Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Adaptív vonaljavító megvalósítása LMS-algoritmus segítségével

Témalaboratórium dokumentáció
2022/23. I. félév

Jakab Tamás Zoltán

III. évf., villamosmérnök szakos hallgató
BSc, Beágyazott információs rendszerek ágazat

Konzulens:

Dr. Sujbert László

Méréstechnika és Információs Rendszerek Tanszék

Tartalomjegyzék

1 Bevezetés

Napjainkban rendkívül nagy jelentőséggel bír a jelfeldolgozás és a jeltovábbítás során a keletkezett zavarok és a hasznos jel különválasztása. Ezt számos területen használják, mint például a mobiltelefóniában, orvosi eszközök jeleinek zavarelnyomásában és a hadászatban is. Feladatom a félév során az adaptív vonaljavító elméleti alapjának és struktúrájának megértése MATLAB nyelvű környezetben, majd a tanultak szintézise egy digitális jelfeldolgozó kártyán.

2 Pontos feladatkiírás

AZ LMS-algoritmus népszerű, kis számításigényű algoritmus véges impulzusválaszú (FIR) szűrők adaptálására. Az adaptív szűrő egy adott struktúrában modellezési feladatot lát el, de arra is alkalmas, hogy jelek mintáit korábbi értékeiből "megjósolja", azaz predikálja. Ezt a tulajdonságot használja ki az ún. adaptív vonaljavító (adaptive line enhancer, ALE). A zajjal terhelt jelet az adaptív szűrő bemenetére vezetjük, és megkíséreljük a jel késleltetett értékeit predikálni. Attól függően, hogy szélessávú vagy periodikus zaj csökkentése a cél, az adaptív szűrő kimenete vagy az algoritmus hibajelele a rendszer kimenete.

A feladatom a fenti struktúrák megvalósítása MATLAB környezetben, illetve ADSP 21364 lebegőpontos jelfeldolgozó processzoron.

3 Elméleti alapok

3.1 Az LMS (Least Mean Squares) algoritmus

Tekintsük az alábbi ábrát (1. ábra), W egy adaptív szűrő melynek a feladata, hogy a kimenet olyan mértékben állítsa, hogy a mintául vett jel különbségével vett négyzete minimális legyen és a lineáris rendszer adaptív szűrő struktúráját tekintve véges impulzusválaszú legyen. Az együttható vektor meghatározására számos matematikai eljárás lehetséges, vizsgáljuk meg esetünkben az LMS algoritmussal való megvalósítását.

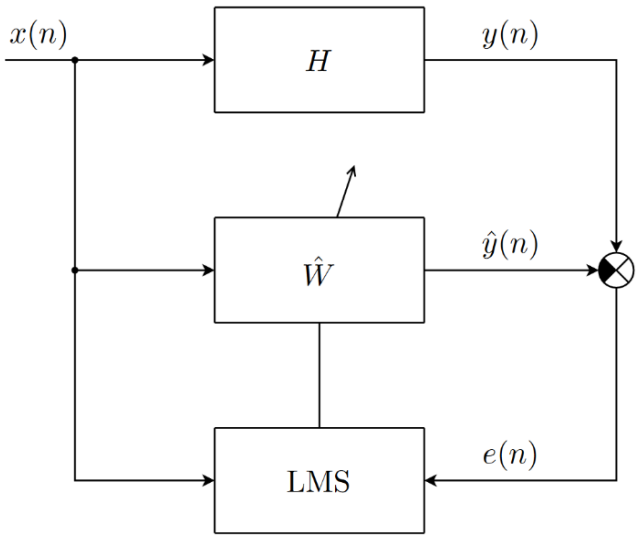
Az LMS algoritmussal rendkívül egyszerűen tudjuk megvalósítani a modellillesztést, mint feladatot. Ha az algoritmus nem állna a rendelkezésünkre ismernünk kellene a rendszer statisztikai jellemzőit és számos bonyolult matematikai egyenletet kellene megoldanunk, aminek a számításigénye hosszabb időbe is beletelhet. Ebből következően feladatunk

szempontjából kiemelkedő jelentőséggel bír az, hogy mindent a lehető legrövidebb idő alatt valósítsunk meg. Amennyibben ezt nem tudnánk megvalósítani, abban az esetben gyorsabban jönne számunkra az adat, mint ahogy azt fel tudnánk dolgozni és ez komoly problémát jelentene.

Az algoritmus egy FIR szűrőt tud megvalósítani, melynek szűrőegyüttható vektorát a rendszer dinamikus változása esetén módosítani is tudja. A módosítás mértékét egy úgynevezett bátorsági tényező határozza meg, melynek jellemző értéke a szűrőegyütthatók számának reciproka körül van. Fontos kiemelni, hogy a FIR szűrő stabilitásából nem következik a rendszer stabilitása, összefüggésben van a bátorsági tényezővel.

A rendszernek négy bemenete és két kimenete van. A bemeneteket a megvalósítani kívánt rendszerátvitel bemenete és kimenete a szűrőegyütthatók száma (milyen pontossággal akarjuk modellezni a rendszert) és a bátorsági tényező képzik. A kimenetek az LMS által hangolt szűrő kimenete, illetve a hibajel, amely az eredeti kimenet és a becsült kimenet különbsége.

Mivel a szűregyütthatók nem fognak állandósulni (folyamatosan valamekkora lépésekkel ugrálnak), így folyamatosan lesz egy pillanatnyi hiba. Ha azonban ezeket a módosításokat hosszabb időintervallumon tekintjük a pillanatnyi hibák kiátlagolódnak.



[1. ábra] LMS algoritmus

$$e(n) = y(n) - \hat{w}^{T}(n) * x(n)$$

$$\hat{w}(n+1) = \hat{w}(n) + 2\mu e(n) * x(n)$$

Jelmagyarázat:

x(n)	A modellezni kívánt rendszer bemeneti **vektora**
y(n)	A modellezni kívánt rendszer kimenet
$\hat{y}(n)$	A szűrő által alkotott kimenet
e(n)	A hibajel
$\hat{w}(n)$	A szűrőegyütthatókat tartalmazó **vektor**
μ	Bátorsági tényező

3.2 Adaptív vonaljavitó

Az adaptív vonaljavítónak jelentős szerepe van a digitális jelfeldolgozásban, mivel kiemelkedő hatékonysággal lehet vele a hasznos jelet és az azt terhelő additív zajt különválasztani. Mivel alkalmazásához szükségünk van az LMS algoritmusra, így azokat a pozitívumokat melyeket korábban megismertünk, ennél a struktúránál is számításba vehetjük (kevés előzetes ismeret szükséges és mellőzhetjük a bonyolult számításokat).

Maga a struktúra rendkívül egyszerű. Egy késleltető és egy lineáris prediktor soros kapcsolása. A lineáris prediktort esetünkben egy FIR szűrő fogja megvalósítani, melyet az LMS algoritmus fog hangolni.

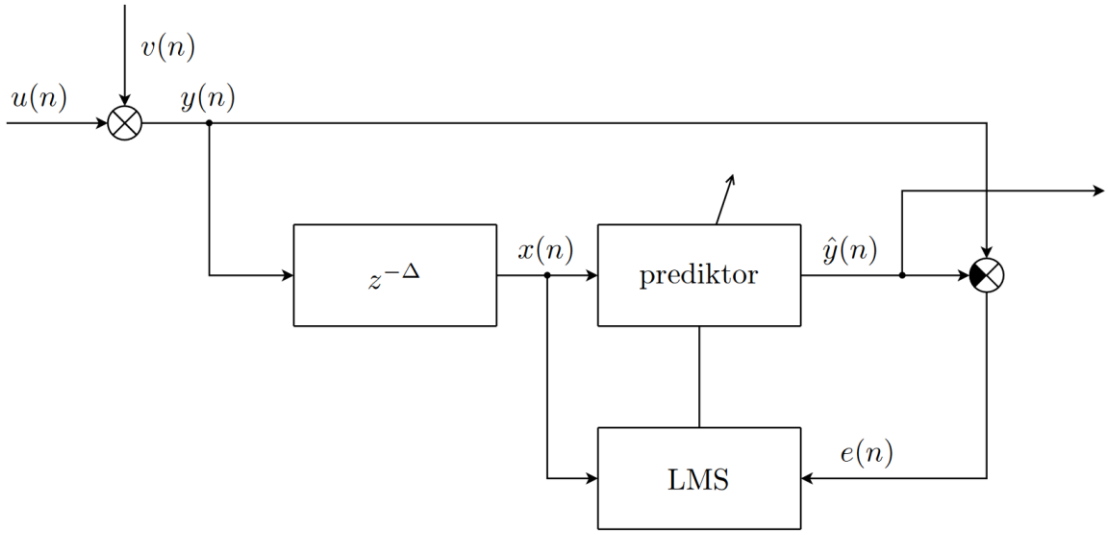
Ahhoz, hogy a zaj szűrésének módszerét és a késleltető szükségességét megértsük két esettel kell foglalkoznunk:

- A zaj a hasznos jelhez képest keskenysávú
- A zaj a hasznos jelhez képest szélessávú

Amennyiben a zajkomponens keskenysávú (például egy szinusz hullám), abban az esetben a késleltetést olyan nagyra kell megválasztani, hogy a hasznos jel és annak késleltetett változata korrelálatlan legyen egymással. Az LMS algoritmus ugyanis a korrelált tényezőt igyekszik átengedni, és a prediktor becsült jele maga a keskenysávú zaj lesz. Ekkor jól

látható, hogy a kimenetünk a hibajel lesz, mely az eredeti jelből és a prediktor kimenetének különbségéből áll.

Amennyiben a zajkomponens szélessávú (például fehérzaj), abban az esetben a késleltetést olyan nagyra kell megválasztani, hogy a zaj és annak késleltetett változata korrelálatlan legyen egymással. Ilyenkor az LMS által vezérelt prediktor kimenetére a jel zajkomponens nélküli része fog kerülni, így a kimenetünk is ez lesz.



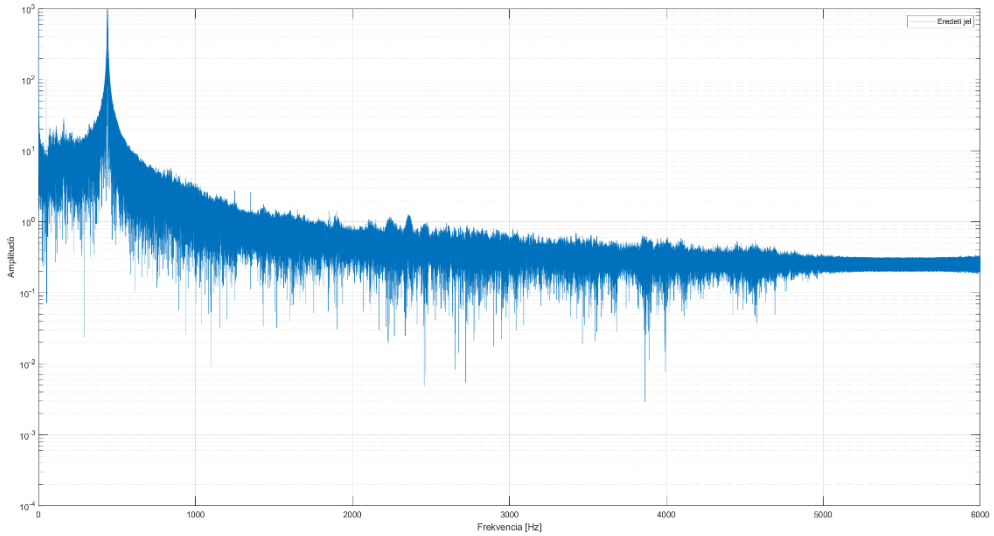
[2. ábra] Adaptív vonaljavító

$$x(n) = y(n - \Delta)$$

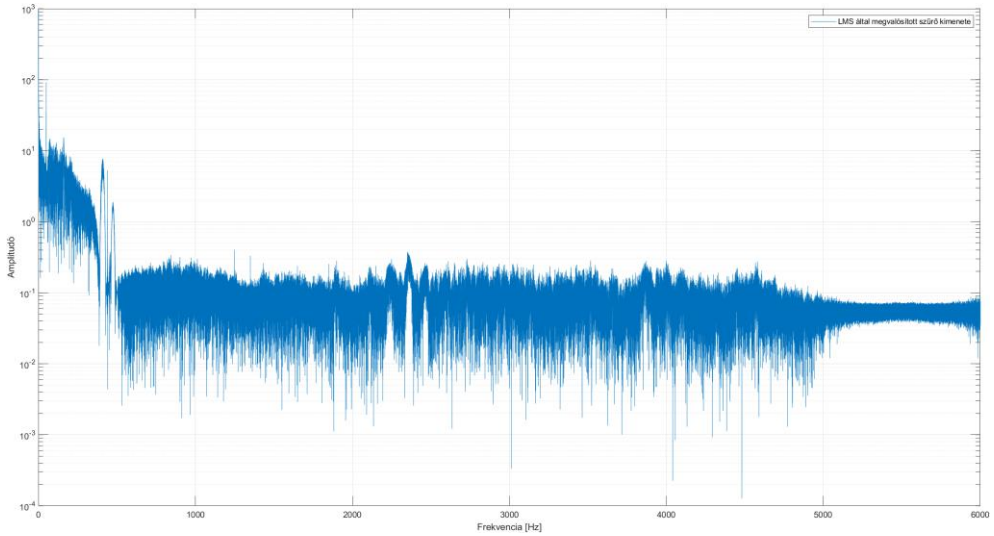
4 Megvalósított feladatok

4.1 FIR szűrő megvalósítása LMS algoritmussal (MATLAB)

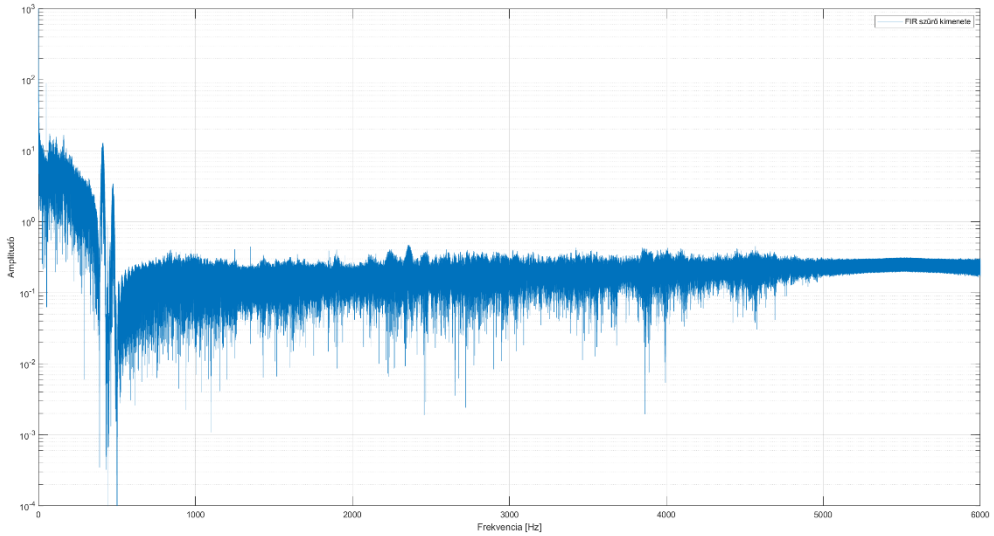
Az LMS algoritmussal megvalósítottam egy FIR szűrőt. Jelnek egy versrészletet választottam, melyet egy 440 Hz-es szinuszjellel zavartam additív módon. A FIR szűrő 1000 db szűrőegyütthatóval rendelkezett, illetve 390 és 490 Hz-es frekvenciák között csillapított. Az LMS algoritmus 1000 db szűrőegyütthatóval igyekezte lemásolni az eredeti szűrőt. A bátorásgi tényező értéke ~10^{-4} volt.



[3. ábra] Az eredeti jel Bode-diagramja



[4. ábra] Az FIR szűrő kimenetének Bode-diagramja



[5. ábra] Az LMS által hangolt szűrő kimenetének Bode-Diagramja

Az amplitúdó értékek 440 Hz-nél:

Jel:	Ábra:	Erősítés:	Erősítés decibelben:
Eredeti jel	[3.ábra]	44731,1	93,01 dB
FIR szűrő kimenete	[4.ábra]	5,18866	14,30 dB
LMS által hangolt szűrő kimenete	[5.ábra]	5,05183	14,07 dB

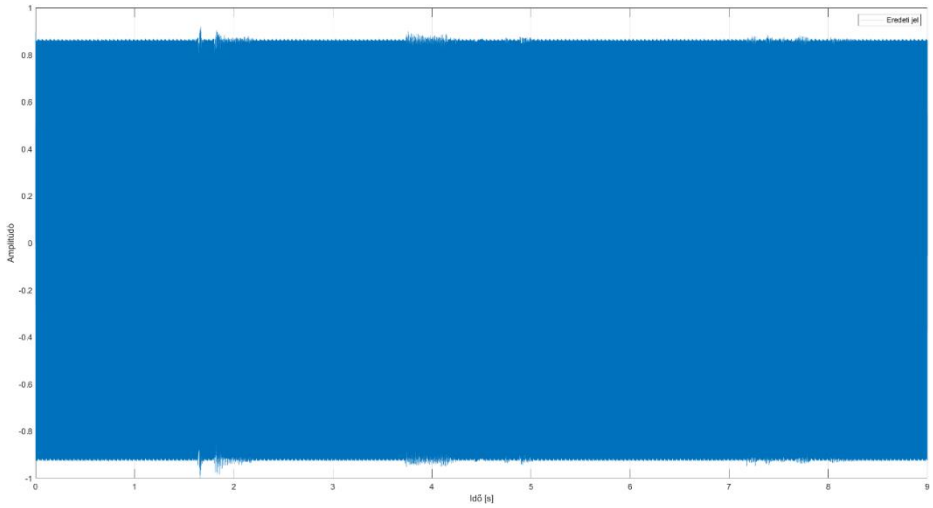
Mint azt ahogyan az alábbi diagramokon is láthatjuk az LMS által hangolt szűrő kiváló módon volt képes lekövetni az eredeti szűrőt. Leellenőriztem számszerűen az értékeket, hogy mennyire változott meg az egyes esetekben a 440 Hz-es tényező értéke, majd meghallgattam a szűrők kimeneteit és meggyőzöttem a feladat sikeres megoldásáról.

A felhasznált kód a 6.1-es fejezetben található.

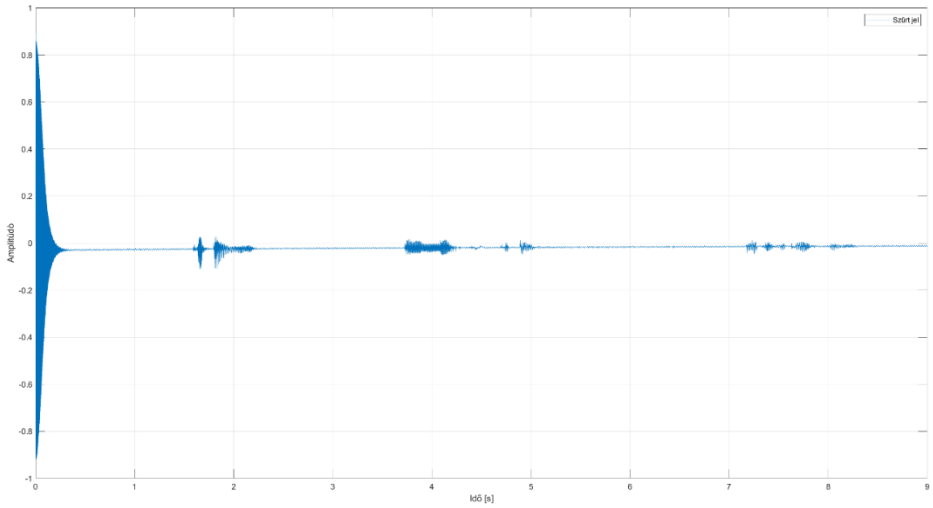
4.2 Adaptív vonaljavító keskenysávú hiba esetén (MATLAB)

Az adaptív vonaljavítót megvalósítottam MATLAB-os környezetben, hogy a kellős struktúrát megfelelő mértékben megismerjem. A zavarásnak egy 440 Hz-es szinuszjelet választottam, melyet additív módon csatoltam a jelhez. A megvalósítás során 1000 db szűrőegyütthatóval dolgoztam, bátorsági tényezőnek pedig 10^{-5} -es értéket, a késleltetésnek pedig 16-os értéket adtam.

Amennyiben az eredeti és a szűrt jel Bode-diagramját vizsgálnánk, nem tapasztalnánk sok különbséget, ugyanis a struktúrának időre van szüksége, hogy a zavarokat kiszűrje a rendszerből, tehát a zavar frekvenciakomponense ugyan úgy megjelenne az ábrán. Számunkra sokkal fontosabbak a jelek idődiagramjai, mivel itt ténylegesen meggyőződhetünk a zavar elnyomásáról.



[6. ábra] A bemenet időfüggvénye



[7. ábra] A kimenet időfüggvénye

Az ábrákon szabad szemmel is jól látható a különbség, hogy kezdetben csak a zaj jelenik meg a diagramon (6.ábra), de a szűrés után körülbelül 200 ms múlva teljesen elnyomjuk a zajt (7.ábra).

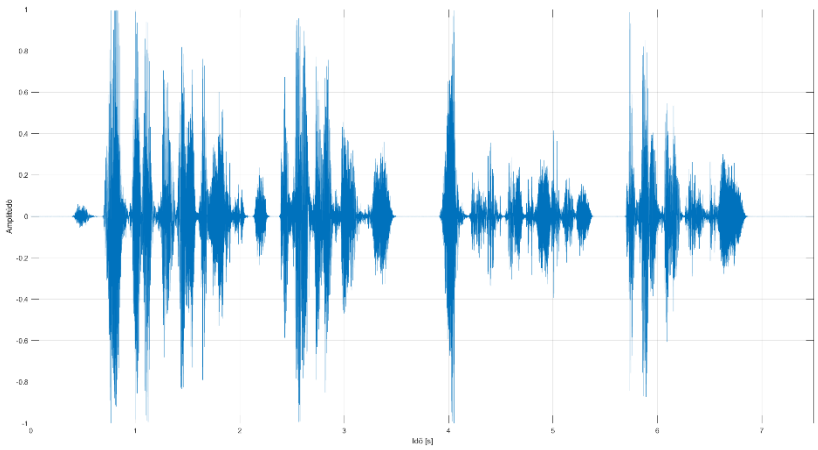
A felhasznált kód a 6.3-es fejezetben található.

4.3 Adaptív vonaljavító szélessávú hiba esetén (MATLAB)

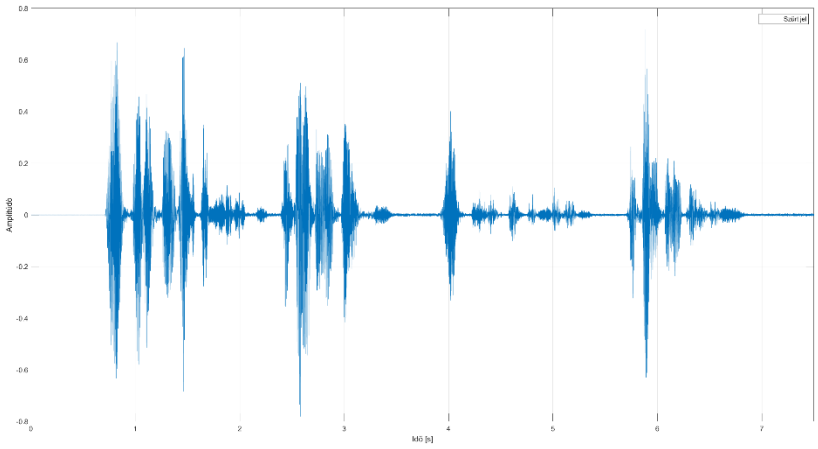
Ezt követően az adaptív vonaljavítót szélessávú additív zaj esetén vizsgáltam meg. A zavarására fehérzajt alkalmaztam. A megvalósítás során 400 db szűrőegyüttható használtam, a bátorsági tényezőnek $2{,}5*10^{-4}$-es érteket, a késleltetésnek pedig 200-as értéket adtam.

Az alábbi esetben sem lenne célszerű a jelek Bode-diagramját vizsgálni, mert most a zajt nem sikerült teljesen elnyomni, csupán csillapítani. A szűrt jelen látható, hogy a hullámosság valamelyest csökkent, ám nem sikerült teljesen elnyomni. Ezt további hangolással még javítani lehetne.

A felhasznált kód a 6.4-es fejezetben található.



[8. ábra] A bemenet időfüggvénye



[9. ábra] A kimenet időfüggvénye

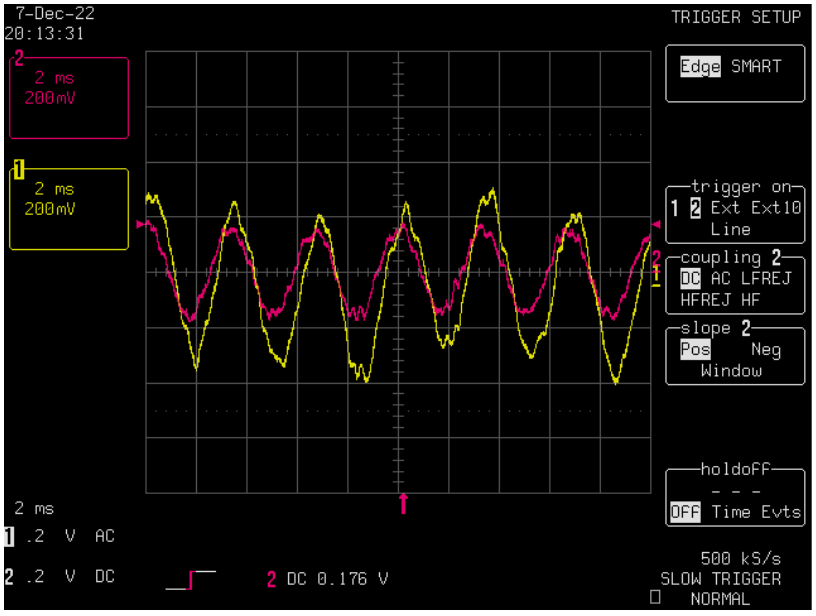
4.4 Adaptív vonaljavitó megvalósítása (DSP)

A tesztelés végeztével elkezdhettem megvalósítani az adaptív vonaljavítót a digitális jelfeldolgozó kártyán (DSP). A kártya pontos típusszáma ADSP-21364, mely rendelkezik lebegőpontos számábrázolással, így kiváló sebességgel képes számolni tört számokat is.

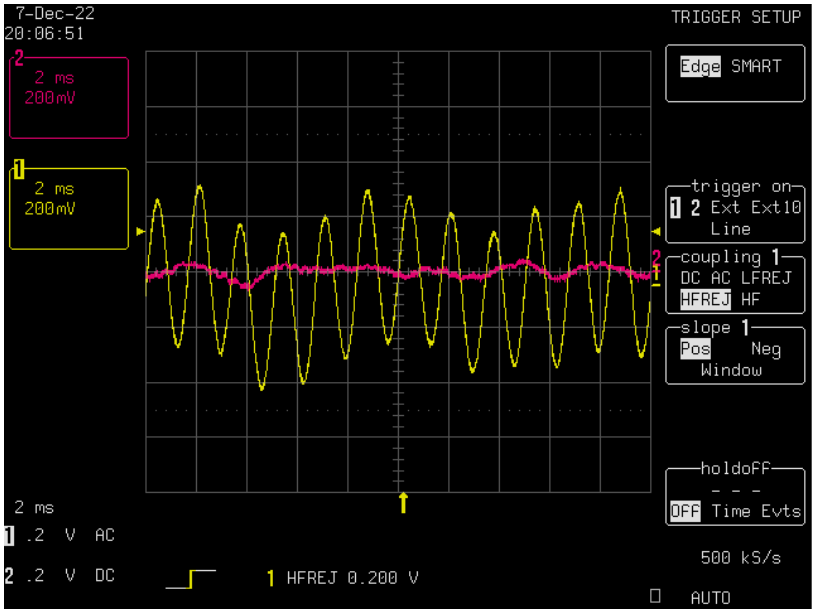
Fejlesztőkörnyezetnek pedig a Visual DSP++ v4.5-öt használtam. A hasznos jelhez additív módon csatoltam szinuszt, mint keskenysávú zajt és fehérzajt, mint szélessávú zajt. A programkód forrása http://home.mit.bme.hu/~orosz/temalabor/ linken, a" FIR temalabor.ZIP" file-ban található, mely még kiegészítésre szorult a feladat szempontjából. A kiegészítés a 6.5-ös fejezetben található.

Az adaptív vonaljavítót először keskenysávú zaj esetén teszteltem. 200 db szűrőegyütthatót, $2{,}5*10^{-5}$-es bátorsági tényezőt és 1-es késleltetést használtam. A szinuszos zavarás frekvenciáját folyamatosan változtattam 100 Hz és 4 kHz között a labor függvénygenerátorának segítségével, hogy megvizsgálhassam a vonaljavitó beállását, majd a kapott jelet oszcilloszkópon ábrázoltam és hangszórón meghallgattam.

Az alábbi képeken a sárga színezésű volt a bemeneti jelünk, a rózsaszín színezésű pedig a kimenő jelünk.



[10. ábra] A jelünk beállás előtt

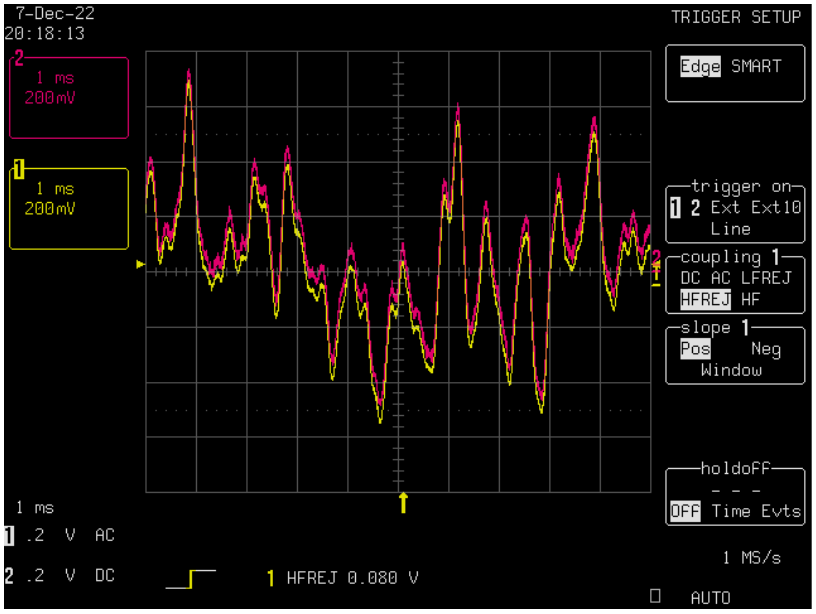


[10. ábra] A jelünk beállás után

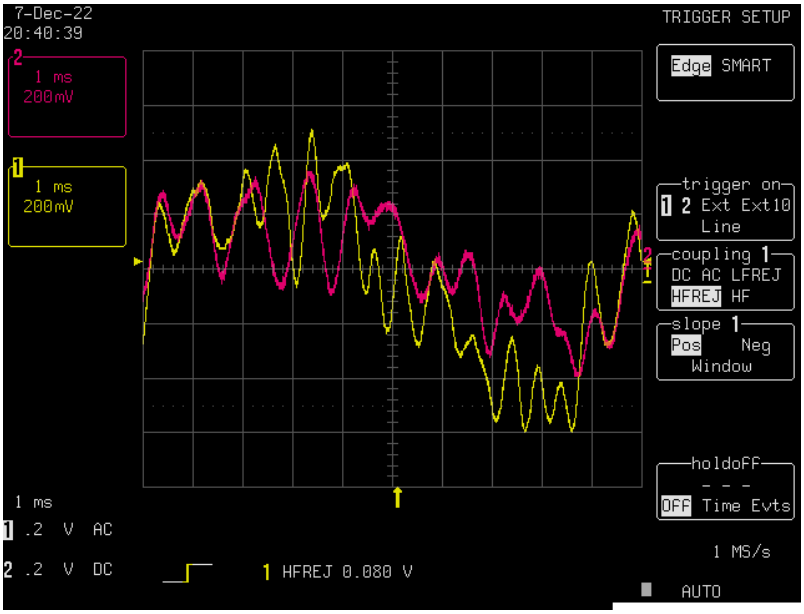
A vonaljavító viszonylag hamar el is nyomta az őt terhelő szinuszos zavarásokat, és melyest „megtanulta” azt a frekvencián az elnyomást, onnantól kezdve a beállás ideje jelentősen csökkent.

Ezt követően megvizsgáltam a vonaljavítót szélessávú zavarás esetén is. A zajt a laborban jelenlévő zajgenerátor segítségével állítottam elő, majd csatlakoztattam a DSP-re. Oszcilloszkóp segítségével ismét megvizsgáltam a jelet és meghallgattam a jelenlévő hangszórókkal.

Az alábbi képeken a sárga színezésű volt a bemeneti jelünk, a rózsaszín színezésű pedig a kimenő jelünk.



[11. ábra] A jelünk szűrés elött



[11. ábra] A jelünk szűrés után

A vonaljavítónak szüksége volt némi időre amíg beállt (~10 sec), de a beállást követően sem tudta megfelelő mértékben csillapítani a zajkomponenst. A megfelelő működéshez még további hangolásra van szükség.

5 Összefoglalás

A féléves munkám célkitűzése a további önálló munkára való felkészülés és a megfelelő gyakorlat elsajátítása a digitális jelfeldolgozás témakörében volt. A munkám során megismerkedtem az LMS algoritmussal és az Adaptív vonaljavítóval. Az LMS algoritmus

megértése létszükséges volt a továbbhaladáshoz, azt a MATLAB keretein belül is teszteltem, és megvalósítottam vele egy sávzáró szűrőt. Ezt követően már elkezdhettem foglalkozni az adaptív vonaljavítóval, melyet aztán egy digitális jelfeldolgozó kártyán is megvalósítottam. A teszteléshez a laboratóriumban lévő ADSP-21364-es kártyát, a jelen lévő oszcilloszkópot, zajgenerátort és hangszórókat használtam. A feladatom megoldását sikeresnek tekintem; rengeteg tapasztalattal és tudással lettem gazdagabb és a vonaljavítót is sikerült megvalósítanom, azonban további célkitűzésként szerepel a vonaljavító továbbfejlesztése szélessávú zaj esetén, illetve szükség lenne a megvalósított C nyelvű algoritmus átírására assembly nyelvre a gyorsabb működés érdekében.

6 Programkódok

6.1 LMS tesztelése (MATLAB)

```
clear all
close all
clc
%% Szűrő
[y, fs] = audioread('C:\Users\Yebume\Documents\MATLAB\Tema Lab\LMS teszt\valami.wav');
N = 2^17;
spektrum = fft(y, N);

f_noise = 440; % Hz
a = 10;
b = 50;
Num = 1000;     % szűrő e ható

h = firpm(Num, [0 f_noise-b f_noise-a f_noise+a f_noise+b fs/2]/(fs/2), [1 0 0 0 0 1], [1 10 1]);
y_filt = conv(y, h);
%soundsc(y_filt*1125, fs)

Signal = y;
%% LMS

M = 1001;            % szűrő ehatók száma
mu = 1/1001;         % bátorsági tényező
L = length(y);       % gerj hossza
w = zeros(M, 1);     % szűrő e.hatók
s = randn(1,L);      % bemeneti jel
%d = y_filt;         % kimeneti jel
d = filter(h,1,s);
x = zeros(M,1);      % forgó vekt, késleltetők imeneti értékét tartalmazza
e = zeros(1,L);      % hiba vektor
%w = h';

    for k = 1:L      % k: futó időváltózó
        x(1)   = s(k);
        y      = w' * x;
        e(k)   = d(k)  - y;
        w      = w  + mu*e(k)*x;
        x(2:M) = x(1:(M-1));
    end

filtfilt = filter(w,1,Signal);

%% Vizsgálat
f = 0 : fs/N : (N-1)*(fs/N);
figure('Name','Az eredeti jel');
legend('Eredeti jel')
xlabel('Frekvencia [Hz]');
xlim([0 6000])
ylabel('Amplitudó [dB]');
ylim([0.0001 1000]);
grid on
semilogy(f, abs(fft(y, N)));

figure('Name','FIR szűrő kimenete');
legend('FIR szűrő kimenete')
```

```
xlabel('Frekvencia [Hz]');
xlim([0 6000])
ylabel('Amplitudó [dB]');
ylim([0.0001 1000]);
grid on
semilogy(f, abs(fft(y_filt, N)));

figure('Name','LMS szűrő kimenete');
legend('LMS által megvalósított szűrő kimenete')
xlabel('Frekvencia [Hz]');
xlim([0 6000])
ylabel('Amplitudó [dB]');
ylim([0.0001 1000]);
grid on
semilogy(f, abs(fft(filtfilt, N)));

semilogy(abs(e))


%soundsc(y, fs)
%soundsc(y_filt, fs)
%soundsc(filtfilt, fs)
```

6.2 Továbbiakban használt LMS algoritmus (MATLAB)

```
function [e,w,yk] = myLMS(M, mu, s, d)
%OUTPUT: [e,w]
%M                  % szűrő együtthatók száma
%mu                 % bátorsági tényező ~(1/M)
%s                  % reprezentálni kívánt rendszer bemenete
%d                  % reprezentálni kívánt rendszer kimenete


L = length(s);      % gerj hossza
w = zeros(M, 1);    % szűrő együtthatók
x = zeros(M,1);     % forgó vektor, késleltetők értékét tartalmazza
e = zeros(1,L);     % hiba vektor
% y                 % szűrő által becsült kimenet
yk = zeros(1,L);

    for k = 1:L     % k: futó IDŐváltózó
        x(1)   = s(k);
        y      = w' * x;
        e(k)   = d(k)  - y;
        w      = w  + mu*e(k)*x;
        x(2:M) = x(1:(M-1));
        yk(k)  = y;
    end
end
```

6.3 Adaptív vonaljavító tesztelése keskenysávú zaj esetén (MATLAB)

```
clear all
close all
clc
```

```
%% Audio file
[Signal, fs] = audioread('valami.wav');
N = 2^17;
spektrum = fft(Signal, N);
L = length(Signal);

% késleltetett jel
K = 16;
Signal_delayed = [zeros(K, 1); Signal(1:end-K)];

%% LMS
N_filt = 1000;
[e, w] = myLMS(N_filt, .01/N_filt, Signal, Signal_delayed);
%soundsc(Signal, fs);
soundsc(e, fs);
cut = [zeros(1, 300), e(300:end)];
%soundsc(cut, fs);

f = 0 : fs/N : (N-1)*(fs/N);
figure('Name','Eredeti jel');
legend('Eredeti jel')
xlabel('Frekvencia [Hz]');
xlim([0 6000])
ylabel('Amplitudó [dB]');
ylim([0.0001 1000]);
grid on
semilogy(f, abs(spektrum));


figure('Name','Szűrt jel');
legend('Szűrt jel')
xlabel('Frekvencia [Hz]');
xlim([0 6000])
ylabel('Amplitudó [dB]');
ylim([0.0001 1000]);
grid on
semilogy(f, abs(fft(e, N)));

figure('Name','Szűrt jel időtartományban');
legend('Szűrt jel')
grid on
xlabel('Idő [s]');
ylabel('Amplitúdó');
xlim([0 9])
plot(t, e)
```

6.4 Adaptív vonaljavító tesztelése szélessávú zaj esetén (MATLAB)

```
clear all
close all
clc
%% Audio file
[Signal, fs] = audioread('WHY.wav');
N = 2^17;
spektrum = fft(Signal, N);
L = length(Signal);
```

```
Noise = randn(L, 1)/50;

SN = Signal+Noise;

% késleltetett jel
K = 200;
SN_delayed = [zeros(K, 1); SN(1:end-K)];

%% LMS
N_filt = 400;
[e, w, yk] = myLMS(N_filt, .1/N_filt, SN_delayed, SN);
%soundsc(SN, fs);
%soundsc(yk, fs);


f = 0 : fs/N : (N-1)*(fs/N);
figure('Name','Eredeti jel');
legend('Eredeti jel')
xlabel('Frekvencia [Hz]');
xlim([0 20000])
ylabel('Amplitudó [dB]');
grid on
semilogy(f, abs(spektrum));


figure('Name','Szűrt jel');
legend('Szűrt jel')
xlabel('Frekvencia [Hz]');
xlim([0 20000])
ylabel('Amplitudó [dB]');
ylim([0.0001 1000]);
grid on
semilogy(f, abs(fft(yk,N)));


t = 0 : 1/fs : 1/fs*(332187-1);
figure('Name','Eredeti jel időtartományban');
legend('Eredeti jel')
grid on
xlabel('Idő [s]');
ylabel('Amplitúdó');
xlim([0 7.5])
plot(t, Signal)

figure(3)
figure('Name','Szűrt jel időtartományban');
legend('Szűrt jel')
grid on
xlabel('Idő [s]');
ylabel('Amplitúdó');
xlim([0 7.5])
plot(t, yk)
```

6.5 Digitális jelfeldolgozó kártyán megvalósított kód (DSP részlet)

```
float SN[N+1];
float error;
float mu;
float merror;


// LMS

        //mu = 0.005/N;         //keskeny
        //mu = 0.0001/N;                //széles
        mu = 0.00008/N; //egész jó
//      mu = 0.000099/N;
        ix = i;
        int K = 21;                     //25 max
                                                //17 jó
                                                //21 eddig a legjobb
        error =  SN[(ix)%N] - out_1;
        merror =mu*error;


        for (j=0; j<N; j++)
        {
                coefs_1[j] = coefs_1[j] + merror*SN[(ix+K)%N];
                ix++;
        }
```

7 Hivatkozások

- Balogh Tibor: Adaptív szűrők vizsgálata. Mérési útmutató. Budapesti Műszaki és Gazdaságtudományi Egyetem.
- http://home.mit.bme.hu/~orosz/temalabor/FIR temalabor.ZIP
- Balogh Tibor: (2008) Adaptív jelfeldolgozás – hallgatói mérés tervezése. Diplomaterv. Budapesti Műszaki és Gazdaságtudományi Egyetem.