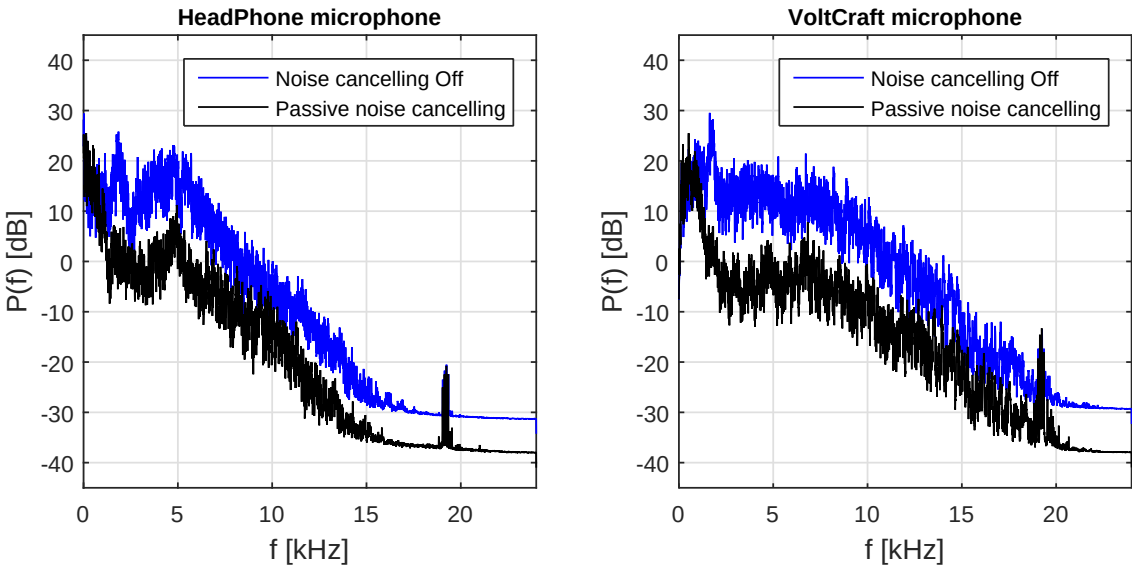
A mérések kiértékelést úgy végeztem el, hogy a teljesítményspektrumok, $P(f)$ integrálját hasonlítottam össze a felvett zajnak és a passzív módszerrel zaj-csökkentett felvételnek. Ezek után a integrálok eredményét dB-be számoltam át, még pedig úgy, hogy a számítás során a viszonyítási alapnak a csillapítatlan zajt vettem, hogy az eredményben egyszerűen megkaphassak egy relatív dB csökkenést a csillapítatlan esethez képest. Ezeket az eredményeket táblázatban foglalom össze a továbbiakban.

$$P_{int} = \frac{2}{f_s} \int_0^{f_s/2} P(f)df \tag{5.1}$$

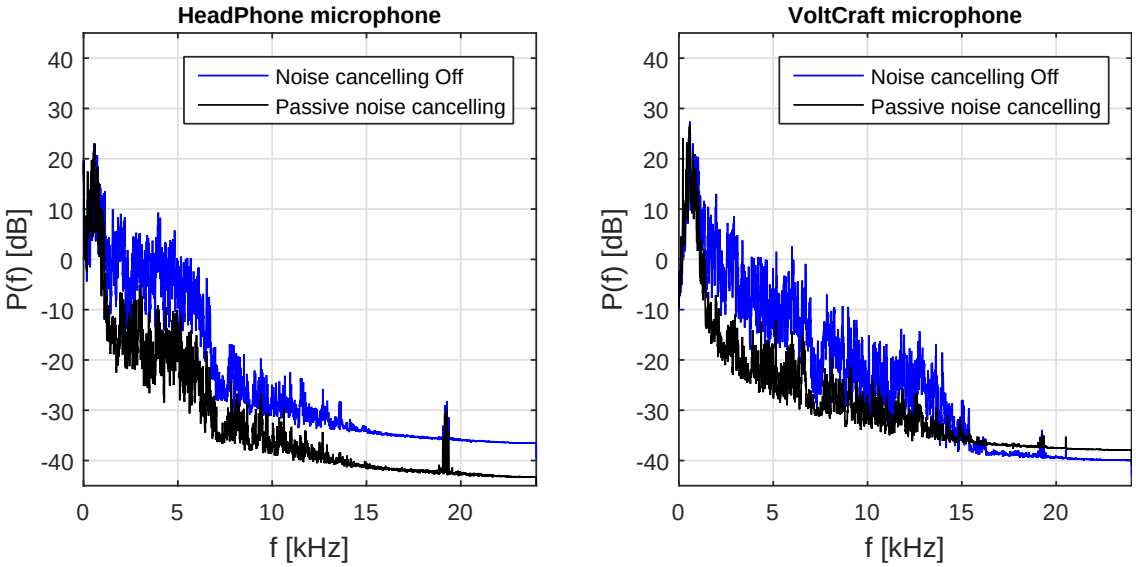
$$P_{dB} = 10 \log_{10} \left(\frac{P_{int,pass}}{P_{int,zaj}} \right) \tag{5.2}$$

A teljesítményspektrumokon jól látszik az az eredmény, hogy a passzív zajelnyomás leginkább a magasabb frekvenciatartományokra koncentrálódik, és a kiértékelések alapján ez kb. $7\,dB$ elnyomást tud biztosítani fehérzaj elnyomása esetén. Csattogó zajok esetén viszont csak igen csekély $1\,dB$ elnyomás tapasztalható. A teljesítményspektrumok alapján ez jól magyarázható azzal, hogy az ilyen fajta zaj sávszélessége éppen abban az alacsony tartományban van, ahol a passzív elnyomás már nem hatékony.

A két mikrofon között is lényeges különbség figyelhető meg. A fejhallgatóba beépített mikrofon sávszelessége $(0-5)\,kHz$ között adódott, míg a zajszintmérő műszer mikrofonja $(0-10)\,kHz$-es tartományban volt használható. Ezen eredmények alapján az aktív zajcsökkentés alatt, a továbbiakban már a bevezetett külső, zajmérő műszer mikrofonját használtam a fejhallgatót is használó elrendezések mérése során is.



5.6. ábra. *Passzív zajelnyomás valamint a Voltcraft SL-400 zajszintmérő műszer és a fejhallgatóba beépített mikrofon teljesítményspektrumának összehasonlítása fehérzaj esetén*



5.7. ábra. *Passzív zajelnyomás valamint a Voltcraft SL-400 zajszintmérő műszer és a fejhallgatóba beépített mikrofon teljesítményspektrumának összehasonlítása csattogó zaj esetén*

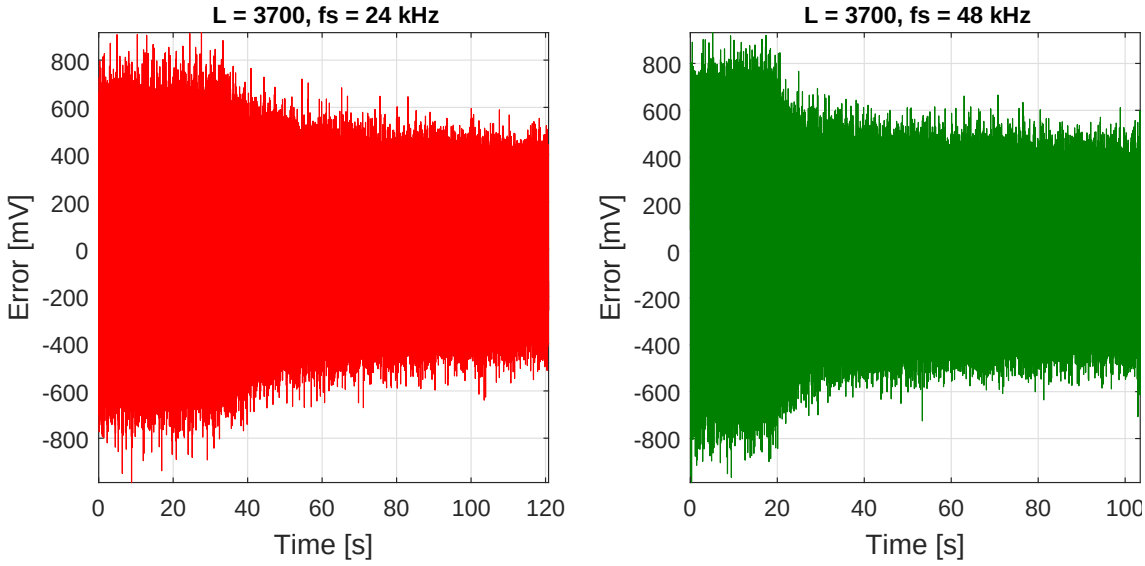
Mérési módszerben használt mikrofon	Fehérzaj elnyomása [dB]	Csattogó zaj elnyomása [dB]
Zajmérő mikrofon	-7.4919	-0.73122
Fejhallgató mikrofon	-6.904	-0.91896

5.1. táblázat. *Passzív zajcsökkentés esetén, a teljesítményspektrumok alapján számított relatív dB csökkenések*

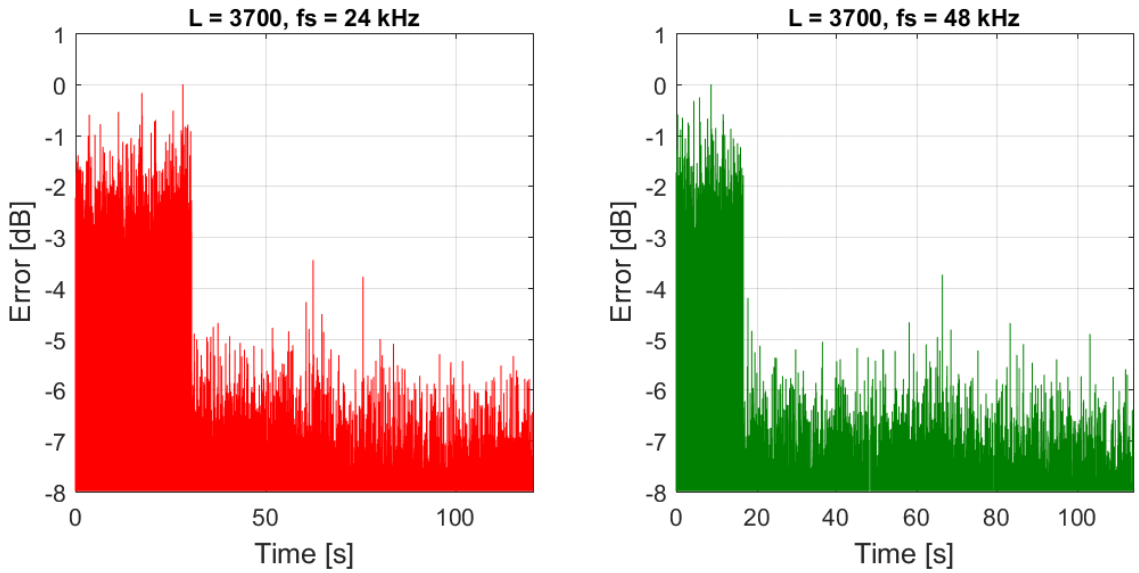
5.3. Fehérzaj aktív zajcsökkentésének eredménye

Az aktív zajcsökkentési mérések alatt használtam fel az előzőekben már bemutatott két mérési elrendezést. Méréseket végeztem a szabadon lévő hibamikrofon elrendezéssel és az aktív zajcsökkentéssel ellátott fejhallgató használatával is, amikor a passzív elnyomás is működik a zajcsökkentés alatt. A legjobb eredmények mind a két elrendezés esetén $L = 3700$ nagyságú adaptív FIR-szűrőre voltak tapasztalhatóak. Ezek az eredmények a következő alfejezetekben tekinthetők meg, a többi mérési eredményemet a melléklet tartalmazza. A kiértékeléseket és ábrázolásokat az előző fejezetben már használt módszer szerint végeztem itt is.

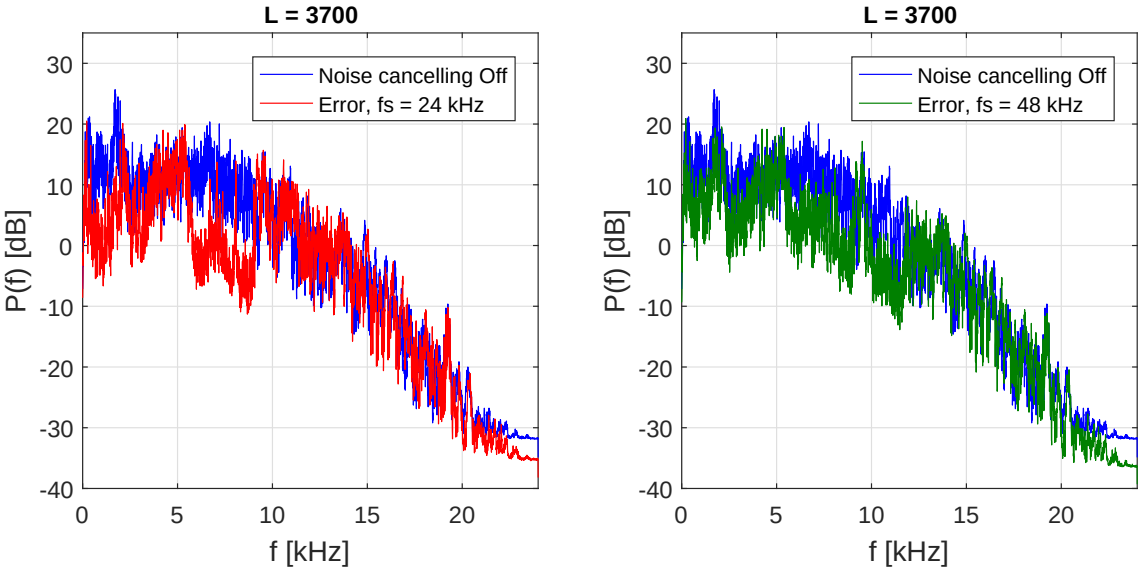
5.3.1. Aktív zajcsökkentő fejhallgató, $L = 3700$, $\mu = 0.004$, $a = 0.1$, $M = 900$



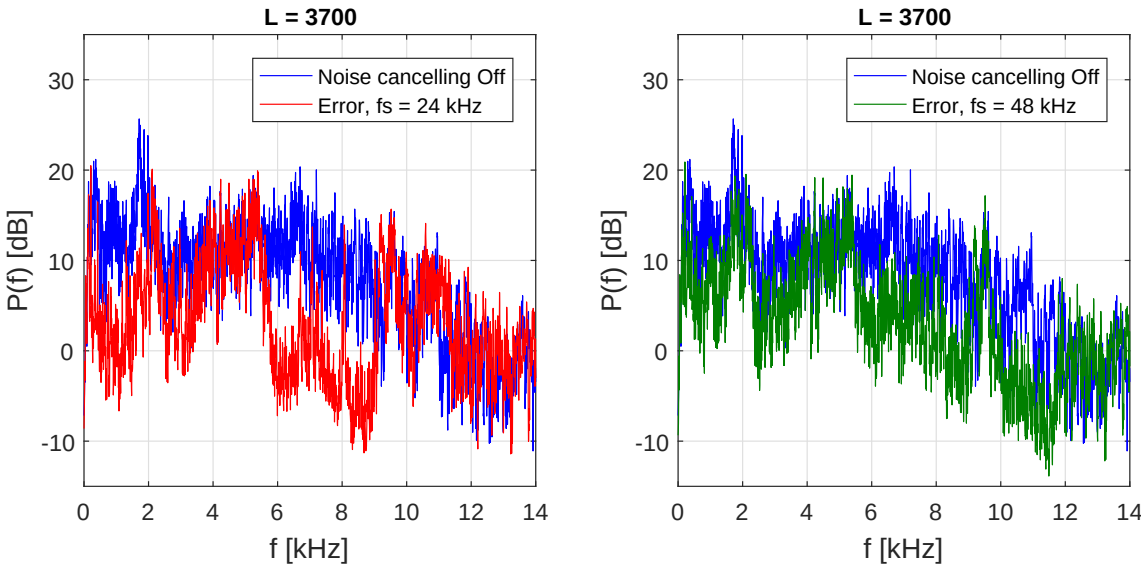
5.8. ábra. *Hibalecsengés, $L = 3700$*



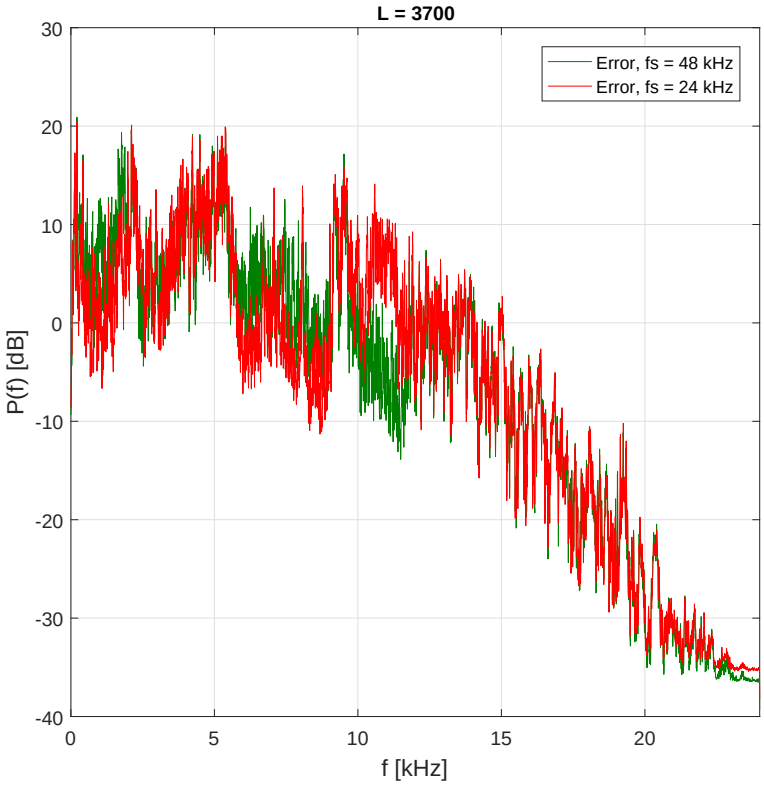
5.9. ábra. *Hibacsökkenés mértéke dB-ben,* $L = 3700$



5.10. ábra. *Hibajel teljesítményspektruma,* $L = 3700$

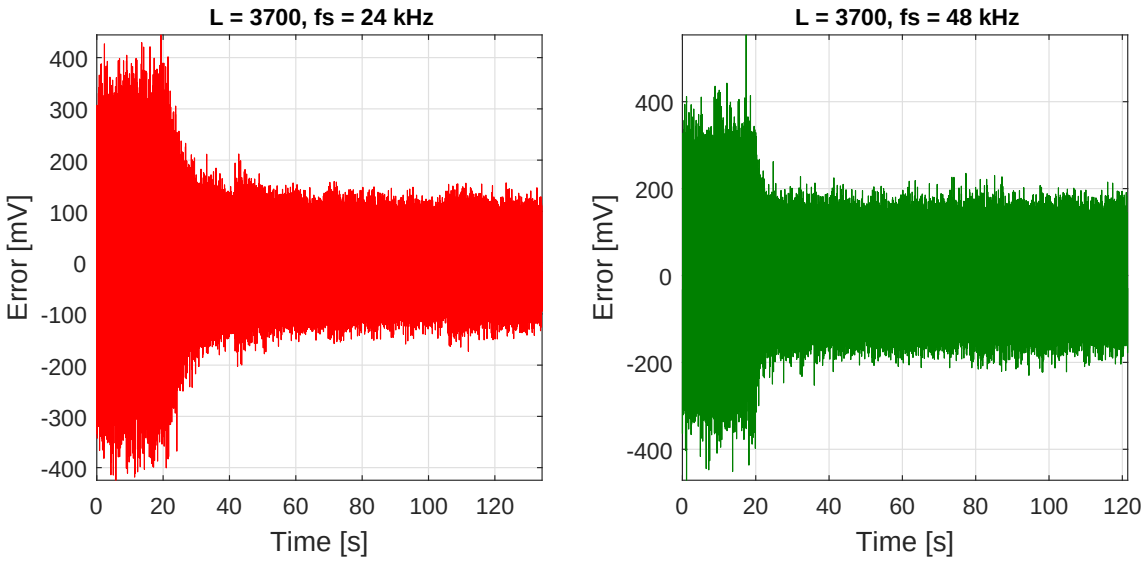


5.11. ábra. *Hibajel teljesítményspektruma (0-15) kHz tartományban,* $L = 3700$

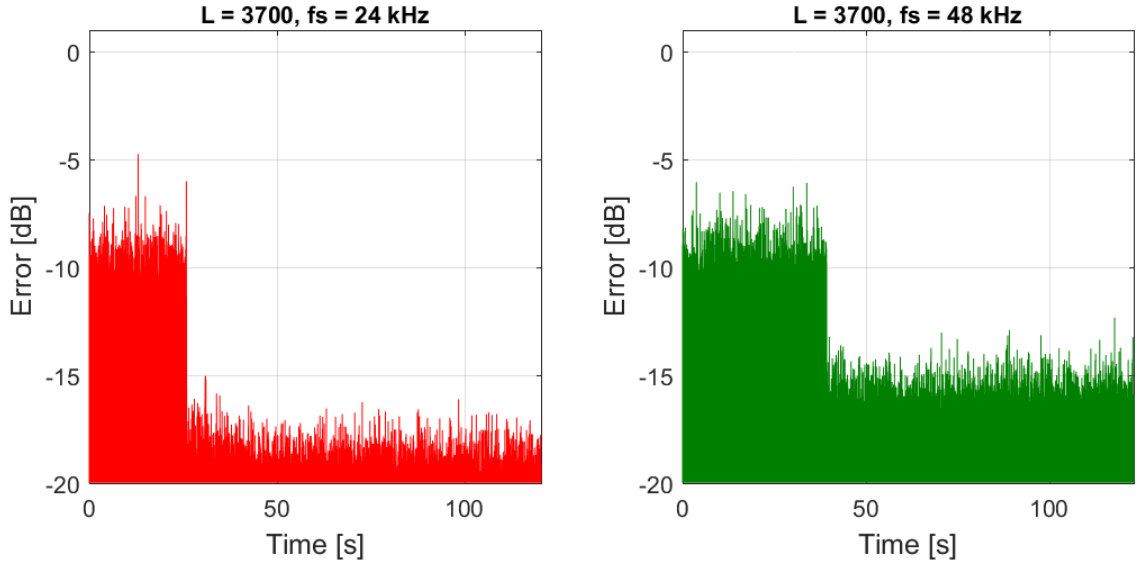


5.12. ábra. *24 kHz és 48 kHz mintavételi frekvenciák mellett működő aktív zajcsökkentés összehasonlítása,* $L = 3700$

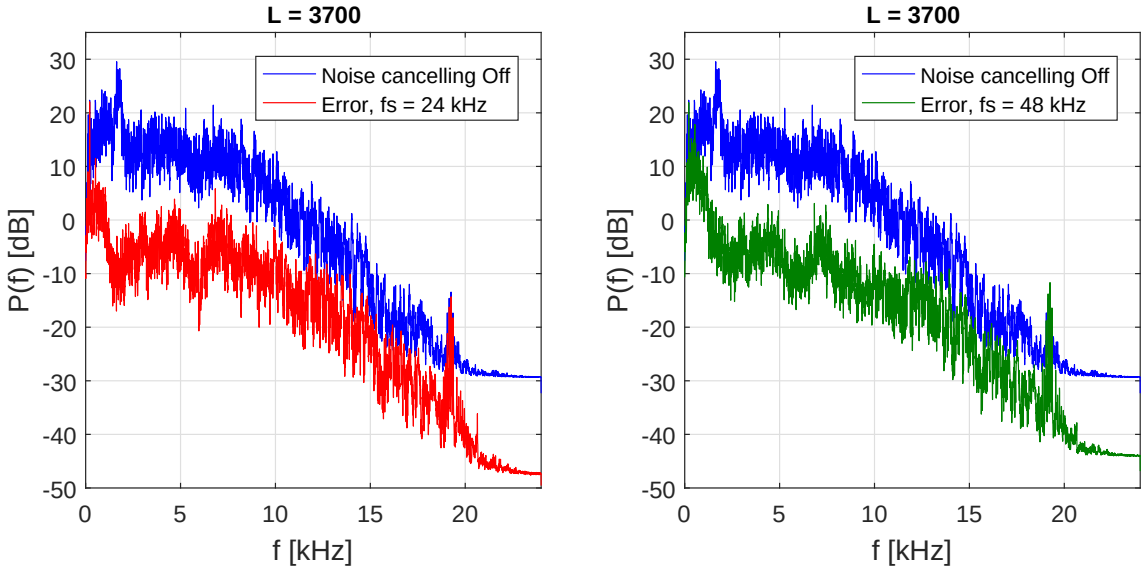
5.3.2. Aktív zajcsökkentő fejhallgató, $L = 3700$, $\mu = 0.004$, $a = 0.1$, $M = 900$



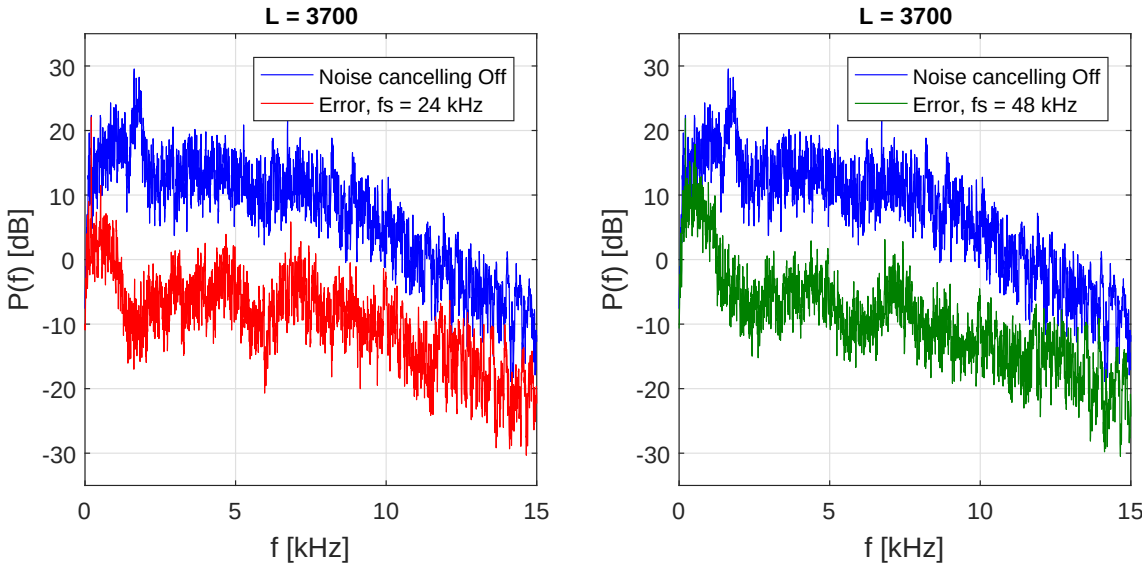
5.13. ábra. *Hibalecsengés, $L = 3700$*



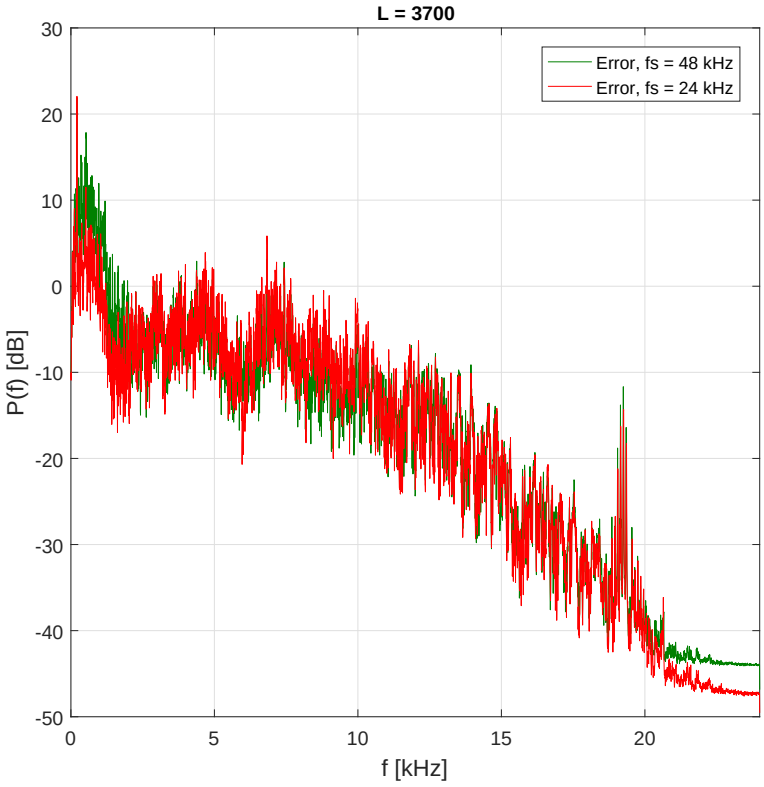
5.14. ábra. *Hibacsökkenés mértéke dB-ben, $L = 3700$*



5.15. ábra. *Hibajel teljesítményspektruma,* $L = 3700$



5.16. ábra. *Hibajel teljesítményspektruma (0-15) kHz tartományban,* $L = 3700$

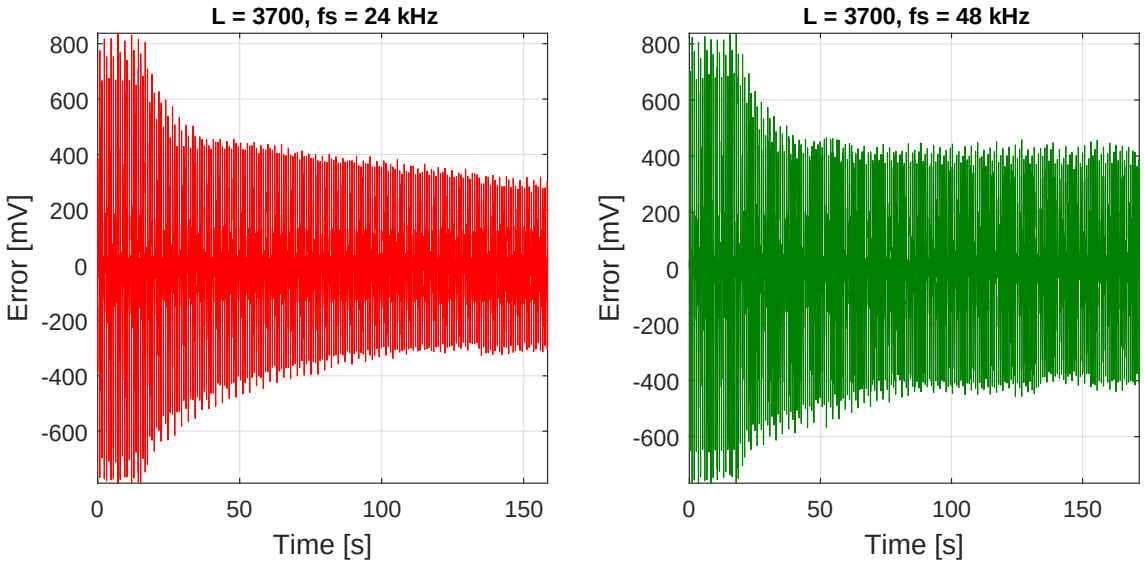


5.17. ábra. *24 kHz és 48 kHz mintavételi frekvenciák mellett működő aktív zajcsökkentés összehasonlítása,* $L = 3700$

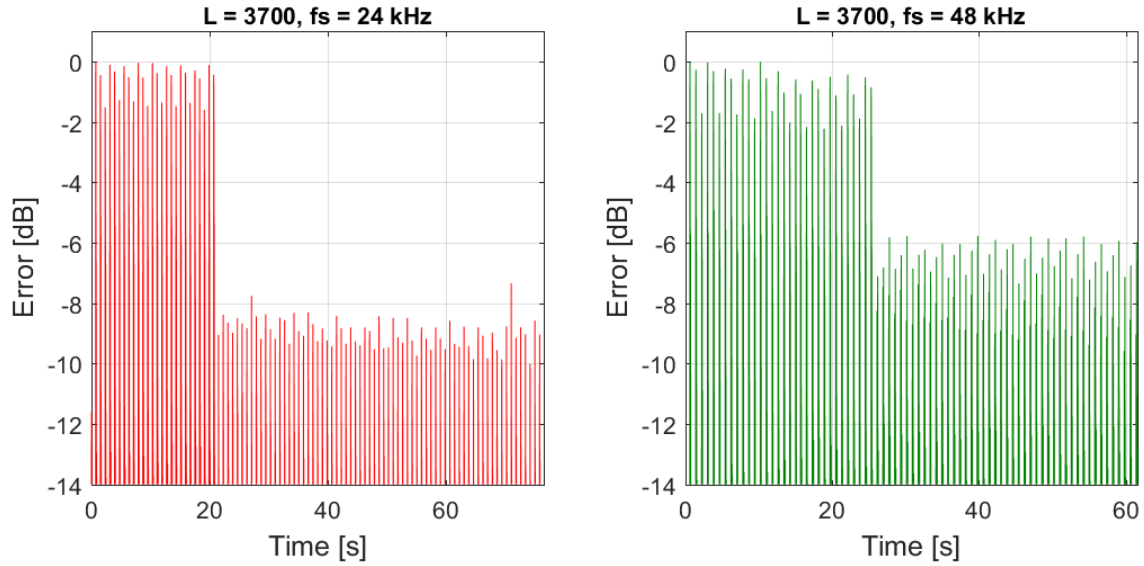
5.4. Csattogó zajok aktív zajcsökkentésének eredménye

A csattogó zajok aktív zajcsökkentését is a már említett két fajta elrendezésben vizsgáltam. A legjobb eredmények ismét az $L = 3700$ nagyságú adaptív FIR-szűrőre voltak tapasztalhatóak. Ezeket az eredményeket ismét a következő alfejezetekben mutatom be, és a többi mérési eredményemet a melléklet tartalmazza.

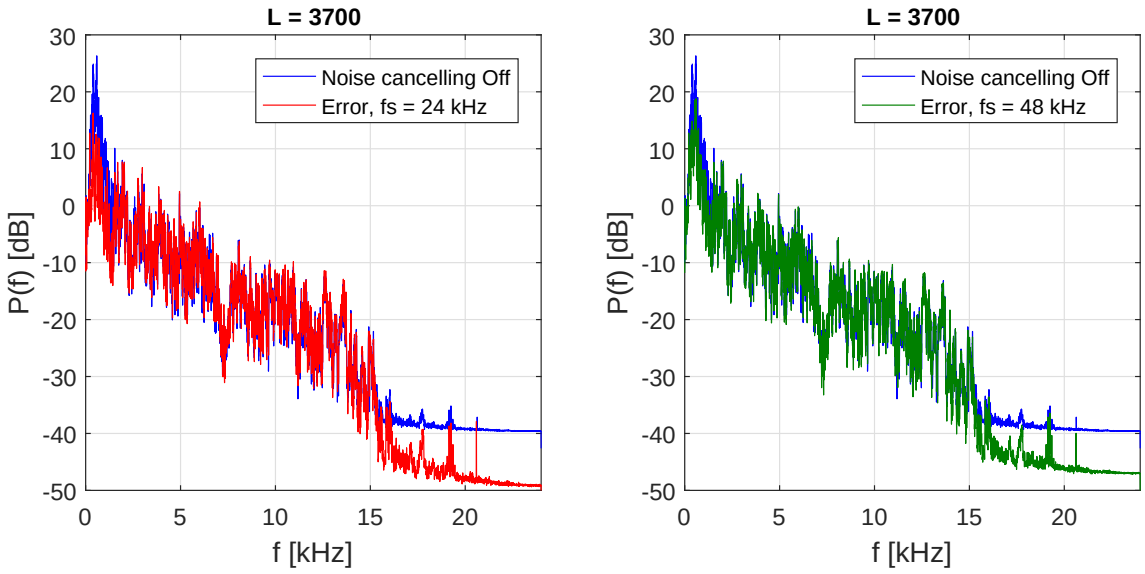
5.4.1. Szabadon lévő hibamikrofon, $L = 3700$, $\mu = 0.005$, $a = 100.0$, $M = 900$



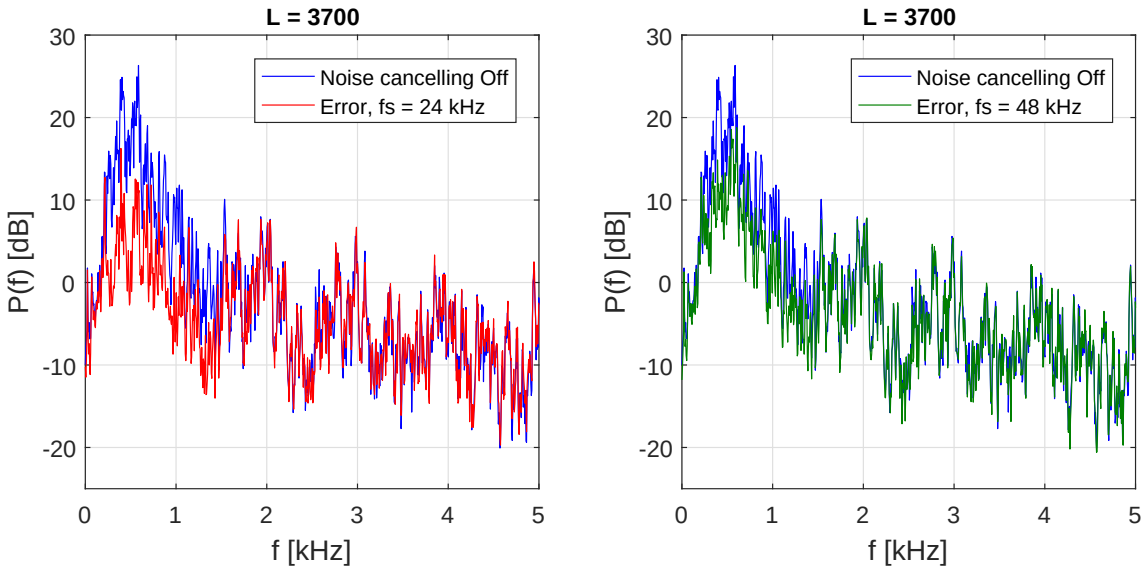
5.18. ábra. *Hibalecsengés, $L = 3700$*



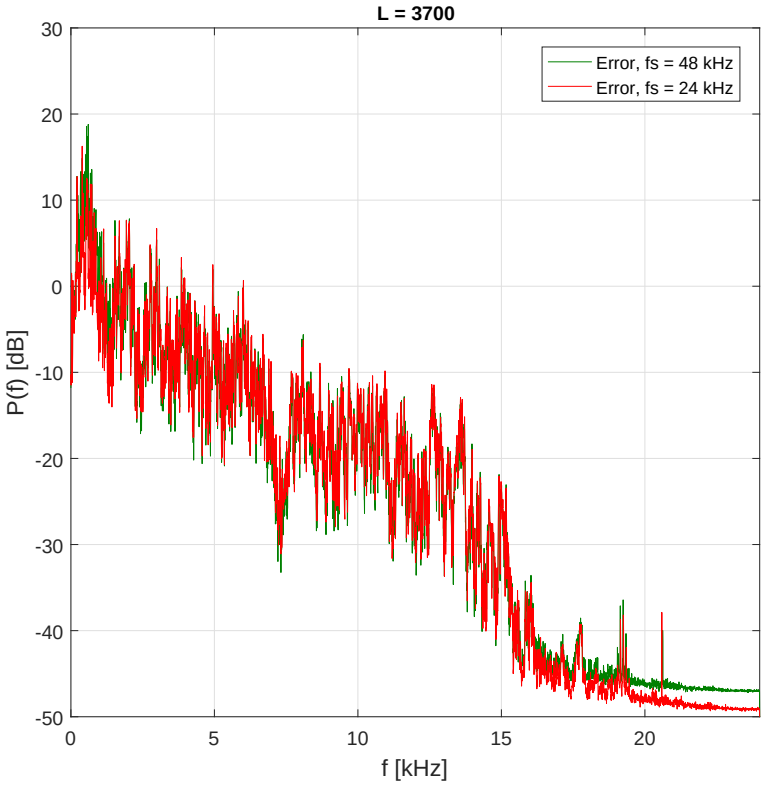
5.19. ábra. *Hibacsökkenés mértéke dB-ben, $L = 3700$*



5.20. ábra. *Hibajel teljesítményspektruma,* $L = 3700$

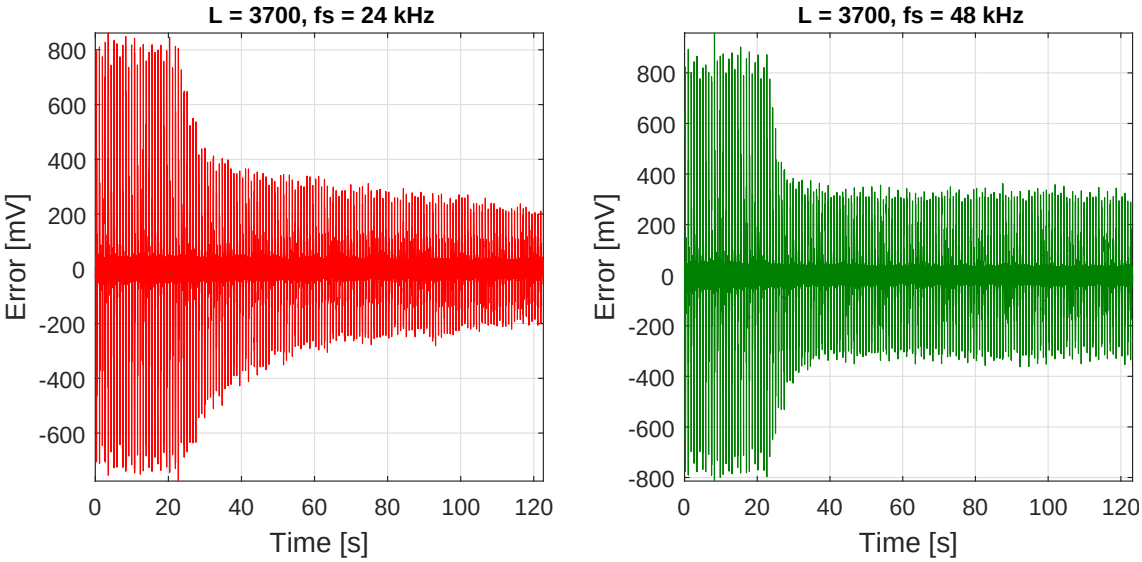


5.21. ábra. *Hibajel teljesítményspektruma (0-5) kHz tartományban,* $L = 3700$

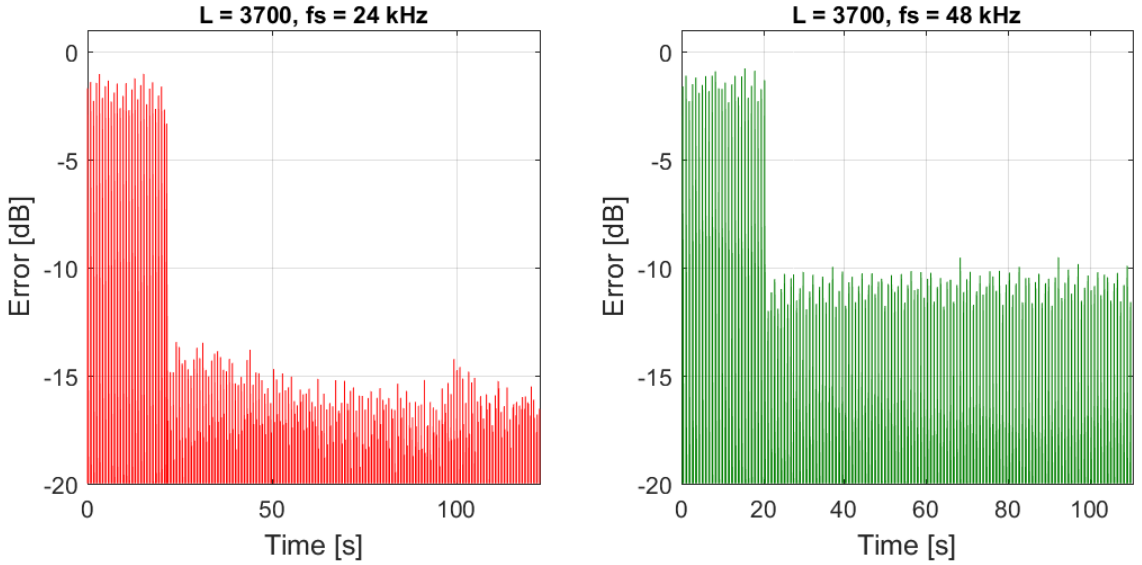


5.22. ábra. *24 kHz és 48 kHz mintavételi frekvenciák mellett működő aktív zajcsökkentés összehasonlítása,* $L = 3700$

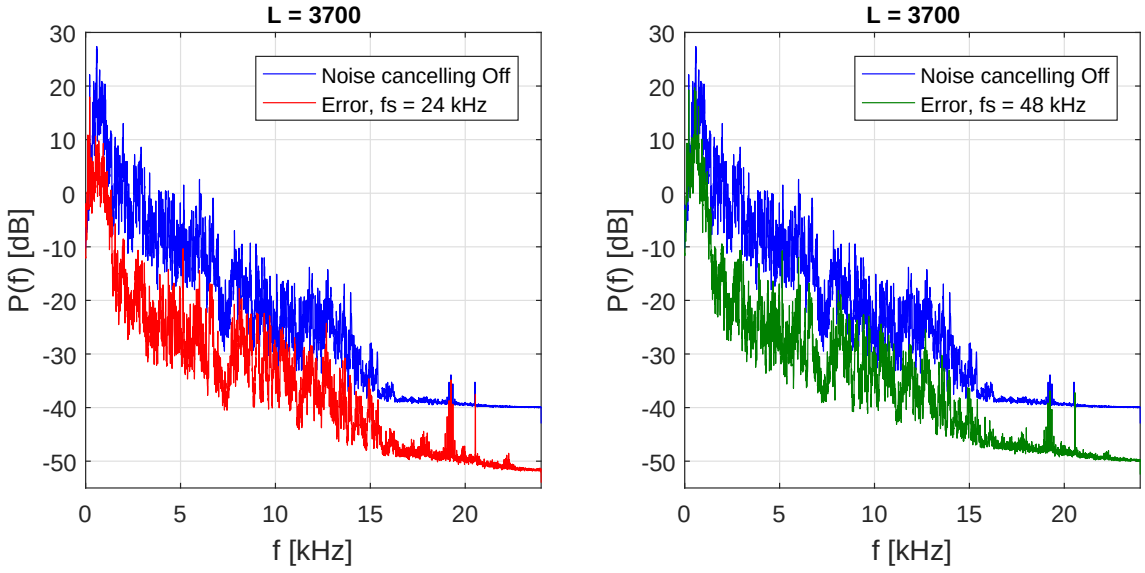
5.4.2. Aktív zajcsökkentő fejhallgató, $L = 3700$, $\mu = 0.005$, $a = 100.0$, $M = 900$



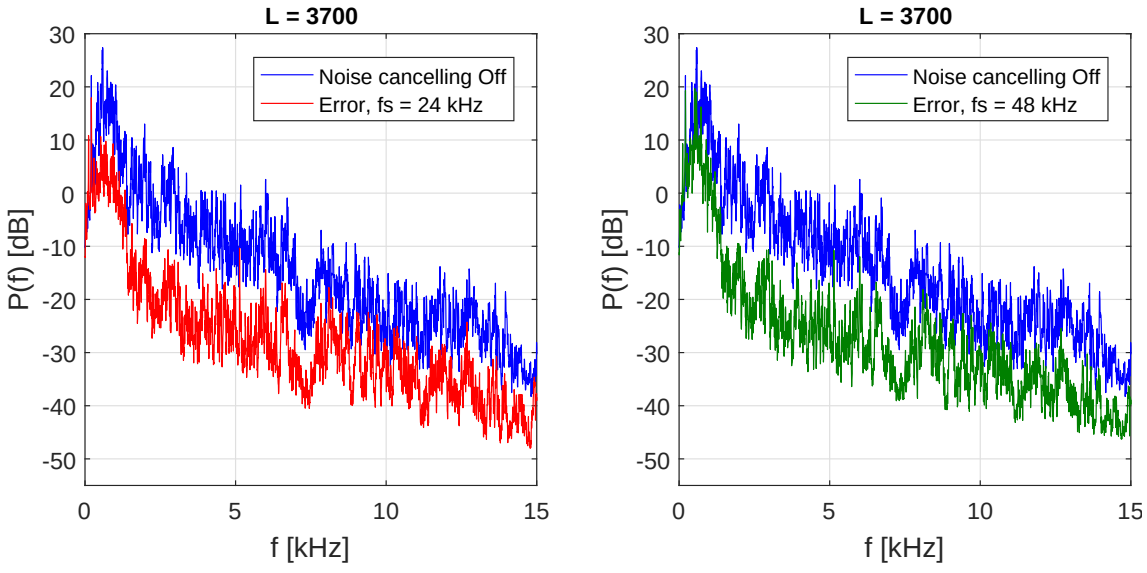
5.23. ábra. *Hibalecsengés, $L = 3700$*



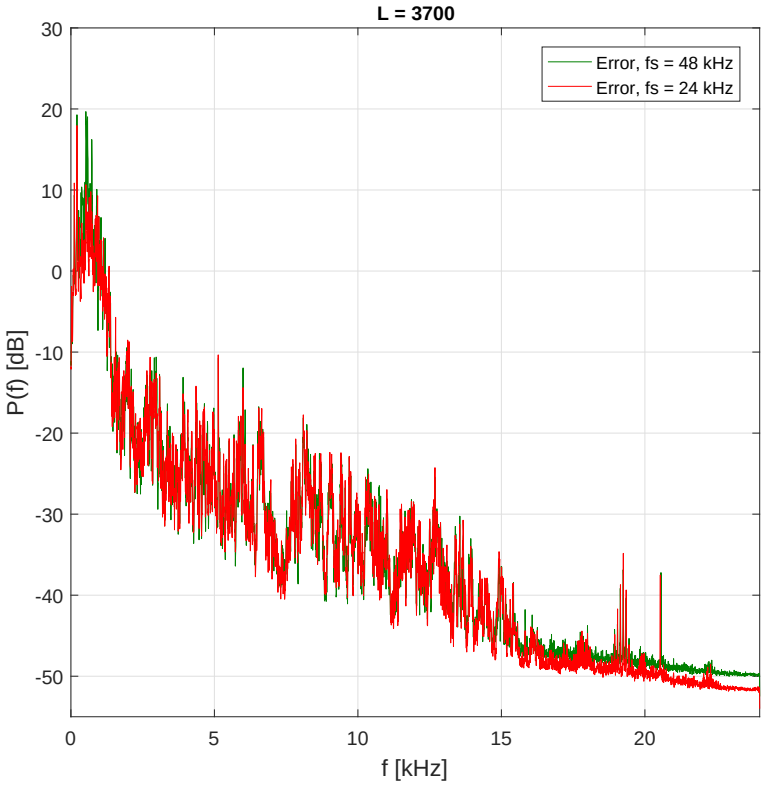
5.24. ábra. *Hibacsökkenés mértéke dB-ben, $L = 3700$*



5.25. ábra. *Hibajel teljesítményspektruma,* $L = 3700$



5.26. ábra. *Hibajel teljesítményspektruma (0-5) kHz tartományban,* $L = 3700$



5.27. ábra. *24 kHz és 48 kHz mintavételi frekvenciák mellett működő aktív zajcsökkentés összehasonlítása,* $L = 3700$

5.5. Aktív zajcsökkentési eredmények összegzése

Az eredmények elemzését követően mind a fehérzaj mind pedig a csattogó zajok zajcsökkentésénél jól látható a teljesítményspektrumokban, hogy az aktív zajcsökkentésnek leginkább az alacsony frekvenciás tartományokban van fő szerepe, ahol a zajok teljesítménye a legerősebb. A mérések igazolták azt is, hogy minél nagyobb adaptív FIR-szűrő együttható mellett egyre jobb aktív zajcsökkentés érhető el, ahogy ez várható is volt.

A mérések másik feltűnő érdekesség még az, hogy a mintavételi frekvencia optimális megválasztás nagyban növelheti még az adaptív szűrő zajelnyomási képességét. Az eredményeket összefoglaló 5.2. és 5.3. táblázatokban ez jól észrevehető mind a fehérzajú mind pedig a csattogó zajú zajcsökkentési mérések során, hogy néhány dB-el jobb zajelnyomás érhető el azonos adaptív FIR-szűrő együtthatószám alatt, ha a jelfeldolgozó rendszer mintavételi frekvenciáját az elnyomandó szélessávú zajhoz igazítjuk. Ez az érdekes eredmény a 24 kHz és 48 kHz mintavételi frekvenciák mellett működő aktív zajcsökkentések összehasonlító ábráinál is nagyon szépen észrevehetők. Ez a jelenség véleményem szerint azzal magyarázható, hogy ekkor az éppen használt adaptív FIR-szűrő együtthatószáma jobban tud koncentrálódni a mintavételi frekvenciával meghatározott kisebb sávszélességű tartományban, és jobb modellt tud alkotni az elsődleges és másodlagos út adaptálása során.

Adaptív FIR-szűrő együtt-hatószáma	$L = 3700$		L = 1900		L = 960	
f_s	24 kHz	48 kHz	24 kHz	48 kHz	24 kHz	48 kHz
Szabadon lévő hibamikrofon	-4.21 dB	-4.32 dB	-4.36 dB	-2.65 dB	-3.06 dB	-1.63 dB
Aktív zajcsökkentő fejhallgató	-17.47 dB	-14.14 dB	-12.74 dB	-9.35 dB	-9.52 dB	-7.54 dB

5.2. táblázat. *Aktív zajcsökkentés elnyomási eredményei fehérzaj esetén dB-ben*

Adaptív FIR-szűrő együtthatószáma	$L = 3700$		L = 1900		L = 960	
f_s	24 kHz	48 kHz	24 kHz	48 kHz	24 kHz	48 kHz
Szabadon lévő hibamikrofon	-8.76 dB	-5.73 dB	-5.84 dB	-3.98 dB	-3.71 dB	-1.97 dB
Aktív zajcsökkentő fejhallgató	-12.54 dB	-7.95 dB	-8.60 dB	-4.43 dB	-4.31 dB	-1.88 dB

5.3. táblázat. *Aktív zajcsökkentés elnyomási eredményei csattogó zaj esetén dB-ben*

A magasabb mintavételi frekvenciát használt mérések eredményeiben csupán annyi

előny figyelhető meg, hogy a hibalecsengés gyorsabbnak adódott. Ezekből az aktív zajcsökkentő eredményekből az a konklúzió vonható le, hogy a hatékony zajelnyomás elérése érdekében csak az adaptív FIR-szűrő együtthatószámának növelése még önmagában nem a legjobb zajelnyomást teszi lehetővé. A legjobb zajelnyomási eredmények érdekében jó, ha tisztában vagyunk az elnyomandó zajok sávszélességével, és azokhoz mérten választjuk meg a jelfeldolgozó rendszer mintavételi frekvenciáját, mivel ezzel még jó néhány plusz dB elnyomás érhető el az aktív zajcsökkentés alatt.

További érdekes mérési eredményeket az adaptív FIR-szűrő együtthatószámának nagy mértékű megnövelésével lehetne még elérni. A 4.3 fejezetben említett, blokkos végrehajtásokból adódó memóriakapacitási limit miatt csak $L = 3700$ a maximálisan használható együtthatószám a bemutatott aktív zajcsökkentési rendszeremben, ezzel szemben az időtartománybeli FxLMS algoritmussal működő aktív zajcsökkentés az ADSP-21364 jelprocesszoron 6-7 ezer együttható használatát teszi lehetővé. Lényegesen több memóriakapacitás mellett megnyílna a lehetőség igazolni a 4.2. ábrán bemutatott elméleti számításokat, miszerint az idő- és frekvenciatartománybeli késleltetésmentesen működő adaptív FIR-szűrő gyorsabb, mint számos más időtartománybeli eljárás. Végezetül még a másik érdekes és hasznos kísérlet az lehetne, hogy a csattogó zajok esetén tapasztalható monoton növekedő elnyomási különbség a 24 és 48 kHz mintavételi mérésekben, milyen kapcsolatban van az adaptív FIR-szűrő együtthatószámának növelésével.

Összefoglalás

Dolgozatomban az adaptív FIR-szűrők hatékony megvalósításának módját vizsgáltam, amivel az aktív zajcsökkentő alkalmazásokban használt nagy fokszámú adaptív FIR-szűrők együtthatóigényének problémája megoldható. Munkám során körüljártam, hogy az idő- és frekvenciatartományban milyen eljárások szolgálnak az adaptív FIR-szűrők implementálására és ezek milyen hátrányokkal és előnyökkel járnak egymáshoz képest.

A szakirodalmat körüljárva egy olyan idő- és frekvenciatartományban működő eljárást dolgoztam ki, amivel megvalósíthatók a nagy fokszámú késleltetésmentesen működtethető adaptív FIR-szűrők. Elméleti számításaimmal igazoltam ennek hatékonyságát, és egy módszert is bemutattam, amivel egy adott processzoron és hardverkörnyezetben megbecsülhető rögzített blokkméret mellett a maximálisan elérhető együtthatók száma az adaptáció során, majd egy jelprocesszoron a helyes működést demonstráltam.

Dolgozatom végét egy olyan zajcsökkentő rendszer alkalmazásával zártam le, amely csak ezen hatékony struktúra használatával válik elérhetővé a frekvenciatartományban működtetve. A fejlesztések során természetesen újabb és újabb a hatékonyságot növelhető ötletek fogalmazódtak meg. Közvetlen továbblépési lehetőség ugyanennek az algoritmusnak az ADSP Sharc processzorcsalád valamely több memóriával rendelkező tagján történő megvalósítása. Ez esetben érdemi programfejlesztés nélkül lehet újabb eredményeket elérni. További cél lehet a más platformon pl. FPGA-n, grafikus processzoron történő megvalósítás.

Irodalomjegyzék

[1] John J. Shynk. Frequency-domain and multirate adaptive filtering. *IEEE Signal Processing Magazine*, 9(1):14–37, January 1992.

[2] W. G. Gardner. Efficient convolution without input-output delay. *Audio Eng. Soc.*, 43(3):127–136, March 1995.

[3] Oppenheim and Shafer. *Digital Signal Processing.* Prentice-Hall, New Jersey, 1975.

[4] E. Ferrara. Fast implementation of lms adaptive filters. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(4):474–475, August 1980.

[5] David S. McGrath. *Method and apparatus for filtering an electronic environment with improved accuracy and efficiency and short flow-through delay.* Lake Dsp Pty Limited, March 1996. URL: https://www.google.com/patents/US5502747.

[6] Wenshun Tian. *Method for efficient and zero latency filtering in a long impulse response system.* Stmicroelectronics Asia Pacific Pte Ltd., December 2012. URL: https://www.google.com/patents/US8340285.

[7] Peng Lia and Xun Yu. Active noise cancellation algorithms for impulsive noise. *Mechanical Systems and Signal Processing*, 36(2):630–635, April 2013.

[8] Tukey John W. Cooley James W. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, April 1965.

[9] E. O. Brigham. *The Fast Fourier Transform.* Prentice-Hall, Englewood Cliffs, New Jersey, 1974.

[10] BME MIT Tanszéki Munkaközösség. Digitális jelfeldolgozás, 2006.

[11] Peter N. Stearns Bernard Widrow. *Adaptive Signal Processing.* Prentice-Hall, New Jersey, 1985.

[12] Balogh Tibor Sujbert László. Adaptív szűrők vizsgálata - 3. mérés, 2010.

[13] D. R. Morgan S. M. Kuo. Active noise control: a tutorial review. *Proceedings of the IEEE*, 87(6):943–973, June 1999.

[14] Nelson P.A Elliot S.J. Active noise control. *Signal Processing Magazine, IEEE*, 10(4):12–35, October 1993.

[15] National Instruments. *Least Mean Squares Algorithms (Adaptive Filter Toolkit).* National Instruments Ltd, June 2008.

[16] Aurelio Uncini. *Fundamentals of Adaptive Signal Processing.* Springer, Signals and Communication Technology Series, London, 2015.

[17] Eugene Walach Bernard Widrow. *Adaptive Inverse Control: A Signal Processing Approach, Reissue Edition.* Wiley-IEEE Press, New Jersey, 2007.

[18] P. A. Nelson and S. J. Elliott. *Active Control of Sound.* Academic Press, San Diego, CA, 1992.

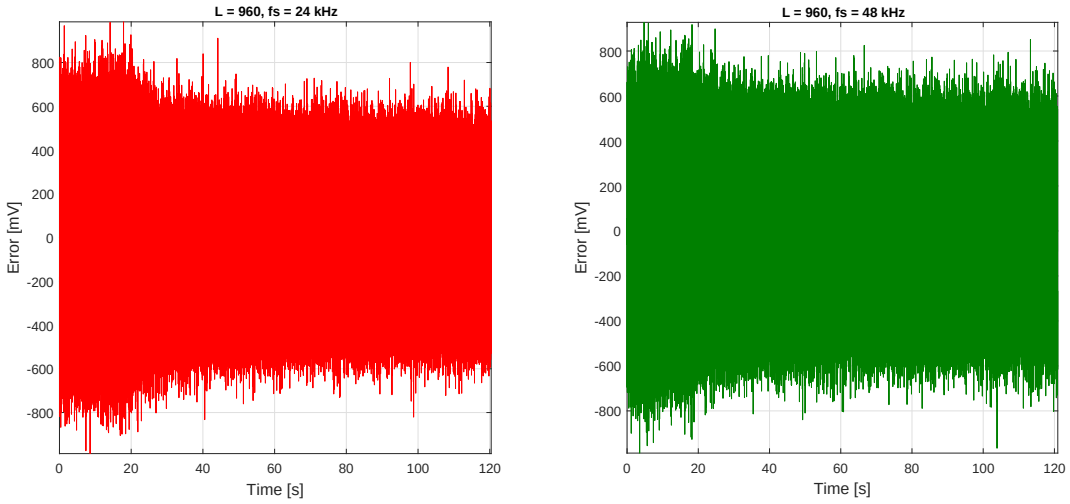
[19] C. H. Hansen and S. D. Snyder. *Active Control of Noise and Vibration.* CRC Press, 2nd edition, 2012.

[20] P. A. Nelson and S. J. Elliott. *The application of adaptive filtering to the active control of sound and vibration.* University of Southampton, 1985.

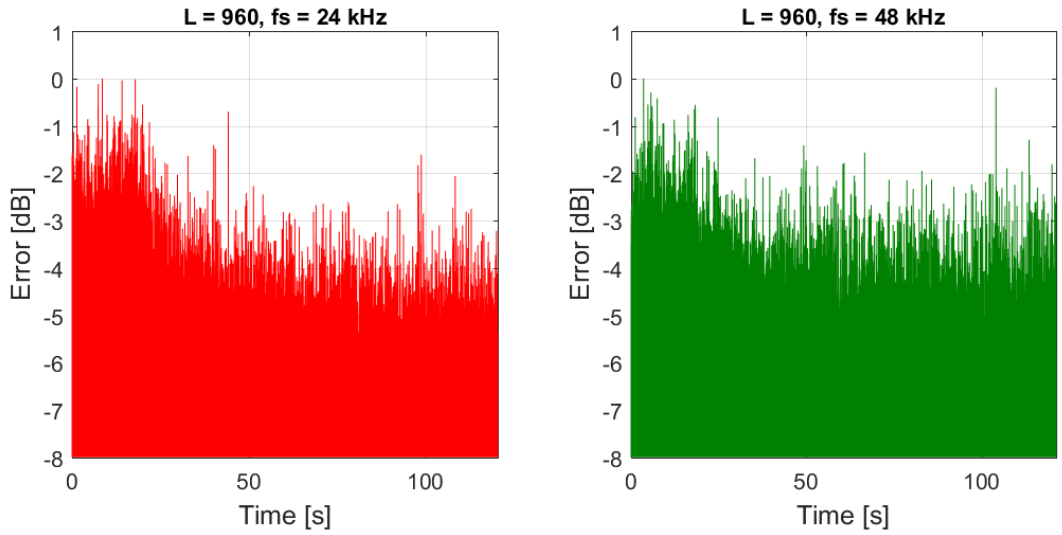
Melléklet

M.1. Fehérzaj aktív zajcsökkentésének eredménye

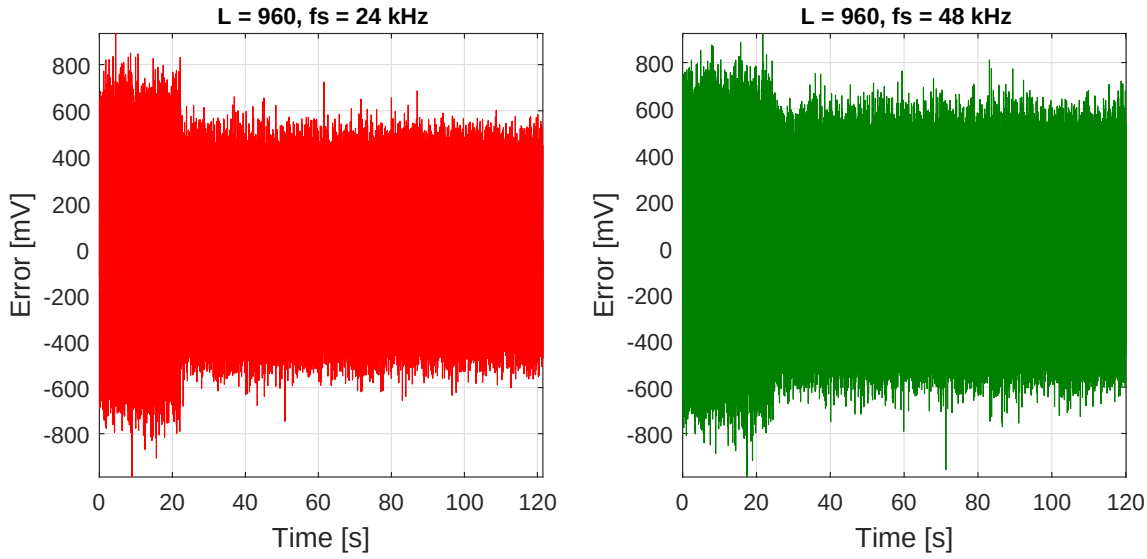
M.1.1. Szabadon lévő hibamikrofon, $L = 960$, $\mu = 0.004$, $a = 0.1$, $M = 900$



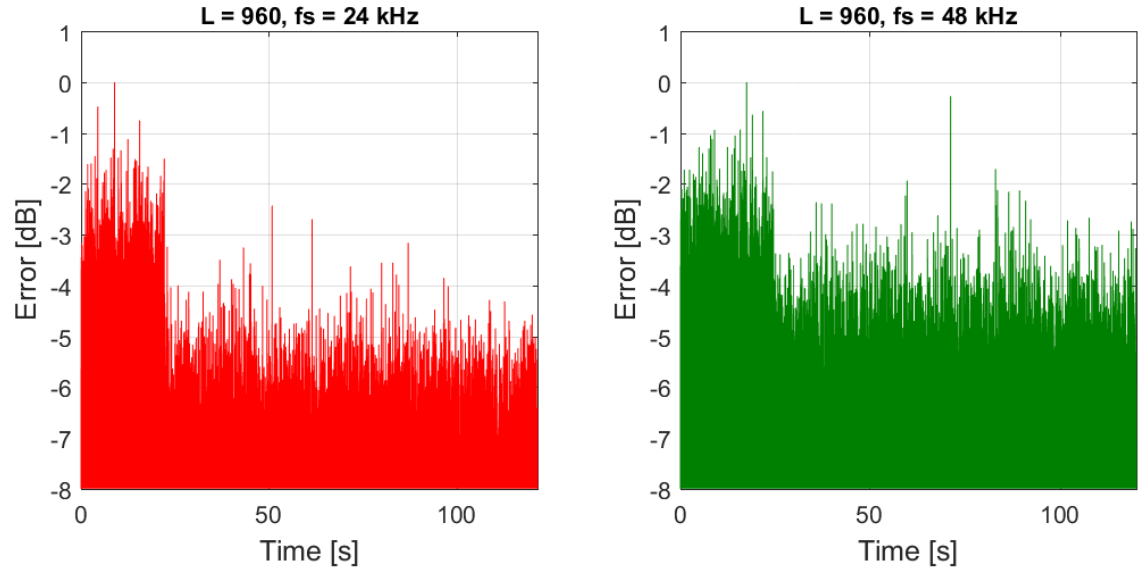
M.1.1. ábra. *Hibalecsengés,* $L = 960$



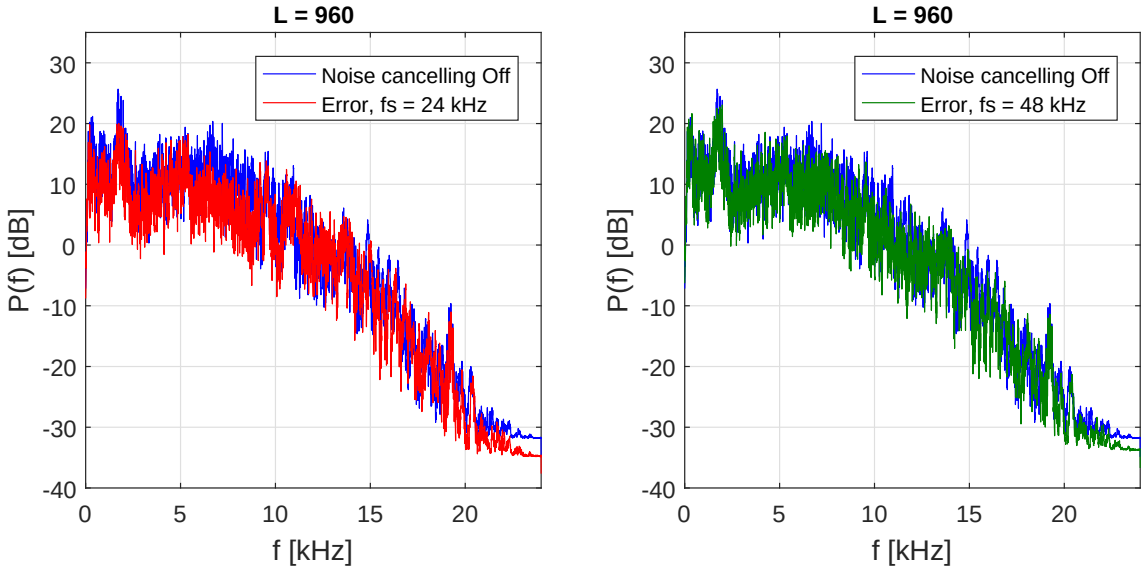
M.1.2. ábra. *Hibalecsengés dB-ben,* $L = 960$



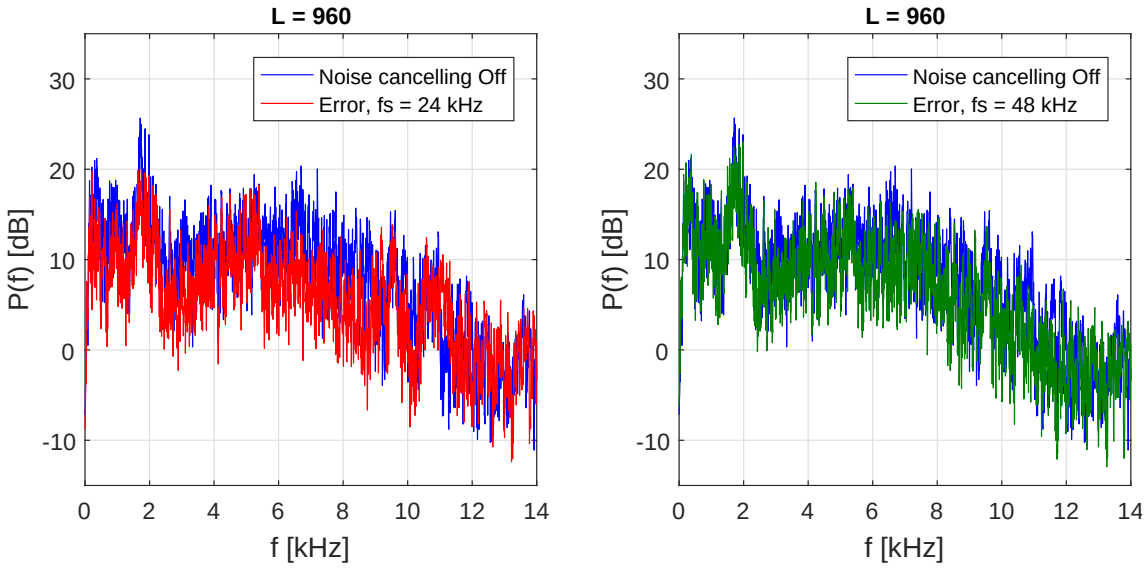
M.1.3. ábra. *Hibacsökkenés mértéke,* $L = 960$



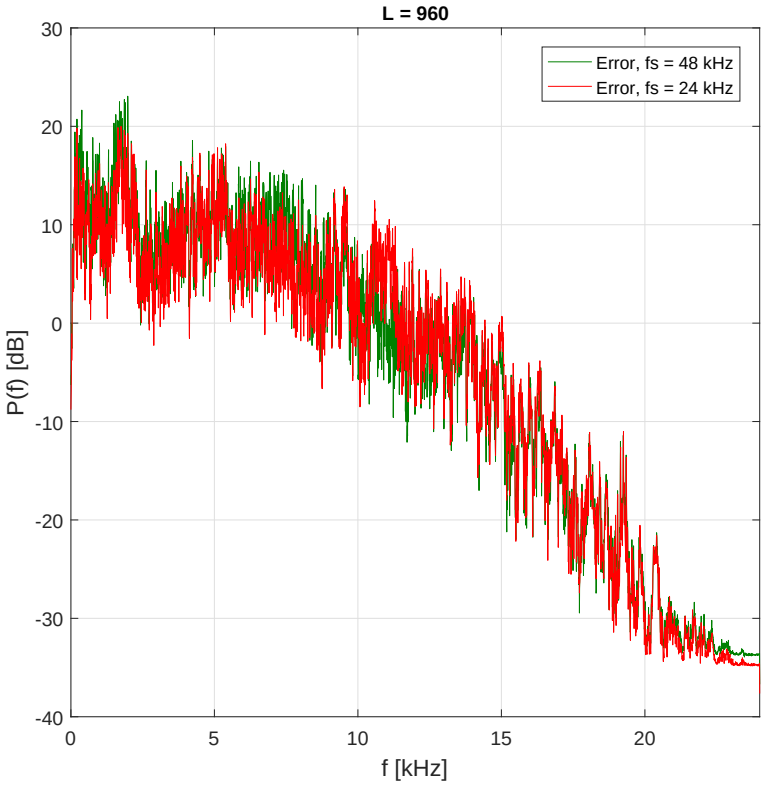
M.1.4. ábra. *Hibacsökkenés mértéke dB-ben,* $L = 960$



M.1.5. ábra. *Hibajel teljesítményspektruma,* $L = 960$

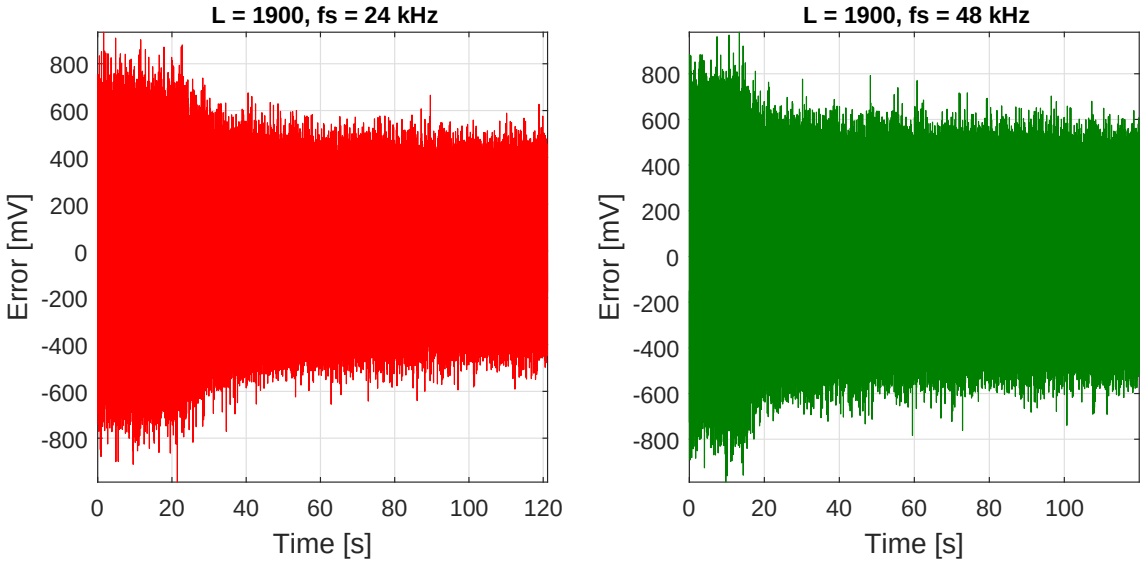


M.1.6. ábra. *Hibajel teljesítményspektruma (0-14) kHz tartományban,* $L = 960$

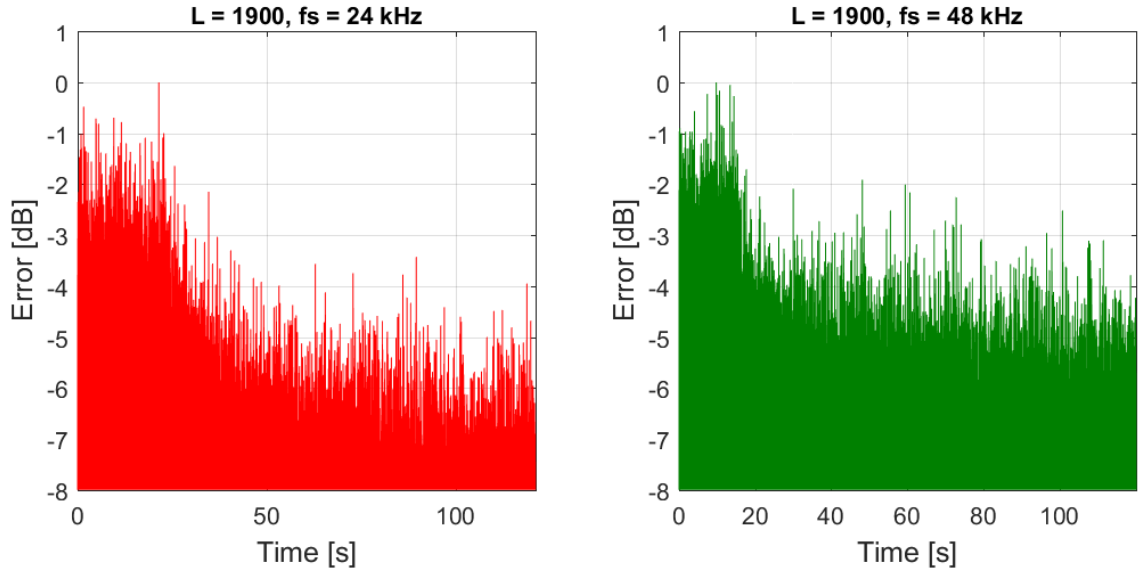


M.1.7. ábra. *24 kHz és 48 kHz mintavételi frekvenciák mellett működő aktív zajcsökkentés összehasonlítása,* $L = 960$

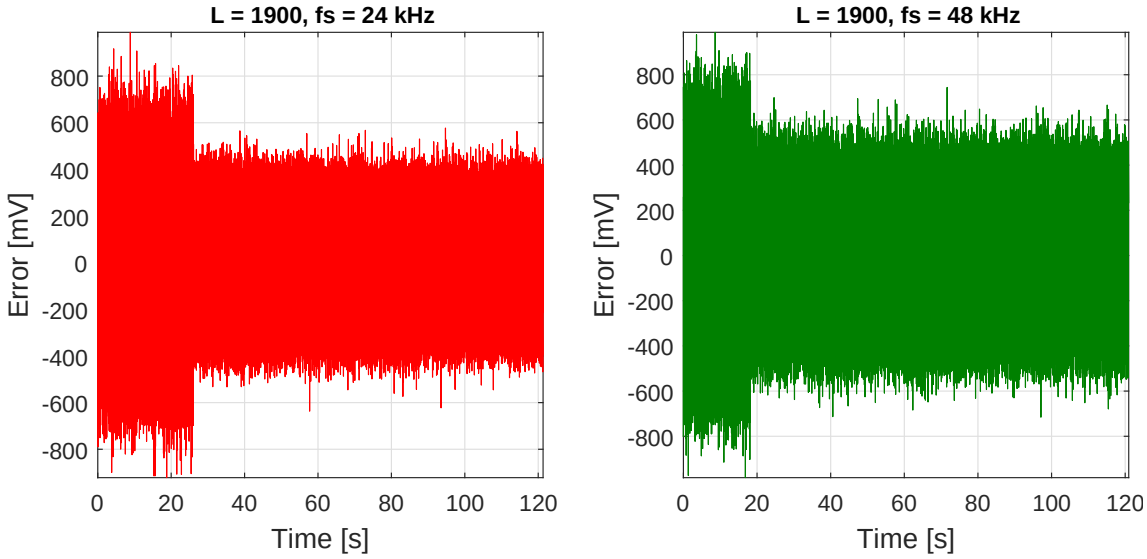
M.1.2. Szabadon lévő hibamikrofon, $L = 1900$, $\mu = 0.004$, $a = 0.1$, $M = 900$



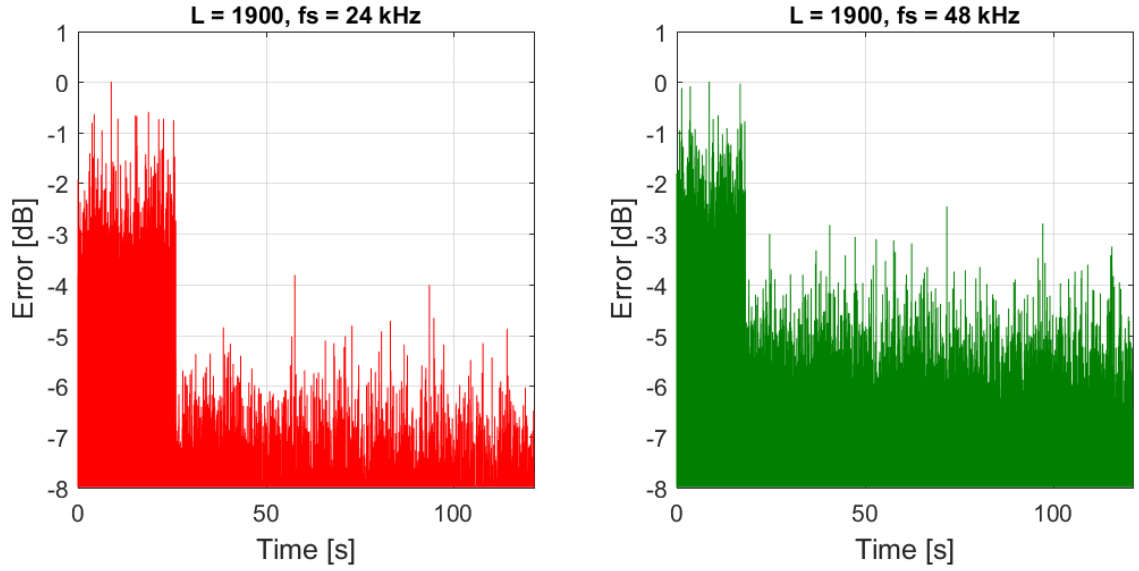
M.1.8. ábra. *Hibalecsengés,* $L = 1900$



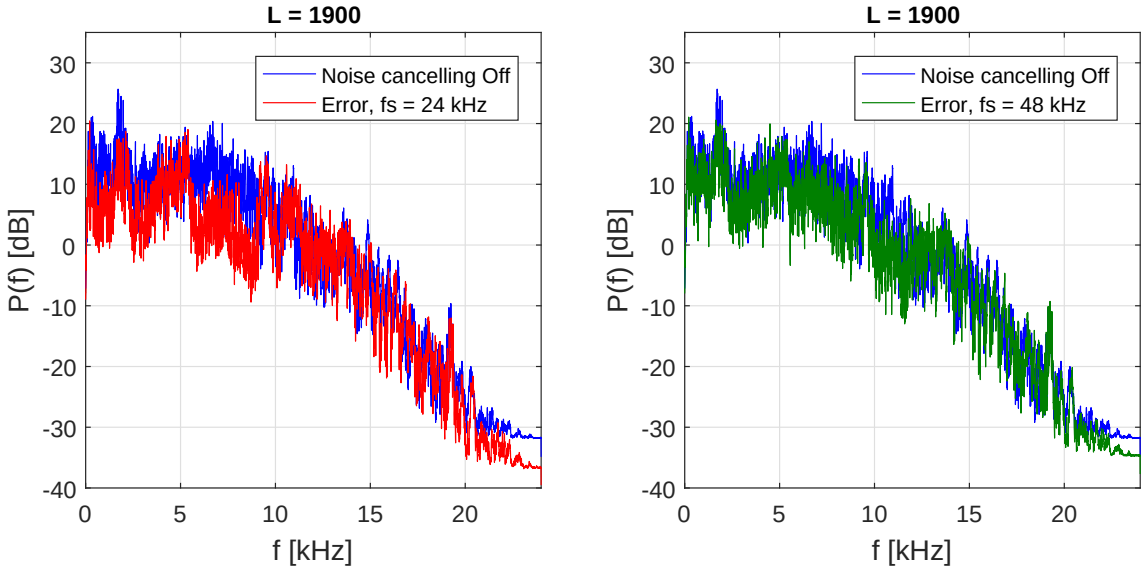
M.1.9. ábra. *Hibalecsengés dB-ben,* $L = 1900$



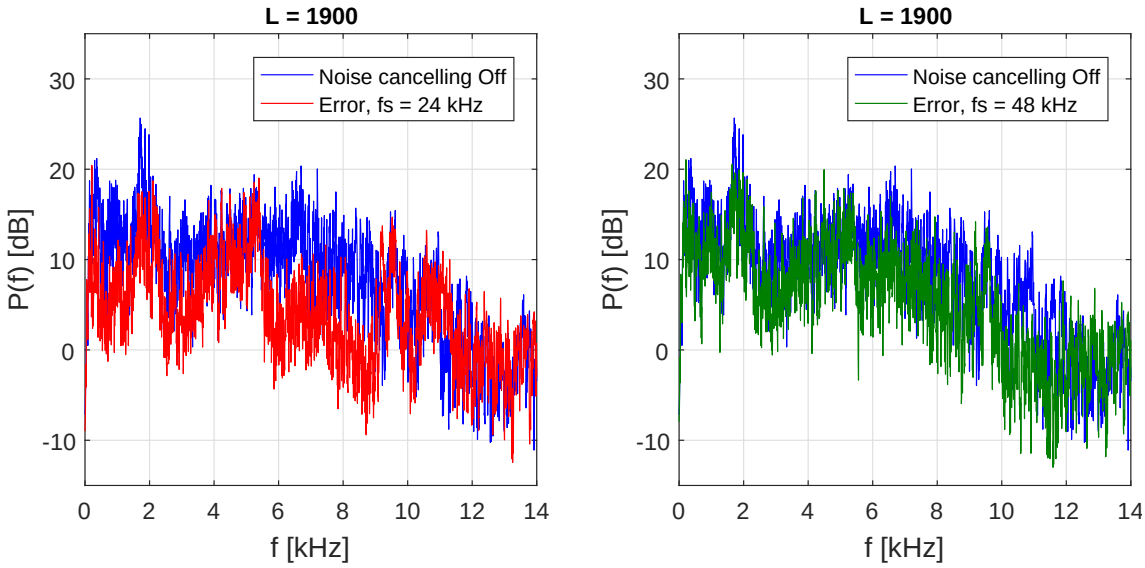
M.1.10. ábra. *Hibacsökkenés mértéke,* $L = 1900$



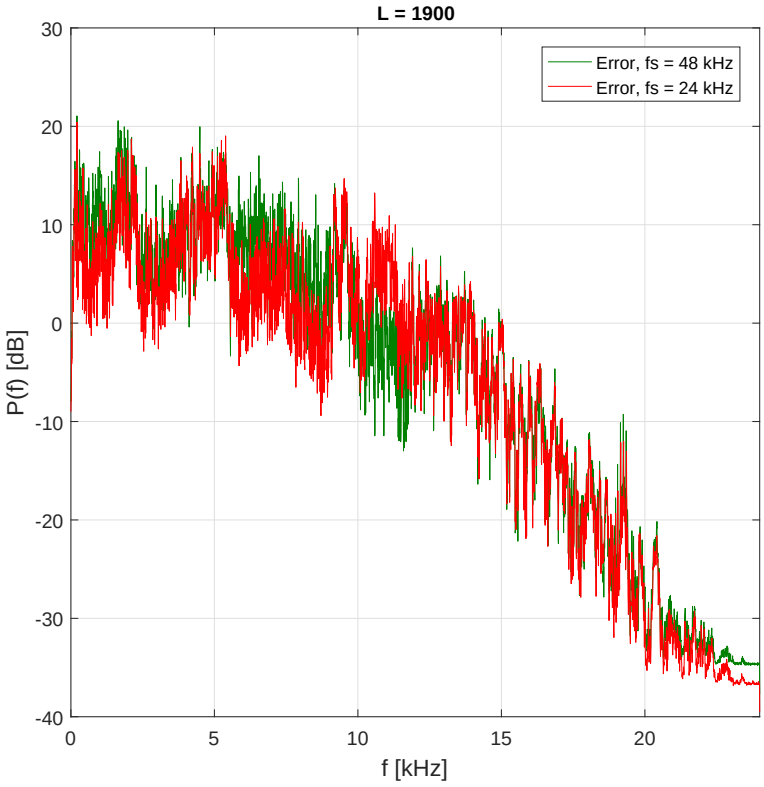
M.1.11. ábra. *Hibacsökkenés mértéke dB-ben,* $L = 1900$



M.1.12. ábra. *Hibajel teljesítményspektruma,* $L = 1900$

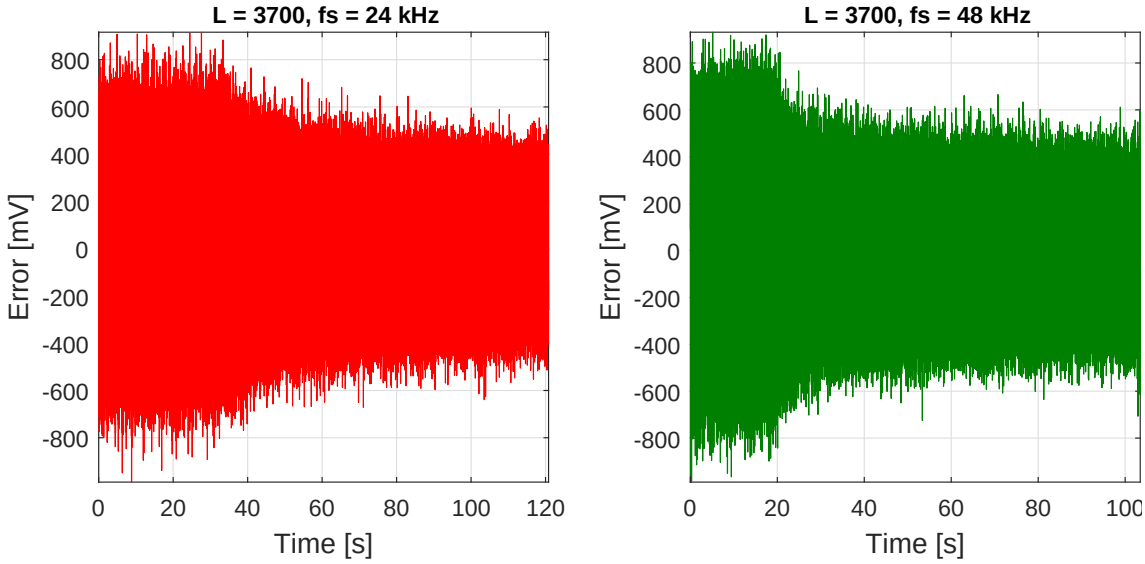


M.1.13. ábra. *Hibajel teljesítményspektruma (0-14) kHz tartományban,* $L = 1900$

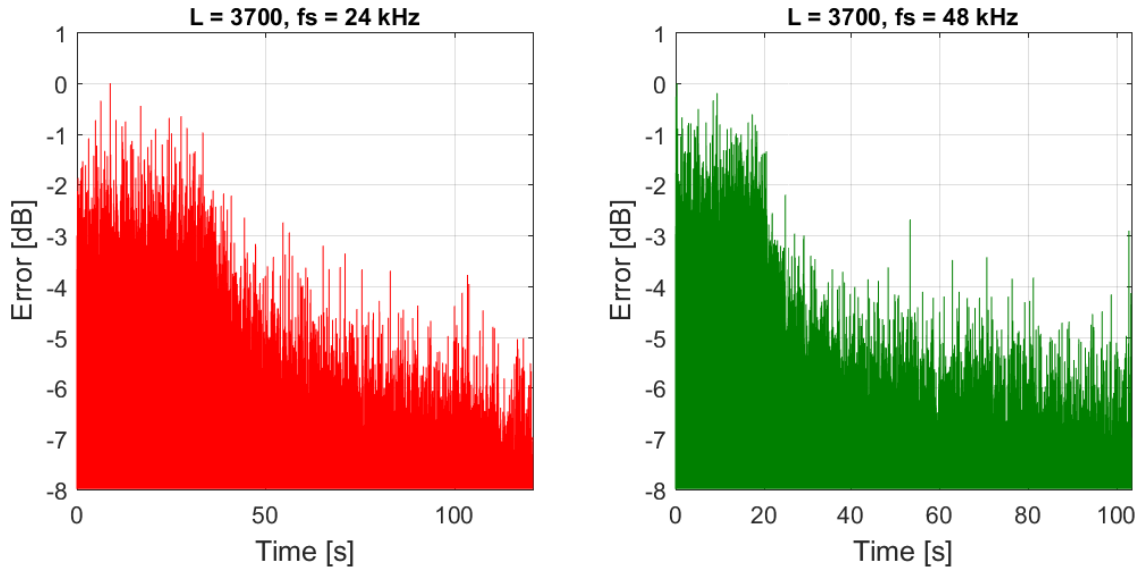


M.1.14. ábra. *24 kHz és 48 kHz mintavételi frekvenciák mellett működő aktív zajcsökkentés összehasonlítása,* $L = 1900$

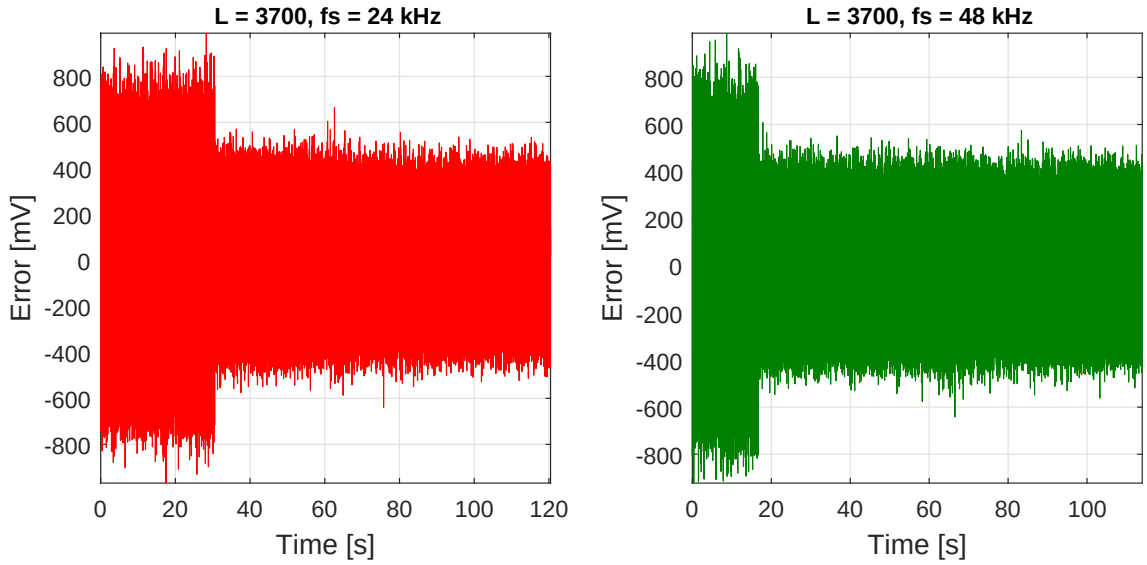
M.1.3. Szabadon lévő hibamikrofon, $L = 3700$, $\mu = 0.004$, $a = 0.1$, $M = 900$



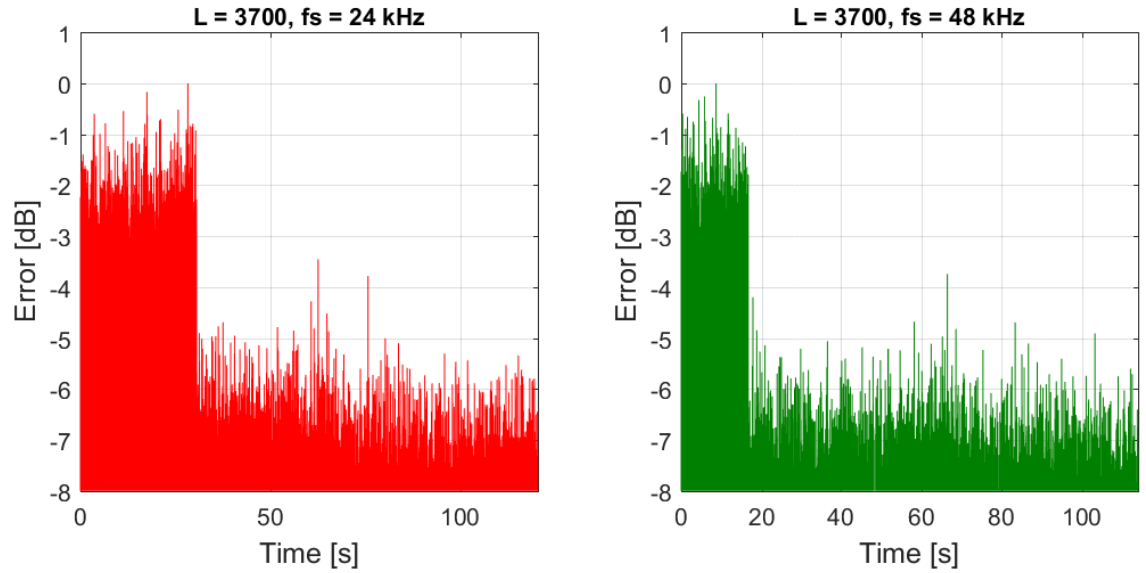
M.1.15. ábra. *Hibalecsengés, $L = 3700$*



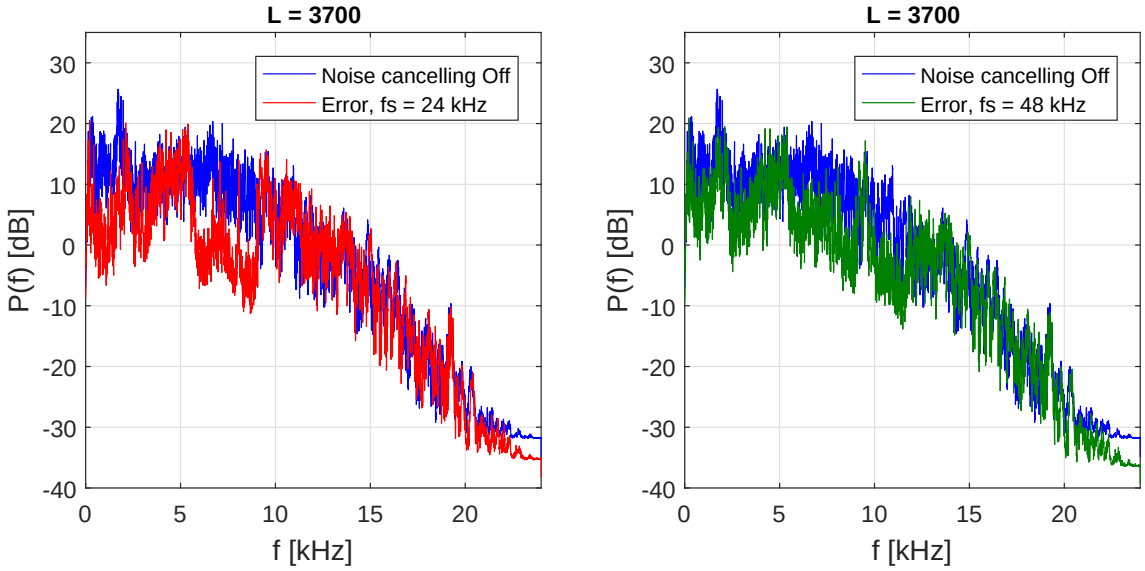
M.1.16. ábra. *Hibalecsengés dB-ben, $L = 3700$*



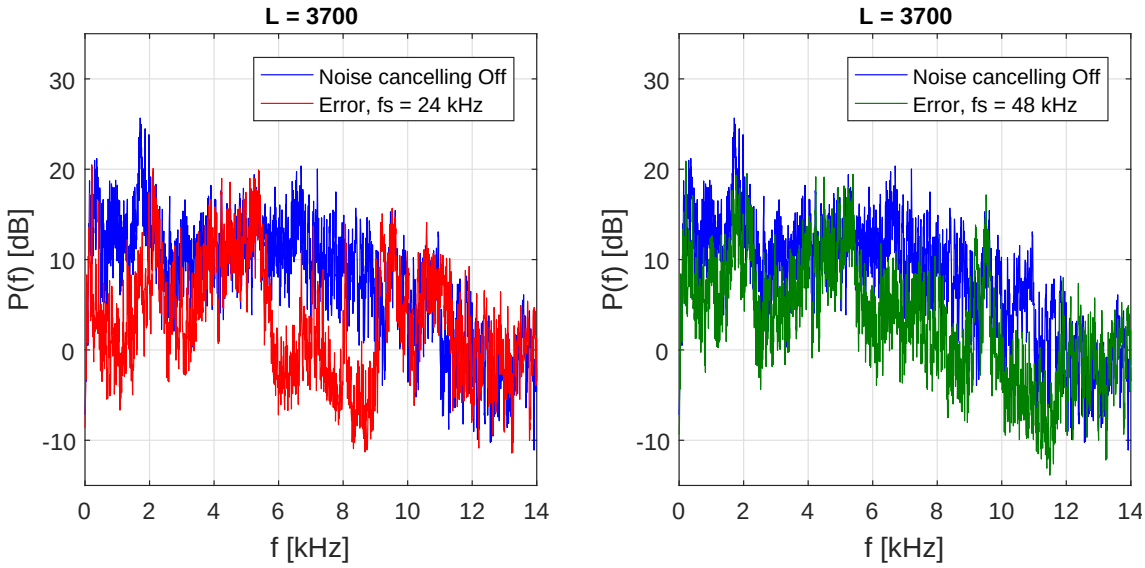
M.1.17. ábra. *Hibacsökkenés mértéke,* $L = 3700$



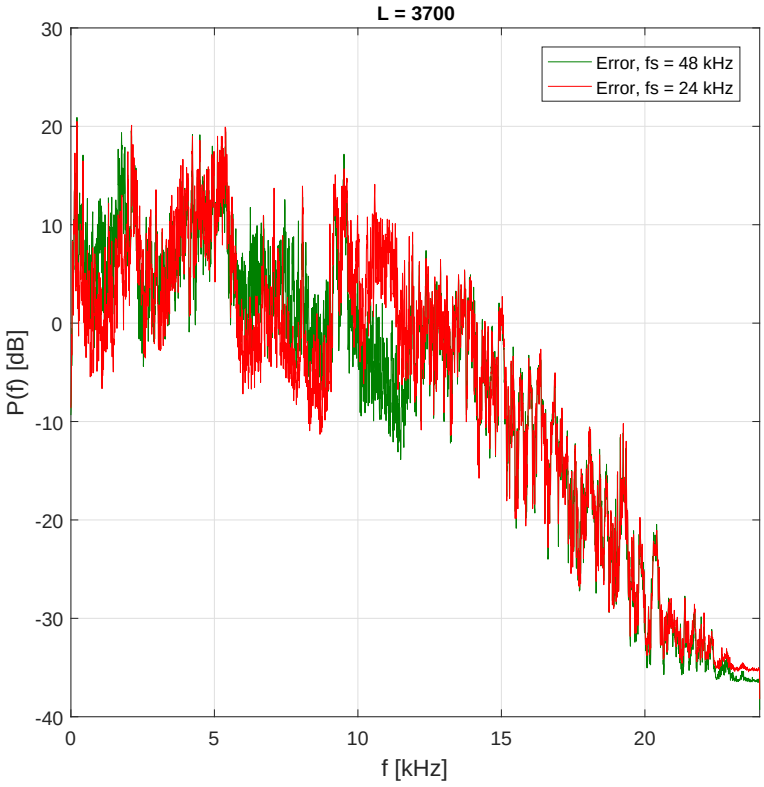
M.1.18. ábra. *Hibacsökkenés mértéke dB-ben,* $L = 3700$



M.1.19. ábra. *Hibajel teljesítményspektruma,* $L = 3700$

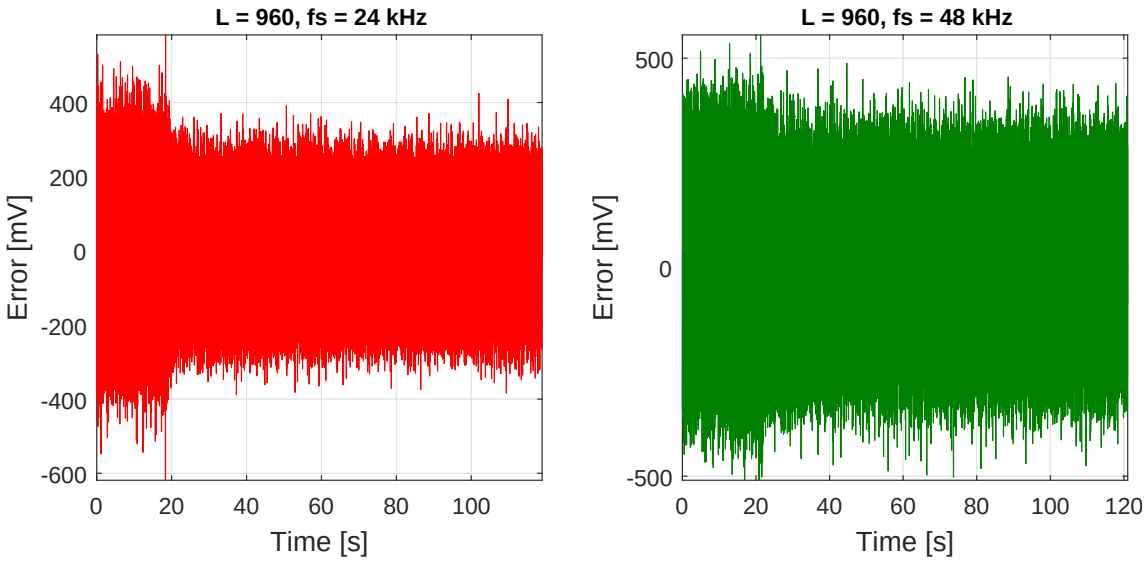


M.1.20. ábra. *Hibajel teljesítményspektruma (0-14) kHz tartományban,* $L = 3700$

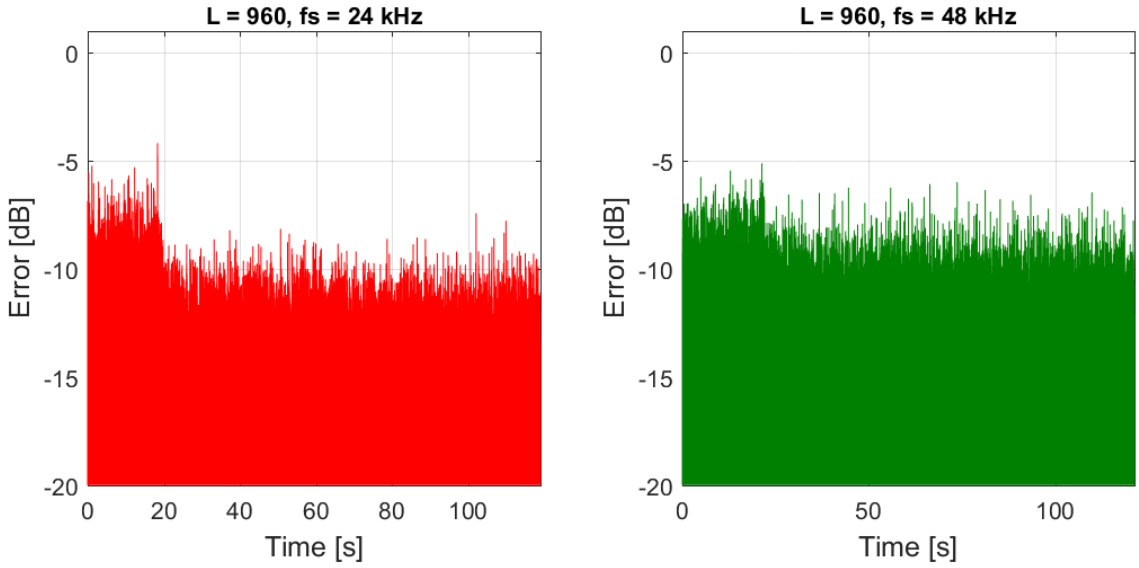


M.1.21. ábra. *24 kHz és 48 kHz mintavételi frekvenciák mellett működő aktív zajcsökkentés összehasonlítása,* $L = 3700$

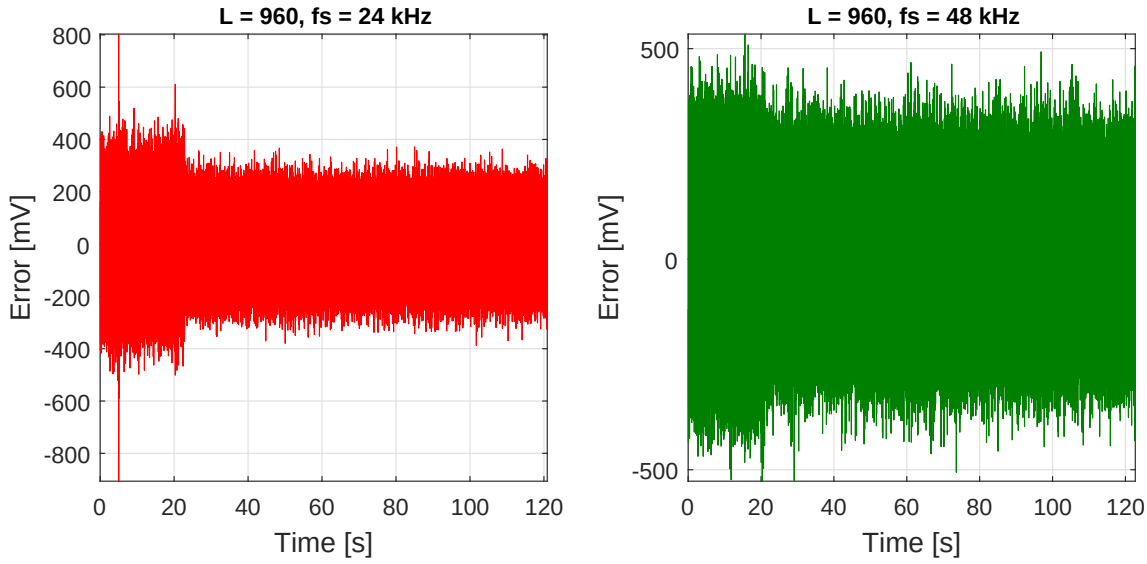
M.1.4. Aktív zajcsökkentő fejhallgató, $L = 960$, $\mu = 0.004$, $a = 0.1$, $M = 900$



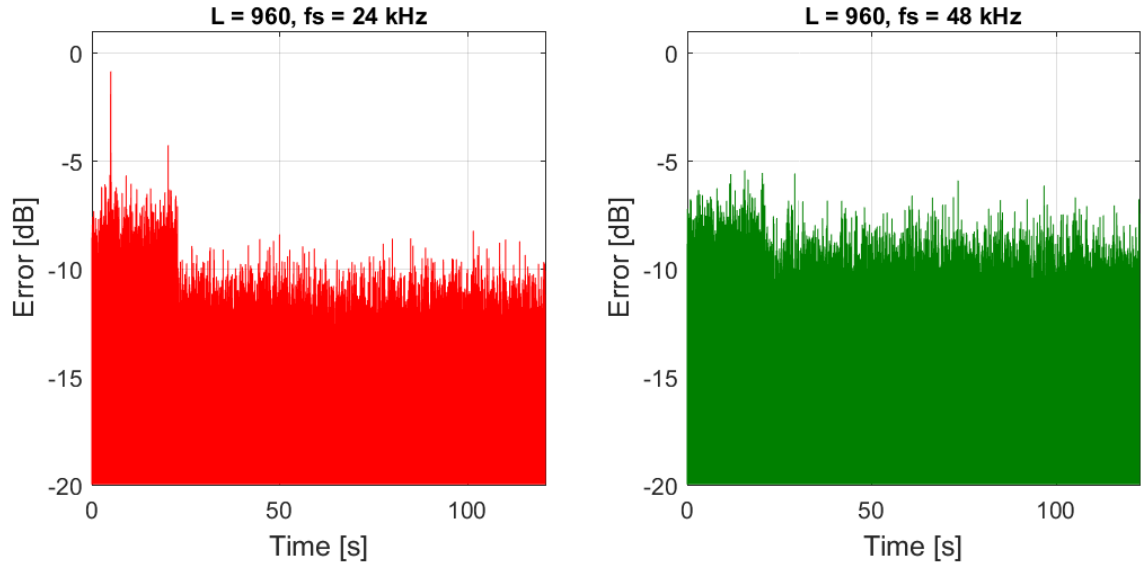
M.1.22. ábra. *Hibalecsengés,* $L = 960$



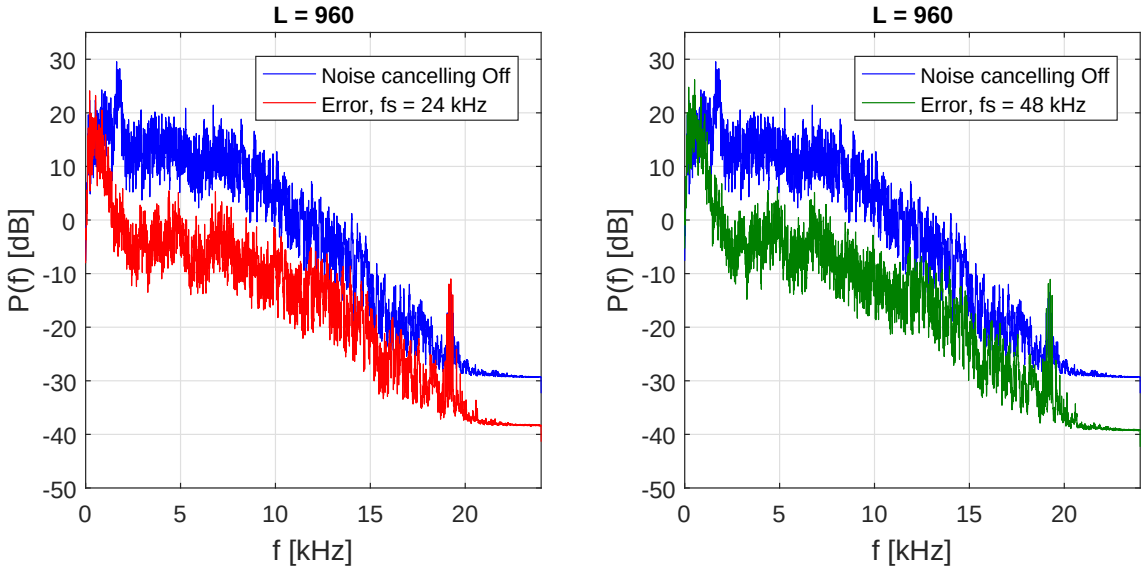
M.1.23. ábra. *Hibalecsengés dB-ben,* $L = 960$



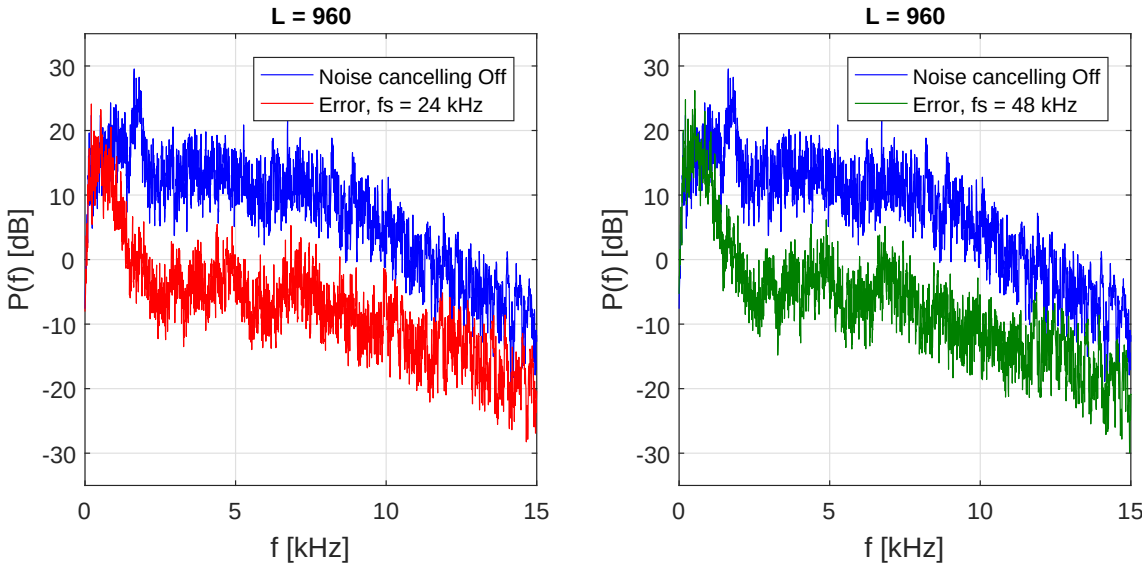
M.1.24. ábra. *Hibacsökkenés mértéke,* $L = 960$



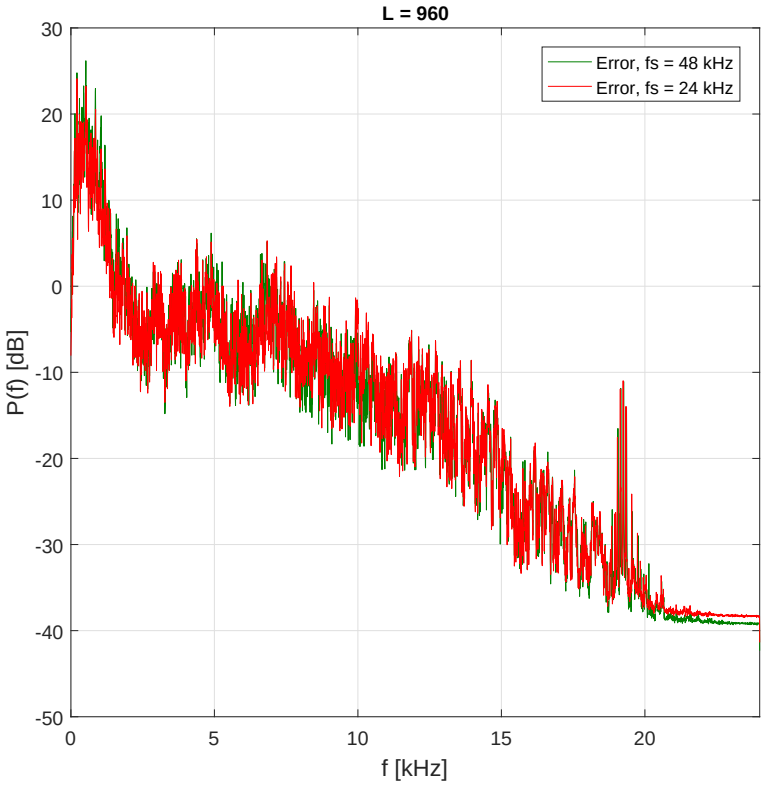
M.1.25. ábra. *Hibacsökkenés mértéke dB-ben,* $L = 960$



M.1.26. ábra. *Hibajel teljesítményspektruma,* $L = 960$

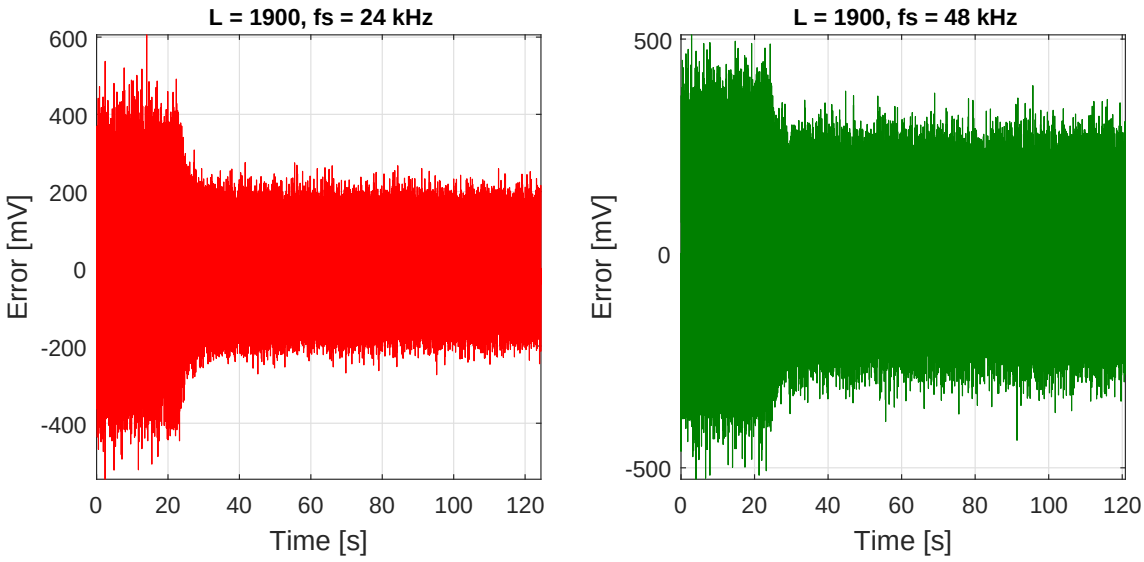


M.1.27. ábra. *Hibajel teljesítményspektruma (0-14) kHz tartományban,* $L = 960$

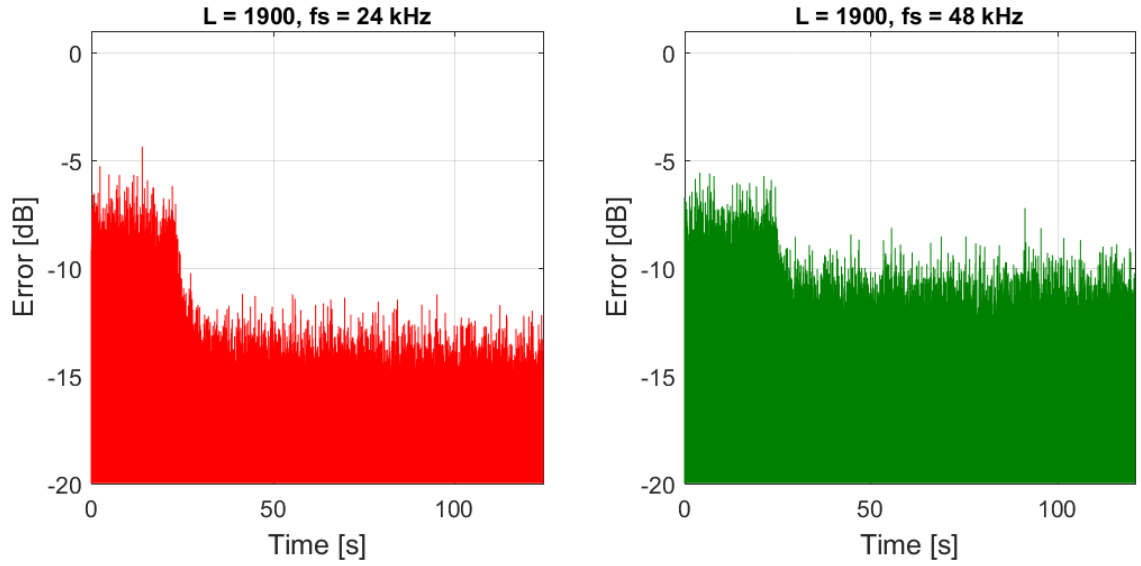


M.1.28. ábra. *24 kHz és 48 kHz mintavételi frekvenciák mellett működő aktív zajcsökkentés összehasonlítása,* $L = 960$

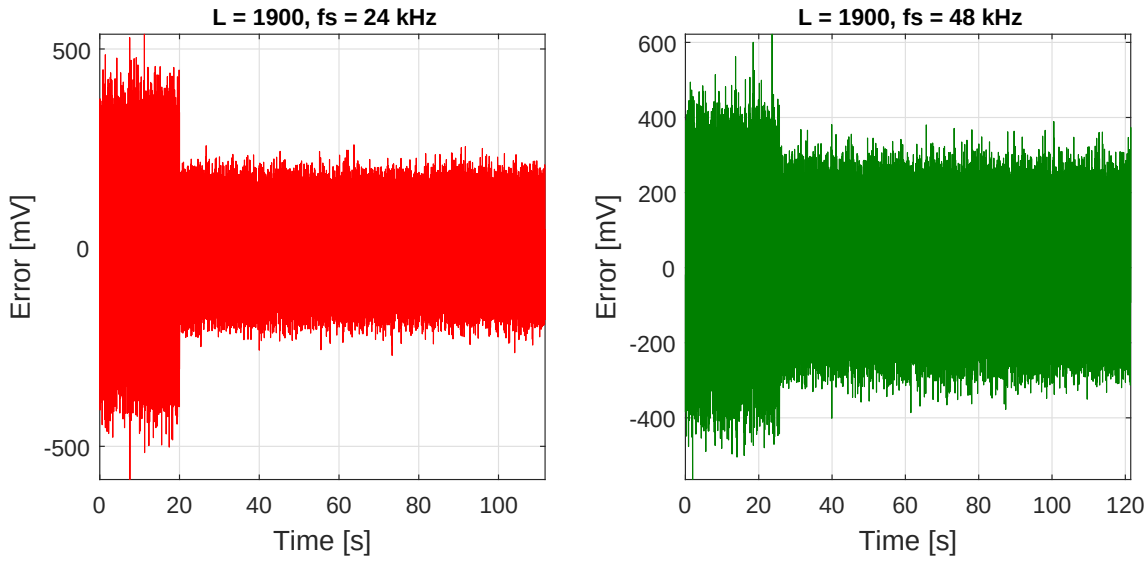
M.1.5. Aktív zajcsökkentő fejhallgató, $L = 1900$, $\mu = 0.004$, $a = 0.1$, $M = 900$



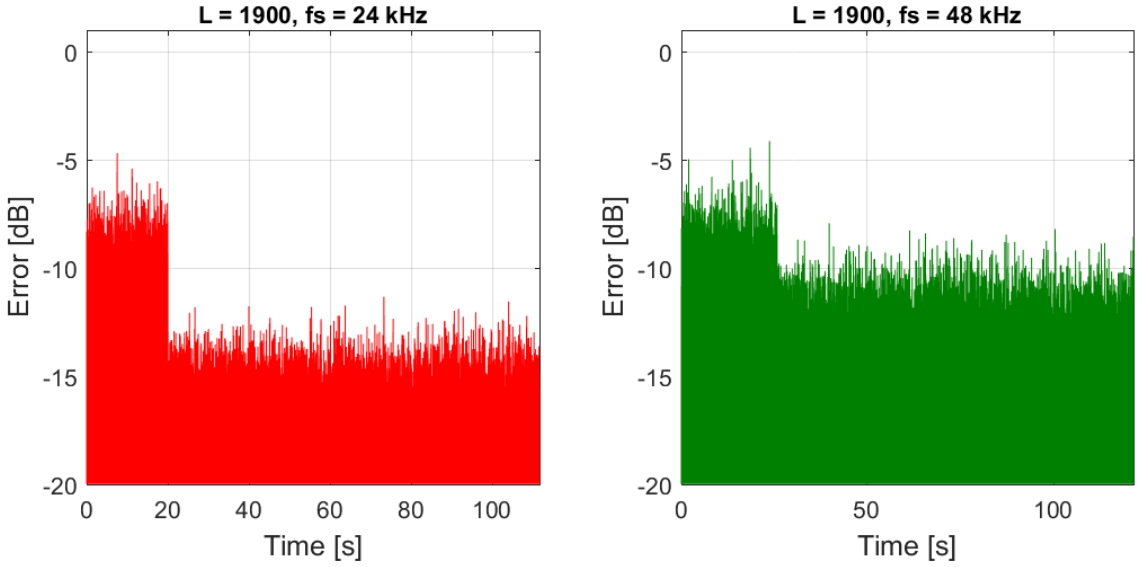
M.1.29. ábra. *Hibalecsengés,* $L = 1900$



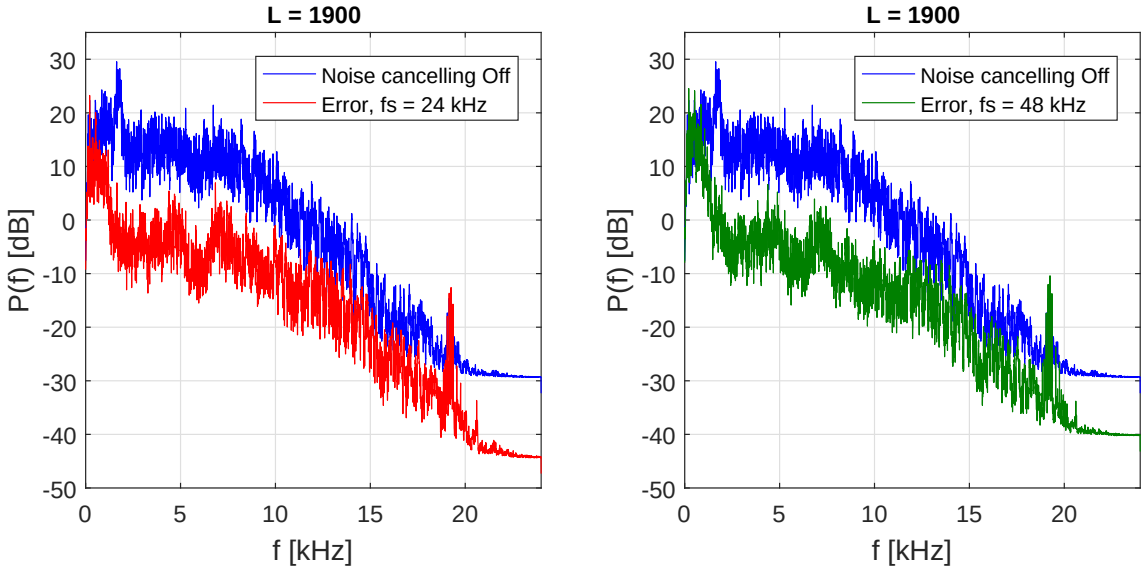
M.1.30. ábra. *Hibalecsengés dB-ben,* $L = 1900$



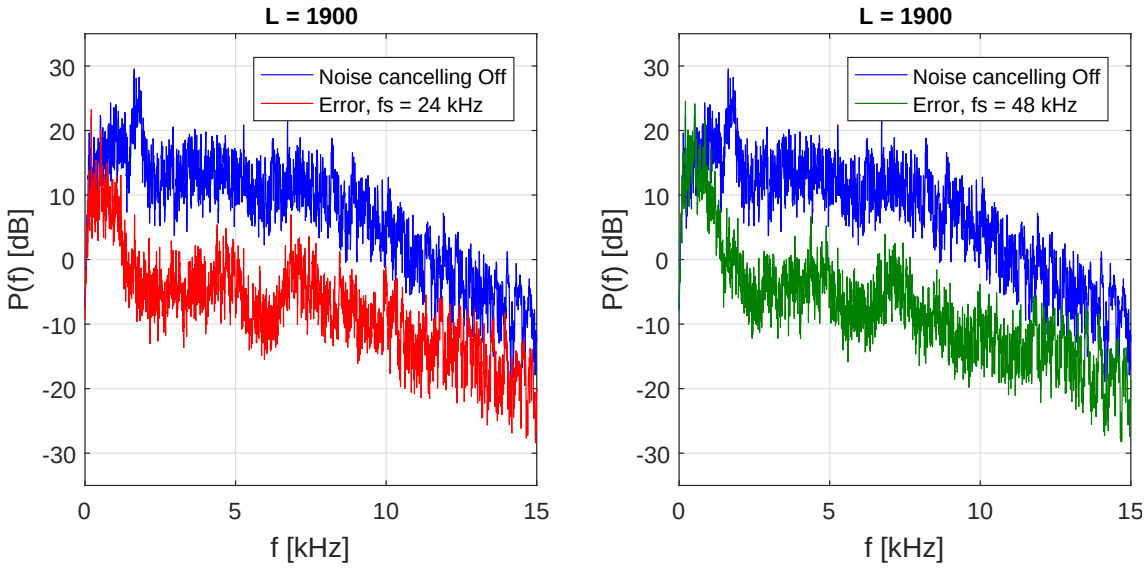
M.1.31. ábra. *Hibacsökkenés mértéke,* $L = 1900$



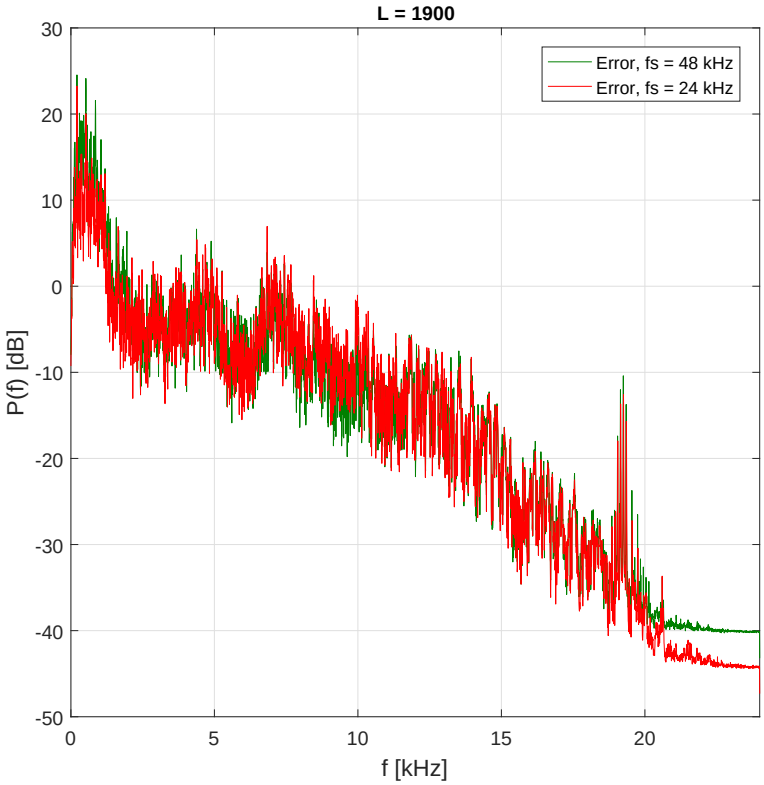
M.1.32. ábra. *Hibacsökkenés mértéke dB-ben,* $L = 1900$



M.1.33. ábra. *Hibajel teljesítményspektruma,* $L = 1900$

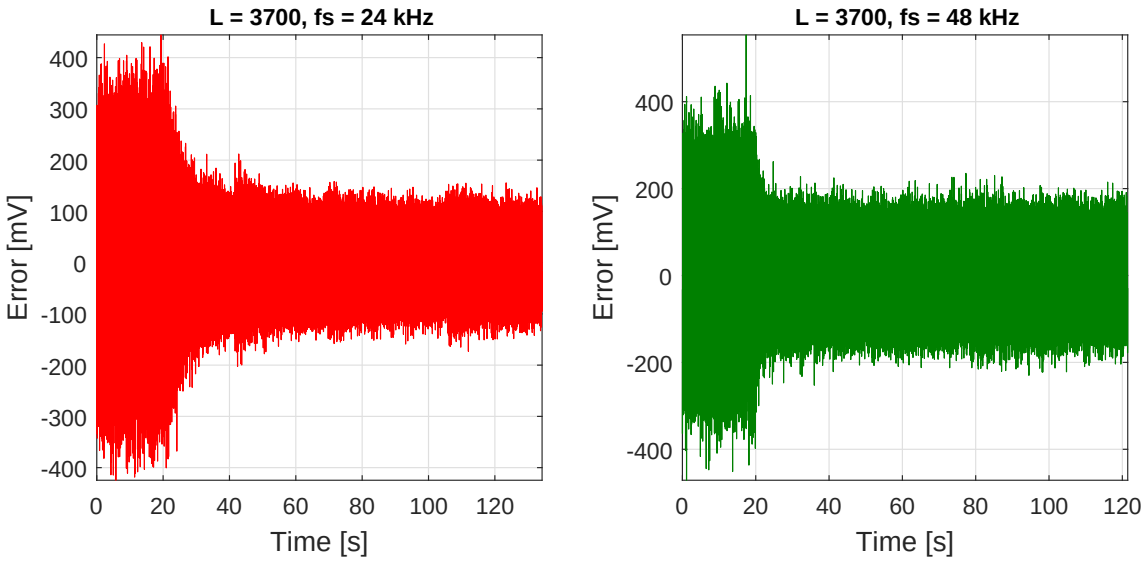


M.1.34. ábra. *Hibajel teljesítményspektruma (0-14) kHz tartományban,* $L = 1900$

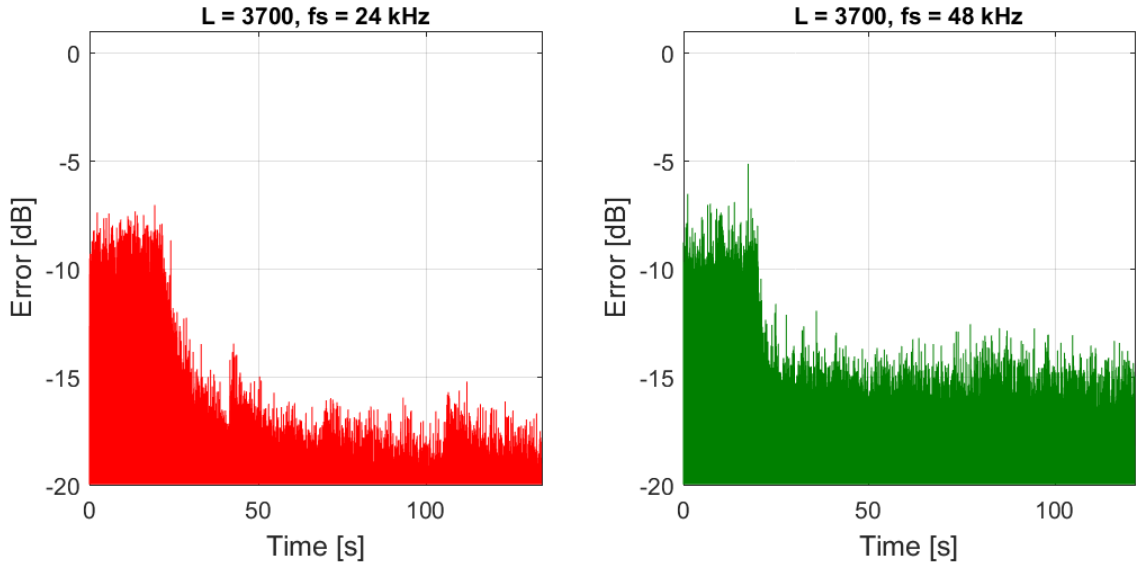


M.1.35. ábra. *24 kHz és 48 kHz mintavételi frekvenciák mellett működő aktív zajcsökkentés összehasonlítása,* $L = 1900$

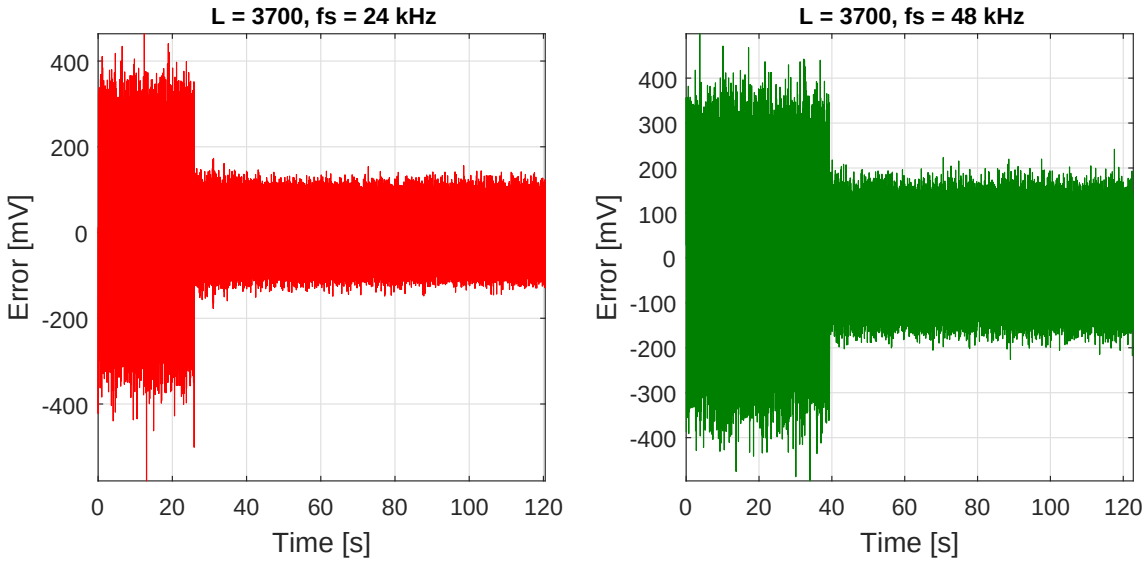
M.1.6. Aktív zajcsökkentő fejhallgató, $L = 3700$, $\mu = 0.004$, $a = 0.1$, $M = 900$



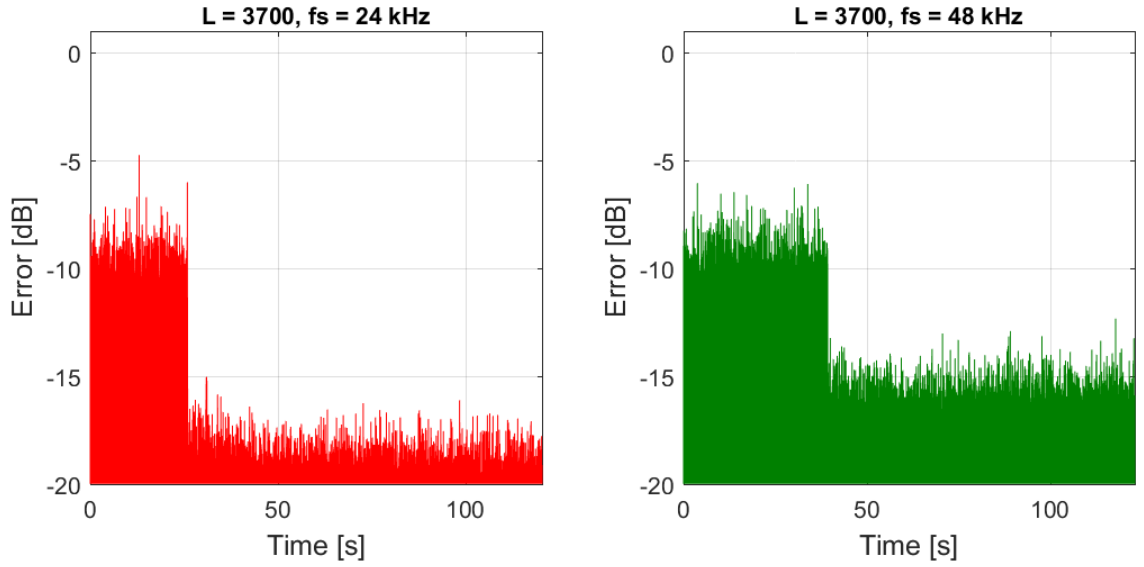
M.1.36. ábra. *Hibalecsengés,* $L = 3700$



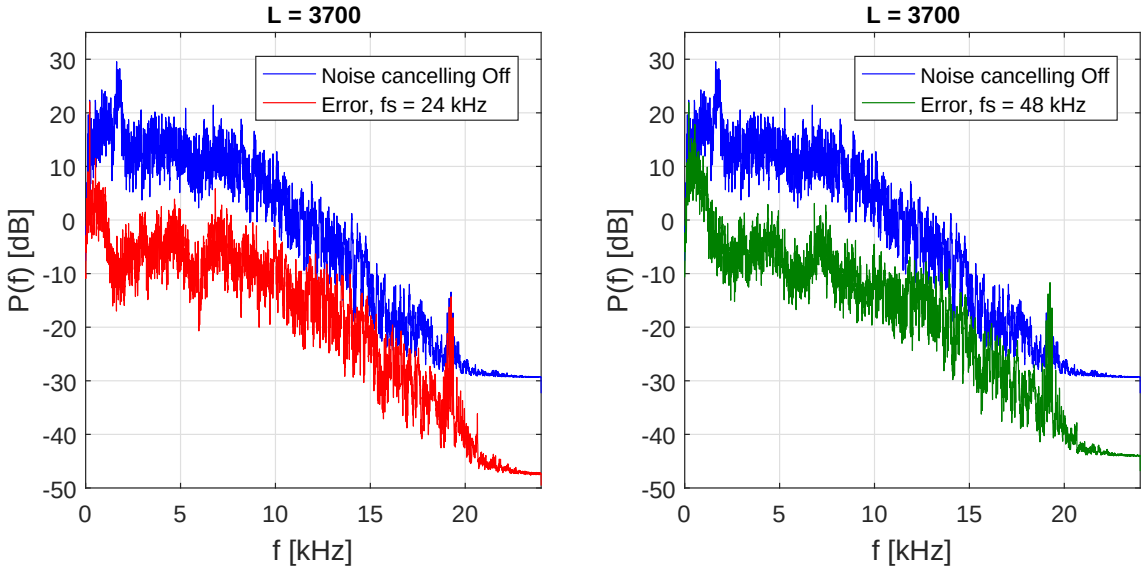
M.1.37. ábra. *Hibalecsengés dB-ben,* $L = 3700$



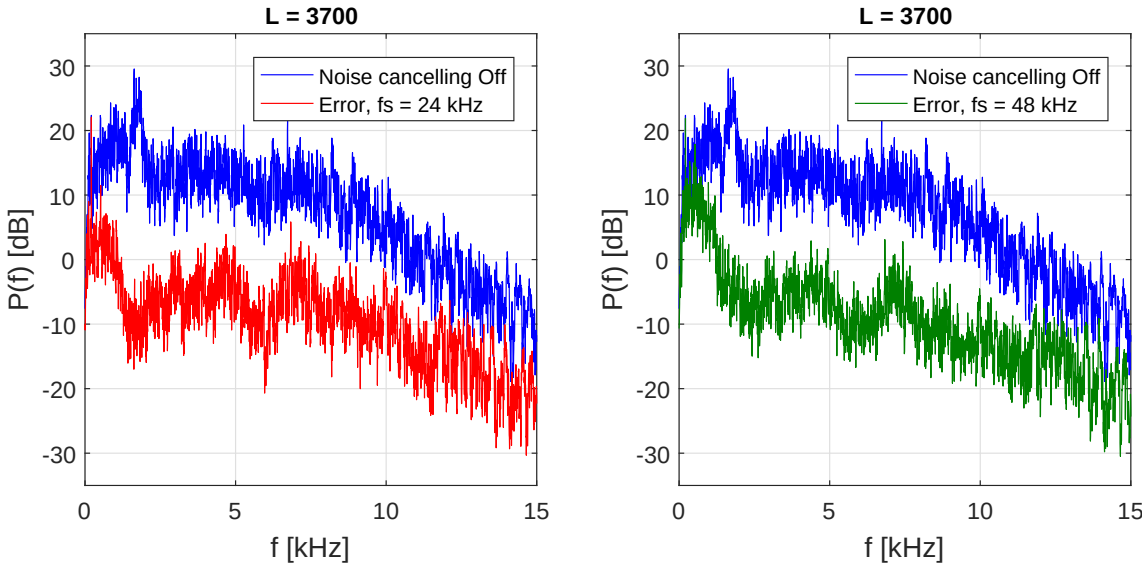
M.1.38. ábra. *Hibacsökkenés mértéke,* $L = 3700$



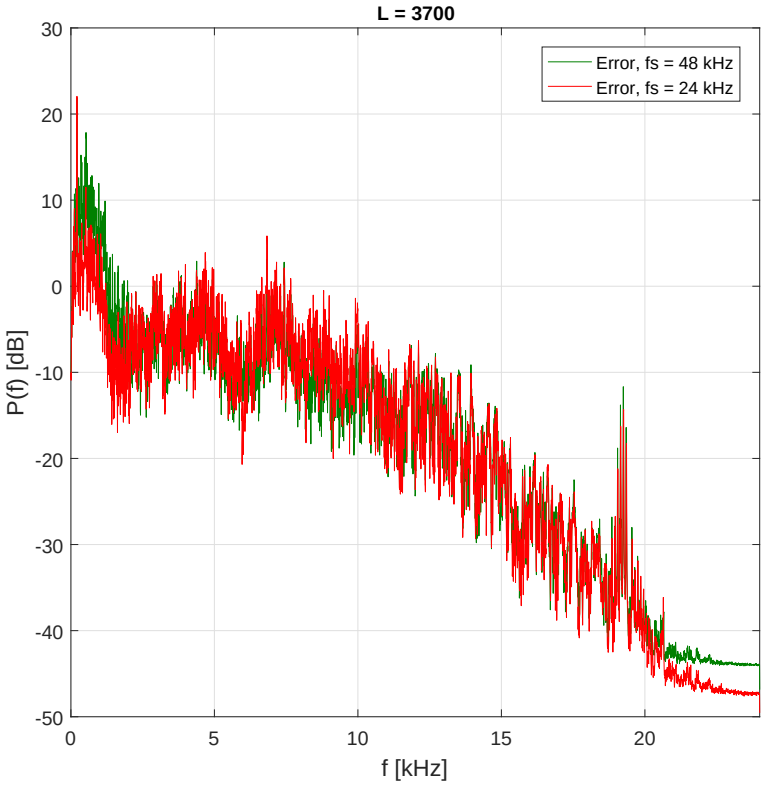
M.1.39. ábra. *Hibacsökkenés mértéke dB-ben,* $L = 3700$



M.1.40. ábra. *Hibajel teljesítményspektruma,* $L = 3700$



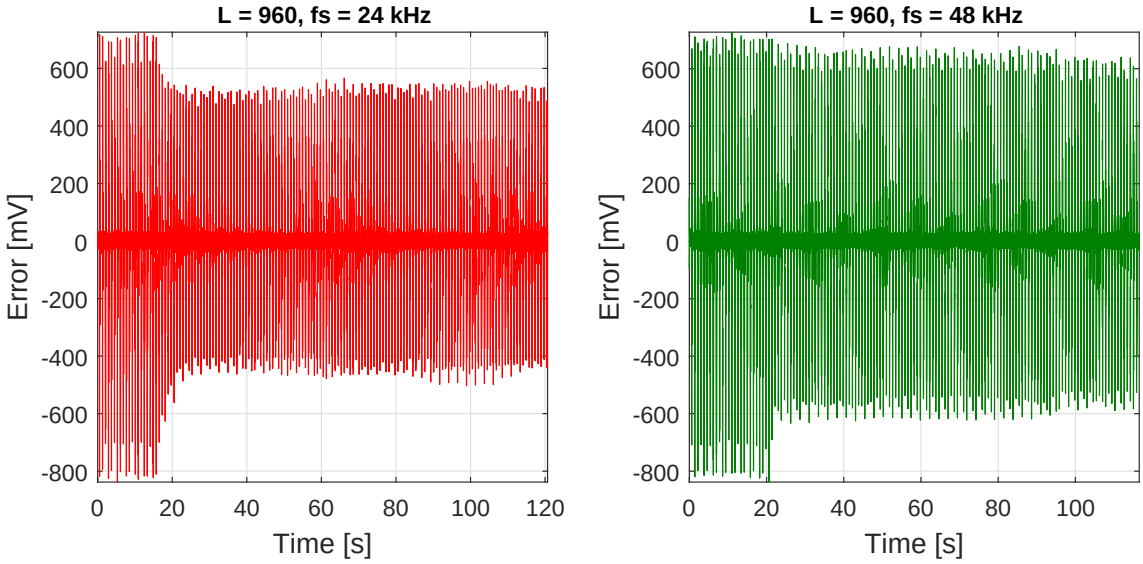
M.1.41. ábra. *Hibajel teljesítményspektruma (0-14) kHz tartományban,* $L = 3700$



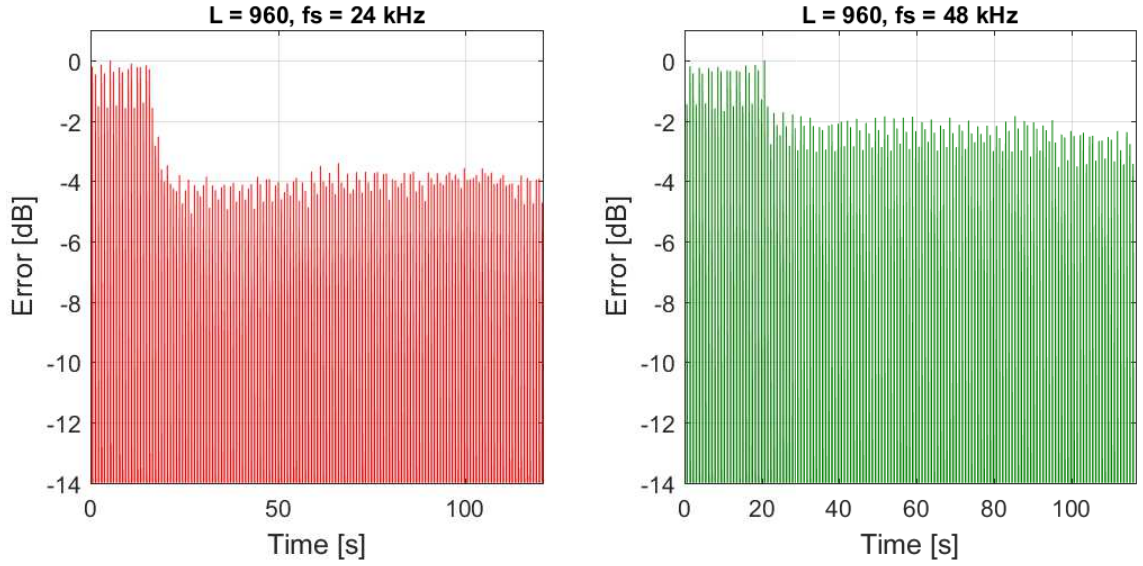
M.1.42. ábra. *24 kHz és 48 kHz mintavételi frekvenciák mellett működő aktív zajcsökkentés összehasonlítása,* $L = 3700$

M.2. Csattogó zajok aktív zajcsökkentésének eredménye

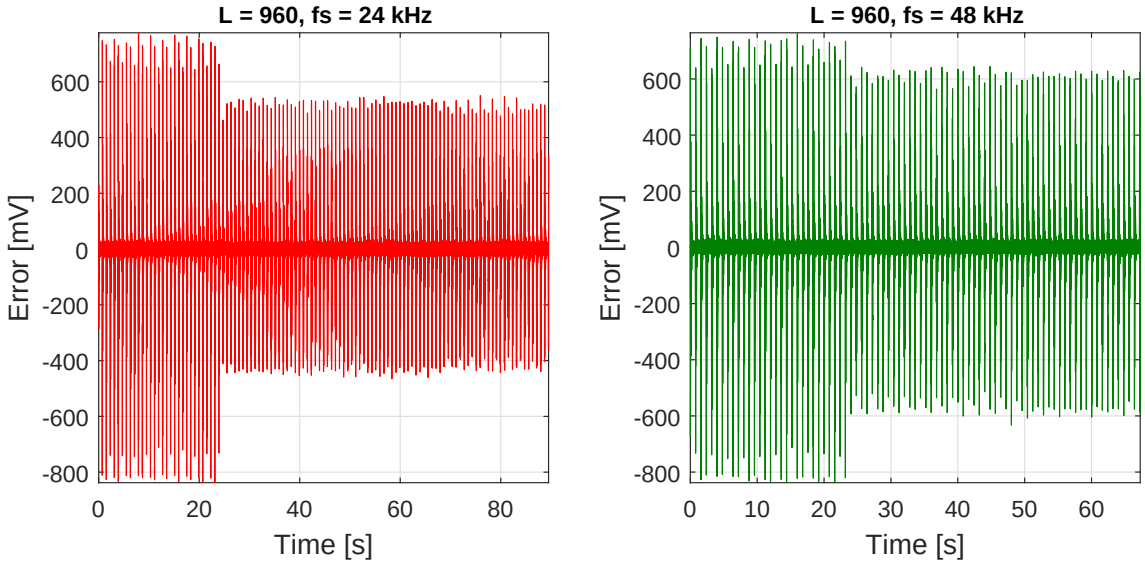
M.2.1. Szabadon lévő hibamikrofon, $L = 960$, $\mu = 0.03$, $a = 100.0$, $M = 900$



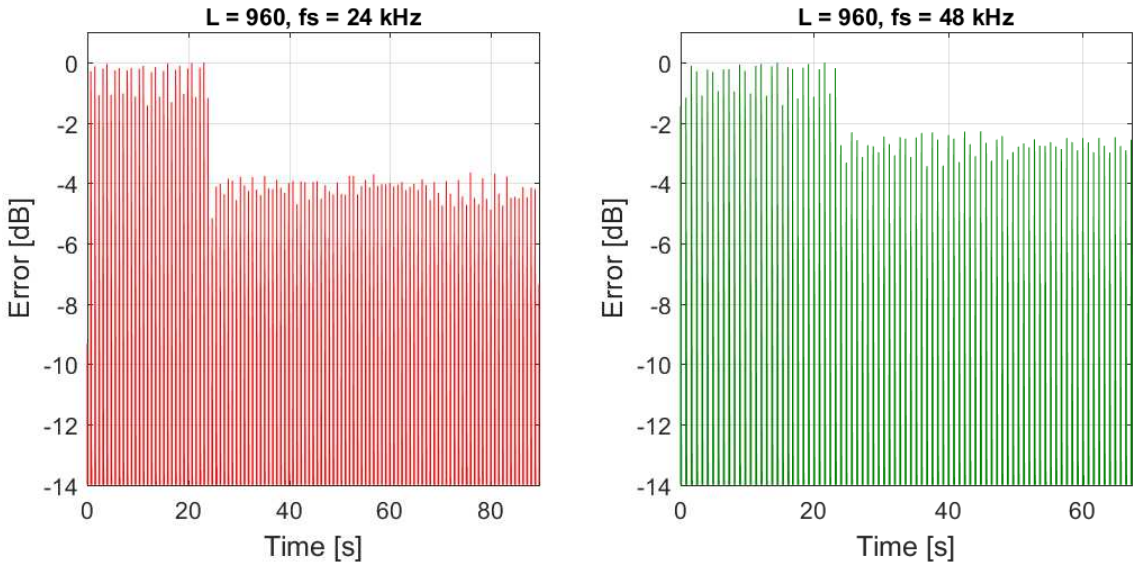
M.2.1. ábra. *Hibalecsengés,* $L = 960$



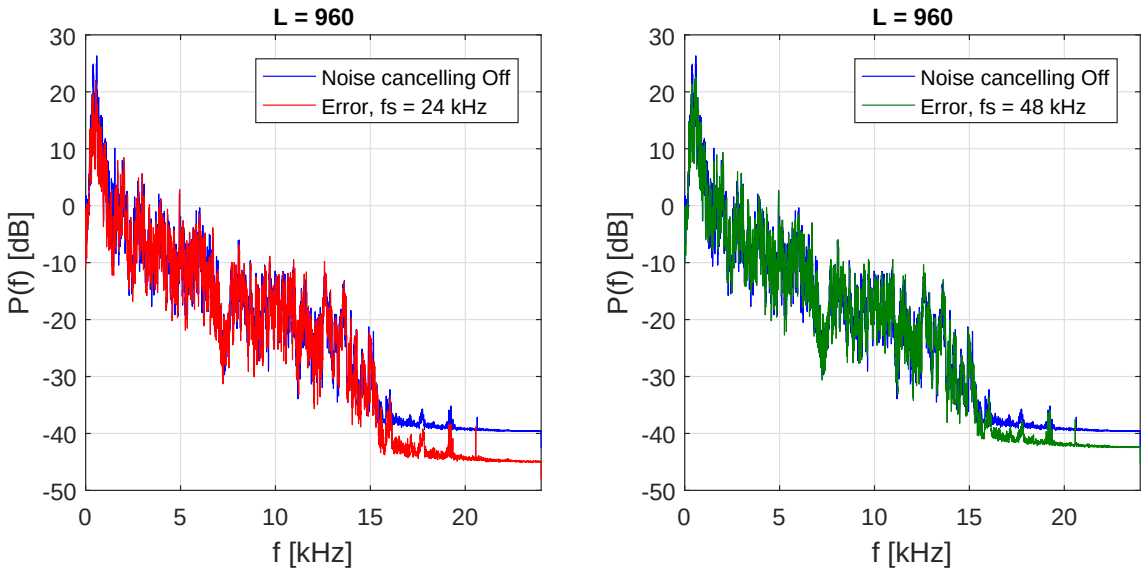
M.2.2. ábra. *Hibalecsengés dB-ben,* $L = 960$



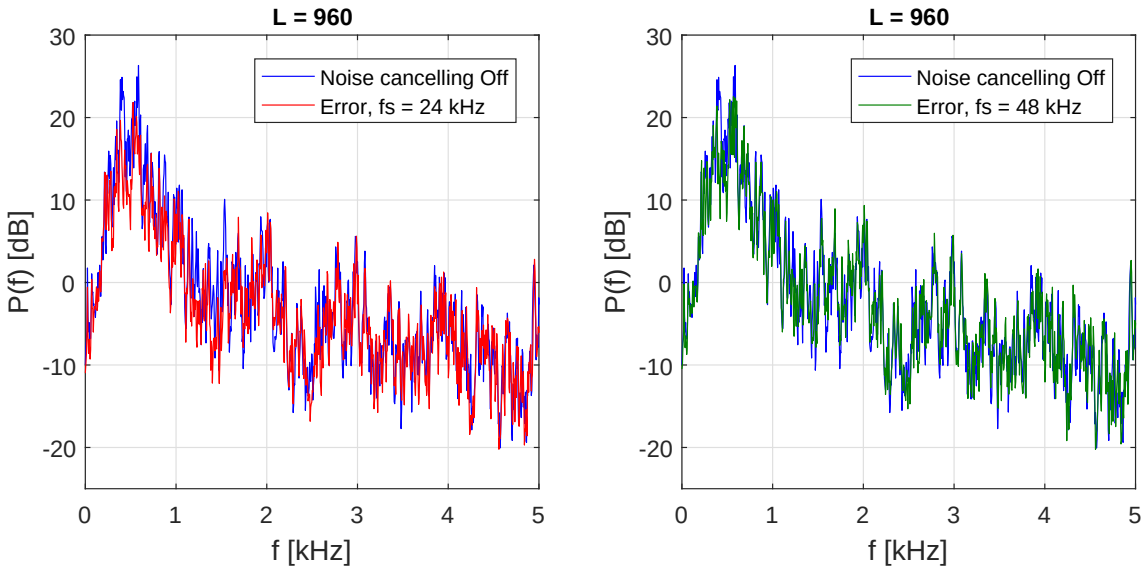
M.2.3. ábra. *Hibacsökkenés mértéke,* $L = 960$



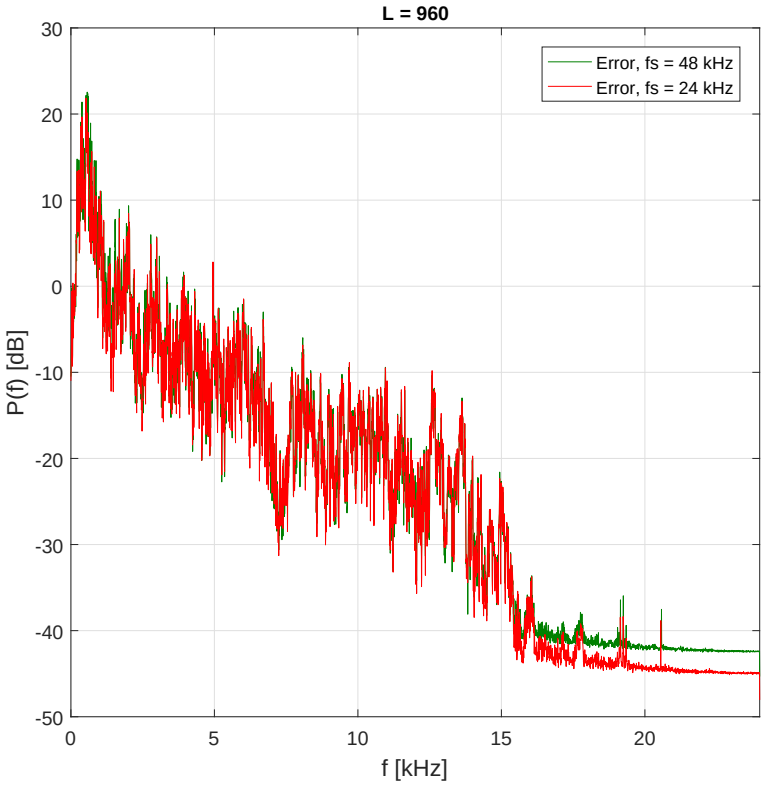
M.2.4. ábra. *Hibacsökkenés mértéke dB-ben,* $L = 960$



M.2.5. ábra. *Hibajel teljesítményspektruma,* $L = 960$

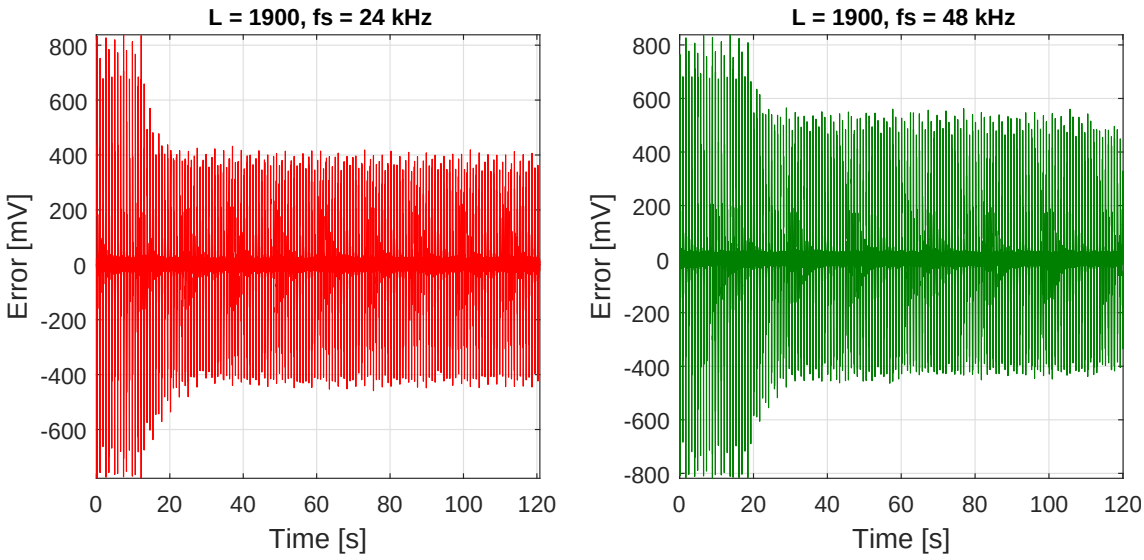


M.2.6. ábra. *Hibajel teljesítményspektruma (0-5) kHz tartományban,* $L = 960$

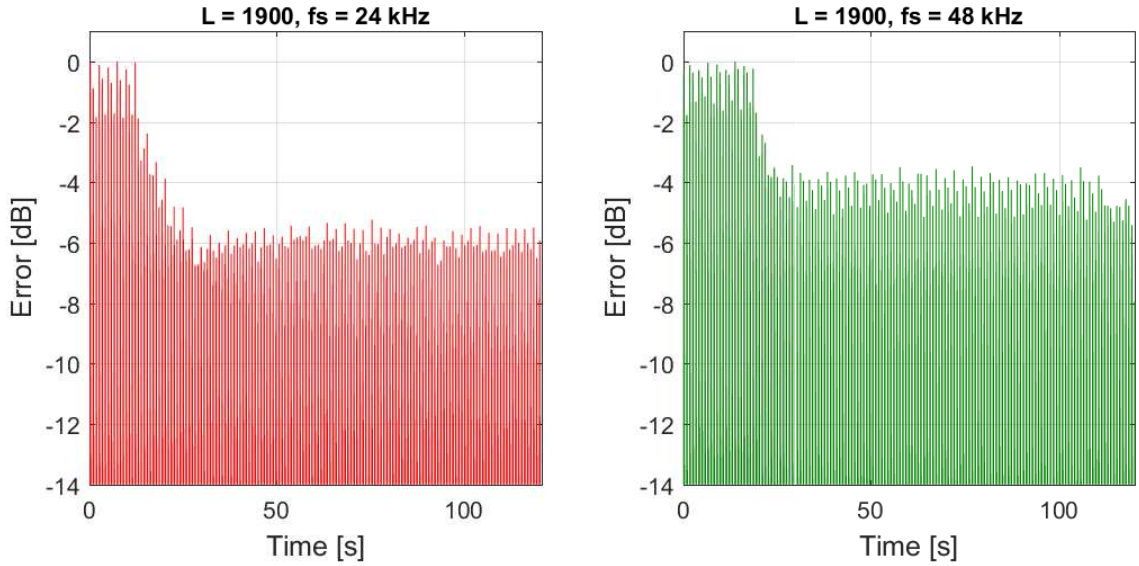


M.2.7. ábra. *24 kHz és 48 kHz mintavételi frekvenciák mellett működő aktív zajcsökkentés összehasonlítása,* $L = 960$

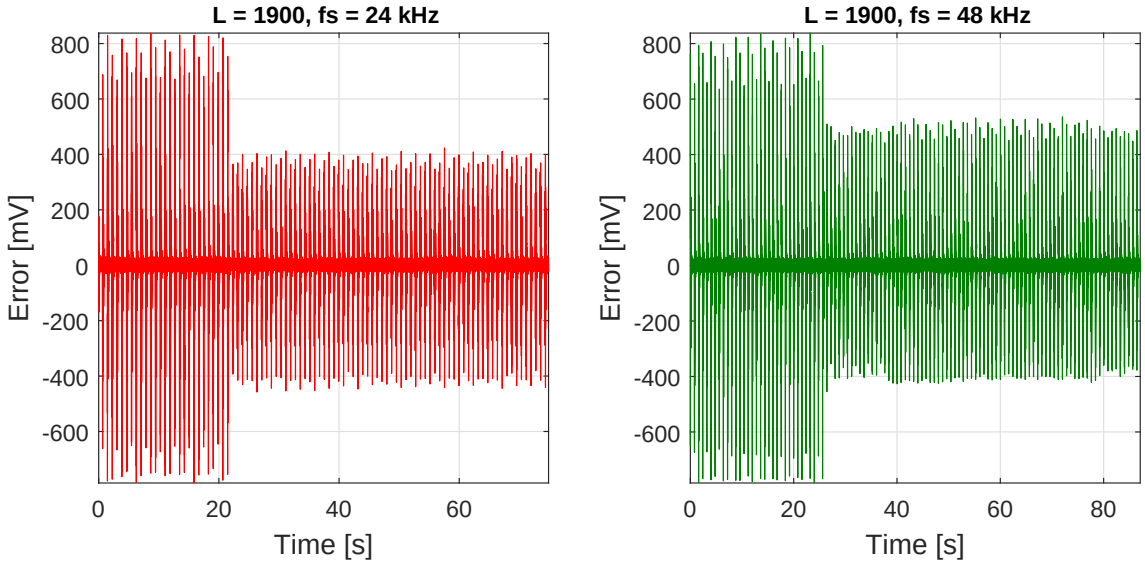
M.2.2. Szabadon lévő hibamikrofon, $L = 1900$, $\mu = 0.02$, $a = 100.0$, $M = 900$



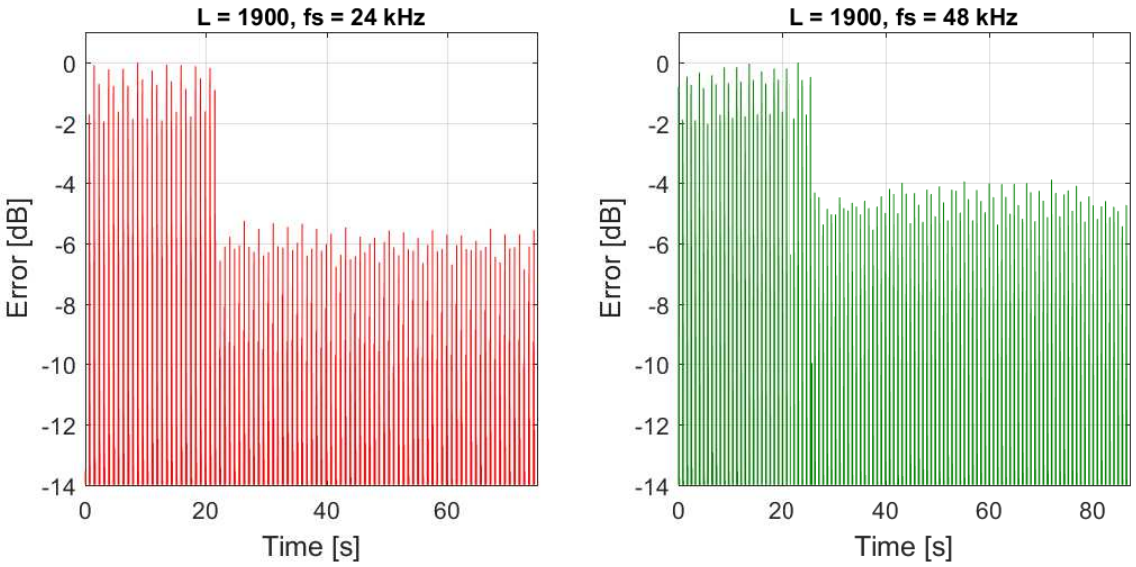
M.2.8. ábra. *Hibalecsengés,* $L = 1900$



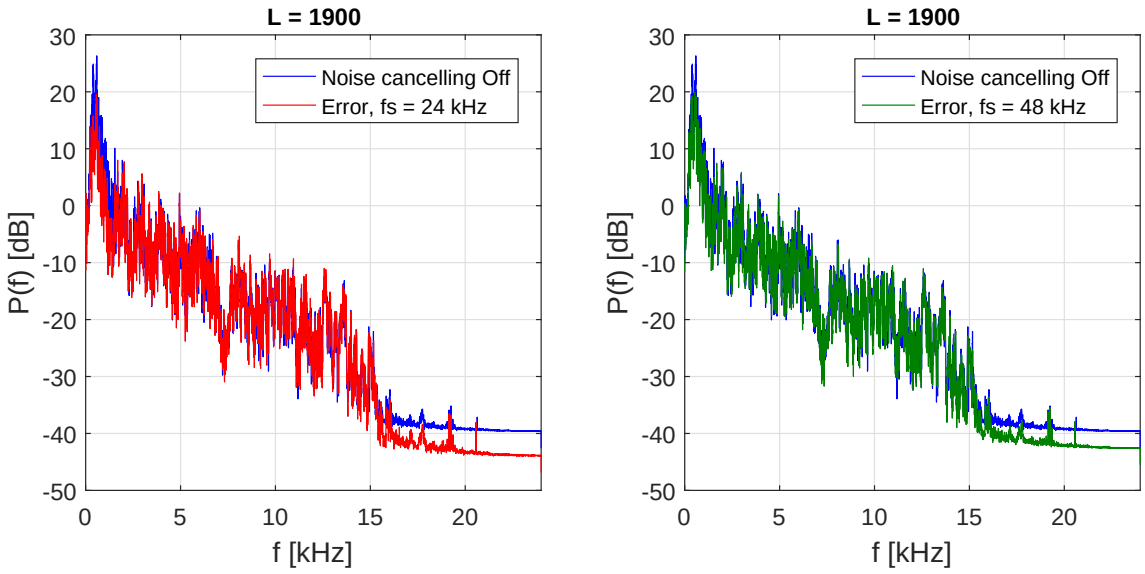
M.2.9. ábra. *Hibalecsengés dB-ben,* $L = 1900$



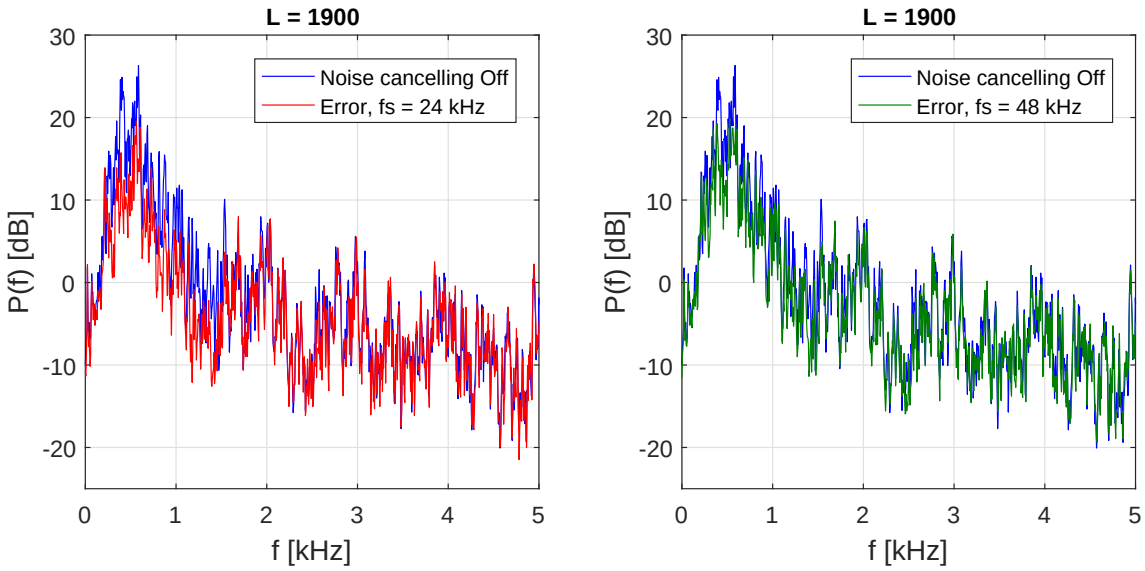
M.2.10. ábra. *Hibacsökkenés mértéke,* $L = 1900$



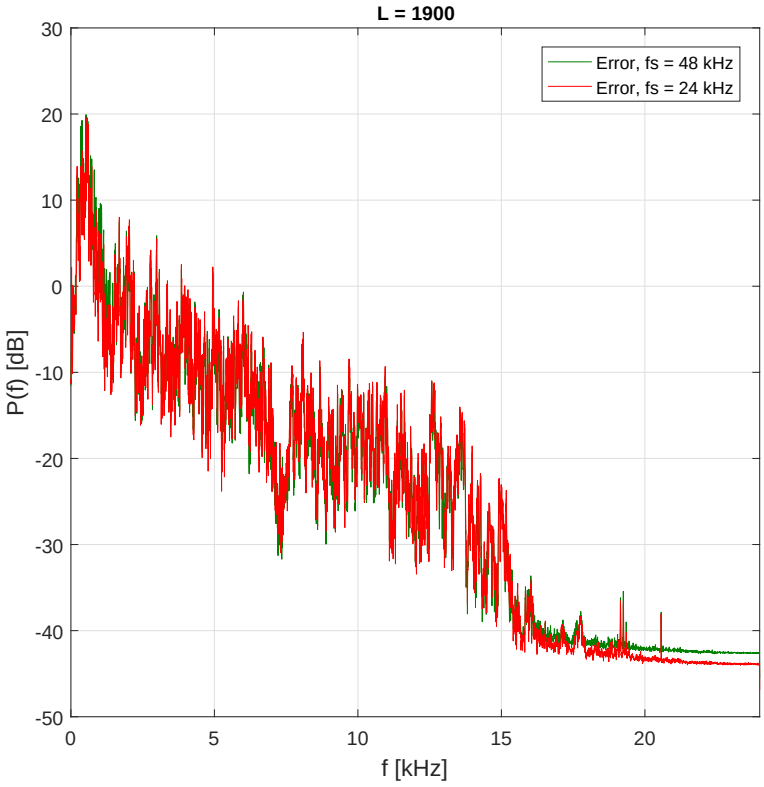
M.2.11. ábra. *Hibacsökkenés mértéke dB-ben,* $L = 1900$



M.2.12. ábra. *Hibajel teljesítményspektruma,* $L = 1900$

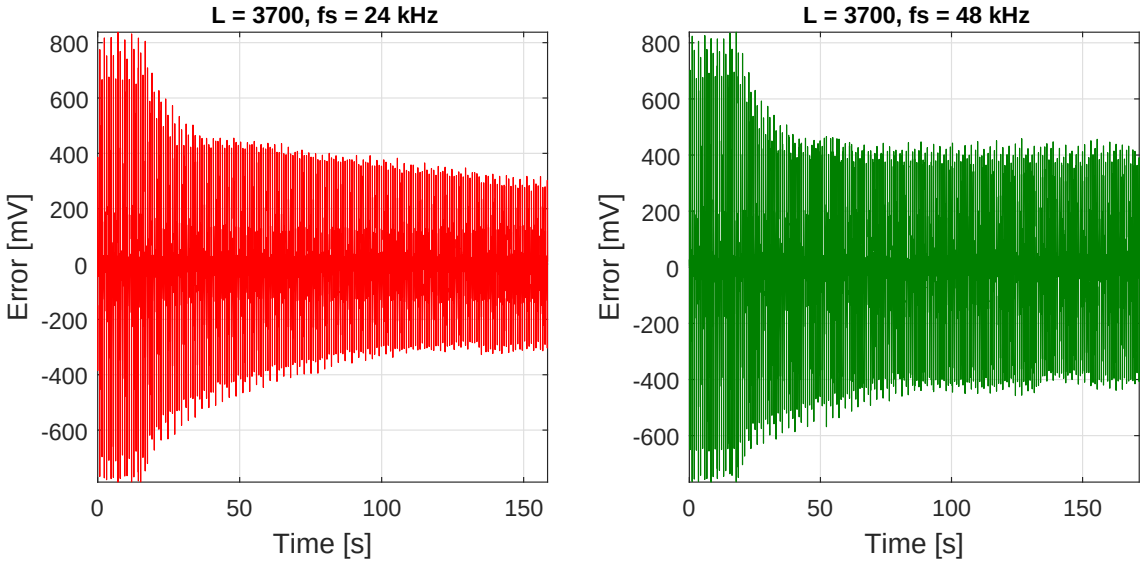


M.2.13. ábra. *Hibajel teljesítményspektruma (0-5) kHz tartományban,* $L = 1900$

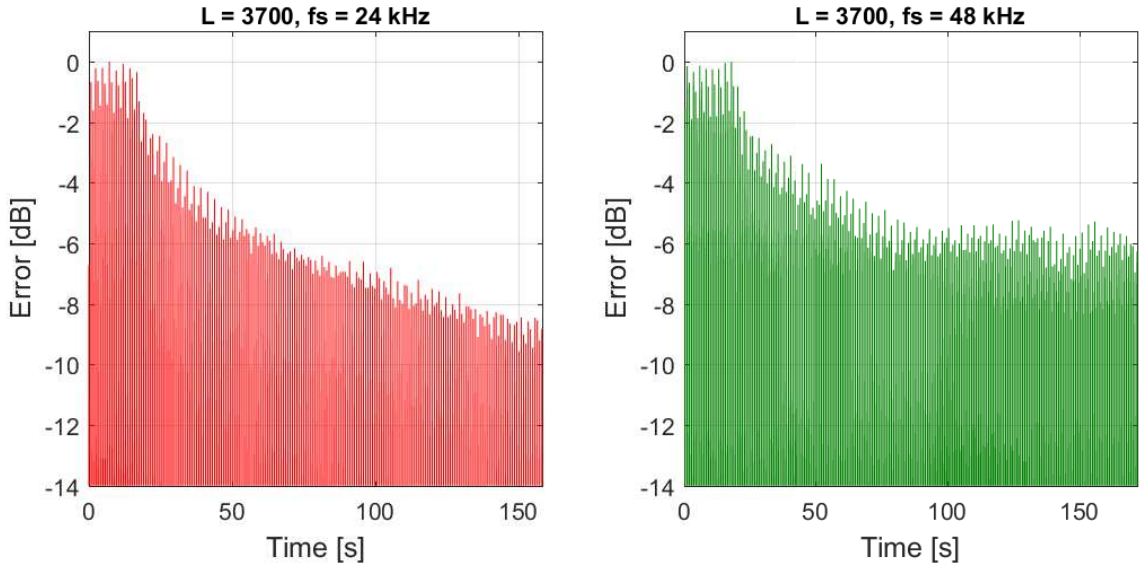


M.2.14. ábra. *24 kHz és 48 kHz mintavételi frekvenciák mellett működő aktív zajcsökkentés összehasonlítása,* $L = 1900$

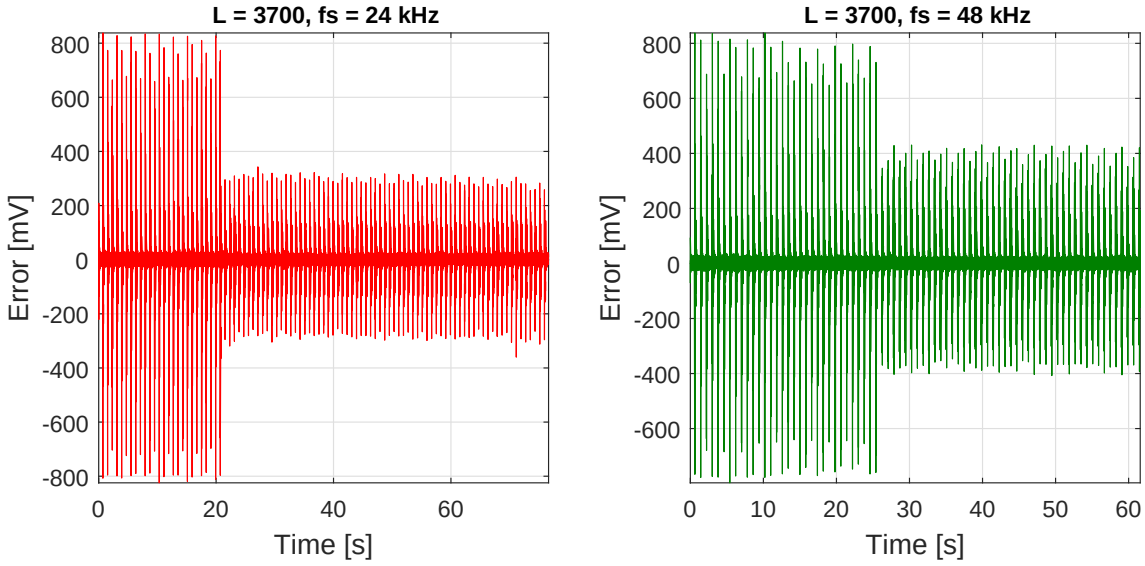
M.2.3. Szabadon lévő hibamikrofon, $L = 3700$, $\mu = 0.005$, $a = 100.0$, $M = 900$



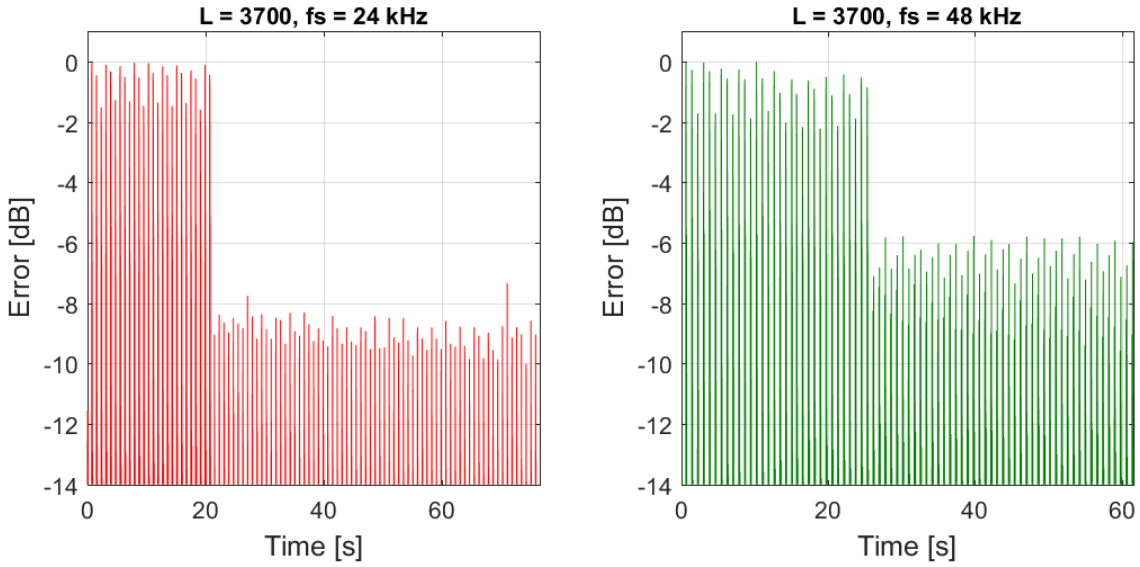
M.2.15. ábra. *Hibalecsengés,* $L = 3700$



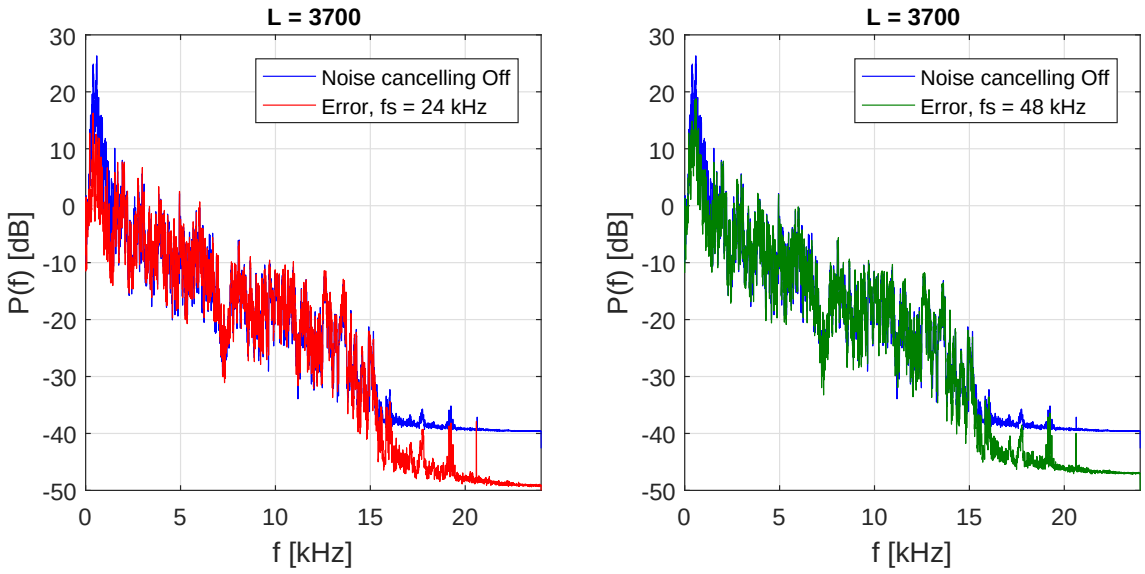
M.2.16. ábra. *Hibalecsengés dB-ben,* $L = 3700$



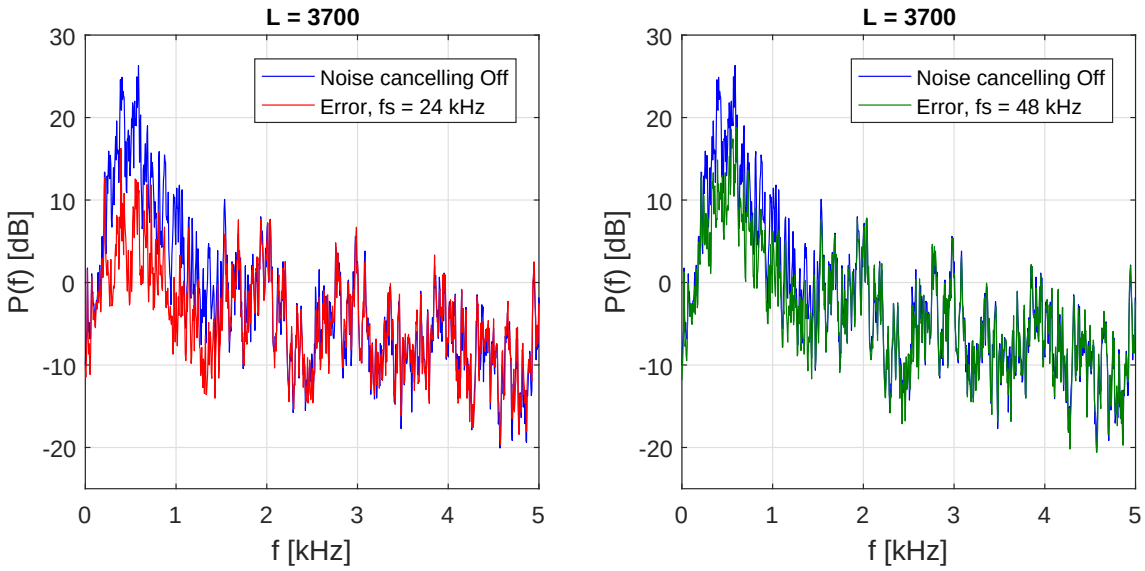
M.2.17. ábra. *Hibacsökkenés mértéke,* $L = 3700$



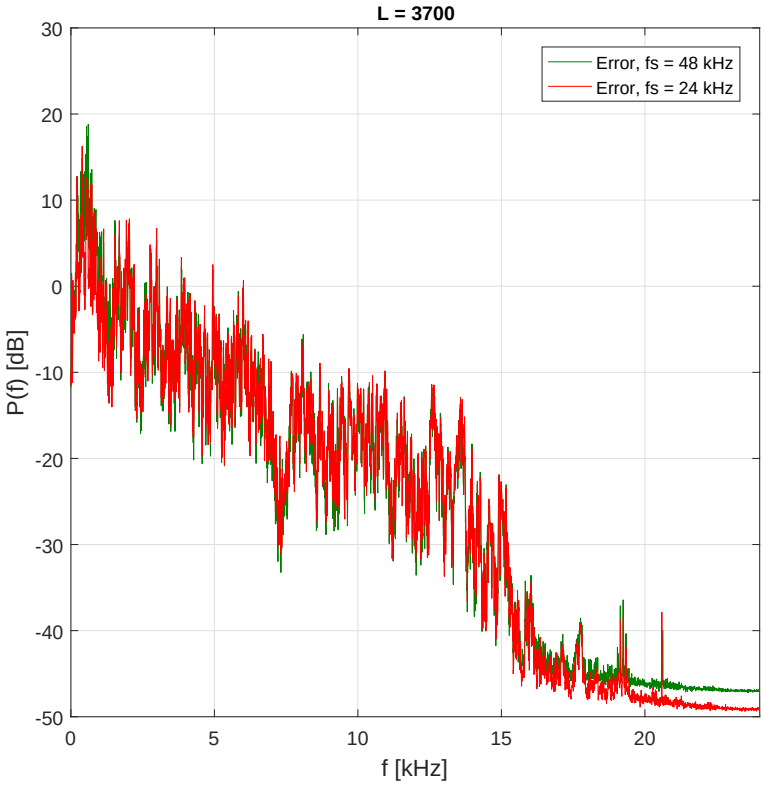
M.2.18. ábra. *Hibacsökkenés mértéke dB-ben,* $L = 3700$



M.2.19. ábra. *Hibajel teljesítményspektruma,* $L = 3700$

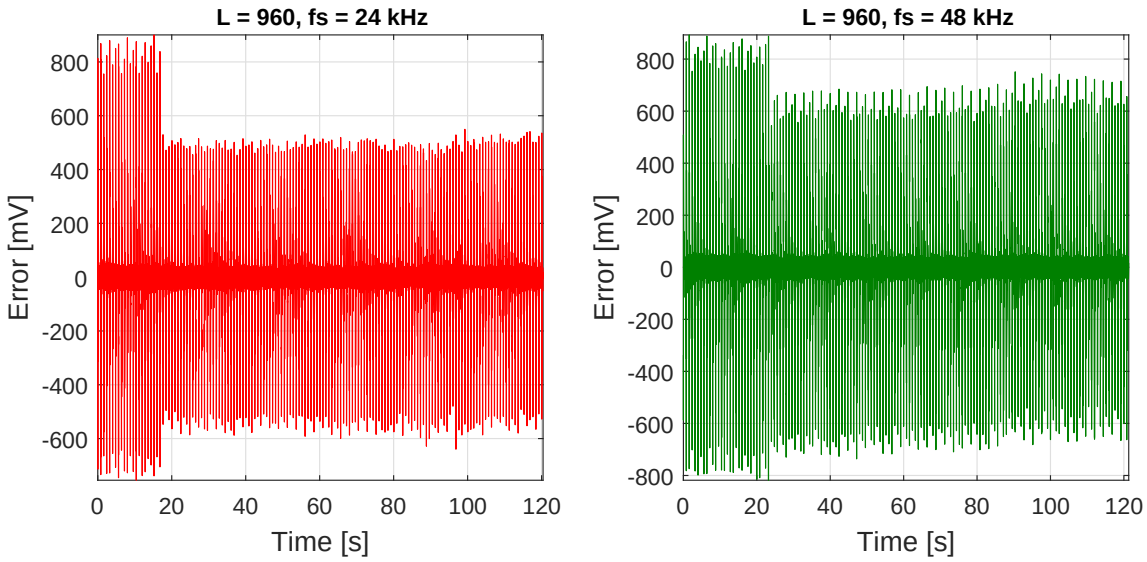


M.2.20. ábra. *Hibajel teljesítményspektruma (0-5) kHz tartományban,* $L = 3700$

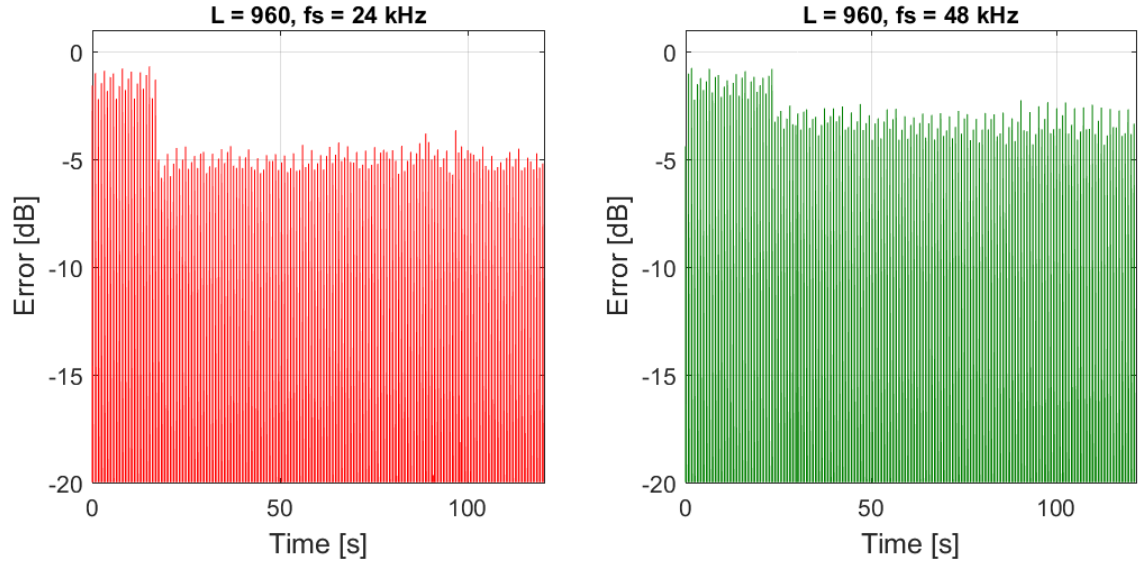


M.2.21. ábra. *24 kHz és 48 kHz mintavételi frekvenciák mellett működő aktív zajcsökkentés összehasonlítása,* $L = 3700$

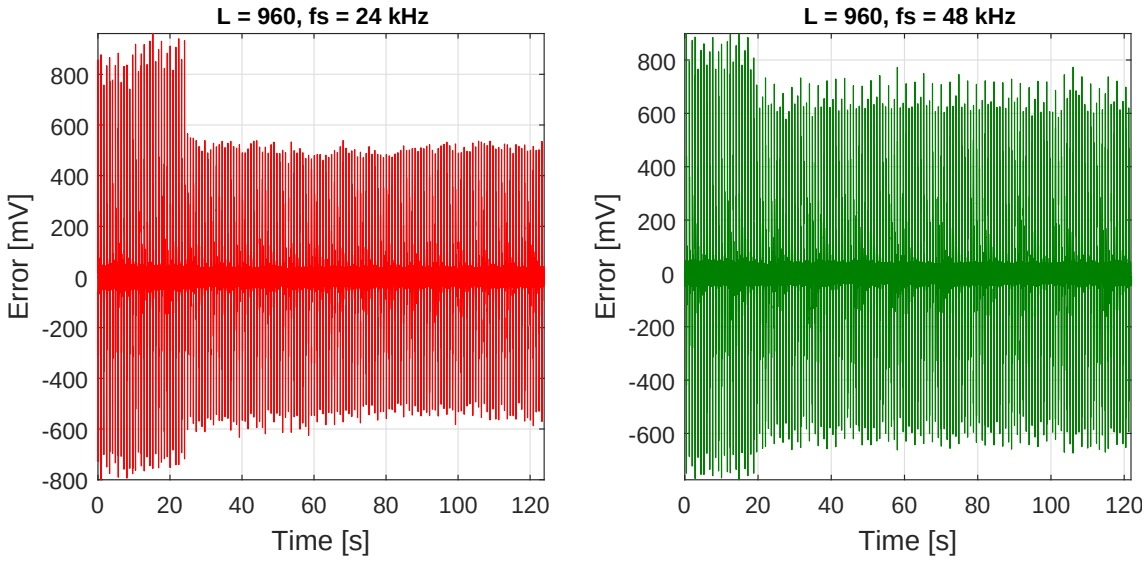
M.2.4. Aktív zajcsökkentő fejhallgató, $L = 960$, $\mu = 0.03$, $a = 100.0$, $M = 900$



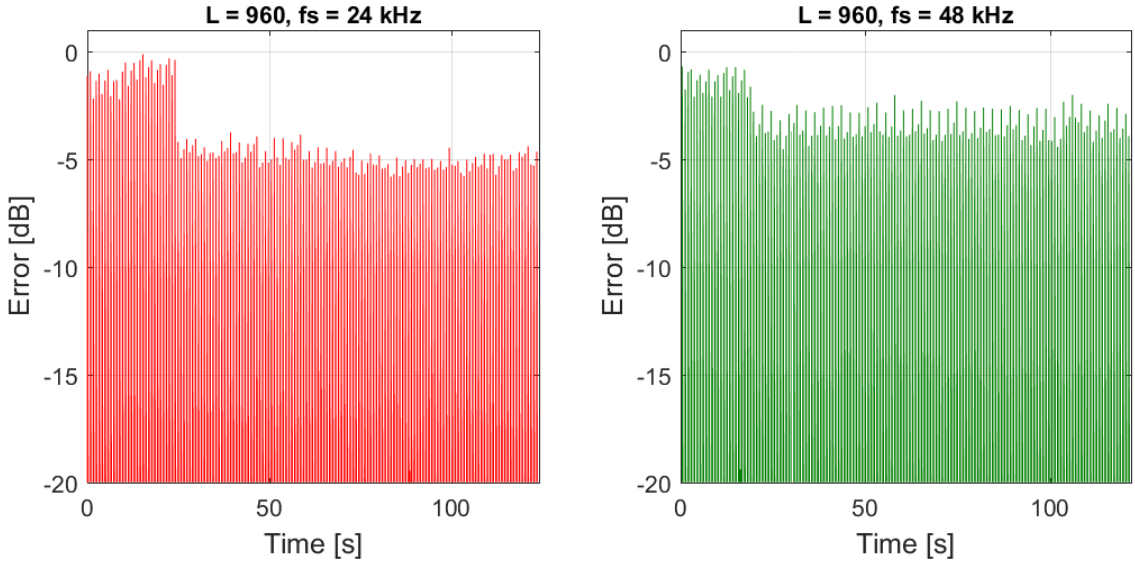
M.2.22. ábra. *Hibalecsengés,* $L = 960$



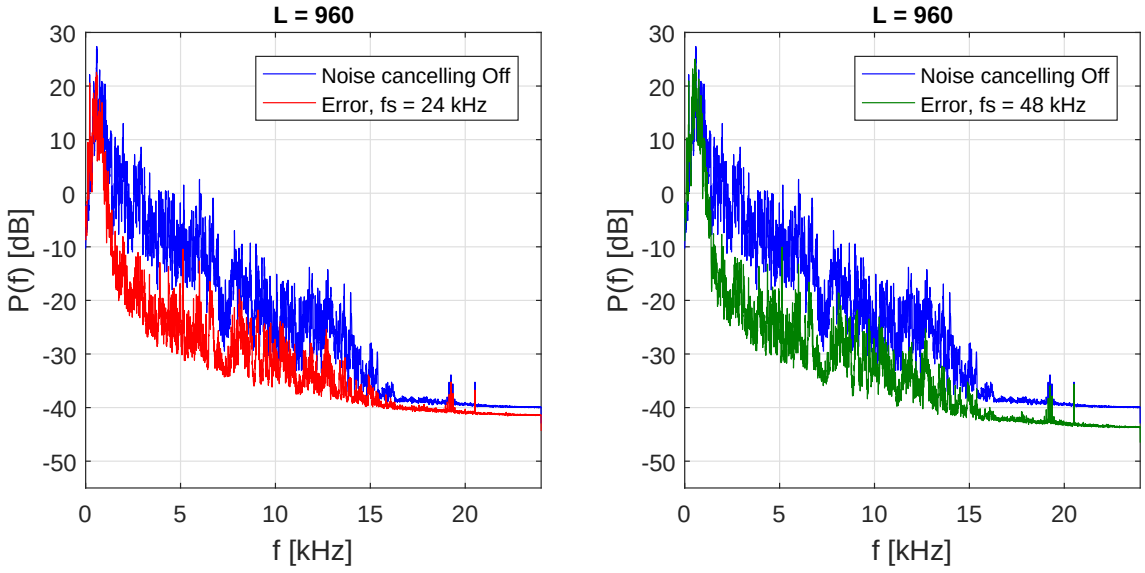
M.2.23. ábra. *Hibalecsengés dB-ben,* $L = 960$



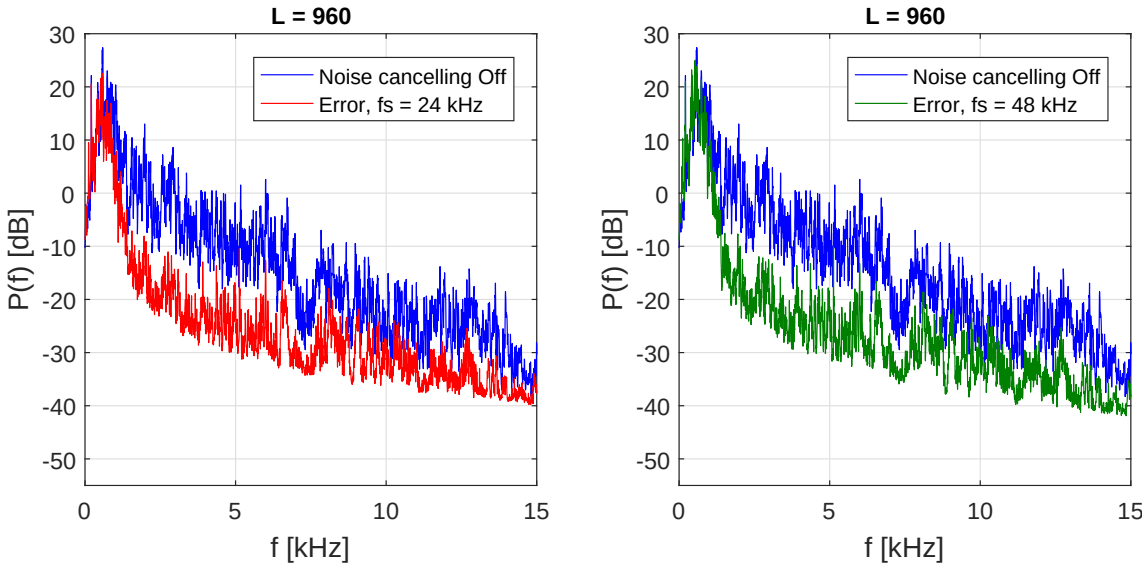
M.2.24. ábra. *Hibacsökkenés mértéke,* $L = 960$



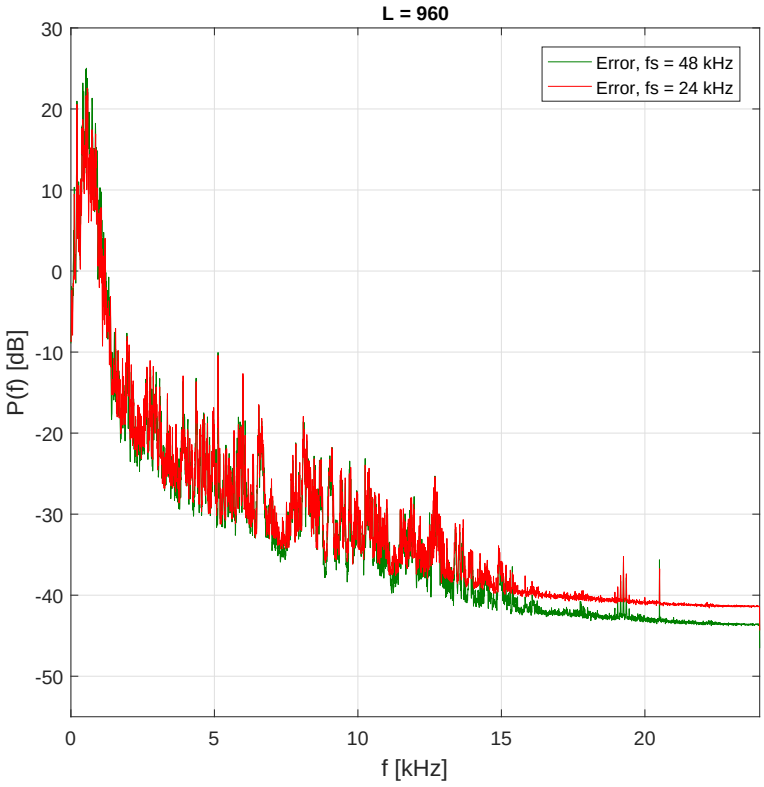
M.2.25. ábra. *Hibacsökkenés mértéke dB-ben,* $L = 960$



M.2.26. ábra. *Hibajel teljesítményspektruma,* $L = 960$

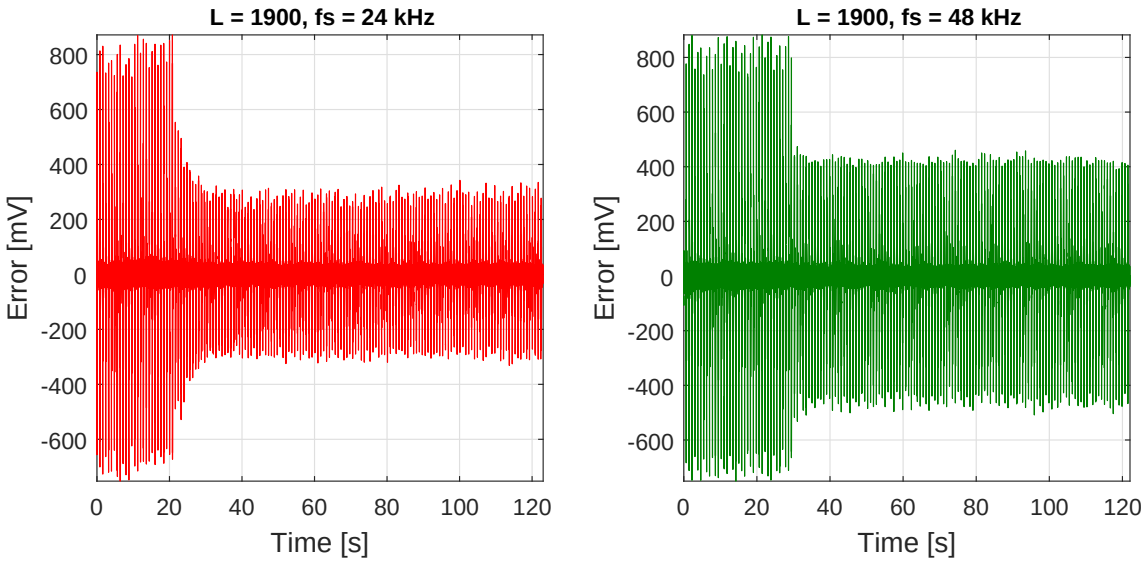


M.2.27. ábra. *Hibajel teljesítményspektruma (0-5) kHz tartományban,* $L = 960$

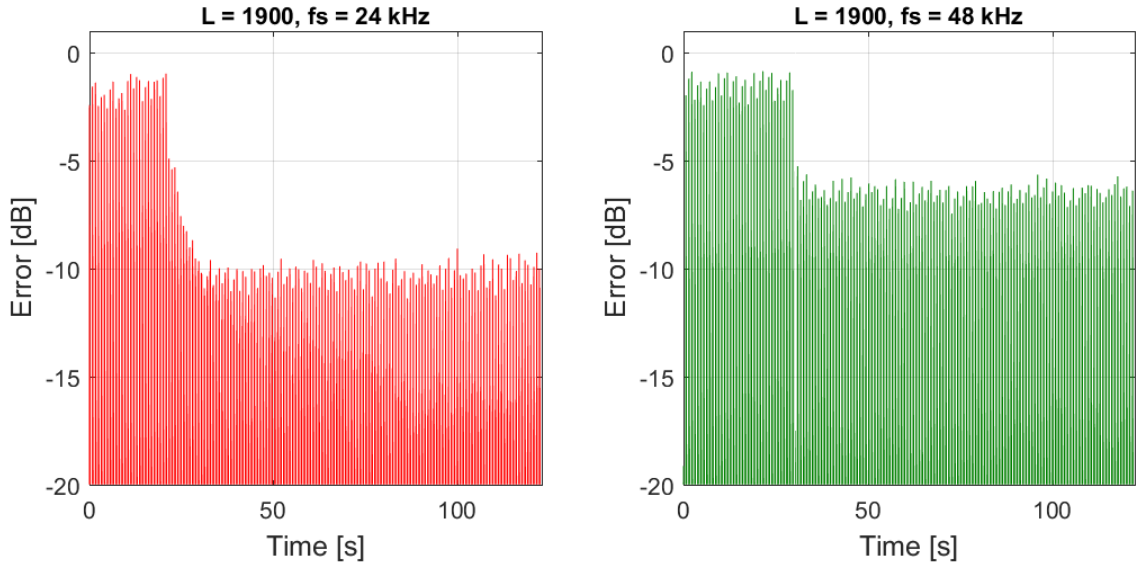


M.2.28. ábra. *24 kHz és 48 kHz mintavételi frekvenciák mellett működő aktív zajcsökkentés összehasonlítása,* $L = 960$

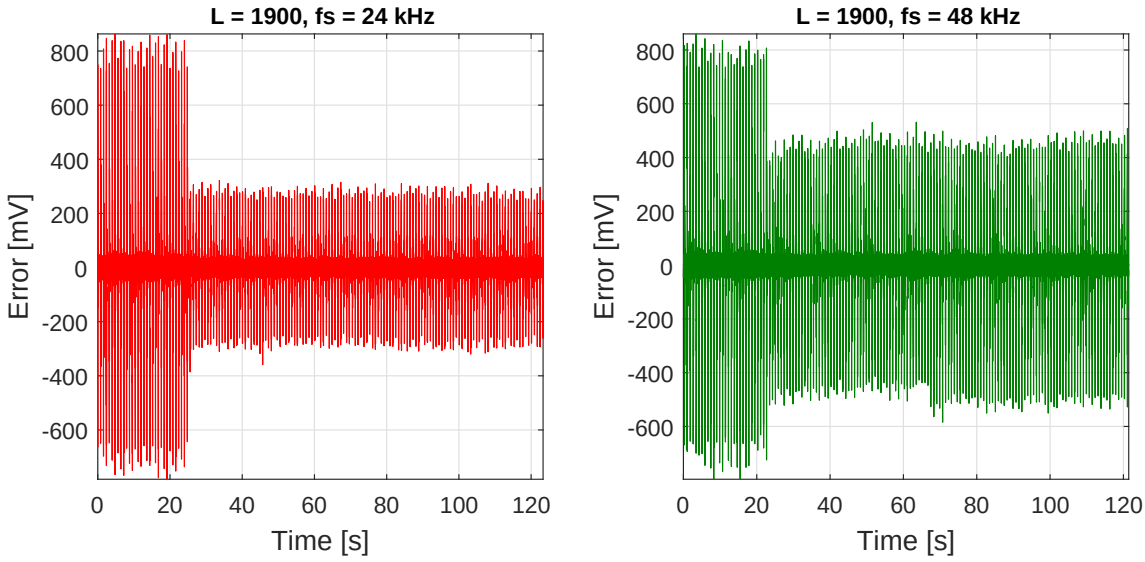
M.2.5. Aktív zajcsökkentő fejhallgató, $L = 1900$, $\mu = 0.02$, $a = 100.0$, $M = 900$



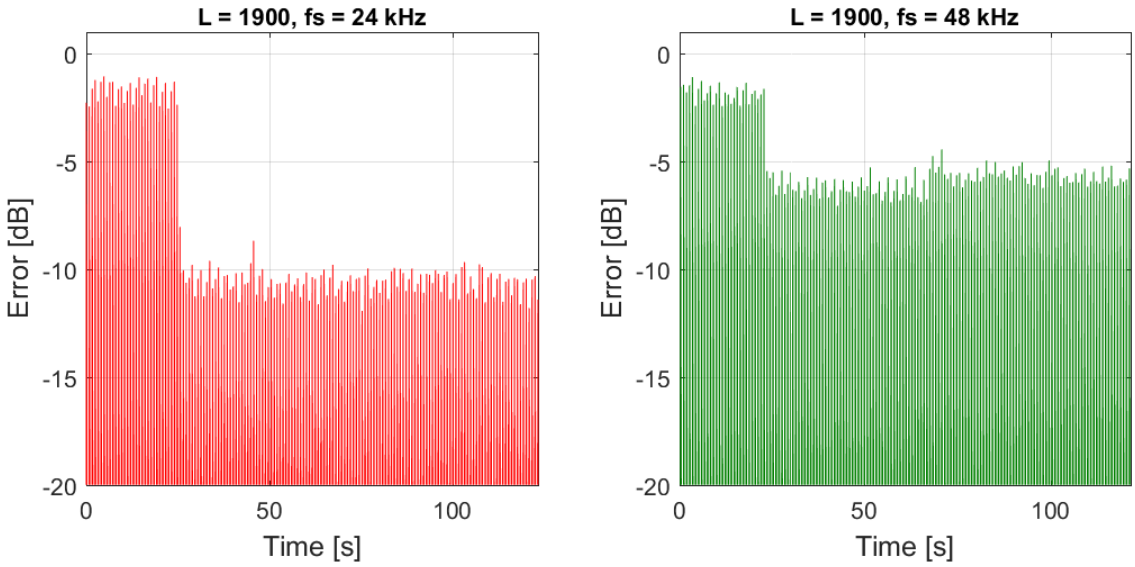
M.2.29. ábra. *Hibalecsengés,* $L = 1900$



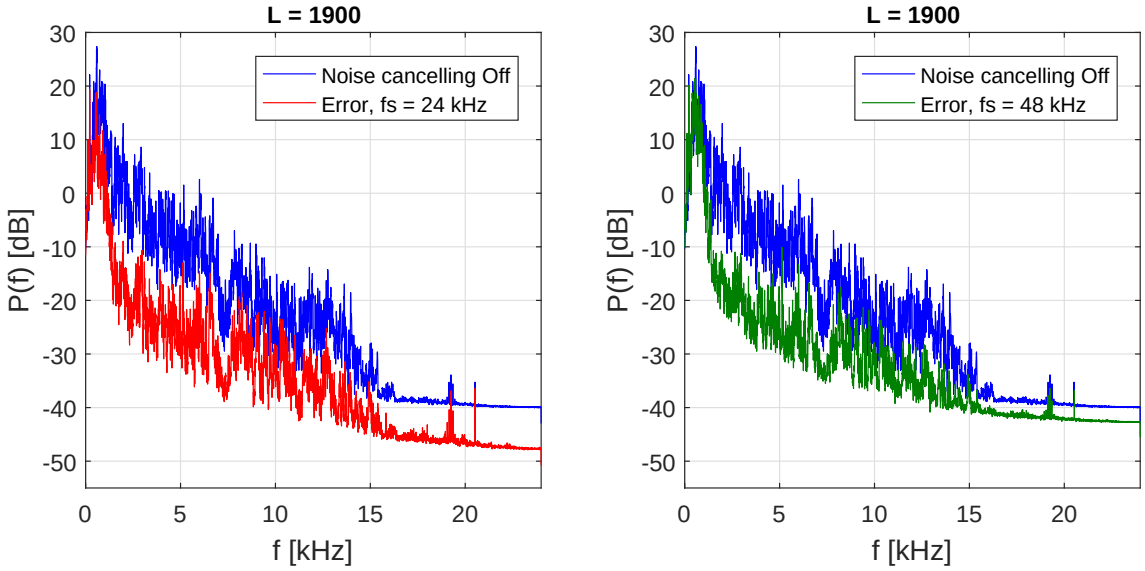
M.2.30. ábra. *Hibalecsengés dB-ben,* $L = 1900$



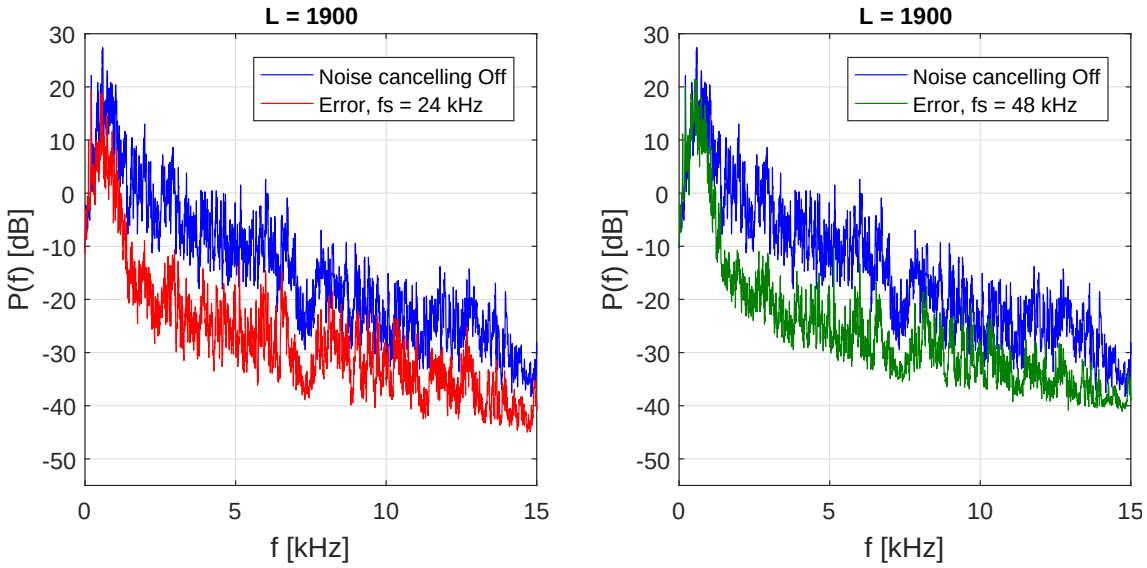
M.2.31. ábra. *Hibacsökkenés mértéke,* $L = 1900$



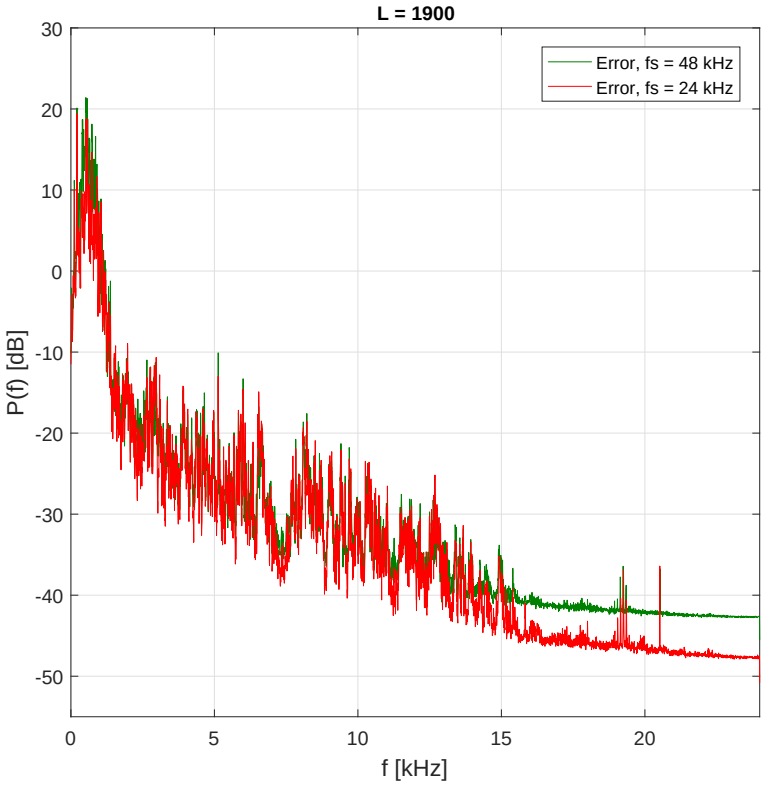
M.2.32. ábra. *Hibacsökkenés mértéke dB-ben,* $L = 1900$



M.2.33. ábra. *Hibajel teljesítményspektruma,* $L = 1900$

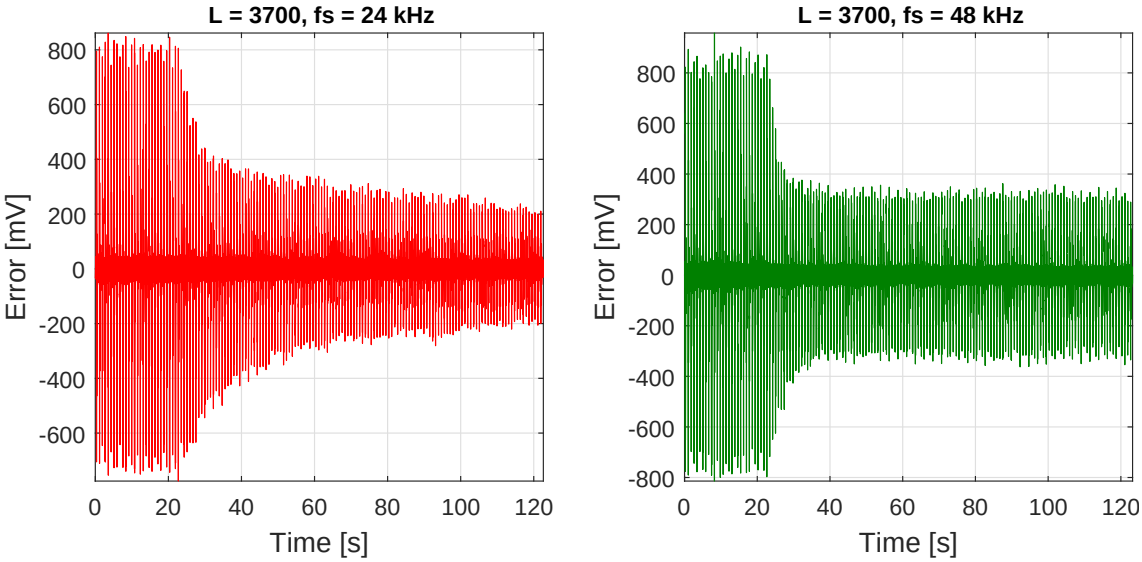


M.2.34. ábra. *Hibajel teljesítményspektruma (0-5) kHz tartományban,* $L = 1900$

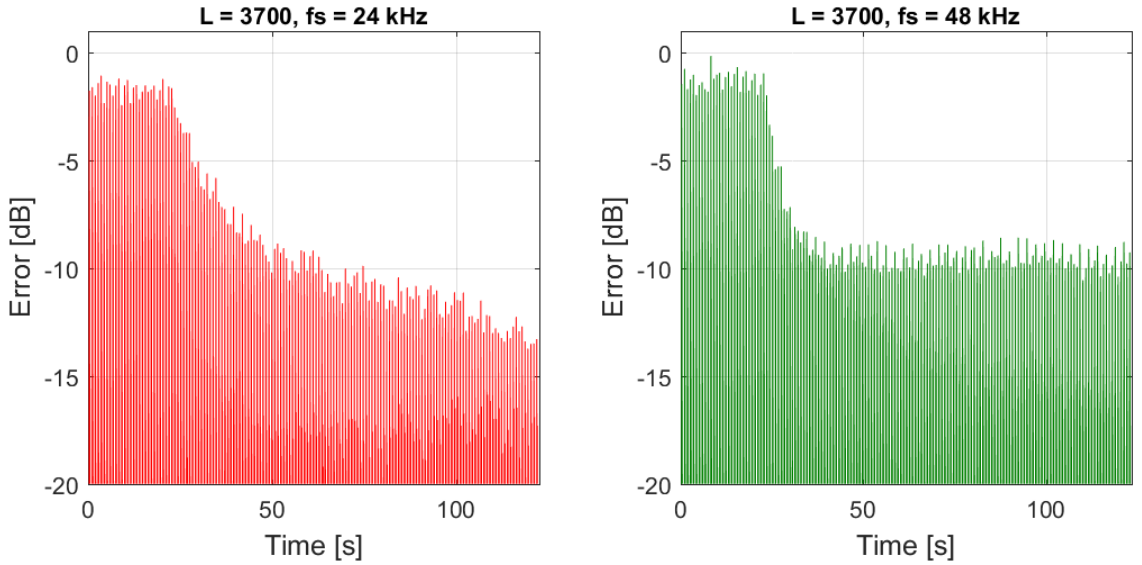


M.2.35. ábra. *24 kHz és 48 kHz mintavételi frekvenciák mellett működő aktív zajcsökkentés összehasonlítása,* $L = 1900$

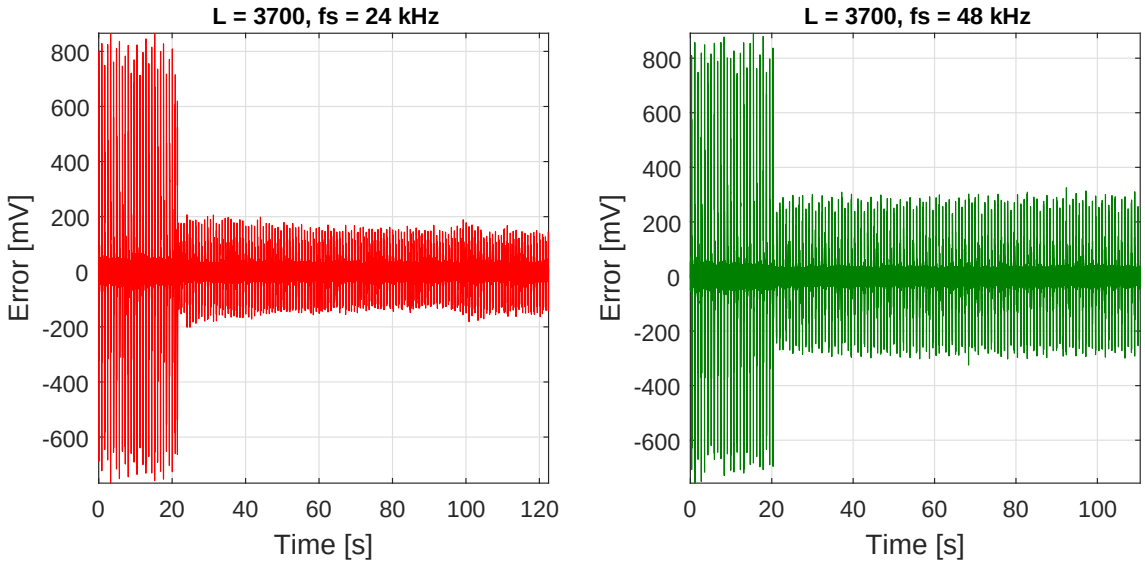
M.2.6. Aktív zajcsökkentő fejhallgató, $L = 3700$, $\mu = 0.005$, $a = 100.0$, $M = 900$



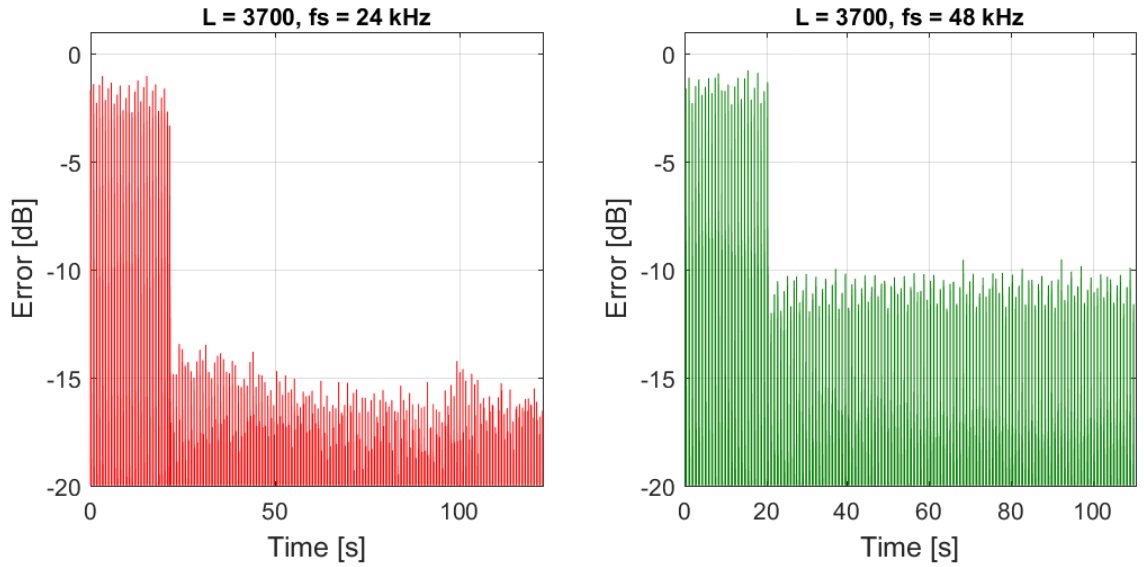
M.2.36. ábra. *Hibalecsengés,* $L = 3700$



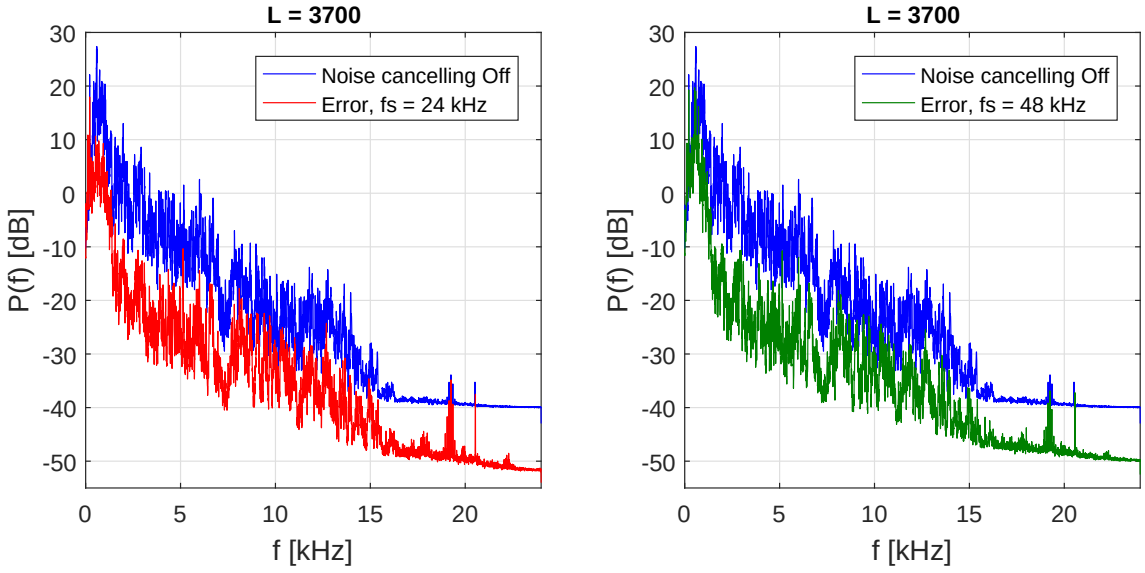
M.2.37. ábra. *Hibalecsengés dB-ben,* $L = 3700$



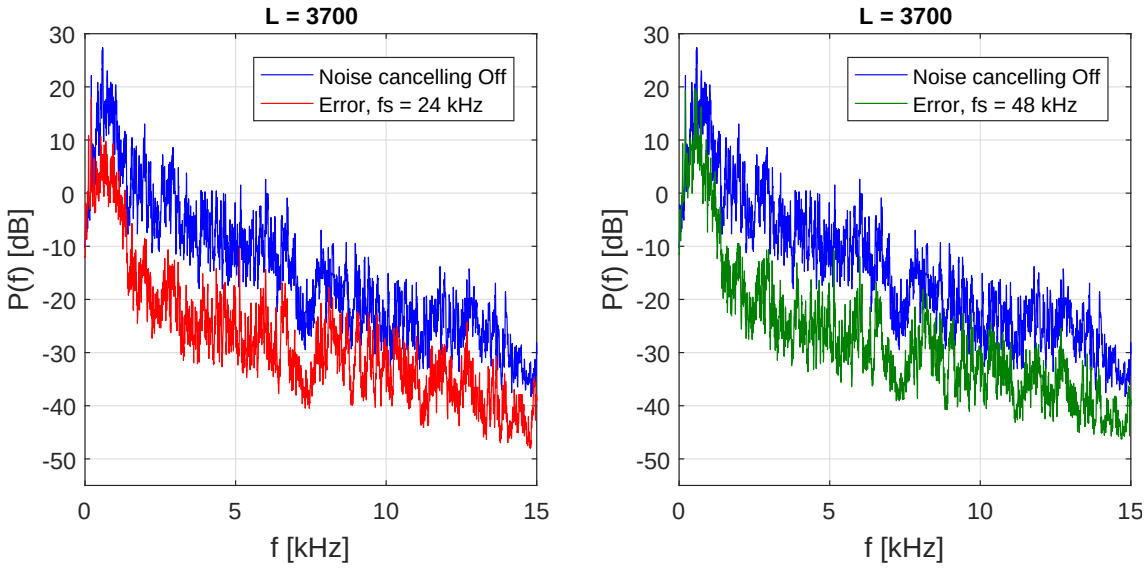
M.2.38. ábra. *Hibacsökkenés mértéke,* $L = 3700$



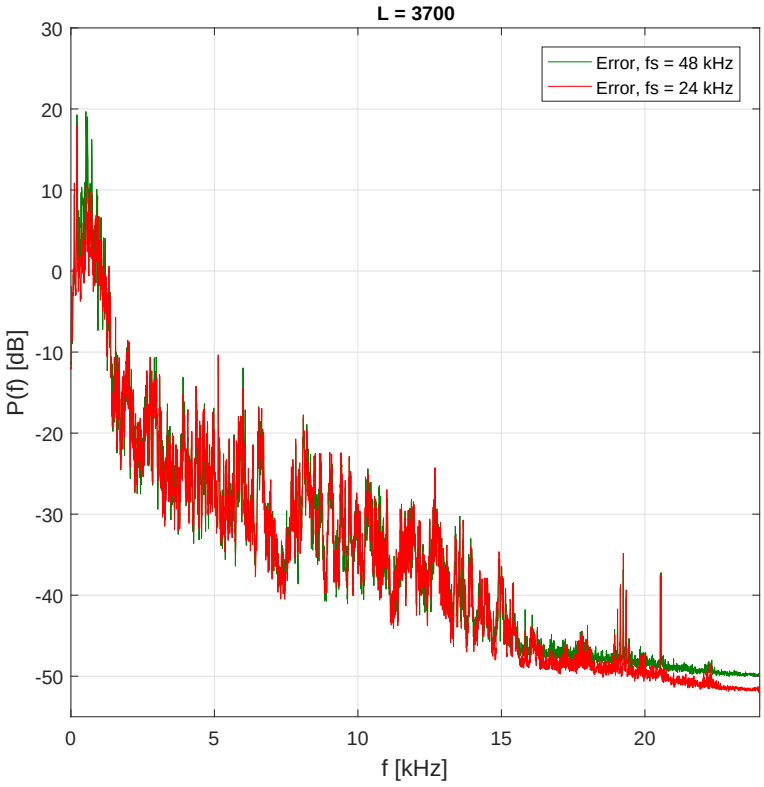
M.2.39. ábra. *Hibacsökkenés mértéke dB-ben,* $L = 3700$



M.2.40. ábra. *Hibajel teljesítményspektruma,* $L = 3700$



M.2.41. ábra. *Hibajel teljesítményspektruma (0-5) kHz tartományban,* $L = 3700$



M.2.42. ábra. *24 kHz és 48 kHz mintavételi frekvenciák mellett működő aktív zajcsökkentés összehasonlítása,* $L = 3700$