TREBALL DE FI DE CARRERA

TÍTOL DEL TFC: Aplicació d'algoritmes iteratius per a la cancel·lació d'interferències en senyals d'àudio

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Sistemes de Telecomunicació

AUTOR: Xavier Damunt Masip

DIRECTOR: Antonio Pascual Iserte

DATA: 30 de juny de 2006

Títol: Aplicació d'algoritmes iteratius per a la cancel·lació d'interferències en senyals d'àudio

Autor: Xavier Damunt Masip

Director: Antonio Pascual Iserte

Data: 30 de juny de 2006

Resum

El principal objectiu d'aquest treball de fi de carrera és estudiar el comportament del filtratge adaptatiu per a la cancel·lació d'una interferència en senyals d'àudio. Per aquest estudi hem programat amb Matlab diversos algoritmes adaptatius FIR que, tot i les diferències que hi ha entre ells amb la manera d'aconseguir l'objectiu fixat, tenen una estructura similar.

Tal i com veurem a l'inici de cada capítol, la principal premissa per al funcionament d'aquests algoritmes és obtenir d'alguna manera el so interferent separat del so útil. En programació no ha suposat cap dificultat ja que tots els escenaris eren muntats per nosaltres, amb la qual cosa sempre teníem el so interferent. Ara bé, en un cas real pot ser més complicat, ja que és necessari captar d'alguna forma el so interferent des de dues fonts. Una font és aquella que després la voldrem eliminar i que per lògica estarà acompanyada d'un senyal que volem recuperar sense interferències; i l'altre font ha de ser un lloc on només captem el volem eliminar.

Si tenim això, *només* cal aplicar l'algoritme. De manera resumida, podem dir que el filtre de l'algoritme s'adaptarà per tal d'aconseguir que el so interferent d'una font acabi sent igual a la de l'altra, per així poder-les cancel·lar per complet mitjançant una simple resta. Això seria trivial si de les dues fonts extraguéssim exactament el mateix so interferent, però això pràcticament mai serà així. Problemes com retards, variacions en el canal que provoquin variacions en la font, soroll gaussià, i un llarg etcètera ens els podríem trobar en un entorn real i per tant seran tinguts en compte en els diferents capítols.

Per la cerca de la solució a aquests problemes utilitzarem diferents algoritmes adaptatius: LMS, NLMS, DSD, LRS, GRS i RLS. L'explicació de cada un d'ells es troba al primer capítol, així com una primera presa de contacte amb cada algoritme amb un escenari senzill on s'ha d'eliminar un soroll aleatori.

L'estructura dels capítols respon a la presentació d'escenaris on es posa a prova vàries de les situacions abans esmentades, seguit d'unes conclusions.

Per últim, es fa una breu introducció al filtratge IIR, presentant un escenari que a priori un filtre FIR no podria solucionar.

Title: Application of iterative algorithms for interference cancellation in audio signals

Author: Xavier Damunt Masip

Director: Antonio Pascual Iserte

Date: June, 30th 2006

Overview

The main objective of the TFC is to study adaptive filtering techniques to cancel signal interferences. Some FIR adaptive algorithms have been programmed using Matlab. These algorithms have differences between themselves but they have a similar structure.

As we can see at the beginning of every chapter, if we want to use these algorithms we will achieve the signal interferences without the signal that we want to hear. Obviously, performing that is not difficult because the scenarios were designed by us and we could program anything we want. But in a real case, it could be more complicated, because is necessary to have the signal interference from two different sources. One of these sources is only the interference and the other one is the interference and something that we want to hear without interferences.

If all we said before is possible, we *only* have to use the algorithm. Concisely, the algorithm's filter will adapt so; the interference of one source will be the same as the other source. Then we will subtract one source from the other one. The result will be the signal that we want to hear without interferences. This would be easy if the two sources would be exactly from the same place, but that happens rarely. Problems like delays, the variability of the channel, the gaussian noise, etc, will always be present in a real scenario so we will evaluate them in different chapters.

To solve these problems we will use different adaptive algorithms: LMS, NLMS, DSD, LRS, GRS and RLS. An explanation of every one of them can be found in the first chapter, and also a first test with a simple scenario where the algorithms have to remove a random noise.

The structure of the following chapters is the presentation and test of scenarios with the situations that we have talked about before. At the end of the chapter we present some conclusions.

Finally, we introduce the IIR filtering with and scenario that a priori the FIR's algorithms cannot solve.

ÍNDEX

INTRODUCCIÓ

El principal objectiu d'aquest treball de fi de carrera és estudiar el comportament del filtratge adaptatiu per a la cancel·lació d'un senyal interferent. Hem programat amb Matlab diversos filtres adaptatius FIR: el LMS (Least Mean Square), el NLMS (Normalized Least Mean Square), el DSD (Differencial Steppest Descent), el LRS (Linear Random Search), el GRS (Guided Random Search) i el RLS (Recursive Least Squares). A tots aquests algoritmes els hem posat a prova tenint en compte la majoria de problemes que podem tenir en un cas real, per tal d'avaluar-los i treure'n conclusions.

Cada capítol és un escenari diferent on alguns d'aquests problemes a que fèiem referència seran usats per veure quins algoritmes els suporten millor i a partir d'això poder comparar-los i extreure'n conclusions. El primer capítol però, abans de sotmetre als algoritmes a les proves, fem una explicació de cada un d'ells per poder entendre'n el funcionament. De manera molt compacta, podem dir que per l'entrada del filtre ha d'entrar quelcom similar a allò que volem eliminar; és a dir, la interferència. Gràcies a aquest filtratge aconseguirem que allò que era similar sigui, idealment, exacte i per tant, si restem el que està contaminat (que ho tenim a la referència) a el que tenim a la sortida del filtre, eliminem la part no desitjada que apareix juntament amb el senyal útil.

Un cop fet això, utilitzem un escenari de mostres aleatòries per tenir un primer contacte amb els algoritmes. A partir d'aquesta prova, podem treure unes primeres conclusions, que no són definitives però que ja serveixen per posicionar una mica els algoritmes: Hi ha algoritmes de primera i segona categoria. Comparant la velocitat de convergència, la de simulació i el desajust, per sobre de tots els altres hi ha el LMS i el NLMS, seguit per el RLS, el LRS i el GRS. A molta distància queda el DSD, que és, el pitjor amb diferència per a l'aplicació que s'està estudiant.

En els següents capítols augmentem la dificultat que han de superar els filtres per aconseguir eliminar el soroll interferent. En el segon tenim una cançó on hi ha dos tons que no ens la deixa sentir bé. Mitjançant el filtratge dels algoritmes iteratius aconseguim eliminar els tons amb pocs coeficients, punt clau si volem que el temps de simulació sigui baix. Evidentment es necessiten pocs coeficients perquè dos tons (o sigui, sinus) en l'espectre freqüencial són dues deltes que poden ser filtrades fàcilment sense cap restricció més. Per això, en el tercer capítol hem repetit l'experiència però introduint soroll a l'entrada. Això fa que ara, a les freqüències on no hi ha les deltes hagi d'intentar obligatòriament no deixar passar res, perquè en cas contrari deixa passar soroll i no podem sentir la cançó. Per això en aquest escenari és necessari dotar als algoritmes de més coeficients, ja que així a la vegada els dotem de més graus de llibertat per poder aconseguir una resposta freqüencial adequada a cada moment depenent de les freqüències de les interferències i el soroll. Aquestes proves reafirmen en part les conclusions del primer capítol: LMS i NLMS estan per sobre els altres i DSD i GRS per sota.

Al quart capítol tractem de programar un escenari que s'adeqüi més a quelcom que ens podríem trobar a la realitat. Molts cops hem vist com un periodista que

està a un circuit automobilístic deixem de sentir-lo quan passa un cotxe a gran velocitat a prop d'ell. Hem simulat un escenari molt similar, on a més hem introduït retards relatius entre el senyal útil contaminat i la mostra d'interferència, variació del canal i diferent amplitud; perquè en un entorn real on necessitaríem dos micròfons tot això ens ho trobaríem. Després de provar tots els algoritmes a les conclusions trobem que tots ells poden, amb major o menor fortuna, eliminar el soroll dels cotxes amb pocs coeficients.

El cinquè capítol és un nou pas en la dificultat. Fins ara, un cop els filtres estan adaptats ja no es desadapten, donat que no hi ha un canvi brusc a l'entrada. Amb aquest nou escenari forcem que això passi amb salts de freqüència. A l'entrada, hi entrem durant tot el temps un conjunt de deltes que coincideixen amb les freqüències que els salts de la referència poden tenir. Depenent de la freqüència dels salts de freqüència i de la distància entre les freqüències susceptibles de fer els canvis trobem més o menys dificultat per als algoritmes. En aquest cas, el LMS segueix sent dels millors juntament amb el LRS, seguit pel NLMS i el RLS.

L'últim capítol dedicat als filtres adaptatius FIR segueix amb l'idea de l'anterior capítol però si abans els canvis eren bruscs i cada cert temps, ara el canvi és continu i suau. Això obliga al filtre a adaptar-se constantment, i depenent de la rapidesa amb que variï el filtre podrà o no seguir aquests canvis. En aquest cas, el RLS és el millor, seguit del NLMS i el LMS.

Per últim, es fa una breu introducció al filtratge IIR, presentant un escenari que a priori un FIR no podria solucionar. Primer programem l'algoritme anomenat Feintuch's IIR LMS, que és una aproximació vàlida al IIR LMS, que més tard també programem. Al capítol en el qual parlem del filtratge IIR veurem que tant l'un com l'altre algoritme milloren a un filtratge FIR per a aquest escenari, però no són uns resultats especialment bons. A part, també veurem que el IIR LMS no millora a l'altre algoritme IIR, a pesar que el aquest és una versió aproximada del primer.

CAPÍTOL 1. PRESENTACIÓ DELS ALGORITMES

Primerament explicarem l'estructura que tot filtre té. Els filtres consten d'una entrada on es troba la font del senyal interferent, la sortida que és l'entrada un cop l'entrada ha estat filtrada i la referència on hi ha la font de senyal útil contaminat. De la resta de la referència i la sortida en surt l'error, que a la vegada serveix per realimentar el filtre donant-li informació de com ha de filtrar l'entrada. Amb aquest serà l'esquema bàsic de tots els capítols.

Per provar els algoritmes sense tenir res de l'explicat abans, ens crearem un vector d'entrada de N nombres aleatoris. Aquest vector el convolucionarem amb un filtre de tres coeficients coneguts. El resultat d'aquesta convolució serà la referència per al nostre filtre adaptatiu de tres coeficients també. Així, el nostre filtre haurà d'arribar a ser igual als 3 coeficients usats en la convolució. En el moment que els dos filtres siguin iguals, l'error serà zero.

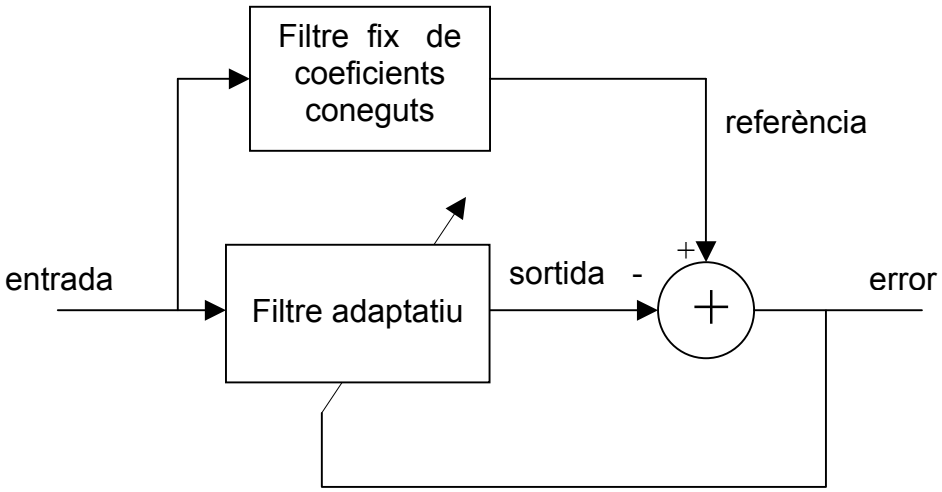


Fig. 1.1 Esquema bàsic dels algoritmes amb senyal d'entrada aleatori

El senyal d'entrada serà de mil mostres (excepte pels algoritmes DSD i GRS que necessiten una entrada més llarga) i aleatori, però serà el mateix sempre per a tots els assaigs. El filtre fix també serà el mateix sempre (filtre de tres coeficients: 1, 0'54, -0'2) i el filtre adaptatiu començarà des de tres zeros (excepte el DSD, que no pot començar des de 0).

Els resultats que es mostraran seran la convergència del valor absolut de l'error a zero amb totes les variables susceptibles de ser canviades per nosaltres amb el valor òptim, fixant-nos en la rapidesa que hi arriba i en el desajust quan el filtre ja ha convergit provocat generalment pel pas d'adaptació. Si s'escau, es mostraran també altres gràfiques amb els valors no òptims. Les gràfiques d'evolució de la convergència depenent d'aquests valors els trobareu a l'annex.

D'altra banda, durant aquest punt s'esmentarà sovint el valor òptim de diverses variables totes amb un significat diferent dins de cada algoritme. Aquest valor

òptim es refereix en les condicions abans esmentades, ja que en posteriors punts podrem veure com al canviar les condicions, varien també les constants en el seu punt òptim.

1.1. Algoritme LMS (Least Mean Square)

1.1.1. Explicació

Aquest algoritme utilitza el mètode del gradient. Això significa que si tenim les corbes de l'error que depenen dels coeficients del filtre, podem calcular el gradient per trobar la direcció i el sentit de l'error mínim, que coincideix amb la del gradient però amb el signe canviat. Per entendre això, podem pensar en el cas més senzill, que és el d'un filtre adaptatiu d'un coeficient. D'aquesta manera, tenim en una gràfica de dues dimensions una corba que defineix la relació entre la potència de l'error i el valor de l'únic coeficient del filtre. Cal pensar que, per a cada iteració estarem més a prop de l'error mínim. Doncs bé, si calculem el gradient d'aquesta corba, la direcció del gradient ens assenyalarà a l'error mínim, amb el signe canviat.

El procediment és relativament senzill. Cal recórrer el vector de l'entrada. Per a cada posició d'aquest vector es treballarà amb la mostra en qüestió i les N anteriors, on N és el número de coeficients que té el filtre. Amb les mostres seleccionades es calcula l'error restant de la referència la sortida (que és la multiplicació dels coeficients amb les mostres de l'entrada abans esmentades).

$$E_k = Ref_k - h_k * x_k \qquad \textbf{(1.1)}$$

On E és l'error, Ref és la referència, h el filtre i x l'entrada. La multiplicació del filtre per l'entrada és la sortida del filtre. Ara ja només ens queda aplicar la regla d'aprenentatge.

$$h_{k+1} = h_k + \mu * x_k * E_k \qquad \textbf{(1.2)}$$

On μ s'anomena pas d'adaptació i és un escalar que fa avançar amb més o menys rapidesa l'algoritme fins a trobar el filtre desitjat. Una μ massa alta fa que l'algoritme faci els passos en la direcció adequada massa grans fins al punt que no arriba a convergir. Per contra, una μ massa petita farà que l'algoritme tardi més en convergir. El fet que utilitzem una estimació instantània del gradient fa que l'error final sigui major que l'òptim. Aquest augment de l'error també el controla el pas d'adaptació. Per a més informació, veure [1] i [2].

A partir de l'estructura bàsica de LMS, es poden fer retocs per tal de trobar algoritmes computacionalment més ràpids i que a la vegada no es perdi gaires prestacions. N'hem programat un que li hem dit LMS-2, i el trobareu a l'annex.

1.1.2. Convergència

El LMS és un dels algoritmes que convergeix més ràpidament sota les condicions abans esmentades. Amb el valor òptim de μ, en aquest cas 0'3, podem considerar que ha convergit aproximadament a la mostra 25 (figura 1.2).

Hem comprovat també que valors per sota de 0'005 i majors de 0'6 aquest algoritme deixa de funcionar. Per valors llunyans del valor òptim, tal com 0'01, l'empitjorament de l'algoritme es fa palès en termes de velocitat de convergència, com demostra la figura 1.3. En qualsevol cas, podem veure que el desajust pràcticament no existeix, ja que la fluctuació entorn de l'error mínim (el zero en aquest cas) és despreciable, ja que es mou pels -140 dB, encara que la μ sigui la convergent més gran possible.

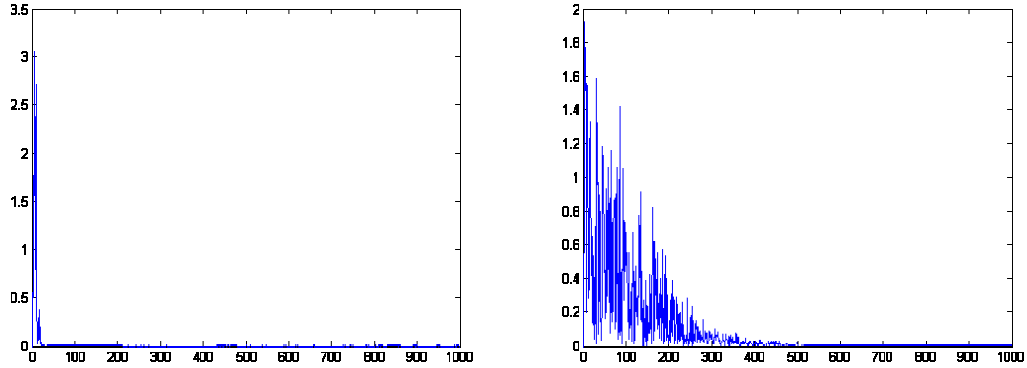


Fig. 1.2 i 1.3 Algoritme LMS amb μ = 0'3 (òptima) i μ = 0'01

1.2. Algoritme NLMS (Normalized Least Mean Square)

1.2.1. Explicació

Aquest algoritme parteix del LMS, però la única diferència és que en comptes de tenir un pas d'adaptació fix, a cada iteració el va calculant. Per fer-ho, calculem cada cop la multiplicació del vector d'entrada (de longitud igual al número de coeficients del filtre, igual que el LMS) per si mateix transposat. D'aquesta manera calculem la norma d'aquest vector al quadrat, i l'anem guardant en una variable P_x. Així doncs, per a cada iteració tenim un valor de P_x, però també tenim la memòria de tots els P_x calculats anteriorment. La manera més eficient d'actualitzar el valor de P_x, és fer una mitjana ponderada entre la norma del vector instantània i la memòria de la pròpia norma, que ve donada pel factor β de la següent manera:

$$P_x(n) = (1 - \beta)\, P_x(n - 1) + \beta\, ||x(n)||^2 \qquad \textbf{(1.3)}$$

L'elecció d'aquest factor dependrà principalment de la naturalesa de les dades de l'entrada. Per exemple, per dades de naturalesa canviant caldrà donar-li més pes a la norma instantània que no pas a la memòria. Un cop tenim calculat P_x, només cal calcular la μ, que no és més que $2*\alpha / P_x$ (amb aquest càlcul es pot demostrar que sempre hi ha convergència). α és un valor a escollir entre 0 i 1 i que ens dóna un compromís entre la velocitat d'adaptació i el nivell de desajust (el desajust és la fluctuació que tenim a l'error quan hi ha una μ prou gran de manera que fa que l'error oscil·li proper al zero, però sense arribar a divergir). Amb el càlcul de la μ a cada iteració s'aconsegueix una convergència més ràpida i segura, ja que no hi ha perill de tenir una μ massa gran.

1.2.2. Convergència

Donat que en aquest algoritme el pas d'adaptació es recalcula a cada iteració, no podem estudiar la convergència del NLMS a partir d'aquesta variable; si no que ho hem de fer a partir dels valors que conformen la μ inicial i indiquen en quin grau i atenent a quins valors ha de variar, α i β.

Per mostrar com afecta la variació d'aquests valors, en deixarem un de fix. Primerament deixarem β fixa a 0'6, un valor que s'ha comprovat que no és extrem i per tant no invalidarà els resultats. De fet, just el contrari, ja que, donat que β pot estar compresa entre els valors 0 i 1, 0'6 és un valor proper al valor mig, però que dóna una mica més de pes a la memòria de la norma del vector que a la norma instantània del vector. Més tard, quan deixem α fixada, donarem més detalls del perquè de l'elecció d'aquest valor.

Observem que per valors propers a 1 la velocitat de convergència és menor degut al fort desajust que trobem. Amb el valor òptim obtenim la figura 1.4, on es pot veure que la convergència és pràcticament instantània. Si la comparem amb la figura 1.2, que és la convergència de l'algoritme LMS amb el seu valor òptim, podem veure que és pràcticament igual.

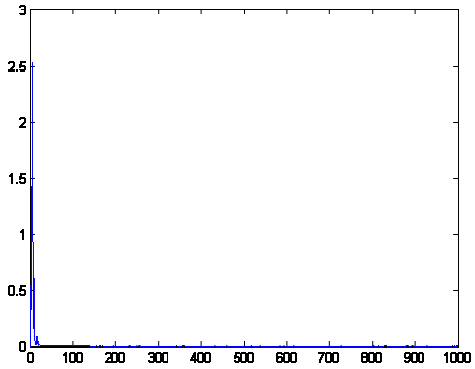


Fig. 1.4 Algoritme NLMS amb α = 0'3 (òptima) i β fixada a 0'6

Amb α fixada al valor òptim, la rapidesa en la convergència del NLMS pràcticament no varia encara que variem la β. El perquè d'això, i de les respostes que hem deixat pendents en els anteriors paràgrafs, és la naturalesa de l'entrada que tenim per experimentar amb els algoritmes. Com que β actua sobre què se li dóna més pes, si a la memòria o al càlcul instantani, i la naturalesa de la nostra entrada no varia al llarg del temps, tan se val a què se li dóna més pes. Per això quan fixem β no és rellevant amb quin valor ho fem; i també el fet de tenir l'entrada que tenim dificulta que podem veure la millora qualitativa de l'algoritme NLMS vers el LMS.

1.3. Algoritme DSD (Differencial Steepest Descent)

1.3.1. Explicació

És un altre algoritme que, com el LMS, es basa en el càlcul del gradient. La diferència rau en que aquest l'estima independentment per a cada coeficient del filtre, que equival a fer derivades parcials per a trobar el gradient. Per fer-ho, s'agafa cada coeficient en l'instant actual i se li suma i posteriorment se li resta una pertorbació aleatòria. A partir d'aquest nou coeficient es calcula l'error talment com al LMS. Amb això podem calcular el gradient (1.4) operant amb els errors del càlcul del coeficient més una pertorbació ($E_{1,k}$) i menys la mateixa pertorbació ($E_{2.k}$).Llavors ja només cal seguir la regla d'aprenentatge (1.5).

$$\nabla = \frac{E_{1,k} - E_{2,k}}{2^* \delta} \qquad \textbf{(1.4)}$$

$$h_{k+1} = h_k - \frac{\mu * \nabla_k}{2} \qquad \textbf{(1.5)}$$

On ∇ és el gradient i δ és la pertorbació, que sempre serà proporcional al valor del coeficient. També seran proporcionals els coeficients inicials del filtre, que s'inicialitzaran aleatoris però amb relació entre la potència del senyal d'entrada i la de la referència. Hem de dir però que després de fer les comprovacions pertinents, hem vist que fent això no es millora d'una manera evident respecte una inicialització aleatòria pura.

1.3.2. Convergència

Com ara veurem, l'algoritme DSD és el més lent convergint, i a sobre té força problemes amb el desajust, ja que és particularment alt amb els passos d'adaptació superiors a 0'1.

Com sempre, amb una μ més alta tindrem més velocitat a canvi de més desajust. Així doncs, tot i que amb unes mateixes condicions inicials podem tenir resultats diferents a conseqüència de la inicialització aleatòria i de la pertorbació que utilitza, una resposta típica amb la μ que creiem òptima és la

de la figura 1.5. Aquest cas ha succeït amb unes pertorbacions aleatòries que han ajudat força, ja que si bé considerem que ha convergit a la mostra 250 aproximadament, en mitjana convergeix a la 400. Si intentem aconseguir més velocitat augmentant la μ veiem el que ens passa a la figura 1.6, un augment del desajust.

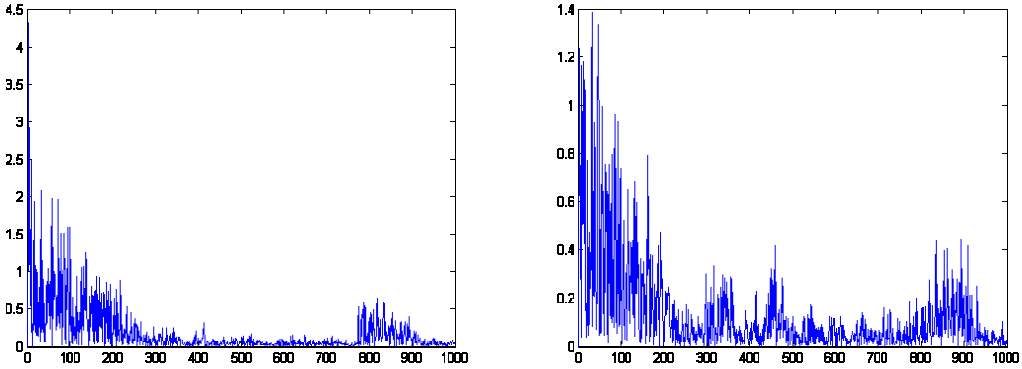


Fig. 1.5 i 1.6 Algoritme DSD amb μ = 0'08 (òptima) i amb μ = 0'1

1.4. Algoritme LRS (Linear Random Search)

1.4.1. Explicació

La idea bàsica d'aquest algoritme adaptatiu ja no és la cerca del gradient per a arribar a l'error mínim. El mètode és molt més senzill, ja que es tracta simplement de generar una pertorbació aleatòria que se suma als coeficients actuals del filtre adaptatiu, de la següent manera:

$$h_{proposat} = h_k + \mu * \delta_k \qquad \textbf{(1.6)}$$

Ara cal calcular l'error que es faria amb aquests coeficients proposats i comprovar que aquest error és més petit que l'error que faríem amb el filtre provinent de l'iteració anterior. Si el filtre proposat veiem que està en la bona direcció per tal de trobar l'error mínim, actualitzem el filtre i passem a la següent iteració.

En cas que no millori es poden fer dues coses. O passar a l'iteració següent sense fer res, esperant que la propera pertorbació sí que estigui en la direcció adequada per a arribar a convergir; o seguir utilitzant la pertorbació anterior per avançar cap a l'error mínim. Hem implementat els dos casos i com es podrà veure a continuació el segon cas, que li direm LRS-2, convergeix més ràpidament, com era d'esperar.

1.4.2. Convergència

Un cop més, ens trobem davant el compromís entre la rapidesa en la convergència i el desajust. Cal dir que en aquest algoritme el desajust es fa especialment palès, fins al punt que amb segons quins valors de μ és difícil saber en quin moment podem considerar que l'algoritme ha convergit. Així que quan decidim la μ òptima, cuidarem especialment l'aspecte del desajust.

Per decidir quin és aquest valor, hem anat provant i comprovant quina era la reacció de l'algoritme. Hem vist que per valors del pas d'adaptació més petits que 0'01 l'algoritme no convergeix, així com per valors majors que 0'07, on el desajust és tan gran que no podem considerar que l'algoritme convergeixi (a partir de 0'04 ja és massa gran, però es pot endevinar certa convergència). Així doncs, considerem que la μ òptima és de 0'03, ja que convergeix ràpidament i el desajust no és encara molt important. Ho podem veure a la figura 1.7.

Pel que fa al que hem denominat LRS-2, és una millora d'aquest últim algoritme però es comporta de manera diferent als altres, depenent de la μ que utilitzem en cada cas. Podríem dividir els possibles valor de μ en tres zones de diferent comportament. En un extrem, entre 0'005 i 0'03 trobem una zona on el LRS-2 podríem dir que es comporta de manera similar als altres algoritmes (figura 1.8): com més alta és la μ més ràpid convergeix però més desajust trobem.

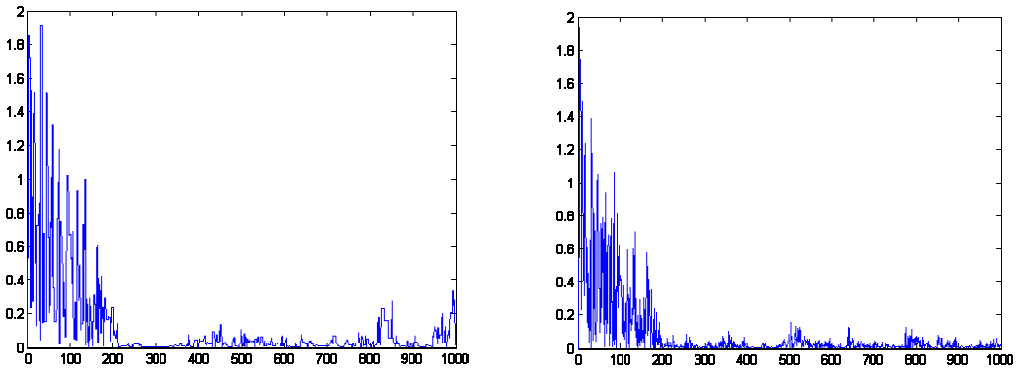


Fig. 1.7 i 1.8 Algoritme LRS amb μ = 0'03 (òptima) i LRS-2 amb μ = 0'02 (zona sub-òptima)

Per contra, a l'altre extrem, entre 0'1 i 0'4 (més amunt ja no convergeix), trobem que quan més augmentem la μ més ràpid convergeix però en cap cas veiem desajust (figura 1.9). Qualsevol d'aquestes dues zones comentades es donen per bones, amb preferència per l'última d'elles, ja que a part del fet que no hi ha desajust, en mitjana convergeix més ràpid. Diem en mitjana perquè, com que aquest algoritme genera una pertorbació aleatòria, cada assaig varia.

La zona que falta per explicar es troba entre les dues anteriors, i sembla ser una mescla entre aquestes dues. Ens poden passar tres coses, que

convergeixi de la manera que ho fa la primera zona, amb l'afegit de tenir un desajust molt alt; que convergeixi com a la segona zona, per tant funcionarà correctament; o que ho faci com a la primera i després com a la segona (figura 1.10). Aquesta zona no és recomanable, perquè tant pot ser que convergeixi com que no.

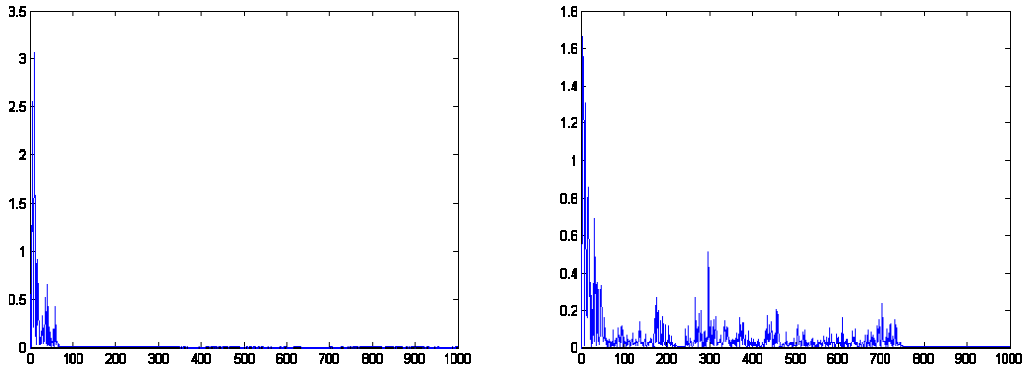


Fig. 1.9 i 1.10 Algoritme LRS-2 amb μ = 0'4 (μ òptima dins de la zona òptima) i μ = 0'06 (zona no recomanable)

1.5. Algoritme GRS (Guided Random Search)

1.5.1. Explicació

Podríem dir que l'algoritme GRS és una millora del LRS, ja que aprofita la idea de trobar una direcció correcta per aconseguir l'error mínim que s'utilitza en el LRS però, quan troba aquesta direcció la manté i suma a μ a cada iteració un factor β per avançar més ràpidament, fins que comprova que la direcció ja no és correcte. En aquest cas, crea una nova pertorbació a cada iteració fins a trobar-ne una que ens serveixi per seguir una nova direcció que ens faci anar fins l'error mínim.

1.5.2. Convergència

Igual que l'algoritme NLMS, no podem fer un estudi de la convergència de l'algoritme GRS vers la μ perquè aquesta pot variar a cada iteració, depenent de si el càlcul de l'error està en la direcció correcta, que en aquest cas la μ augmenta, o no ho està, que llavors la μ torna a tenir el valor inicial. Així doncs, haurem de fer-ho fixant-nos amb la μ inicial i amb la β, que és el valor amb què augmenta la μ cada iteració en que la direcció apunta cap a l'error mínim.

Començarem per deixar fixada la μ inicial. Donat que la β és un multiplicador de la μ inicial en el cas que estem en la direcció correcta, $\beta > 1$. En aquest cas i amb μ inicial fixada a 0'05, podem veure que el mínim el trobem al voltant de 2, tot i que la incertesa que crea el fet de treballar amb pertorbacions fa que el

marge d'error a l'hora de trobar el valor òptim sigui alt. També podem veure que hi ha un mínim relatiu al voltant de 15. Trobareu la gràfica a l'annex, com les altres que fan referència a l'evolució de la convergència depenent del pas d'adaptació. A continuació mostrem l'evolució de l'error quadràtic per l'algoritme GRS amb els valors òptims, en un dels casos que ha convergit amb certa celeritat.

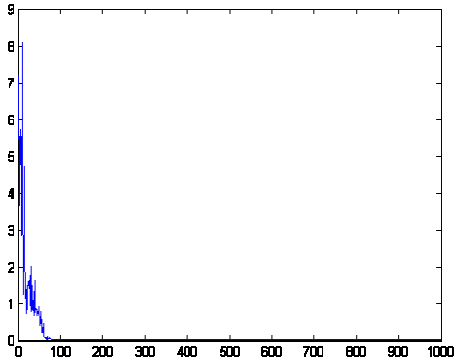


Fig. 1.11 Algoritme GRS amb μ inicial = 0'05 i β = 2

1.6. Algoritme RLS (Recursive Least Squares)

1.6.1. Explicació

Aquest algoritme utilitza un mètode diferenciat als anteriors que mereix una explicació prèvia.

Hi ha una manera de calcular el filtre òptim fàcilment (extret de [1]), el problema és que per calcular-lo necessita tenir tot el vector de dades i el de referència disponible, per tant el filtre òptim l'obtens quan tota l'entrada ha passat pel filtre, o sigui que és un càlcul *offline* (1.9). Consisteix en trobar la matriu de correlació de dades i la correlació creuada entre les dades i la referència, que són respectivament:

$$R = \frac{1}{M} * \sum_k x_k * x_k^T \qquad \textbf{(1.7)}$$

$$P = \frac{1}{M} * \sum_k x_k * Ref_k \qquad \textbf{(1.8)}$$

$$h_{òptim} = R^{-1} * P \qquad \textbf{(1.9)}$$

On M és el número de mostres. Doncs bé, l'algoritme RLS es basa en aquests càlculs, aprofitant-los per a aconseguir un filtre que s'adapti a mida que es van rebent les dades de l'entrada i la referència, i no al final. El procediment és, primerament, acumular un número relativament petit de mostres de l'entrada

comparada amb la longitud total del vector d'entrada per poder fer el càlcul de la part acumulada de la correlació de dades, R, tal i com s'ha explicat. Cal tenir en compte que el número de vectors a promitjar per calcular R ha de ser més gran que el número de coeficients del filtre. Com que l'ordre de la matriu depèn del nombre de coeficients del filtre, és possible tenir problemes depenent de la naturalesa de l'entrada i el nombre de coeficients que escollim, ja que podem arribar a tenir una matriu amb files linealment dependents i per tant no invertible.

Un cop tenim la inversa, es calcula l'error que tenim amb el procediment habitual (1.1), però la regla d'aprenentatge varia (1.10). La idea és que ara minimitzem un promitg d'errors fets anteriorment fins l'instant actual (1.11).

$$h_{k+1} = h_k + K_k * E_k \quad \textbf{(1.10)}$$

$$R_{k+1} = \beta * R_k + (1 - \beta) * x_k * x_k^T \quad \textbf{(1.11)}$$

$$K_k = \frac{(1-\beta) * R_k^{-1} * x_k}{\beta + (1-\beta) * (x_k^T * R_k^{-1} * x_k)} \quad \textbf{(1.12)}$$

On K_k és el vector de guany que es calcula a (1.12). D'aquesta manera actualitzem R^{-1} sense haver-la de calcular cada cop ja que és molt costós, podent així calcular (1.9) a cada iteració.

1.6.2. Convergència

Si fem l'estudi de la convergència depenent del número de mostres que s'utilitzen a l'inici per calcular (1.7) trobem que com més gran sigui el nombre de mostres emmagatzemades a l'inici de la recepció del vector x, millor serà el càlcul inicial de R^{-1}, així que cal pensar que seria un encert agafar més mostres M per a aquest càlcul. I de fet és així com podrem comprovar a l'apartat 2.6. No obstant, per una entrada aleatòria, el RLS disminueix la seva velocitat de convergència lleugerament a mida que augmentem el valor de M. Així doncs, ara el valor òptim és 3 mostres per un filtre adaptatiu de tres coeficients.

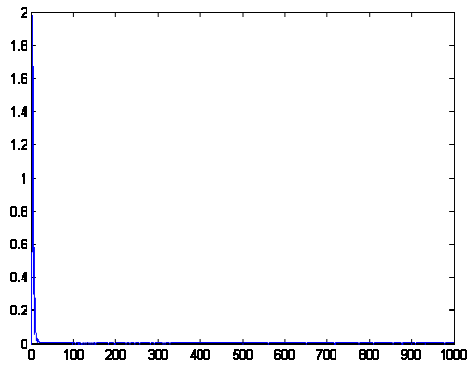


Fig. 1.12 Algoritme RLS amb M = 3 (òptim)

1.7. Primera comparació dels algoritmes

Un cop hem repassat tots els algoritmes, és el moment de fer una comparació per saber, amb les condicions que hem donat al principi, quins són els que s'adapten millor i per tant convergeixen més ràpidament, quins tenen menys desajust o quins es poden calcular més ràpidament. Per fer-ho, les simulacions de cada un dels algoritmes seran amb els seus valors òptims. Escollir-los, com s'ha vist anteriorment, sovint és un compromís entre dos factors que precisament volem avaluar; i per tant una bona elecció serà aquella que farà perdre prestacions a ambdós factors però sense perjudicar-ne un excessivament.

A la figura 1.13 compararem les velocitats de convergència. Es pot veure que hi ha grans diferències entre els més ràpids, NLMS, RLS, LMS i GRS, i el més lent, que és el DSD. A la figura 1.14 veiem la comparació del desajust un cop han convergit. Per una millor presentació, mostrem els dB en positiu, així que com més alt, menys desajust tindrà l'algoritme.

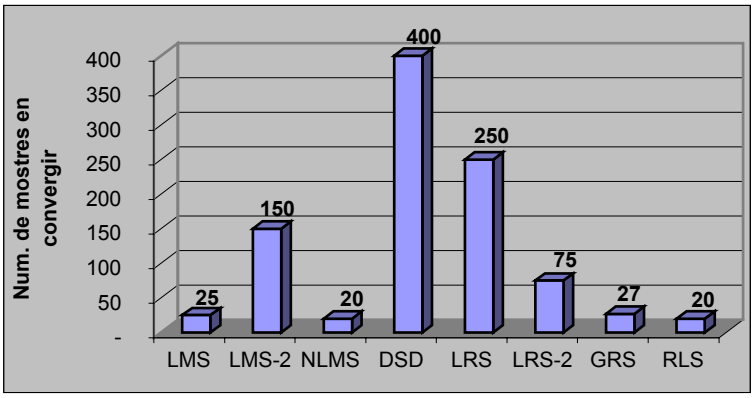


Fig. 1.13 Comparació de les velocitats de convergència

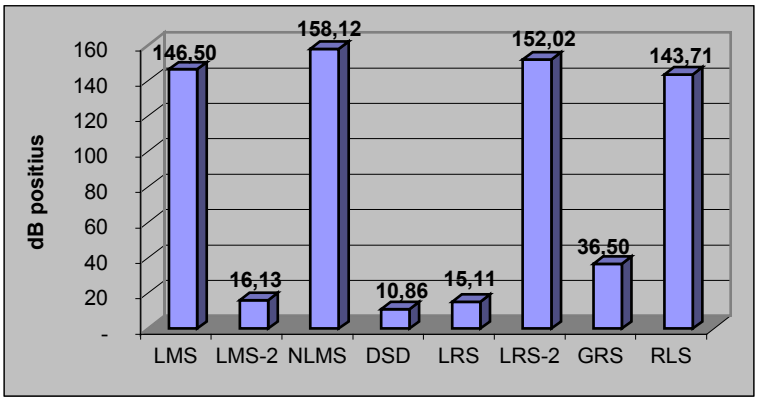


Fig. 1.14 Comparació del desajust

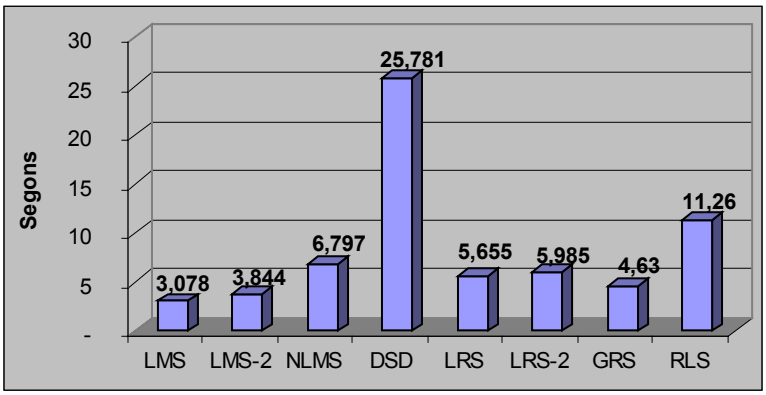


Fig. 1.15 Comparació del temps de la simulació

Per les comparacions hem realitzat les simulacions amb cent mil mostres perquè així igualem totes les llargades de les entrades i podem comparar amb igualtat el temps de simulació, ja que fins ara el GRS l'havíem utilitzat amb un factor M més de mostres ja que el necessita perquè el GRS només avança cap a l'error mínim cada M mostres. Així, aconseguim veure que si tots els algoritmes han convergit abans de que s'acabi el vector d'entrada i també podem comparar en igualtat el temps de simulació.

Un cop hem vist tot les comparacions, és hora de treure'n conclusions. Sempre tenint en compte les condicions inicials, ja que amb d'altres els resultats poden variar, podem dir quins són els algoritmes que millor s'adapten a aquestes circumstàncies. És evident que n'hi ha tres que sobresurten per sobre de la resta en totes les comparacions: el LMS és el més ràpid amb el temps de simulació, i dels millors en tot l'altre, el NLMS és el millor pel que fa el desajust i a la velocitat de convergència però el tercer més lent amb el temps de simulació, i el RLS és, com el NLMS, el més ràpid a convergir, tot i que encara és més lent que aquest en el temps de simulació i no té tantes bones prestacions en el desajust. Més enrere ja queda el LRS-2, o el GRS, que és bo amb el temps de convergència però amb les altres dues coses falla. En últim lloc trobem el DSD, que és el pitjor amb tot.

Menció a part mereixen el LMS-2 i el LRS. El primer es va crear amb la idea d'aconseguir un estalvi de memòria i/o una major rapidesa computacional a canvi d'una pèrdua de velocitat de convergència i potser també de desajust. Doncs bé, les dues coses queden confirmades; la pèrdua de prestacions ho veiem de manera directe amb el resultat de les comparacions i les millores s'aconseguirien en un estalvi de bits a nivell *hardware*. Com que els nostres escenaris seran implementats només en *software*, aquesta millora no la podrem veure. Com que els dos algoritmes són similars, d'ara en endavant estudiarem el LMS. També deixarem d'usar el LRS, ja que s'ha complert el que avançàvem en el subapartat 1.4.2: una millor convergència. A més la petita pèrdua en el temps de simulació el recuperem amb una amplia millora del desajust. Per tant, a partir d'ara només s'utilitzarà el LRS-2.

CAPÍTOL 2. ELIMINAR TONS

Per tal de posar a prova tots els algoritmes amb una situació més real que la que teníem fins ara, definirem un nou escenari. Primerament, a la referència utilitzarem una cançó. Aquesta cançó li sumarem uns tons de forma que interfereixin l'entrada i així el filtre tingui quelcom a eliminar. Pel que fa a l'entrada, també canvia. Necessitem que hi entri allò que volem eliminar, en el cas que ens pertoca, els tons. Però per fer-ho més difícil, la amplitud i la fase dels tons les variarem respecte als que entren per la referència, i així comprovarem el potencial d'aquests algoritmes.

Per últim, l'error que trobarem ja no haurà de ser una convergència fins a zero, si no que haurem de poder sentir la cançó sense la interferència, gràcies al filtre, que haurà de tenir més coeficients, com més complexa sigui la interferència a eliminar. En principi, amb dos tons fixes amb quatre coeficients n'hi hauria d'haver prou.

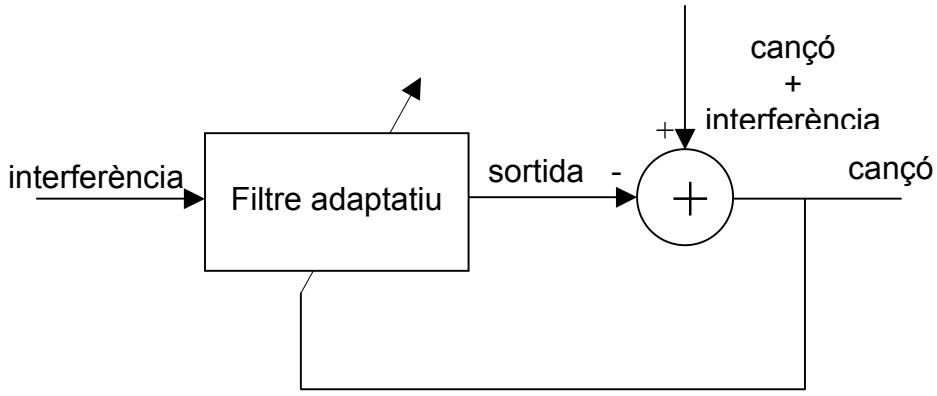


Fig. 2.1 Esquema bàsic dels algoritmes amb senyal d'entrada una interferència (tons), volent recuperar una cançó

A continuació passem a explicar els passos previs per aconseguir l'escenari esmentat. Un cop tenim el vector de mostres d'una cançó; cal construir els tons, que no seran res més que uns sinus a la freqüència desitjada (hem usat a 441 Hz i 2205 Hz). Per a l'amplitud dels mateixos, per no posar-ne una d'arbitrària, farem que tingui relació amb la potència de la cançó mitjançant la relació senyal a soroll. Ara ja podem muntar la referència, que és la suma del vector de mostres de la cançó amb els vectors dels tons. A l'entrada hi posarem els tons amb fase zero i sense el retoc de l'amplitud que hem comentat.

2.1. Algoritme LMS eliminant tons

Amb l'algoritme LMS obtenim uns resultats molt bons, ja que amb 4 coeficients per filtre els tons ja desapareixen, si bé és cert que tarda molt en eliminar-los per complet (quasi fins al final de la cançó de 200k mostres no deixem de

sentir-los). Amb 5 coeficients, la qualitat augmenta ja que sentim els tons a les primeres mostres. Podem veure-ho a les figures 2.2 i 2.3. Amb 5 coeficients la delta més important es veu reduïda en un 75%. Que la delta disminueixi vol dir que ha necessitat menys mostres per eliminar-la i per tant la seva magnitud es veu disminuïda en la representació de l'error.

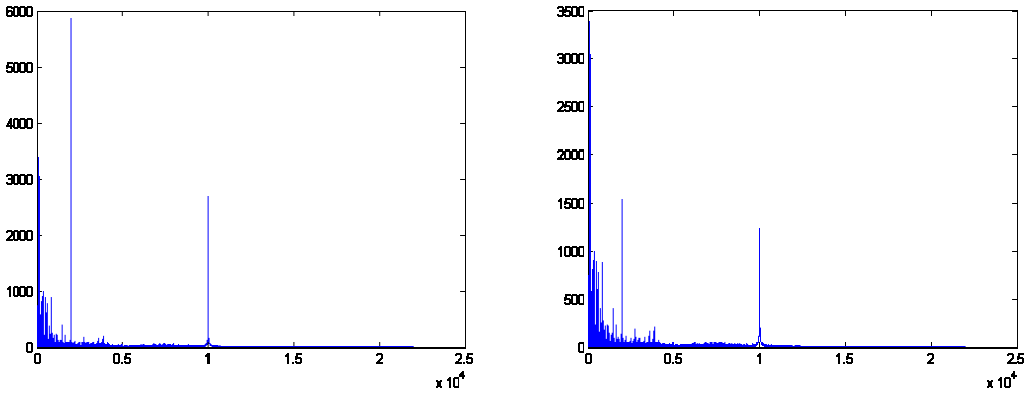


Fig. 2.2 i 2.3 Comparació de l'espectre freqüencial de l'error de l'algoritme LMS amb 4 i 5 coeficients

Per aconseguir aquest nivell de filtratge hem hagut de retocar novament el valor de μ, ja que el valor òptim trobat anteriorment ara no és vàlid. Al capítol 1 era de 0'3 i ara és de 0'001. Veiem a la figura 2.4 si simulem amb el valor anterior. No cal dir que la cançó que recuperem se sent fatal.

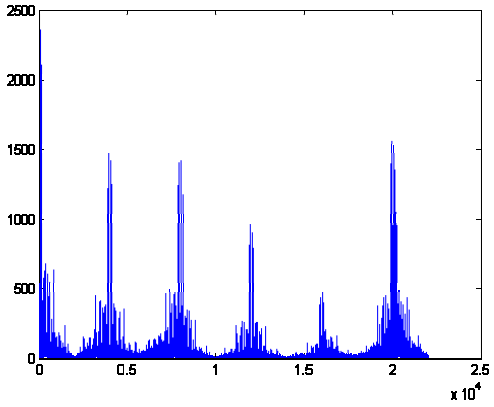


Fig. 2.4 Error de LMS en el domini freqüencial amb μ = 0'3 (abans òptima) i cinc coeficients

2.2. Algoritme NLMS eliminant tons

Amb NLMS ens passa el mateix que amb LMS. Primer, comprovem que el que abans eren valors òptims ara són valors que no recuperen la cançó, i hem vist

que succeeix el en domini freqüencial quelcom similar al que tenim a la figura 2.4. Així doncs, després de fer diverses proves hem arribat a la conclusió que el valor òptim de α és 0'01; mentre que β, si α és òptima, el seu valor és poc important. Però en cas que no ho fos, hem notat que per valors més alts s'allarga el temps que tarda en eliminar per complet el to.

Al igual que LMS, també obtenim millora amb cinc coeficients respecte quatre, tot i que amb quatre el comportament ja és força bo, ja que amb un NLMS de quatre coeficients i valors òptims tenim un resultat molt semblant a un LMS amb cinc coeficients i valors òptims. Si posem un coeficient més, l'eliminació dels tons és pràcticament instantània, només cal mirar la figura 2.5 per veure com pràcticament no es veu la delta, cosa que vol dir que l'elimina ràpidament.

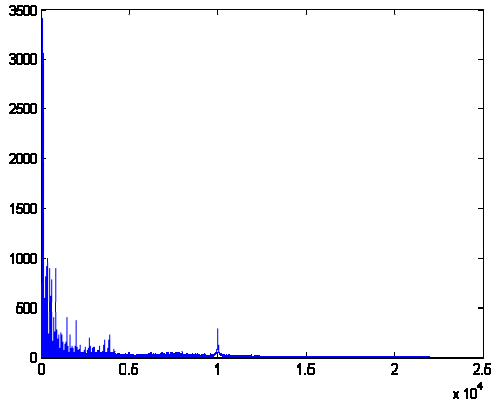


Fig. 2.5 Error de NLMS en el domini freqüencial amb valors òptims i 5 coeficients

2.3. Algoritme DSD eliminant tons

Un cop més, aquest algoritme queda per sota en prestacions que els altres que estem analitzant. Per començar, és el que més tarda en aconseguir una cançó totalment neta d'interferències. Si es vol tenir una rapidesa similar a la dels altres coeficients hem de posar 15 coeficients com a mínim. Amb 5 coeficients aconsegueix eliminar-lo gairebé al final de la porció de cançó que utilitzem (200k mostres), i amb 4 coeficients el to el va disminuint però s'acaba estabilitzant i no l'arriba a eliminar mai.

Tot això suposant que escollim una μ de 0'005, el valor òptim en aquest cas. L'altre possibilitat és utilitzar-ne una de 0'025. El temps que tarda en convergir disminueix, però a canvi tenim una cançó un pèl distorsionada i més si tenim pocs coeficients. Si augmentem la μ la distorsió és més alta, mentre que si augmentem el nombre de coeficients redueix la interferència que distorsiona la cançó però no fins al punt d'eliminar-la per complet.

La figura 2.6 mostra el domini freqüencial en el millor dels casos, que comparant amb les altres gràfiques veiem que el DSD no té bones prestacions.

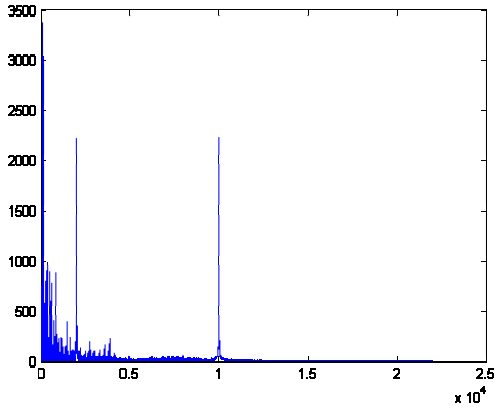


Fig. 2.6 Error de DSD en el domini freqüencial amb valors òptims i 15 coeficients

2.4. Algoritme LRS-2 eliminant tons

Un cop més tenim un compromís entre la velocitat de convergència i la qualitat del so obtingut, probablement degut al desajust provocat per una μ massa gran. Con en els altres algoritmes, la que abans era la μ òptima ara fa que l'algoritme no convergeixi, i hem de posar-ne una tres ordres de magnitud més petita.

Amb μ = 0'0005 tenim una velocitat de convergència acceptable, però bastant més lenta comparada amb LMS o NLMS, i un soroll romanent quasi imperceptible. Si el volguéssim eliminar hauríem de disminuir el valor de μ a 0'0001, però llavors el to ja tarda massa en desaparèixer.

Pel que fa al nombre de coeficients, fer-ho amb quatre o cinc només varia el temps de convergència. Creiem que amb cinc és raonablement millor, tot i que veiem a la figura 2.7 unes deltes degut a que s'ha fet la fft de tot el vector d'error, i per tant es veuen les deltes de l'inici, quan encara no han estat eliminades.

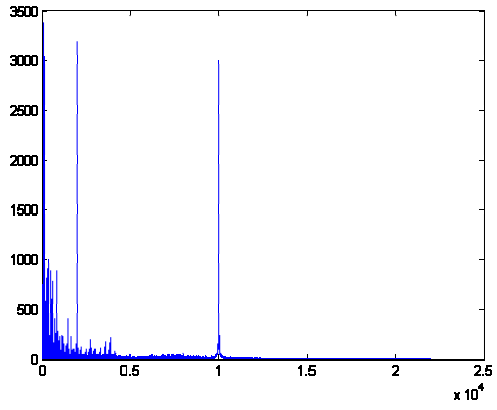


Fig. 2.7 Error de LRS-2 en el domini freqüencial amb valors òptims i cinc coeficients

2.5. Algoritme GRS eliminant tons

Per a la utilització d'aquest algoritme en el nou escenari hem necessitat modificar lleugerament el codi. Fins ara, només calculàvem l'error cada cop que actualitzàvem el valor dels coeficients del filtre, que era cada M mostres. Com que abans es tractava d'una entrada aleatòria no hi havia cap problema si fèiem això; però ara l'error, com que serveix per adaptar el filtre i també per escoltar la música resultant, necessitem calcular-lo a cada iteració. I això és el que fem, actualitzem l'error a cada mostra i els coeficients cada M mostres.

Quan estudiem el comportament de l'algoritme en funció del valor de M ens n'adonem que al principi semblava beneficiós una M alta degut a que així es calculava l'error quadràtic mitjà amb un número més alt de mostres. Però ara que calculem l'error a cada mostra veiem que és més beneficiós actualitzar més sovint el filtre (M petita). Fins al punt que el millor comportament del GRS és amb M=2 quan, per la manera que està dissenyat el codi, així és com el filtre adaptatiu s'actualitza a cada iteració. Després de diverses proves, hem trobat que amb la M = 2, una μ inicial de 0'0001, una β de 1'25 i 5 coeficients del filtre és la manera que l'algoritme ens doni les millors prestacions possibles.

Aquests valors s'han escollit tenint en compte que una β més gran provoca més rapidesa en la convergència però també més desajust; que una M alta, tal i com hem explicat ara mateix, disminueix la qualitat de la cançó de forma fefaent, així com la velocitat de simulació; i amb quatre coeficients no és possible en cap cas eliminar per complet els tons. Aquí tenim l'espectre freqüencial resultant, amb un efecte mirall en l'espai comprès entre 0 i $F_s/2$.

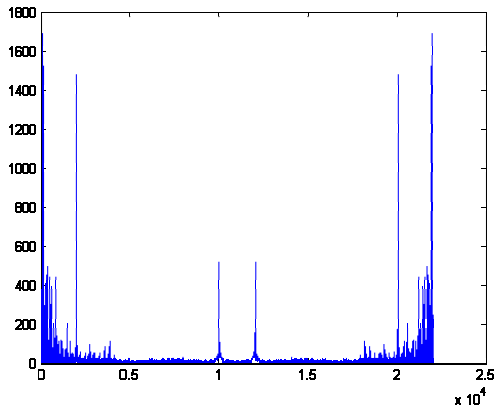


Fig. 2.8 Error de GRS en el domini freqüencial amb valors òptims i cinc coeficients

2.6. Algoritme RLS eliminant tons

A l'hora d'utilitzar aquest algoritme hem tingut molts problemes. Tal i com estava dissenyat el codi, si simulàvem la situació actual, l'algoritme aparentment funcionava correctament a l'inici de la cançó eliminant per complet

els tons però arribava un moment que divergia molt ràpidament i deixàvem de sentir la cançó. Trobareu més informació a l'annex. Un cop solucionat això, hem pogut veure que l'algoritme RLS no aconsegueix eliminar per complet els tons amb tres coeficients, com era d'esperar després de l'experiència amb els altres algoritmes. Amb quatre coeficients podem sentir la cançó totalment neta.

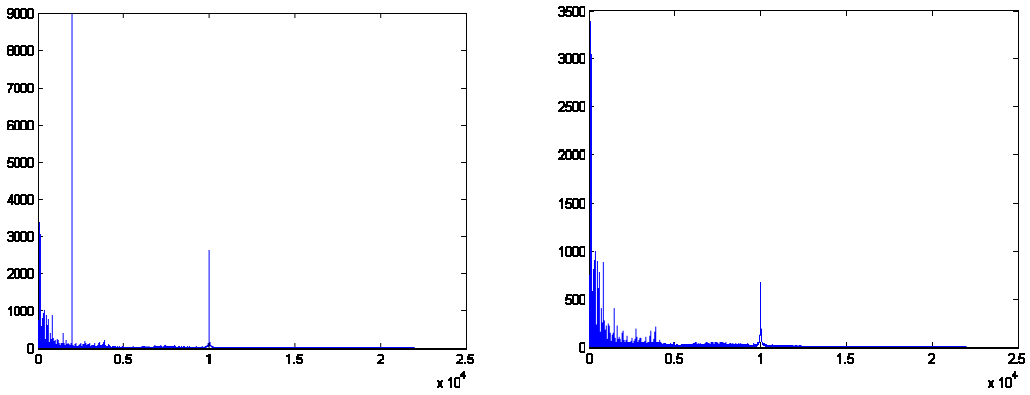


Fig. 2.10 i 2.11 Error de RLS en el domini freqüencial amb valors òptims i 3 coeficients. A la dreta, amb 4 coeficients

2.7. Comparació dels algoritmes eliminant tons

Igual que a les conclusions extretes de les comparacions del primer punt, aquí veiem que hi continuen havent algoritmes de primera i de segona categoria. El LMS, el NLMS i el RLS eliminen els tons amb rapidesa i sense deixar-ne rastre, si bé que el LMS amb menor rapidesa. Un esglaó per sota trobem el LRS-2, el GRS, que el seu funcionament és bo, tant en rapidesa com en desajust, però no arriben al nivell dels dos primers. I en últim lloc i destacat tenim el DSD, ja que si volem aconseguir una cançó totalment neta de tons, el temps que tarda en aconseguir-ho és extremadament alt en comparació amb els altres algoritmes.

Als millors algoritmes també hem vist que no els cal que el filtre adaptatiu tingui gaires coeficients. De fet, es pot demostrar que idealment amb quatre coeficients és suficient per eliminar dos tons. Ara bé, els únics que realment eliminen els tons amb quatre coeficients són el NLMS, el LMS i el RLS. Els algoritmes de segona fila en necessiten cinc perquè amb quatre només disminueixen els tons. El DSD, si volem que es comporti de manera similar als altres, necessita d'uns 15 coeficients. No cal dir que quan menys coeficients millor, ja que a més coeficients més càrrega computacional.

Destacar que en tots els algoritmes, el valor de les constants que en el primer punt havíem decidit òptimes, ara, al canviar el tipus d'entrada i referència, aquests valors els hem hagut de retocar. En general, les μ i altres constants les hem disminuït dos ordres de magnitud aproximadament.

CAPÍTOL 3. ELIMINAR TONS I SOROLL

En aquest punt introduirem el soroll per fer encara més real les simulacions. Ens crearem un vector de mostres aleatòries amb una potència controlada per nosaltres mitjançant el valor de la relació senyal a soroll de l'entrada. Això farà que, com és lògic, a l'algoritme li sigui més feixuc aconseguir la neteja de la cançó; i a partir d'això aconseguirem una avaluació més realista dels algoritmes. El valor de la SNR que utilitzarem serà de -1 dB, ja que complirà els objectius descrits sense arribar a emmascarar els tons o la cançó. Més endavant, a les conclusions, explicarem què passa si usem un soroll massa potent.

La principal diferència amb les anteriors simulacions serà que el soroll només estarà present a l'entrada del filtre adaptatiu, de tal manera que aquest haurà de discernir entre la interferència i el soroll per adaptar-se i aconseguir eliminar la interferència que tenim a la cançó, ja que el soroll no el tindrem a la referència. Per entendre-ho millor, el soroll no és quelcom que s'hagi d'eliminar si no que un impediment més per a què el filtre adaptatiu elimini la interferència produïda pels tons, tot i que evidentment el filtre ideal no deixarà passar el soroll perquè quan es resti la cançó de la sortida no quedi soroll a l'error. Així doncs, l'esquema queda de la següent manera.

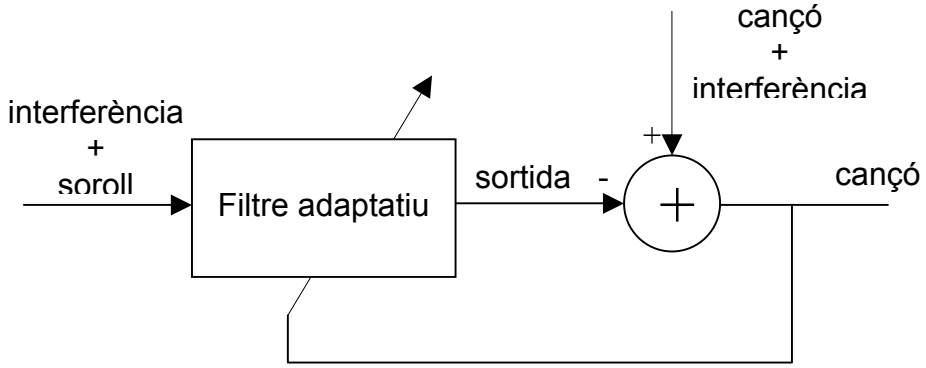


Fig. 3.1 Esquema bàsic dels algoritmes amb senyal d'entrada una interferència i soroll, volent recuperar una cançó

Els tons continuaran sent els mateixos que en el punt anterior, 441 Hz i 2205 Hz, amb la mateixa amplitud. També la cançó serà la mateixa i ambdós vectors continuaran sent de la mateixa llargada.

Com més tard es podrà comprovar, el soroll passarà a ser el principal problema. Això és perquè, per molt restrictiu que sigui el filtre deixarà passar soroll, encara que sigui només per l'ample de banda a on hi ha les deltes (ja que és impossible que un filtre sigui tan restrictiu que sigui capaç de separar la delta del soroll que té immediatament al costat seu freqüencialment parlant). A més, el filtratge de les deltes (entenent el filtratge com la part que ens interessa, no la despreciable), suposa allà a on només hi ha soroll uns lòbuls

que li permeten el pas; o el que és el mateix, volem filtrar per quedar-nos només amb les deltes, i aconseguim les deltes i una porció del soroll que no és eliminat degut als lòbuls i a l'ample de banda esmentat. Per tal de disminuir aquests lòbuls i aquest ample de banda, podem dotar al filtre de més coeficients. Com més coeficients té, més graus de llibertat té el filtre per decidir quina part de l'espectre freqüencial filtra i quina no. En el cas que expliquem això es tradueix en més graus de llibertat per tal de fer l'ample de banda més petit i/o tractar de disminuir els lòbuls que deixen passar el soroll.

En aquest apartat, per comparar els algoritmes en les mateixes condicions, considerarem que el número mínim de coeficients del filtre necessaris per a cada algoritme és aquell número més baix que faci que els tons s'eliminin completament. A partir d'aquí podrem comparar el grau de soroll que deixa passar. Així comparem el nombre de coeficients necessaris per aconseguir l'objectiu primari i també quin és el preu a pagar per aconseguir-ho.

3.1. Algoritme LMS eliminant tons i soroll

Si provem d'utilitzar l'algoritme amb 4 o 5 coeficients veurem que el resultat obtingut és molt dolent, ja que escoltarem soroll però també els tons que volíem eliminar. Algú pot pensar que perquè no elimina els tons ara, si abans amb els mateixos coeficients ho aconseguia. Això és perquè l'algoritme "decideix" que el millor resultat possible amb aquests coeficients és eliminar una porció de tons i una de soroll; ja que si centrés tots els esforços en l'eliminació dels tons la quantitat de soroll a la sortida seria tal que el resultat obtingut seria pitjor. Ho podem veure a la figura 3.2.

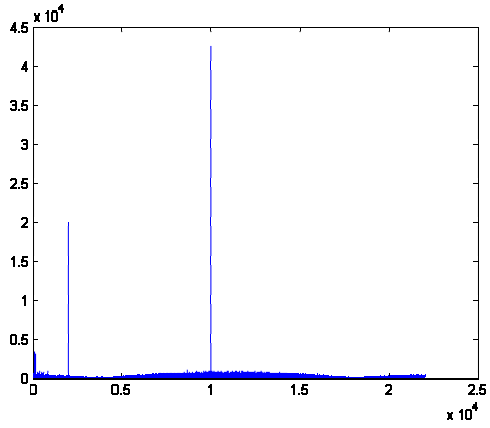


Fig. 3.2 Espectre freqüencial de l'error amb l'algoritme LMS de 5 coeficients

Si comparem la figura 3.2 amb la 2.3 veurem que ara ha filtrat pitjor. Així que haurem d'augmentar el nombre de coeficients. Després de diverses simulacions designem aproximadament als 30 coeficients com el nombre que fa que deixem de sentir els tons.

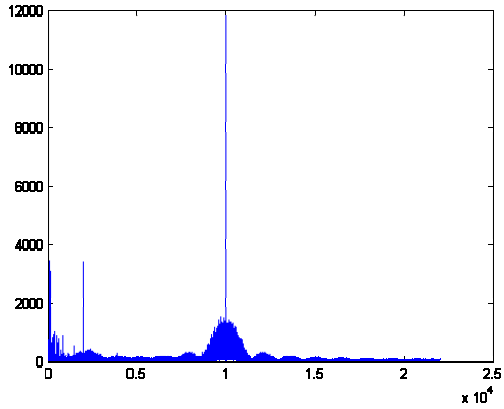


Fig. 3.3 Espectre freqüencial de l'error amb l'algoritme LMS de 30 coeficients

A tall d'exemple, mostrem les gràfiques amb 200 coeficients; i al costat la forma que el filtre adaptatiu assoleix en l'última iteració. Està clar que a més coeficients millor resultat obtindrem degut als graus de llibertat que dotem al filtre, però la limitació és el temps de simulació que augmenta a cada coeficient. Noti's que el filtre a la zona del to tingui un ample de banda molt estret.

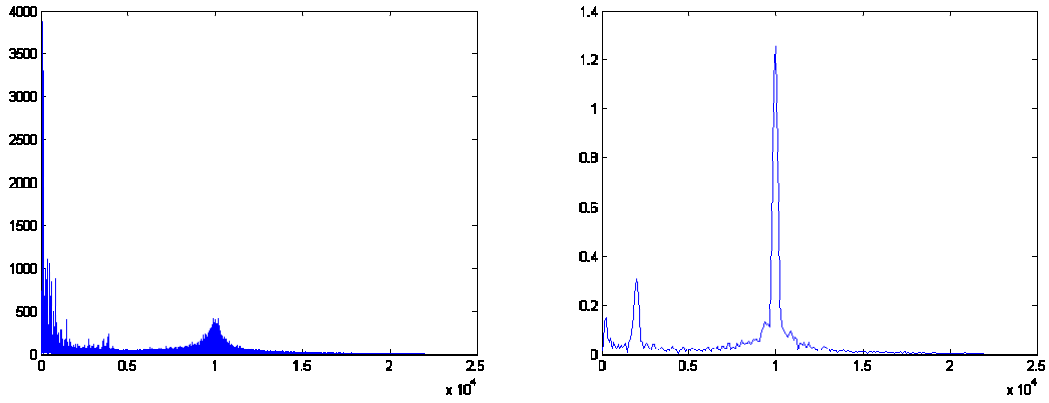


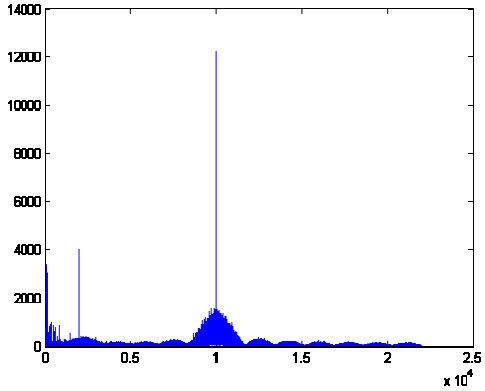
Fig. 3.4 i 3.5 A l'esquerra espectre freqüencial de l'error amb l'algoritme LMS de 200 coeficients i espectre del filtre a la última iteració

3.2. Algoritme NLMS eliminant tons i soroll

Hem seguit el mateix procediment per al NLMS. Hem anat comprovant el resultat auditivament fins que hem considerat que no sentíem els tons. Això ha estat als 25 coeficients. Al fer la gràfica de l'error en el domini freqüencial, hem vist que la del NLMS amb 25 coeficients i la del LMS a 30 coeficients és similar.

A l'escoltar-ho percebem que el soroll sona diferent respecte al que sentíem amb el LMS. Això és degut a que els lòbuls de cada un dels algoritmes filtren algunes freqüències més que d'altres. No ens decantem per cap dels dos en quan a més molest, ja que és difícil decidir-ho auditivament i a part això

coincideix amb la gràfica perquè no es veu que un tingui una major potència vers la cançó respecte l'altre. A continuació mostrem l'error i el filtre definitiu en el domini freqüencial del NLMS de 25 coeficients.



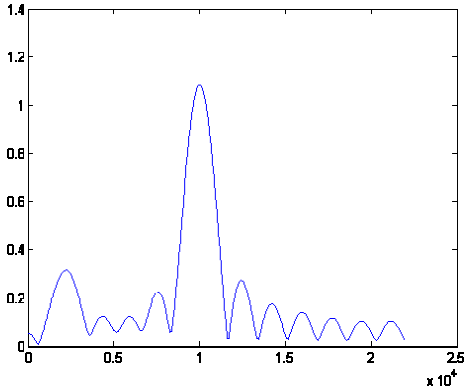


Fig. 3.6 i 3.7 A l'esquerra l'espectre freqüencial de l'error amb l'algoritme NLMS de 25 coeficients i el filtre a la última iteració

Com a curiositat, hem provat el comportament amb unes components no òptimes. Amb una α més gran empitjora, sentim tant soroll que emmascara la cançó per complet. Però el més curiós és amb una α més petita (per exemple, 0.00005): Com que tarda més en convergir, al principi de la cançó la sentim amb tons i sense soroll. A mida que va convergint, pot anar eliminant els tons però a la vegada va deixant passar soroll. Així, l'efecte que tenim és una gradual disminució dels tons a la vegada que un augment del soroll.

3.3. Algoritme DSD eliminant tons i soroll

Hem comprovat que amb 25 coeficients, el DSD tarda en eliminar els tons, però ho acaba fent. De fet, és possible que realment no els elimini completament si no que la seva progressiva disminució arribi un moment que quedi emmascarada pel soroll. Aquesta és l'explicació de com un algoritme que abans necessitava el triple de coeficients que els altres, ara pugui "eliminar" els tons amb els mateixos coeficients que el més eficient.

I és que el soroll aquí si que és molt molest. Per poder plasmar sobre el paper aquest important augment respecte, per exemple, el NLMS farem el següent. Li restarem a l'error dels dos algoritmes (DSD i NLMS ambdós amb 25 coeficients) la cançó original, sense interferències de cap tipus. Així podrem comparar el soroll que queda a l'error de cada algoritme, mostrant una porció del domini freqüencial de l'error; allà on, després de restar la música, només hi pot haver soroll. Als voltants dels 1100 Hz reuneix aquestes característiques. També mostrem una porció de la mateixa zona de l'entrada, per poder demostrar que els dos algoritmes han reduït part del soroll, gràcies als graus de llibertat que li hem donat amb els 25 coeficients.

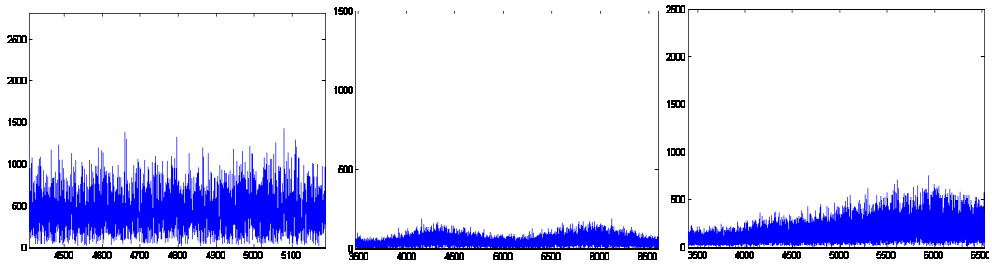


Fig. 3.8, 3.9 i 3.10 De dreta a esquerra, soroll de l'entrada, del NLMS i del DSD

Es pot veure que els dos han reduït el soroll, però el NLMS més que el DSD. Donat que els dos tenen el mateix nombre de coeficients, podem concloure que el primer és més eficient ja que sap utilitzar millor els coeficients per eliminar els tons sense deixar passar tant soroll. Un altre paràmetre que amb el DSD cal tenir en compte és el temps de simulació. Per a 25 coeficients, tarda aproximadament sis minuts, quan amb el LMS no acostuma a durar més de 20 segons.

3.4. Algoritme LRS-2 eliminant tons i soroll

Un cop més hem seguit el mateix procediment que els altres algoritmes i hem trobat un nombre de coeficients similars, 30. Calcularem ara quin tant per cent de tons s'han eliminat respecte els tons de la referència. Ja avancem que el resultat és similar en tots els algoritmes, i per això només es detalla en aquest.

Si fem la gràfica de la referència menys la cançó, tindrem únicament els tons; d'aquesta manera podem veure l'amplitud dels tons. D'altra banda, fem la gràfica de l'error menys la cançó, per veure com han quedat els tons després de l'actuació del filtre adaptatiu. En aquesta gràfica (figura 3.12) també podrem veure la porció de soroll que ha passat gràcies als lòbuls. No cal dir que la millora és considerable, ja que si ens fixem en el to amb més amplitud, veiem que passa de $9 \cdot 10^4$ a $12 \cdot 10^3$.

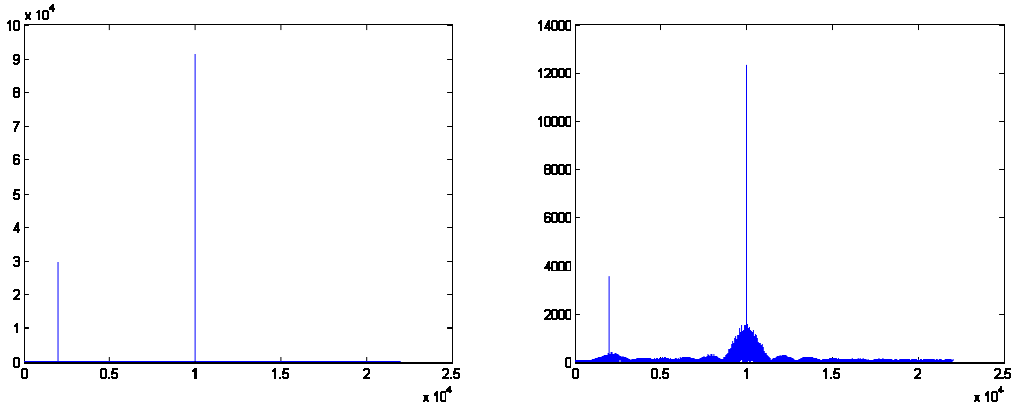
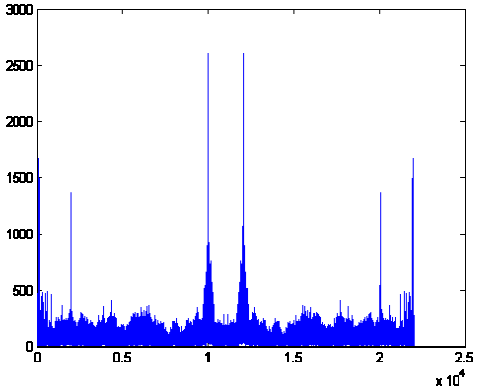


Fig. 3.11 i 3.12 A l'esquerra tenim la referència menys la cançó i a la dreta l'error menys la cançó

3.5. Algoritme GRS eliminant tons i soroll

Si busquem el número de coeficients que fa que els tons deixin de sentir-se, la resposta la trobem al voltant dels 30 coeficients. Ara bé, aquests coeficients no són suficients per sentir la cançó amb nitidesa. El soroll a la sortida és tal que amb prou feines se sent la cançó. El pitjor de tot és que per més coeficients que s'utilitzi, el soroll no disminueix. Com a mostra les figures 3.13 i 3.14, on es veu l'error en el domini freqüencial amb 100 coeficients i la forma del filtre a l'última iteració.



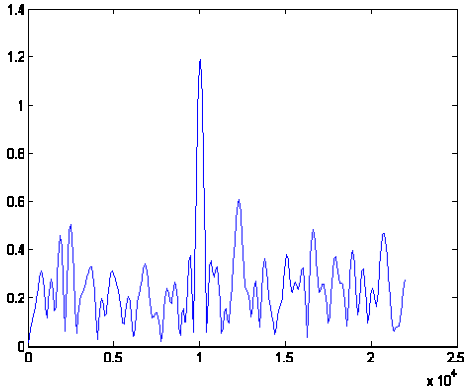


Fig. 3.13 i 3.14 Error amb el filtre de 100 coeficients i el filtre adaptatiu a la última iteració

Per fer-nos una idea, la qualitat del so filtrat pel GRS ha baixat de forma considerable fins al punt de ser semblant al del DSD, que en tots les anteriors punts era el pitjor. No obstant encara manté la velocitat de simulació en els paràmetres que fins ara li eren habituals.

3.6. Algoritme RLS eliminant tons i soroll

Si posem 25 coeficients la cançó obtinguda està totalment neta de tons, un cop ha convergit l'algoritme. A més, el soroll tot i existir és molt poc molest, deixant-nos sentir la cançó amb una qualitat prou bona comparada amb els altres algoritmes. A la figura 3.15 veurem l'espectre de l'error amb 25 coeficients. És un dels millors algoritmes sota l'escenari que estem provant en aquest capítol, per la rapidesa de convergència, per l'eliminació de tons i per la disminució del nivell de soroll.

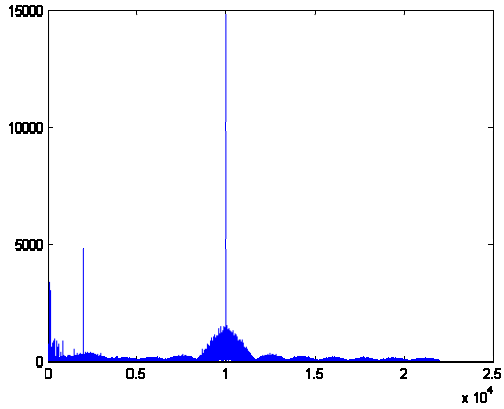


Fig. 3.15 Espectre freqüencial de l'error de l'algoritme RLS amb 25 coeficients

3.7. Comparació i conclusions

Com era d'esperar la qualitat del so ha disminuït molt amb la introducció del soroll. En cap cas es pot aconseguir eliminar tot el soroll i tons, per molts coeficients que tinguin els algoritmes, ja que sempre hi haurà una porció de soroll que no podrà ser filtrada, ja sigui per la impossibilitat de fer un filtre millor o perquè el propi soroll coincideix freqüencialment amb parts que no volem eliminar, com els tons. No obstant, segueixen havent-hi algoritmes que eliminen millor els tons i part del soroll que d'altres. És el cas sobretot del LMS i NLMS. Els altres, especialment el DSD i el GRS, el seu filtratge és bastant pitjor. Paradoxalment, per aquests dos últims algoritmes trobem que el número de coeficients que hem donat per bo és menor o igual que d'altres que funcionen millor, com el LMS per exemple. Això és així perquè el mètode utilitzat per decidir aproximadament el nombre de coeficients necessaris és aquell en què els tons es deixen de sentir. Però com ja hem comentar en l'apartat del DSD, això pot induir a error ja que els mals algoritmes deixen passar una quantitat major de soroll que emmascara els tons. També cal dir que el GRS i el DSD amb més coeficients no milloren substancialment i per tant no té sentit posar més coeficients, mentre que els altres si que milloren, pel que fa a l'eliminació del soroll.

Cal comentar també el perquè de l'elecció del valor de la SNR. Com hem dit a l'inici del punt 3, un soroll massa potent pot comportar problemes. Això és així perquè els nostres filtres davant d'aquest problema, en comptes de filtrar els tons i per tant una porció de soroll però de gran potència, "decideixen" que el millor que poden fer és no deixar passar res i per tant el que succeeix és que a la referència no se li resta res provinent de la sortida del filtre. Així, a l'error tenim la cançó i els tons sense atenuar. Els filtres fan això perquè donada una potència molt alta del soroll, cap intent d'atenuació dels tons provocarà una millora de la relació senyal a interferència, si no que just el contrari.

Per últim, dir que no hem canviat els valors en les constants de cada algoritme respecte el punt anterior, com el pas d'adaptació o similar. Això és degut a que la naturalesa de l'entrada era la mateixa que a l'apartat al que fem referència.

CAPÍTOL 4. ENTREVISTA A UN CIRCUIT AUTOMOBILÍSTIC

Tractant d'arribar a simular situacions el més reals possibles, hem pensat que un cas real potser el d'una entrevista d'un mitjà de comunicació audio-visual a peu de circuit. Sense aplicar l'algoritme, les dues persones que estan conversant queden totalment emmascarades pel soroll que fan un seguit de cotxes quan passen a prop d'ells.

En aquest punt intentarem que gràcies a l'aplicació dels algoritmes puguem sentir l'entrevista. Per aconseguir-ho, l'esquema utilitzat serà el de la figura 4.1.

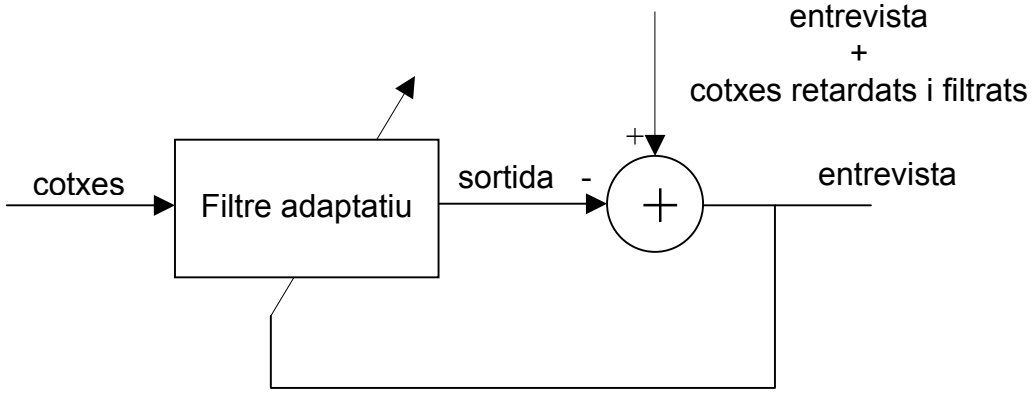


Fig. 4.1 Esquema bàsic dels algoritmes amb senyal d'entrada el soroll produït pels motors dels cotxes, volent recuperar una entrevista

Tal i com estan les coses doncs, caldria tenir un micròfon prop dels interlocutors per obtenir la referència i un altre micròfon una mica més allunyat on només se sentissin els motors dels cotxes (un micròfon d'ambient, per exemple). Aquest serviria per obtenir les dades d'entrada al filtre adaptatiu. Això faria que el soroll dels motors que tenim als dos micròfons no fossin exactament iguals. Per poder simular aquesta diferència hem aplicat als motors de la referència un retard d'un terç de mostra (per evitar que els retards siguin un nombre enter de mostres), els hem convolucionat amb un filtre de tres coeficients i li hem aplicat una amplitud diferent al soroll dels motors de l'entrada.

Aquest nou escenari servirà per veure com reaccionen els filtres adaptatius davant de nous reptes, com els retards, juntament amb d'altres que ja hem experimentat.

4.1. Algoritme LMS eliminant soroll de motors de cotxes

Hem començat fent la simulació amb valors alts de coeficients, esperant que donada la complexitat de l'experiment caldrien bastants coeficients per poder

eliminar el soroll produït pels motors dels cotxes. Hem comprovat que el LMS amb molts coeficients (hem arribat a provar fins a 65) elimina en part el soroll dels cotxes. Però també hem comprovat que a mida que traiem coeficients el so que s'obté millora, fins al punt que hem vist que només amb dos coeficients és la millor manera d'utilitzar l'algoritme LMS. El perquè de que un experiment aparentment complex sigui resolt pels algoritmes amb només dos coeficients es troba al punt 4.7.

Per optimitzar encara més l'algoritme, hem variat el valor del pas d'adaptació. Degut a què el que s'ha de filtrar varia freqüencialment parlant més ràpidament que els tons de l'apartat anterior (de fet els tons no variaven) cal dotar a l'algoritme de més rapidesa en l'adaptació. Això s'aconsegueix amb un augment de la μ.

Tot seguit presentem dues gràfiques de l'evolució de la resposta freqüencial del filtre, amb 2 coeficients i μ = 0'1 (resultat òptim) i amb 30 coeficients i el mateix pas d'adaptació. Per simplificar el càlcul, en aquests tipus de figures no mostrarem l'evolució detallada si no que només mostrarem una de cada cent mostres.

Podem veure la paulatina evolució del filtre entre 0 i 5 kHz, les freqüències desitjades; ja que en les altres freqüències no hi tenim soroll de motors de cotxes, com podrem veure a les figures 4.4 i 4.5.

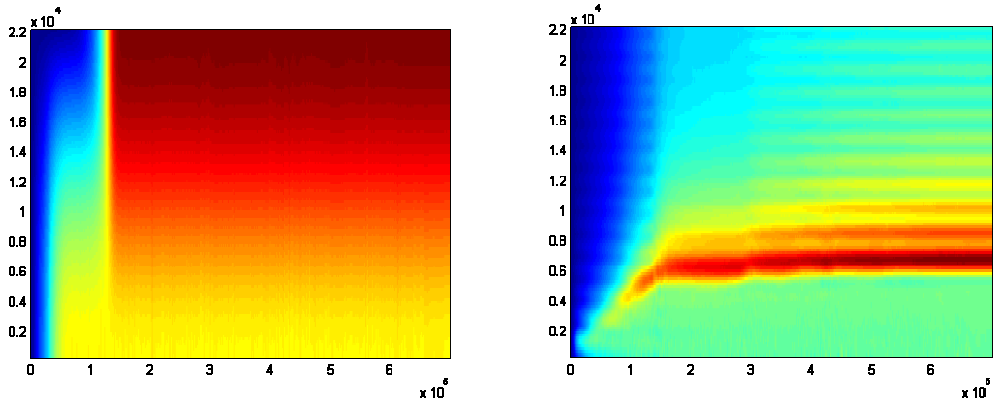


Fig. 4.2 i 4.3 A l'esquerra, evolució del filtre amb 2 coeficients i μ = 0'1 i a la dreta amb 30 coeficients i μ = 0'1

Per finalitzar aquest apartat, mostrem el soroll de només els cotxes a l'entrada i a la sortida (a l'error) per veure quina porció n'ha eliminat. Podem comprovar com ha eliminat una gran quantitat d'interferència ja que aquesta, a la referència, obviant els pics, té un valor que oscil·la entre 5000 i 15000. En canvi, quan a l'error que obtenim li restem l'entrevista ens queda el valor de l'interferència. Aquesta interferència, gràcies a l'acció del filtre ara té un valor aproximat d'entre 200 i 500.

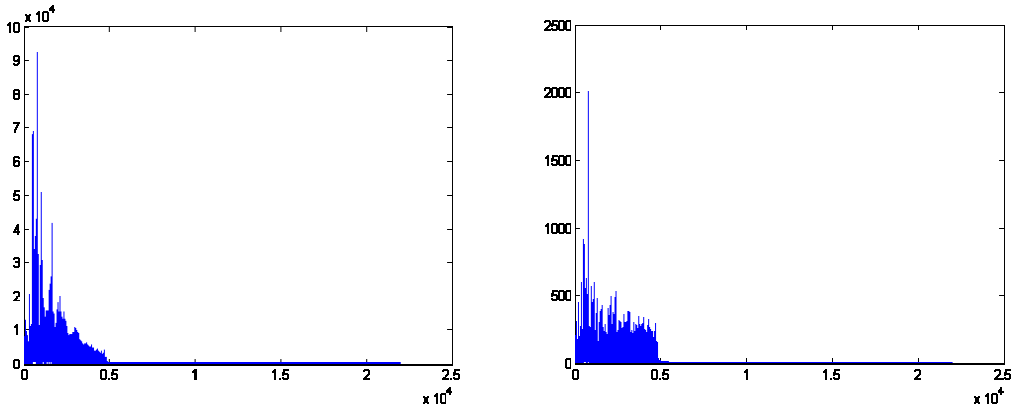


Fig. 4.4 i 4.5 A l'esquerra espectre de la interferència a eliminar (la de la referència) i a la dreta l'espectre de la interferència a l'error

4.2. Algoritme NLMS eliminant soroll de motors de cotxes

Els resultats obtinguts amb aquest algoritme han estat auditivament molt similars al LMS: L'entrevista es pot recuperar de tal manera que es pot entendre tot el que diuen i els cotxes se senten de fons.

La diferència entre els dos algoritmes ha estat que si en el LMS havíem de fer més gran el pas d'adaptació respecte els apartats anteriors, amb el NLMS ha estat el contrari, ja que ara α = 0'0001. Cal dir però que les constants dels dos algoritmes no tenen exactament la mateixa funció i per tant aquesta pot ser una raó de perquè unes hagin variat a l'alça i les altres a la baixa.

Però les gràfiques de l'evolució del filtre adaptatiu demostren que el NLMS convergeix més ràpidament que el LMS, tot i tenir unes constants més petites; tal i com podem veure a la figura 4.6 i comparant-la amb la 4.2, on podem apreciar que el canvi brusc de tonalitat que denota la convergència succeeix abans en l'algoritme NLMS.

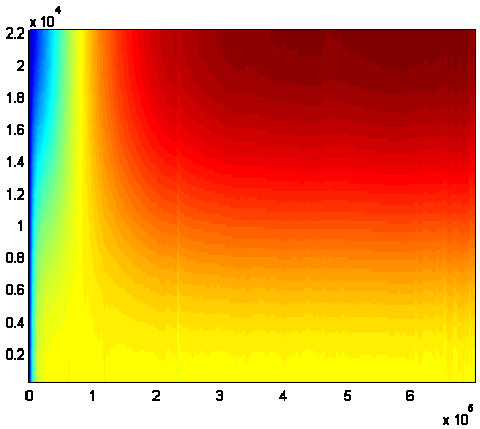


Fig. 4.6 Evolució del filtre amb 2 coeficients i α = 0'0001 i β = 0'001

Per últim, comentarem el valor de β, que també ha disminuït fins a 0'001. Això vol dir que li donem molt pes a la norma de l'entrada del punt actual que no a la memòria. És lògic si pensem que estem en un entorn on l'entrada varia ràpidament.

4.3. Algoritme DSD eliminant soroll de motors de cotxes

Dels anteriors apartats ja sabem que el DSD és normalment l'algoritme que pitjor elimina els sons interferents. No obstant, en el cas que ens trobem, aquest algoritme elimina el soroll dels cotxes amb resultats similars a d'altres algoritmes, i amb només dos coeficients.

Malauradament, per tal que l'algoritme DSD funcioni de la millor manera possible no només hem de donar els valors adequats al pas d'adaptació o escollir el nombre de coeficients correctament, si no esperar també que les successives pertorbacions aleatòries que utilitza l'algoritme per minimitzar l'error siguin favorables i per tant elimini les interferències el més ràpidament possible. A continuació mostrem dos casos amb dos coeficients i μ = 0'0001.

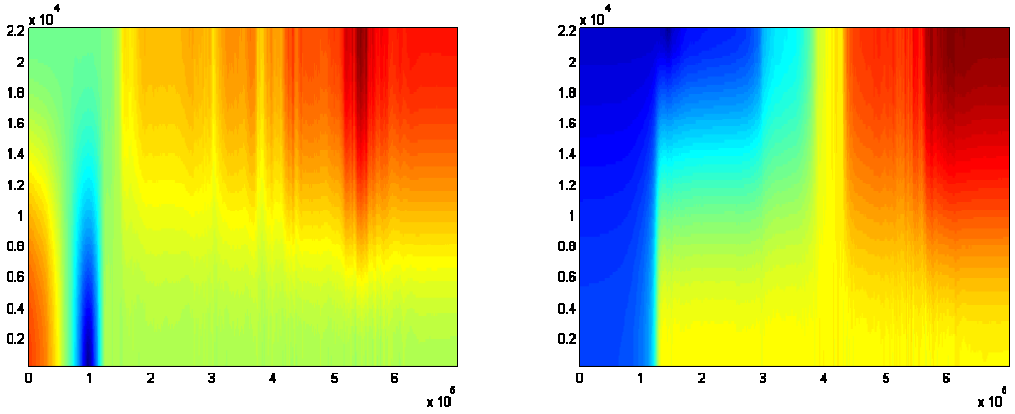


Fig. 4.7 i 4.8 Diferents evolucions del filtre de l'algoritme DSD de 2 coeficients amb μ = 0'0001

En les figures 4.7 i 4.8 tenim un cas com el de l'esquerra on a l'escoltar l'error podem sentir l'entrevista quasi des del principi mentre que en l'altre cas només la podem començar a sentir cap al final. Això és difícil de detectar observant només les figures ja que el fet que només hi hagi freqüències a eliminar fins a 5 kHz fa que els valors de les zones on el filtre no ha de fer res puguin ser elevats i això provoca que a la zona interessant perdem resolució en l'escala de colors.

Si posem més coeficients, a part que es dispara el temps de simulació, notem una progressiva disminució de la qualitat de l'entrevista, si bé cal dir que la diferència entre dos coeficients i tres és inapreciable. A les següents figures mostrem un cas de tres coeficients i un de deu. En el de tres ha convergit ràpid mentre que amb el de deu no ho ha fet fins a la mostra 1500 aproximadament.

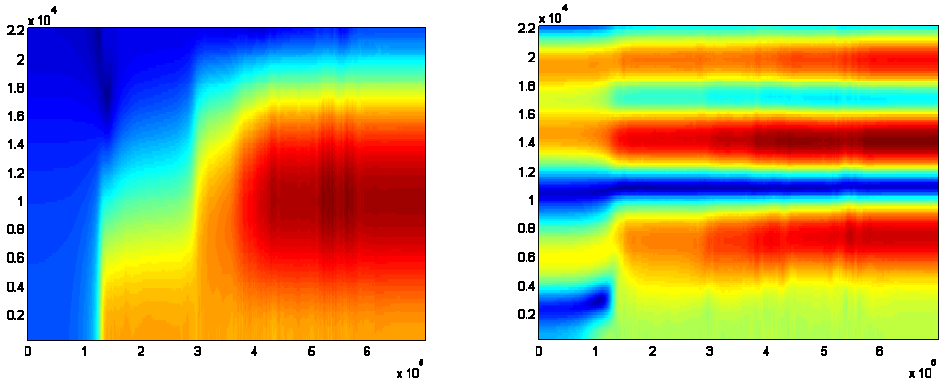


Fig. 4.9 i 4.10 A la dreta evolució del filtre amb 3 coeficients i a l'esquerra amb 10; amb μ = 0'0001

4.4. Algoritme LRS-2 eliminant soroll de motors de cotxes

Ens trobem davant d'un algoritme que té un molt bon comportament. Augmentant la μ fins a 0'0075 com ja hem hagut de fer en alguns algoritmes anteriors, aconseguim una convergència molt ràpida, de tal manera que ens permet escoltar l'entrevista d'inici a fi. A més, si comparem la figura 4.5 amb la figura 4.11 veiem que elimina millor la interferència l'algoritme LRS-2 que el LMS (hem comprovat que el NLMS millora el LMS però no el LRS-2).

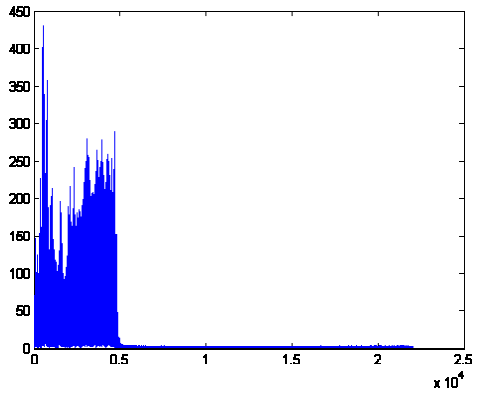


Fig. 4.11 Espectre freqüencial que pertany a la interferència de l'error

A tall d'exemple presentem l'evolució del filtre de l'algoritme LRS-2 amb dos i tres coeficients. Amb això, continuem comprovant que no de moment no hi ha cap algoritme que millori amb l'augment de coeficients; i ja avancem que després de fer les comprovacions pertinents amb els algoritmes que falten tampoc aquests milloren amb més coeficients.

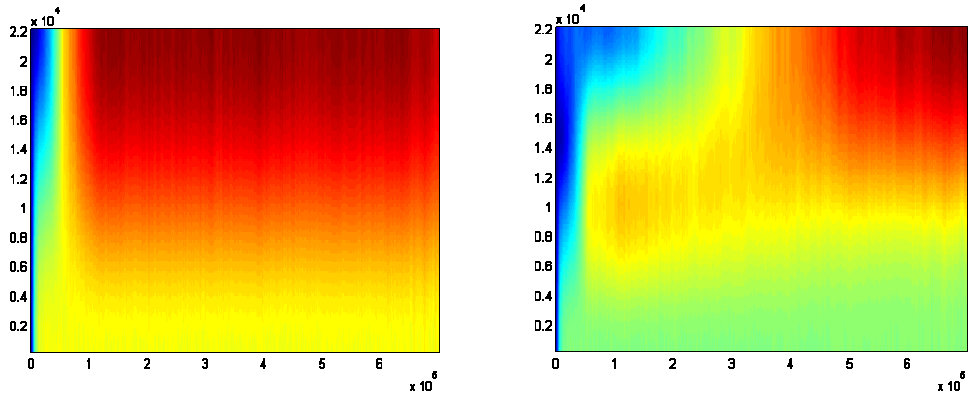


Fig. 4.12 i 4.13 A l'esquerra evolució del filtre de l'algoritme LRS-2 amb 2 coeficients μ = 0'0075 i a la dreta amb 3 coeficients i mateixa μ

4.5. Algoritme GRS eliminant soroll de motors de cotxes

Un cop més ens trobem davant d'un algoritme que convergeix quasi instantàniament, permetent-nos escoltar l'entrevista des del principi. Els cotxes continuen escoltant-se de fons però no molesten de cap manera. A més, no ha calgut retocar el valor de la μ inicial, de la β ni de M; tot i que si disminuïm un ordre de magnitud la μ inicial potser sentim l'entrevista amb una mica més de nitidesa, però això és una sensació subjectiva. Per tant, μ inicial = 0'00001, β = 1'25 i M = 2.

A la següent figura veurem l'evolució del filtre (recordem que l'algoritme GRS té un factor M de mostres menys a l'error). Es pot apreciar que el filtre no varia durant un seguit d'iteracions i després fa un canvi brusc, que queda lleugerament marcat a la figura, i que es pot apreciar per exemple a l'iteració $2'8 \cdot 10^4$ aproximadament. Això succeeix perquè en trobar una direcció adequada cap a l'error mínim la segueix fins que deixa de ser bona, que en aquest moment en busca una altra mitjançant una nova pertorbació aleatòria.

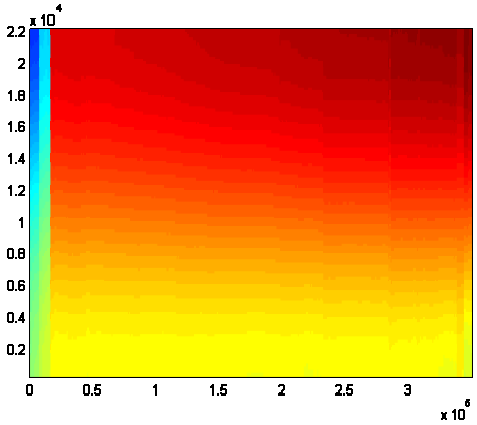


Fig. 4.14 Evolució freqüencial del filtre de l'algoritme GRS amb 2 coeficients, amb μ inicial = 0'00001, β = 1'25 i M = 2

4.6. Algoritme RLS eliminant soroll de motors de cotxes

Amb aquest algoritme tenim una convergència quasi instantània, independentment del valor de M. Amb dos o tres coeficients funciona correctament i podem escoltar amb nitidesa l'entrevista. A les següents figures veiem la similitud entre l'adaptació amb dos i tres coeficients. La diferència aparent provocada pel diferent color de les gràfiques és perquè hi ha una variació de l'escala de colors degut a què a la figura 4.15 hi ha un valor aïllat de magnitud elevada que provoca que les dues gràfiques tinguin marges diferents entre el valor màxim i mínim, cosa que provoca que les dues gràfiques acabin tenint tonalitats diferents. Però si ens fixem en l'evolució de les gràfiques sense tenir en compte el color, si no el canvi de tonalitat, podem concloure que l'adaptació amb dos i tres coeficients és la mateixa.

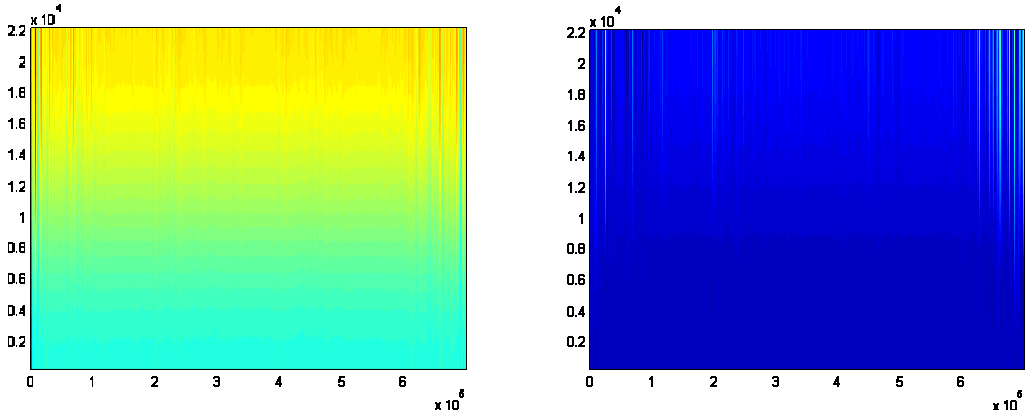


Fig. 4.15 i 4.16 A l'esquerra evolució del filtre de l'algoritme RLS amb dos coeficients i M = 1000 i a la dreta amb 3 coeficients i M = 100

Altres algoritmes, si augmentem els coeficients del filtre adaptatiu la qualitat de l'entrevista disminueix progressivament, a vegades imperceptiblement. En el cas del RLS és diferent. Si posem quatre o més coeficients l'algoritme sembla que elimini el soroll interferent però en un moment donat divergeix i evita que sentim a partir d'aquell moment l'entrevista. Això és degut a el que s'ha explicat a el subapartat 1.6.1., on es feia referència a possibles errors en l'algoritme RLS a l'hora de calcular R mitjançant (1.7), si escollíem inadequadament el nombre de coeficients del filtre.

4.7. Conclusions

Els algoritmes han demostrat tenir capacitat de resposta davant d'un escenari aparentment difícil de resoldre. Però en el fons, els algoritmes només han hagut de trobar la diferència entre els cotxes de la referència amb els de l'entrada per tal de poder cancel·lar l'una amb l'altra. I aquests canvis no eren res més que un filtratge amb un filtre fix de tres coeficients, un petit retard i un canvi d'amplitud. Per tant, tot i la quantitat de canvis entre els dos senyals de cotxes, la diferència real no és tanta.

A posteriori, hem provat els algoritmes amb un retard major, concretament de dos mostres i un terç. Els algoritmes són capaços d'eliminar pràcticament amb la mateixa qualitat aquest retard que l'anterior depenent d'on s'aplica el retard, si a l'entrada o a la referència. Si el retard és a l'entrada (com en l'escenari inicial del punt 4) els algoritmes no poden eliminar la interferència; mentre que si el mateix retard és a la referència si que la poden eliminar, i amb només 4 o 5 coeficients aconseguim una qualitat similar a l'escenari inicial.

La raó per la qual si el retard és a l'entrada no es pot eliminar és que els algoritmes que estem estudiant són FIR (Finite Impulse Response), que vol dir que la sortida del filtre està formada per mostres de l'entrada retardades de la següent manera:

$$y(n) = C_0\, x(n) + C_1\, x(n-1) + \ldots + C_M\, x(n-M) \qquad \textbf{(4.1)}$$

En conseqüència, a la sortida del filtre només hi trobarem una combinació lineal de la mostra actual i mostres retardades. Per tant, en el cas que el retard el posem a la referència, per un retard de D mostres i a la iteració n, el filtre haurà de trobar en la mostra n-D i en les M anteriors quelcom similar a el que tenim a la mostra n de la referència; cosa difícil ja que la iteració n encara no ha succeït a l'entrada.

Per contra, si el que va retardat és l'entrada, amb un retard de D mostres i a la iteració n, si que el filtre podrà trobar quelcom similar a la mostra n-D de la referència entre la mostra n, n-1, ..., n-M.

Pel cas concret que hem experimentat durant tot el punt quatre, el retard era tan petit que tot i estar retardada la interferència a la referència, el filtre és capaç de convergir, ja que és capaç de trobar quelcom molt similar a el que ha d'eliminar en les mostres 1/3 retardades.

CAPÍTOL 5. ALGORITMES ELIMINANT *FREQUENCY HOPPING:* ESCENARI VARIANT

Aquest capítol és un pas endavant més en la dificultat que li plantegem als filtres. Ara la interferència anirà variant de manera que el filtre adaptatiu hagi d'anar convergint a punts diferents a mida que passa el temps. En concret, a la referència hi trobarem un to que cada cert temps canvia de freqüència. Aquest canvi brusc farà que el filtre hagi de trobar la nova freqüència i eliminar-la. A l'entrada del filtre adaptatiu hi posarem un tren de deltes on cada delta és una de les possibles freqüències que la interferència podrà tenir durant els seus canvis de freqüència. També hi afegim soroll per fer més realista l'escenari, ja que sense ell el filtre té total llibertat en les freqüències on no hi ha deltes i, amb conseqüència, el filtre no és gens restrictiu a aquestes zones.

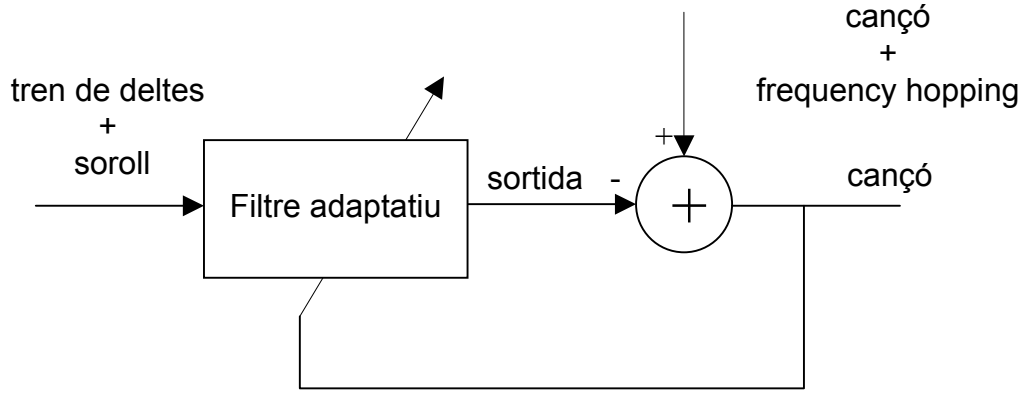


Fig. 5.1 Esquema bàsic dels algoritmes amb senyal d'entrada un tren de deltes i soroll, volent recuperar una cançó

Per a l'estudi d'aquest escenari ens fixarem en dues dades claus: la distància freqüencial entre deltes i la velocitat de canvi del *frequency hopping*. Per poder comparar els algoritmes, considerarem que un to és eliminat quan, abans del salt a una altra freqüència, el to hagi estat eliminat completament, independentment de la quantitat de soroll que tinguem a la cançó.

La distància entre deltes és important perquè a menor distància més restrictiu haurà de ser el filtre, ja que si no ho és prou, al voler eliminar una sola delta, deixarem passar pel mateix lòbul deltes properes, això provocarà que a la cançó de sortida no hi trobem el to a eliminar però si d'altres que el filtre ha hagut de deixar passar per poder eliminar el to inicial. El soroll que trobem a la cançó final la hi trobem d'una manera similar a aquesta, com a conseqüència de la impossibilitat d'eliminar un to amb un lòbul extremadament restrictiu, amb un ample de banda d'una delta.

D'altra banda, la velocitat amb què canvia de freqüència també és important, perquè es pot arribar al cas que variï més ràpid que la velocitat d'adaptació del filtre, amb el qual no tingui temps de convergir.

5.1. Algoritme LMS eliminant salts de freqüència

Primerament estudiarem quants coeficients necessita l'algoritme LMS per eliminar els tons depenent de la distància entre deltes. Amb una distància de 6 kHz, el filtre no té prou graus de llibertat per ser suficientment restrictiu fins que no té 8 coeficients. Amb 8 coeficients la velocitat d'adaptació és molt ràpida i el volum del soroll tot i que és molest ens deixa sentir la cançó correctament. A tall d'exemple, presentem la figura 5.2 i 5.3, on veiem una porció de cançó on hi ha una única freqüència a eliminar.

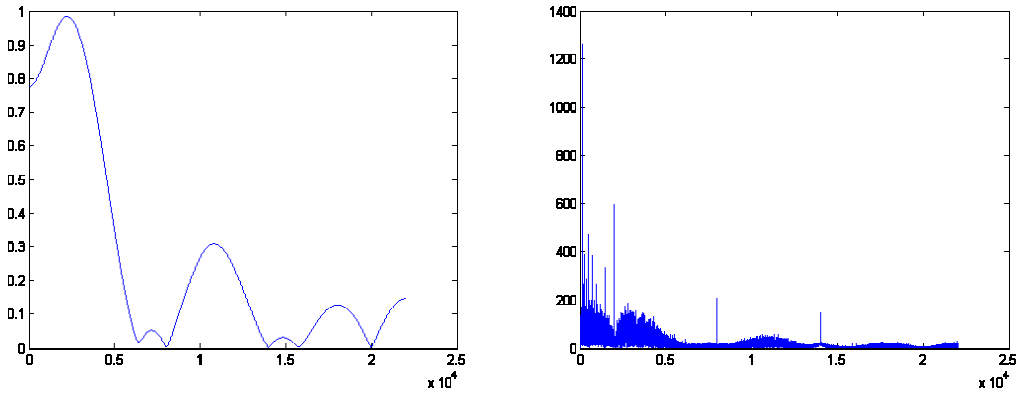


Fig. 5.2 i 5.3 A l'esquerra forma del filtre quan ha d'eliminar un to de 2 kHz. A la dreta l'error en el domini freqüencial

Podem veure quina és la forma que té el filtre quan ha d'eliminar (que és deixar-lo passar per poder-lo cancel·lar amb el to de la referència) un to de 2 kHz i no deixar passar tons de 8, 14 i 20 kHz. Amb aquesta distància l'escenari no és excessivament restrictiu i tal i com hem comentat, amb 8 coeficients n'hi ha prou perquè pugui eliminar el to desitjat, si bé veiem a la figura 5.3 com queden els tons de quan el filtre encara no havia convergit.

A mida que augmentem la dificultat el nombre de coeficients també han d'augmentar. En efecte, amb 4, 2 i 1 kHz de distància necessita 12, 18 i 32 coeficients per eliminar completament els tons.

5.2. Algoritme NLMS eliminant salts de freqüència

Els resultats obtinguts amb aquest algoritme són lleugerament inferiors al LMS si ens fixem amb el nombre de coeficients necessaris per eliminar completament els tons depenent de la distància entre deltes. Si bé amb 6 i 4

kHz el nombre de coeficients coincideixen, amb les següents distàncies el NLMS necessita de 20 i 36 coeficients.

Un exemple que podem posar és el de dotar de menys coeficients al filtre dels que hem comprovat que necessita. En el cas de 2 kHz en necessita 20, a la figura 5.4 veiem què passa si només en posem 14.

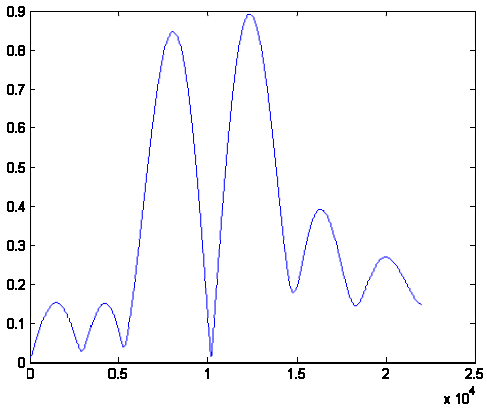


Fig. 5.4 Espectre freqüencial del filtre quan ha d'eliminar un to a 8 kHz

Està clar que té molts problemes per eliminar els tons no desitjats. Si del tren de deltes hauria d'eliminar els tons situats a 2, 4, 6 i 10 kHz, veiem que només pot eliminar-los en part ja que els mínims relatius de la gràfica més propers a aquestes freqüències estan a 0, 2'5, 5 i 10 kHz. És evident que si escoltem l'error no sentirem el to de 8 kHz però si que sentirem els altres tons que només han estat parcialment eliminats, excepte el de 10kHz. Però no només sentirem això, perquè si ens hi fixem, el lòbul més gran no té la funció d'eliminar cap freqüència, si no que allà només hi ha soroll que anirà a parar a la cançó.

5.3. Algoritme DSD eliminant salts de freqüència

El DSD en aquest escenari torna a donar resultats similars als altres algoritmes pel que fa al nombre de coeficients necessaris per eliminar els tons. Val a dir però que la manera escollida per comparar-los el beneficia, ja que no tenim en compte el temps que tarda en eliminar-los (que si que es tindrà en compte en l'altra comparació que es fa en l'apartat 5.7, la de la velocitat d'adaptació). De fet, amb la majoria de casos el DSD elimina per complet els tons poc abans que la freqüència salti (en aquest escenari això succeeix cada 50k mostres). Quan comparem la velocitat d'adaptació aquesta dada es demostrarà. A la figura 5.5 es representa una de cada 50 iteracions, per poder veure l'evolució del filtre.

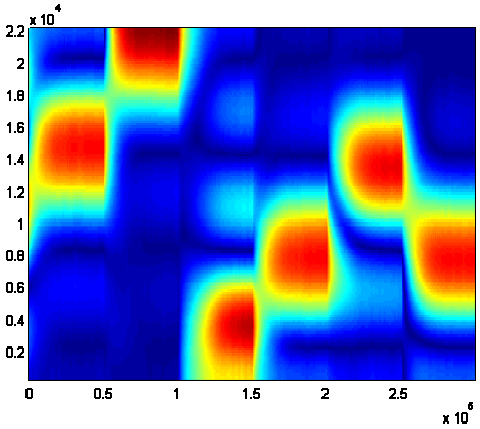


Fig. 5.5 Evolució del filtre de 8 coeficients amb distància entre deltes de 6 kHz, i canvi de freqüència cada cinquanta mil mostres

Podem veure com just després de cada canvi de freqüència el filtre tarda en convergir, com es demostra en les formes arrodonides del l'evolució del filtre, signe que a cada mostra va variant la resposta freqüencial del filtre i per tant encara no ha convergit. Per comparar-ho es pot mirar la figura 5.10 on veiem com l'evolució del filtre de l'algoritme RLS si que convergeix quasi instantàniament després de cada canvi de freqüència. De fet, el DSD és el que té més problemes en aquest sentit.

Finalment, dir que per les distàncies entre deltes que sempre utilitzem, els coeficients necessaris són respectivament 12, 22 i 36 coeficients.

5.4. Algoritme LRS-2 eliminant salts de freqüència

Un cop més hem comprovat que en les distàncies amplies, els diferents algoritmes sempre necessiten el mateix número de coeficients. Vuit per 6 kHz i dotze per 4 kHz. A partir de 2 kHz de marge és quan els millors algoritmes marquen la diferència, i és el que a priori podríem pensar del LRS-2, ja que només necessita 18 coeficients, que és el que menys en necessita juntament amb el LMS. Però cal fer una puntualització; si bé les condicions que es demanaven al principi es compleixen (que els tons siguin eliminats) cal dir que el LRS-2 és el pitjor algoritme pel que fa a quantitat de soroll a l'error. Com podem veure a la figura 5.6 l'espectre freqüencial del filtre permet eliminar el to desitjat i no deixar passar les altres deltes, però amb la forma que té allà on només hi ha soroll ja queda clar que el deixa passar, emmascarant la cançó de l'error.

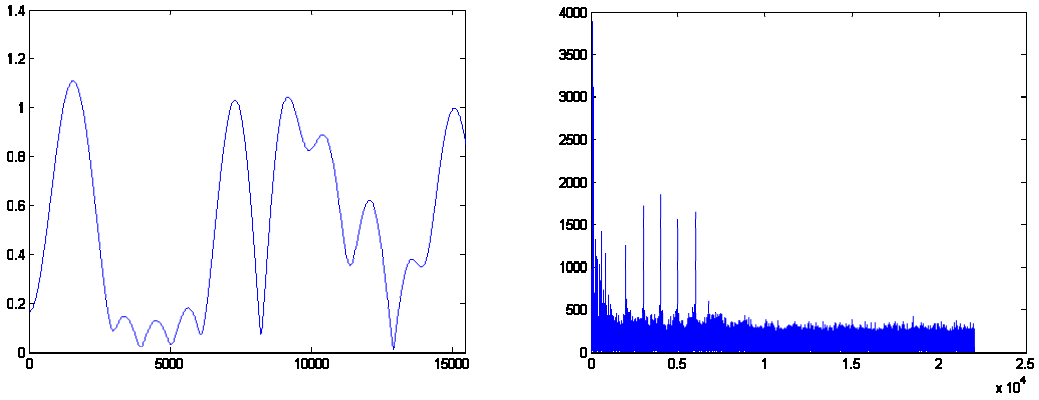


Fig. 5.6 i 5.7 A l'esquerra espectre freqüencial del filtre quan ha d'eliminar un to a 2 kHz. A la dreta espectre freqüencial de l'error

En la porció de la cançó on el *frequency hopping* tenia un to a 2 kHz, el LRS-2 troba com a millor solució la que veiem a la figura 5.6 on veiem el que ja avançàvem en l'anterior paràgraf; la quantitat de soroll que deixa passar a la sortida és massa alta. Auditivament sentim la cançó amb bastant dificultat. El volum de soroll es pot comprovar que és molt alt comparant la figura 5.7 amb la 5.3.

5.5. Algoritme GRS eliminant salts de freqüència

Quan a número de coeficients, és dels que més en necessiten. Fins a 38 pel cas més amb més complicació, si bé és cert que la qualitat obtinguda és superior a d'altres algoritmes que necessiten menys coeficients. A continuació mostrem a la figura 5.8 el cas de 1 kHz de separació.

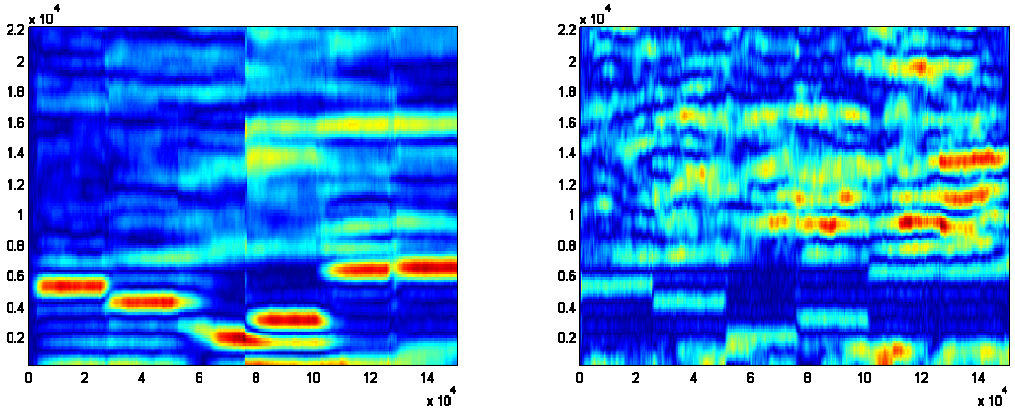


Fig. 5.8 i 5.9 Evolució del filtre de 38 coeficients amb distància entre deltes de 1 kHz i canvi de freqüència cada cinquanta mil mostres, a la esquerra amb μ = 0'0001 i a la dreta amb μ = 0'001

En aquest cas, les freqüències del tren de deltes es troben entre 2 i 6 kHz. Pel que fa al salt de freqüències la seqüència és la de 5, 4, 2, 3, 6 i 6 kHz. Podem veure com amb el pas d'adaptació adequat a cada salt el filtre deixa passar la freqüència desitjada mentre que en els altres tons, allà on just hi ha la delta hi veiem una línia negra que significa que no hi deixa passar res (obviant els errors evidents que fa aquest algoritme). En canvi amb un pas d'adaptació massa alt veiem com el filtre intenta eliminar les deltes necessàries però no ho fa correctament i a sobre deixa passar molt més soroll.

5.6. Algoritme RLS eliminant salts de freqüència

Amb l'escenari que hem estat comentant durant tot el capítol 5 el RLS dóna uns resultats molt bons auditivament parlant. Ara bé, si ens fixem amb els coeficients necessaris per aconseguir aquests resultats podem dir que està en el grup mig: ni és el pitjor ni està al grup capdavanter. I és que quan l'espai entre deltes és gran necessita els mateixos coeficients que la resta d'algoritmes però quan l'espai és més estret, amb 2 kHz necessita 20 coeficients i amb 1 kHz en necessita 36.

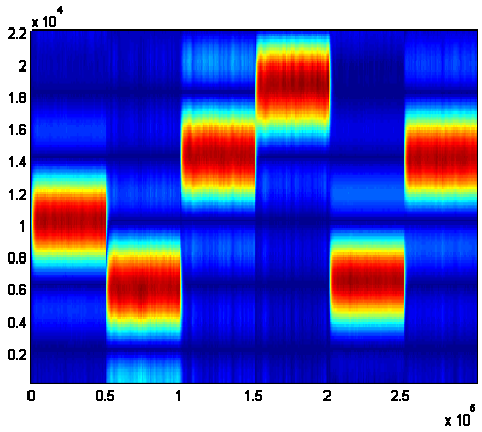


Fig. 5.10 Evolució del filtre de 12 coeficients amb distància entre deltes de 4 kHz i canvi de freqüència cada cinquanta mil mostres

5.7. Comparació i conclusions

En el capítol cinc teníem clar que volíem posar a prova els algoritmes amb escenaris canviants. Primer vam pensar amb posar, enlloc de un salt de freqüències, un *chirp* o to que va variant linealment en el temps. Malauradament els resultats no van ser gaire bons. Trobareu més informació d'això a l'annex.

Cenyint-nos a l'escenari que al final hem presentat a aquest capítol, podem fer una comparació de tots els algoritmes amb relació a la seva capacitat per eliminar els tons depenent de la distància entre deltes que tenim a l'entrada.

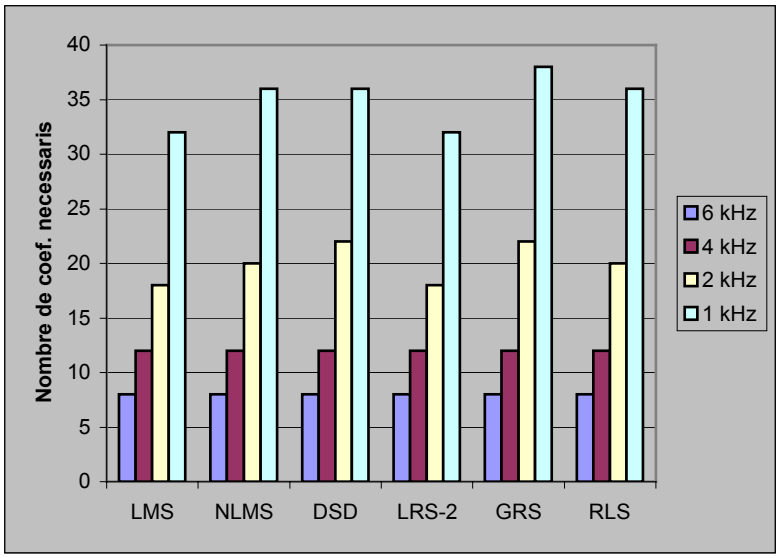


Fig. 5.13 Comparació de la selectivitat dels algoritmes

Així doncs, si ens fixem estrictament amb aquesta dada, podem concloure que els millors algoritmes són el LMS i el LRS-2, i els pitjors són el DSD i el GRS.

D'altra banda, podem comparar també quins algoritmes permeten els canvis de freqüència més ràpids. Escollim el cas de 4 kHz entre deltes ja que així tots els algoritmes estan en igualtat de condicions (tots necessiten 12 coeficients quan hi ha un salt de freqüència cada cinquanta mil mostres, tal i com acabem de veure a la figura anterior). Anirem disminuint progressivament la velocitat del *frequency hopping* de manera que podrem anar veient quins algoritmes són capaços de seguir el ritme i quins no. Considerarem que un algoritme pot seguir aquests canvis quan ha aconseguit eliminar el to abans que canviï a una altra freqüència. A la figura 5.14 podem veure aquesta comparativa, on es veu que els millors i els pitjors coincideixen amb els que havíem trobat amb la comparació de la selectivitat dels algoritmes.

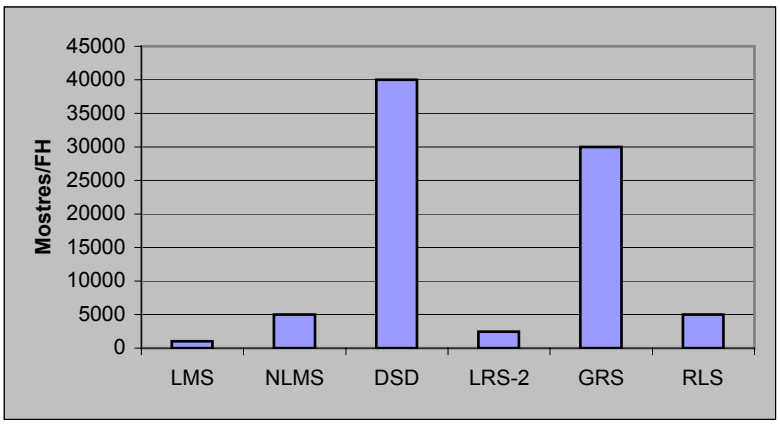


Fig. 5.14 Comparació de la capacitat per seguir els salts de freqüència

CAPÍTOL 6. ALGORITMES AMB LA REFERÈNCIA FILTRADA AMB UN FILTRE CANVIANT: ESCENARI VARIANT

En aquest capítol seguim amb la intenció de continuar provant els algoritmes sota escenaris variants. En el cas del circuit automobilístic, aquest cop, enlloc d'utilitzar salts de freqüència, el que farem és filtrar els cotxes de la referència amb un filtre que a cada iteració va canviant. Aquest canvi no podrà ser ni brusc ni aleatori, ja que en aquest cas els algoritmes no podran eliminar els cotxes.

Fixarem el nombre de coeficients dels filtres adaptatius, per poder comparar quins d'ells, a igualtat de coeficients, és capaç de seguir, amb més o menys encert, el canvi que introdueix el filtre variant. L'evolució la farem amb un filtre de 30 coeficients que durant les dues-centes mil mostres inicia en un valor de coeficients aleatori i va variant linealment fins a un filtre final també aleatori. Per a que els algoritmes puguin seguir aquest canvi, a part de garantir una velocitat dels canvis adequada, cal també aconseguir que la norma del filtre a cada iteració sigui la mateixa. Per aquest motiu el filtre final no acaba sent exactament mai al filtre definit aleatòriament amb antelació, ja que a cada iteració el fet d'ajustar la norma del filtre fa variar l'evolució del propi filtre. A la figura 6.1 trobem com serà l'esquema de l'escenari en qüestió.

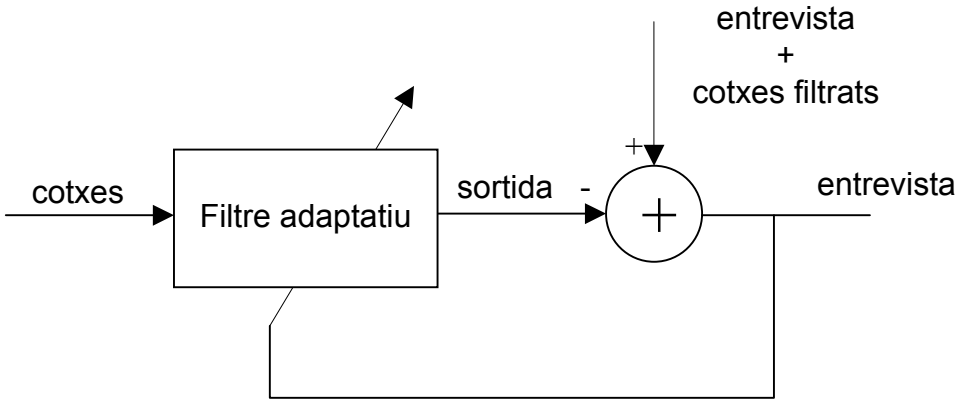


Fig. 6.1 Esquema bàsic dels algoritmes amb senyal d'entrada el soroll produït pels motors dels cotxes, volent recuperar una entrevista

En essència, el que haurà de realitzar el filtre adaptatiu per aconseguir eliminar el soroll de motor dels cotxes és desfer els canvis introduïts pel filtre variant, sempre i quan pugui seguir els canvis iteració a iteració. A priori necessitarien el mateix nombre de coeficients que el filtre que ha introduït els canvis; més tard veurem que no serà exactament així. A la figura 6.2 presentem l'evolució dels coeficients del filtre de la referència.

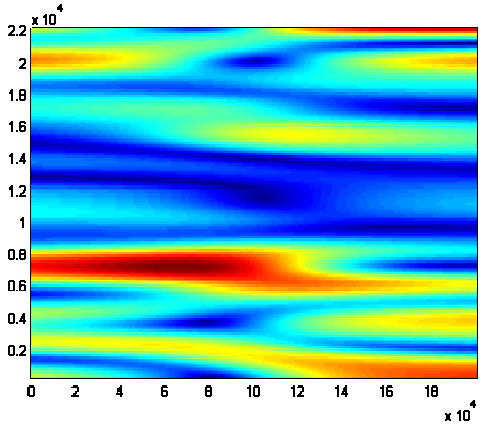


Fig. 6.2 Evolució del filtre variant de 30 coeficients

6.1. Algoritme LMS eliminant sons variants

Amb la velocitat del canvi del filtre que hem comentat a l'inici del capítol, el LMS pot seguir l'evolució d'aquest canvi fins al punt d'eliminar quasi completament el so interferent, ja que només el podem sentir en un discret segon pla, per sota de l'entrevista. El pas d'adaptació escollit ha estat el de μ = 0'01, ja que amb menys el soroll dels cotxes no era tan ben eliminat, i amb més la qualitat de l'entrevista disminuïa.

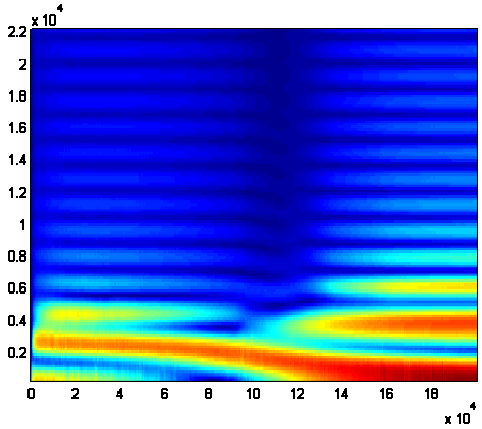


Fig. 6.3 Evolució del filtre LMS amb 28 coeficients i μ = 0'01

El LMS pot filtrar correctament amb dos coeficients menys respecte el filtre variant. Això és així perquè el filtre adaptatiu només elimina la diferència que introdueix el filtre de 30 coeficients a la zona freqüencial on hi havia el soroll dels cotxes (fins a 5 kHz aproximadament). A les altres zones, com que no hi ha res a eliminar la forma del filtre no té transcendència. Per això el filtre pot ser de menys coeficients, ja que no cal que segueixi l'evolució del filtre en totes les freqüències.

6.2. Algoritme NLMS eliminant sons variants

En aquest escenari, l'algoritme NLMS amb les constants adequades (α = 0'01 i β = 0'001) és capaç de reduir el nombre necessari de coeficients en 4 respecte el filtre de la referència, que en tenia 30. Per tant, suposa una millora respecte el LMS i com veurem més endavant, acabarà per ser el segon millor algoritme en l'escenari presentat. A part, no només cal fixar-se amb el nombre de coeficients, ja que també cal dir que auditivament la sensació de qualitat és major que molts altres algoritmes. A la figura 6.4 i 6.5 presentem les evolucions del filtre en el cas de tenir 26 coeficients i de tenir-ne 30.

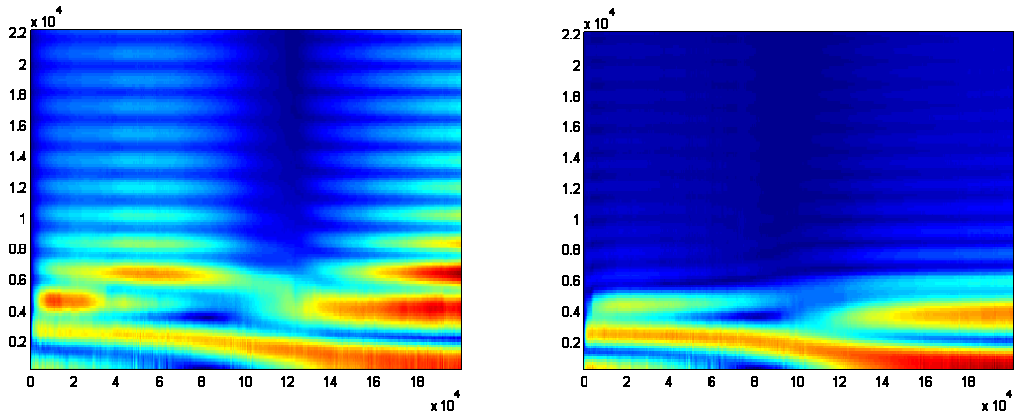


Fig. 6.4 i 6.5 Evolució del filtre NLMS amb 26 i 30 coeficients respectivament, amb α = 0'01 i β = 0'001

La millora que suposa fer-ho amb 30 coeficients respecte 26, és a la capacitat per copiar exactament els canvis introduïts pel filtre variant entre 0 i 5 kHz i no en el fet que en l'altra part de l'espectre els lòbuls siguin més o menys grans. Això seria un problema si a més dels cotxes, a l'entrada del filtre hi hagués soroll.

6.3. Algoritme DSD eliminant sons variants

En la majoria d'escenaris presentats el DSD és el que aconsegueix pitjors prestacions. En aquest, un cop més, l'eliminació del so interferent no arriba fins que dotem al algoritme de 30 coeficients. Amb 29 també se sent bé, exceptuant uns dos segons que és impossible poder entendre que s'està dient a l'entrevista degut a la no eliminació del soroll dels cotxes. A més, cal afegir-li el problema del temps de simulació. Amb 30 coeficients ha tardat set minuts i mig aproximadament per acabar la simulació. A la figura 6.6 mostrem l'evolució del filtre amb 30 coeficients, on es pot veure que el DSD opta per tenir una forma diferent als algoritmes anteriorment presentats allà on no hi ha res a filtrar, a partir dels 5 kHz. Això no seria problema si no fos que a la zona realment important no aconsegueix els resultats desitjats.

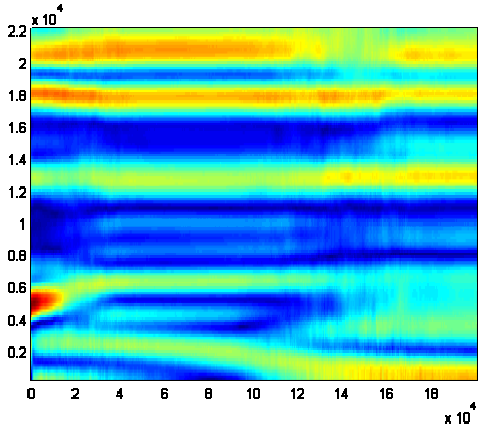


Fig. 6.6 Evolució del filtre DSD amb 30 coeficients i μ = 0'01

6.4. Algoritme LRS-2 eliminant sons variants

Un altre cas en què, com el DSD, a partir dels 5 kHz decideix tenir una forma del filtre que, si bé no afecta al resultat, cal dir que és totalment innecessària. El LRS-2 a partir dels 29 coeficients ja pot eliminar suficientment els soroll dels cotxes com per sentir l'entrevista durant tot el temps, però mai arribant als nivells de qualitat oferts per altres algoritmes com el LMS, NLMS o RLS.

A la figura 6.7 i 6.8 comparem la forma del filtre en l'espectre freqüencial a la mostra cent mil. Ens podem fixar que el valor dels punts dins de la zona compresa entre els 0 i els 5 kHz aproximadament, tenen el mateix valor i per tant podem dir que el filtre adaptatiu està copiant el que l'altre filtre ha fet al soroll dels cotxes de la referència per tal què, al restar les dues interferències, es cancel·lin.

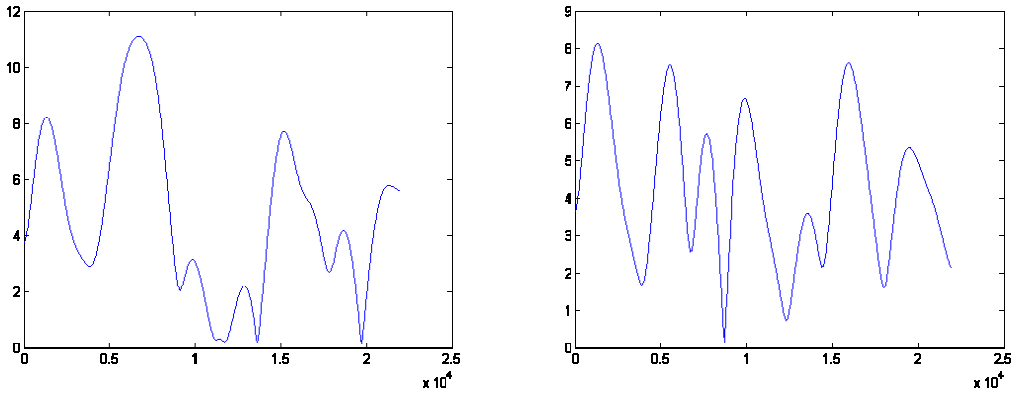


Fig. 6.7 i 6.8 Espectre freqüencial del filtre variant i del filtre adaptatiu amb 29 coeficients i μ = 0'0075 a la mostra cent mil

6.5. Algoritme GRS eliminant sons variants

L'algoritme GRS podem dir que el fet que utilitzi pertorbacions aleatòries fa que s'hagi de tenir en compte que sempre hi ha la possibilitat que tot i utilitzar suficients coeficients el filtre no elimini la interferència. Tenint en compte això, direm que el nombre de coeficients mínim ha de ser de 29. Com a mostra d'aquesta incertesa, mostrem dues figures on, tot i tenir totes dues 29 coeficients, la primera pot eliminar el soroll i l'altra no. En el cas que no aconsegueix eliminar del tot (figura 6.10), es pot arribar a distingir que dins de l'ample de banda d'allà on hi ha el soroll dels motors dels cotxes, els colors que informen de la situació del filtre en aquesta figura no ressalten tant respecte els colors d'allà on no hi ha res, sobretot a l'inici i al final. També es pot apreciar els canvis provocats pel canvi de pertorbació aleatòria (sobretot a la figura 6.9), ja que això és un canvi brusc que es veu reflectit en canvis bruscs en la forma de l'espectre freqüencial del filtre.

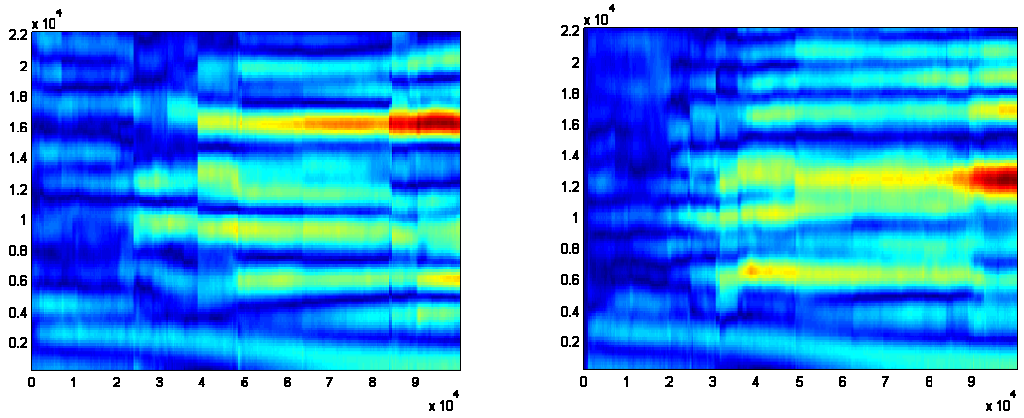
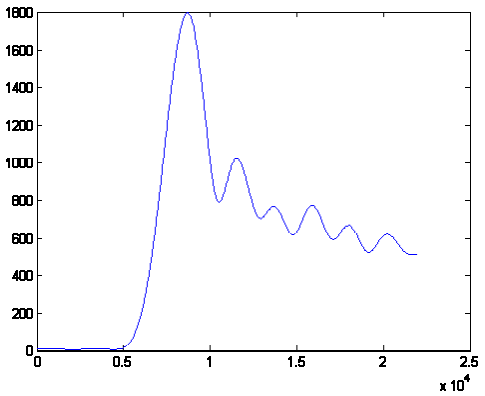


Fig. 6.9 i 6.10 Evolucions del filtre GRS amb 29 coeficients, μ = 0'001 i β = 1'25

6.6. Algoritme RLS eliminant sons variants

Sense dubte, el millor algoritme en aquest apartat. La velocitat de convergència a l'inici és immediata, cosa que es tradueix en què sembla que a l'escoltar l'error ja hagi convergit des de l'inici. A més, la qualitat de l'entrevista no queda malmesa i els cotxes pràcticament no se senten si utilitzem 26 coeficients o més. Tot i això, podem considerar que el filtre ja funciona correctament amb 23 coeficients. A més, té la particularitat d'utilitzar una *estratègia* diferent a l'hora de conformar la forma de l'espectre freqüencial del filtre. Si bé els altres intentaven emular el filtre variant en els primers 5 kHz i en l'altra zona deixar-ho en zero, el RLS el que fa és, evidentment, en l'ample de banda d'interès fer el mateix que els altres però en l'altra zona el filtre obté valors molt alts. Si ens fixem a la figura 6.11, i a la 6.12 després de fer un zoom, podem veure que els valors de la zona d'interès són els esperats.



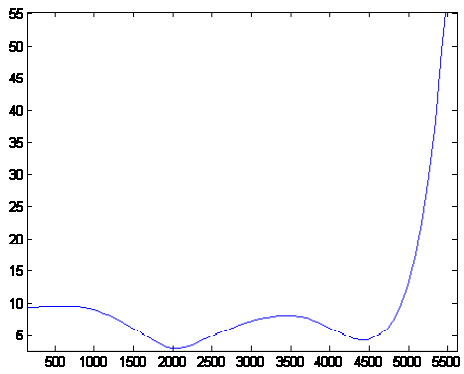


Fig. 6.11 i 6.12 Forma del filtre a la mostra cent cinquanta mil del filtre

6.7. Conclusions

Inicialment, al fer aquest capítol vam començar utilitzant un nombre de coeficients del filtre variant baix (vam provar entre 3 i 10). De seguida vam veure que tots els algoritmes podien eliminar el soroll dels cotxes amb pocs coeficients, fins i tot amb 1 o 2 en segons quins casos. A més, el filtre resultant vèiem que tenia una forma plana. Això succeïa perquè al tenir pocs coeficients feia que el filtre en la zona on la interferència està situada tingués poca variació, ja que els canvis són més suavitzats quan menys coeficients hi ha. Aquests canvis podien ser emulats pels filtres amb un únic valor per tot l'espectre freqüencial ja que el filtre de tres coeficients dins de la zona on hi ha la interferència fa variar tots els punts d'aquesta interferència de manera igual.

Amb la introducció de més coeficients, dins de l'espectre freqüencial del soroll produït pels motors cotxes trobarem més variació que farà que no tota aquesta zona variï igual. Per tant, de manera indirecta afegim dificultat als nostres algoritmes. Això s'ha notat perquè si amb el cas de tres coeficients tots els algoritmes podien recuperar l'entrevista nítidament (amb l'excepció del DSD, que necessitava més coeficients) amb la inclusió de més coeficients hem notat que la diferència entre els algoritmes es fa palesa. De fet, podríem fer una classificació de millor a pitjor en quan a la resolució de l'escenari presentat: En primer lloc es troba el RLS que, amb 23 coeficients pot oferir una qualitat que d'altres no poden ni amb 30, ja que el nivell dels cotxes és baix i l'entrevista queda intacta, cosa que sembla no passar amb la majoria dels altres algoritmes. En segona posició trobaríem el NLMS, que necessita 26 coeficients i la qualitat és acceptable. Prop d'ell, està el LMS, amb 2 coeficients més. Ja més endarrerits queden LRS-2, GRS i DSD, per aquest ordre. Necessiten més coeficients, i la qualitat de l'entrevista queda bastant malmesa en algun d'ells.

Per últim comentar que cap dels algoritmes és capaç de tenir a l'error l'entrevista totalment neta de cotxes (el RLS és el que està més a prop d'aconseguir-ho però se sent lleugerament el so dels cotxes de fons). Per tant, cap algoritme pot reproduir amb exactitud els canvis introduïts per l'altre filtre, tot i que amb els mateixos coeficients teòricament haurien de ser capaços de copiar el filtre anterior i per tant recuperar l'entrevista de manera perfecta.

CAPÍTOL 7. INTRODUCCIÓ ALS ALGORITMES IIR

En aquest capítol hem creat un escenari on teòricament, un filtre FIR hauria de tenir problemes per a eliminar la interferència. Això ho aconseguirem filtrant els cotxes amb un filtre FIR format per dos coeficients, 1 i -0'9. Com que filtrem la interferència amb un filtre FIR, el filtre ideal per desfer aquests canvis és un filtre IIR. Evidentment, si usem un FIR aconseguirem eliminar la interferència però amb major dificultat. De fet, hem comprovat que amb l'algoritme LMS fins que el filtre no té 12 coeficients no es pot percebre amb claredat l'entrevista.

Per comparar si els algoritmes IIR milloren als FIR en l'escenari descrit, utilitzarem primer el filtre LMS dels anteriors apartats amb quatre coeficients i μ = 0'01 per tal de veure fins a quin punt és capaç d'eliminar el so interferent. Comparant la referència i l'error (figures 7.1 i 7.2) podem apreciar que el filtre FIR de l'algoritme LMS elimina lleugerament el soroll dels cotxes. Auditivament però es pot comprovar que l'entrevista es continua sense sentir quan el cotxe passa a prop de l'entrevistador.

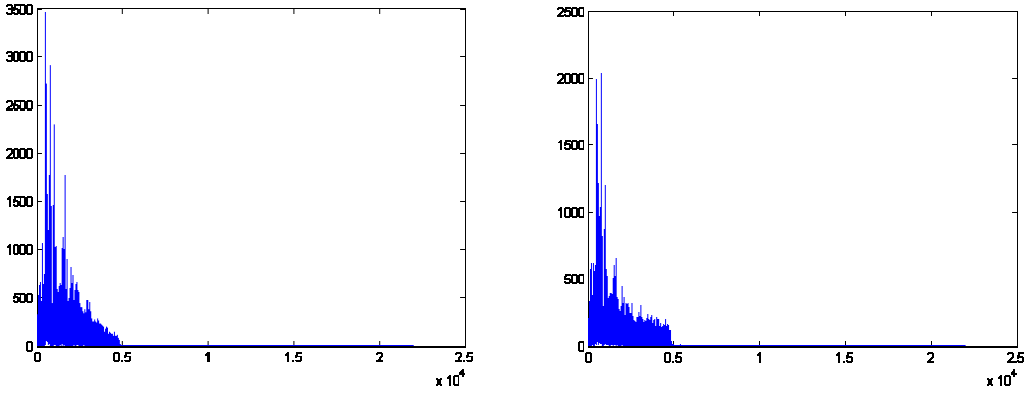


Fig. 7.1 i 7.2 Espectre freqüencial de la referència de l'escenari i error quan utilitzem el filtre FIR de l'algoritme LMS

Un cop hem vist de què és capaç l'algoritme LMS amb quatre coeficients, programem un filtre IIR anomenat Feintuch's IIR LMS, que és una aproximació de l'algoritme IIR LMS (veure [10] i [12]). Com tots els filtres IIR té uns coeficients que tenen la mateixa funció que els coeficients d'un filtre FIR, i uns altres que són la part realimentada. Per tant ara, la sortida del filtre no és només un sumatori de mostres retardades de l'entrada, si no que també de la pròpia sortida:

$$y(n) = C_0\, x(n) + C_1\, x(n-1) + \ldots + C_M\, x(n-M) + \\ + D_1\, y(n-1) + D_2\, y(n-2) + \ldots + D_N\, y(n-N+1) \qquad \textbf{(7.1)}$$

L'actualització dels coeficients en aquest primer algoritme IIR que programem és trivial perquè es fa de la mateixa manera que ho fem amb els algoritmes

FIR. Per tal de poder comparar amb igualtat de condicions amb el FIR, utilitzarem el mateix valor pel pas d'adaptació i el mateix nombre de coeficients (dos *feedback* i dos *feedforward*). Els resultats, com veurem a la següent figura, no són els esperats. El so interferent té pràcticament el mateix nivell que amb el filtre FIR i, efectivament, a l'escoltar-ho sentim el mateix que sentíem abans.

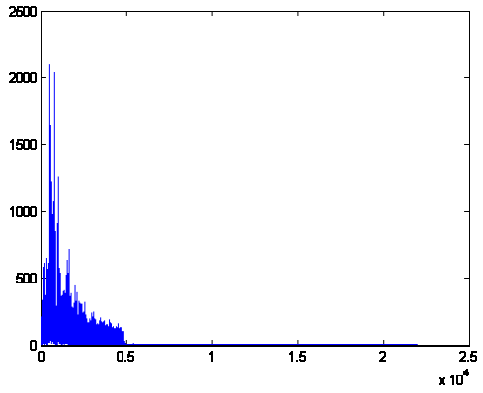


Fig. 7.3 Espectre freqüencial de l'error del filtre Feintuch's IIR LMS

Buscant la millora del resultat de l'escenari programem el IIR LMS per complet, sense cap aproximació. La regla d'aprenentatge és igual que (1.2) però enlloc de μ tenim α pels coeficients de la part FIR i β dels de la realimentada:

$$\alpha_n(k) = x(k-n) + \sum_{J=1}^{M-1} D_J(k) * \alpha_n(k-j) \qquad \textbf{(7.2)}$$

$$\beta_n(k) = y(k-m) + \sum_{J=1}^{M-1} D_J(k) * \beta_m(k-j) \qquad \textbf{(7.3)}$$

La millora no és suficient per sentir bé la cançó, com es pot comprovar:

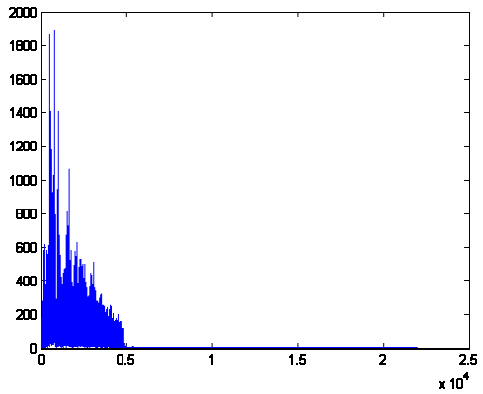


Fig. 7.4 Espectre freqüencial de l'error del filtre IIR LMS

BIBLIOGRAFIA

[1] Lagunas M.A., Apunts de "Procesado de señal" de la ETSETB 3B disponible a http://www-cttc.es/publications/undergraduate cap I, IV i V.

[2] Haykin S., *Adaptive Filter Theory,* Prentice-Hall, second edition 1991.

[3] Irizar Picón A., Apunts de "Tratamiento digital del señal" disponible a http://www.tecnun.com/asignaturas/tratamiento%20digital/frametds5.html

[4] Boyd S., Chua L.O. and Desoer C.A., "Analytical Foundations of Volterra series", *IMA Journal of Mathematical Control & Information* 1*,* 243-282 (1984)

[5] --, Apunts de "Filtrado Adaptativo: El algoritmo LMS", disponible a www.gtas.dicom.unicam.es/miembros/nacho/Doctorado/Calidad/filt_adap.pdf

[6] Dalla Vecchia G., *Una estructura en cascada FIR para predicción lineal adaptiva* disponible a http://iie.fing.edu.uy/ense/asign/tes/ materiales/monografias/PrediccionLineal.pdf

[7] Meler Ferraz L., *Variantes del algoritmo LMS. Aplicación de un sistema cancelador de ecos.* (2005) disponible a http://personal.auna.com/lorenzo.diaz/ ermele/trabajos/trabajo_fin_de_carrera.pdf

[8] López Valcarce R., *Adaptive Algorithms for identification and equalization using recursive filters.* (2001) disponible aquest i d'altres publicacions esmentades més avall a http://www.gts.tsc.uvigo.es/~valcarce/

[9] López Valcarce R., Mosquera C., Pérez González, F., "Hyperstable adaptive IIR algorithms with polyphase structures: Analysis and design"., *IEEE transactions on signal processing* 47 (7), 2043-2046 (1999).

[10] Mosquera C., Gómez J.A., Pérez F., Sobreira M., "Adaptive IIR filters for active control", *Sixth Internacional Congress On Sound And Vibration* (1999) disponible a www.gts.tsc.uvigo.es/gpsc/ publications/mosquera/icsv99.pdf

[11] Apunts de "Adaptive Noise Cancellation" disponible a http://www.owlnet.rice.edu/~ryanking/elec431/algos.html

[12] Stewart B., Apunts de "Adaptive IIR filtering" de *5th IEE Adaptive Signal Processing Professional Development Course* disponible a www.iee.org/oncomms/pn/signalprocessing/bobstewart2.pdf

ANNEXOS

A.1. Evolució de la convergència dels algoritmes

En aquest primer capítol de l'annex mostrarem les gràfiques de convergència d'alguns algoritmes depenent de la constant que defineix el pas d'adaptació.

Primer veiem com l'algoritme LMS té un bon comportament a partir de μ = 0'05 aproximadament; i ho manté fins a 0'5, que és quan ja deixa de convergir.

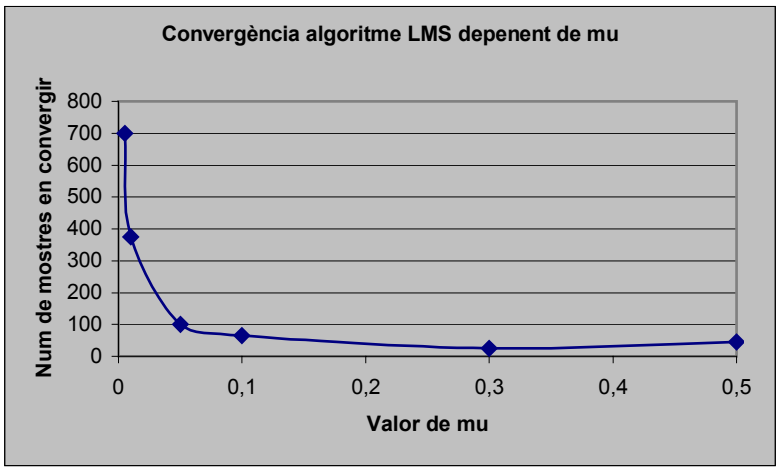


Fig. A.1 Evolució de la convergència de l'algoritme LMS depenent de μ

Pel que fa al NLMS, amb β fixada, hem anat provant diversos valors de α, tenint en compte que, igual que β, els seus valors han d'estar entre 0 i 1. A la figura A.2 en podem veure els resultats.

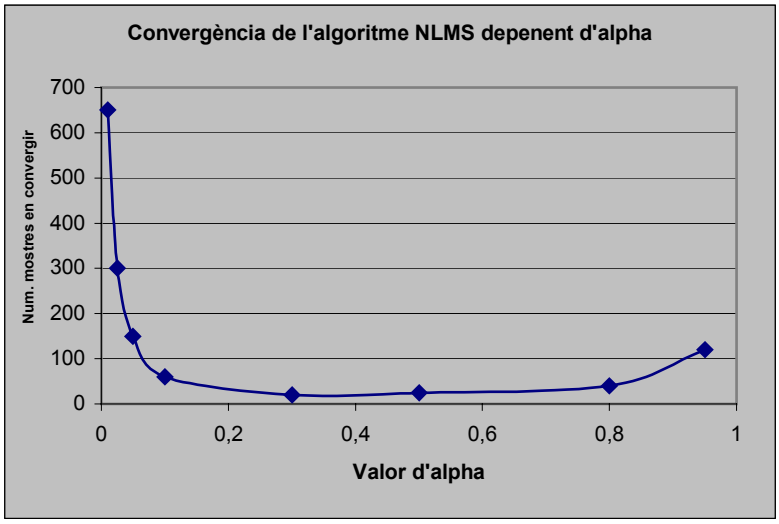


Fig. A.2 Evolució de la convergència de l'algoritme NLMS depenent de α

L'evolució de la convergència amb α fixada no la presentarem, ja que la variació no és significativa, si bé s'observa una millora amb β molt propera a 1; 0'99, per exemple.

L'evolució de la convergència de l'algoritme LRS-2 la mostrem a la figura A.3. Obviem la zona no recomanable que hem explicat al capítol 1.4.2 ja que ja la hem descartat.

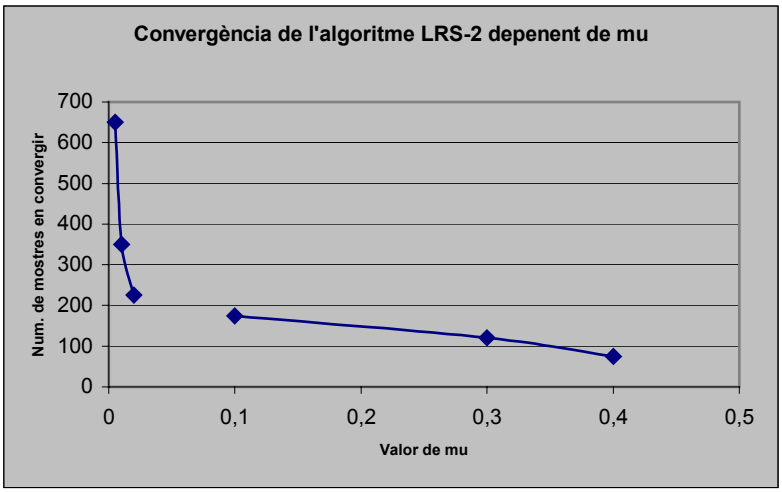


Fig. A.3 Evolució de convergència de l'algoritme LRS-2 depenent de μ

La següent gràfica (figura A.4) tracta de l'algoritme GRS. La gràfica en qüestió ens diu a quina mostra considerem que ha convergit. Però cal tenir en compte que aquest algoritme necessita per a cada iteració M entrades. Així, si M = 10 com el nostre cas, per β = 2, convergeix realment en mitjana a la mostra 270 i no 27 com diu la gràfica.

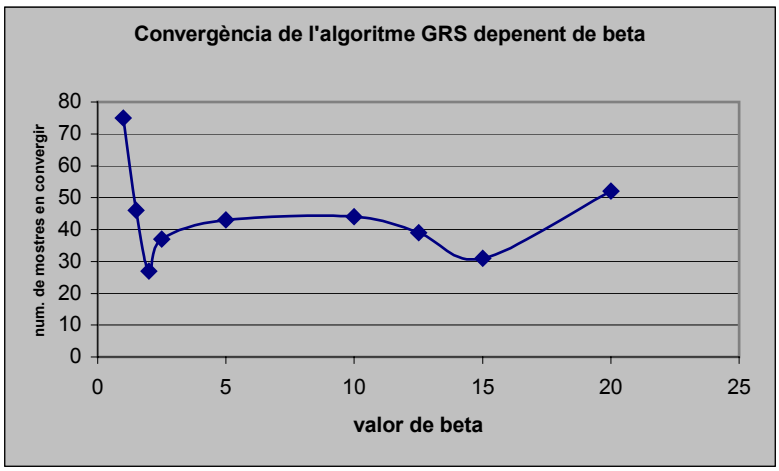


Fig. A.4 Evolució de la convergència de l'algoritme GRS depenent de β

Pel que fa a l'estudi del pas d'adaptació inicial òptim hem conclòs que el valor no és ni molt menys crític. De fet, l'única restricció que posaríem és que tingui un valor entre 0'05 i 0'5. Per sota de 0'05 si que es nota una reducció de la velocitat de convergència, mentre que per sobre de 0'5 en ocasions no convergeix. Per tant, entre aquest rang és preferible escollir la més baixa, ja que amb μ altes es comença a notar el desajust.

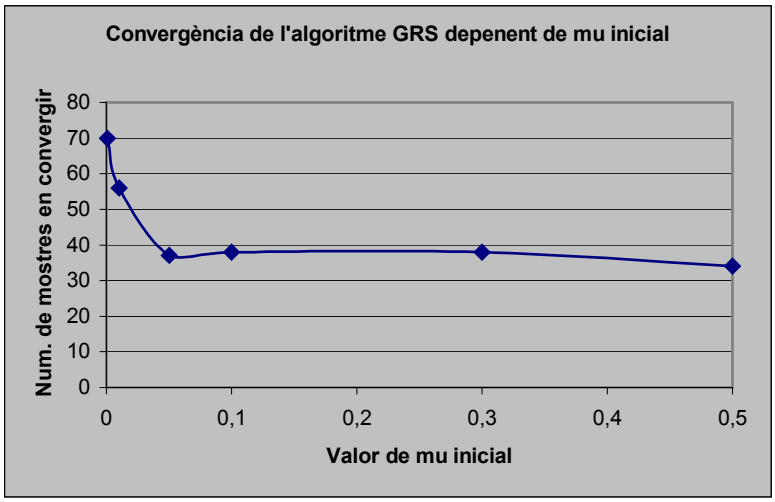


Fig. A.5 Evolució de la convergència de l'algoritme GRS depenent de μ inicial

Per últim, a la figura A.6 trobem l'evolució de la convergència de l'algoritme RLS. Noti's que la línia traçada és pràcticament lineal, la qual cosa vol dir que quan més augmentem el valor de M, més mostres tarda en convergir; ja que M és precisament el número de mostres que s'utilitza al principi de l'algoritme per calcular la matriu de correlació de dades. Aquestes mostres inicials doncs, queden descartades i l'algoritme comença a adaptar-se a partir de la mostra M+1.

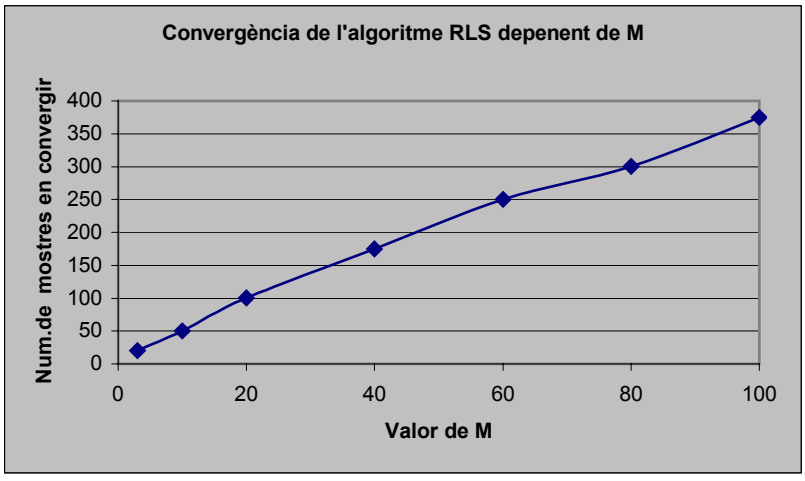


Fig. A.6 Evolució de la convergència de l'algoritme RLS depenent de M

A.2. Explicació del càlcul del desajust

Per comprovar el desajust D, hem creat el següent sumatori:

$$D = \frac{1}{N-1}\sum_{k=1}^{N} | M_{k+1} - M_k | \qquad \textbf{(A.1)}$$

Es tracta d'anar sumant en valor absolut la resta de cada mostra M respecte l'anterior, durant les N últimes mostres; i dividir-ho pel nombre de mostres menys una. Quan hem usat això (en el capítol 1) ho hem fet amb les 200 últimes mostres, d'un total de 1000. Perquè en aquestes últimes mostres ja es pot considerar segur que ja ha convergit.

A.3. Explicació de la variant de l'algoritme LMS: LMS-2

Per programar l'algoritme LMS-2 cal utilitzar l'ordre *sign* per guardar només el signe de l'entrada i de l'error. Això és molt potent ja que aquesta ordre és només un bit, de manera que amb dos bits (dos *sign*) ja tindrem tota la informació necessària per utilitzar la regla d'aprenentatge.

La pèrdua d'informació que creem pel fet de voler tenir un algoritme més ràpid computacionalment afectarà a la rapidesa en la convergència. A part, aquesta pèrdua d'informació fa que ara no sabem en quina mesura hem d'avançar en la direcció adequada per arribar a l'error mínim. Si comparem el LMS amb el LMS-2 ens hem fixat que els valors de la μ en l'algoritme LMS són d'un ordre de magnitud més gran que en el LMS-2. Això és degut a la informació que perdem al utilitzar l'ordre *sign*: Aquesta ordre només ens dóna informació de si el resultat és positiu o negatiu, i ho guarda amb un 1 o -1. Donat que els valors que realment tenim són normalment més petits que 1 en valor absolut, aquest augment d'ordre de magnitud que fem indirectament, l'hem de contrarestar amb una μ més petita. Això es pot demostrar posant la μ òptima del LMS, 0'3, en l'algoritme LMS-2. Es pot comprovar que no convergeix.

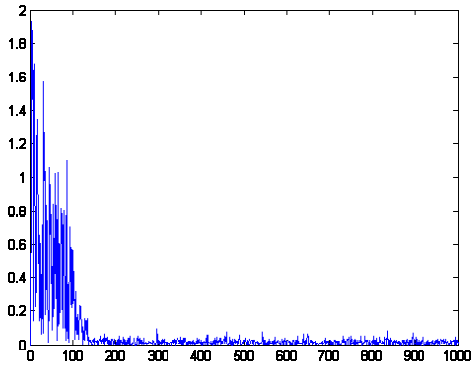


Fig. A.7 Algoritme LMS-2 amb μ = 0'01.

A.4. Explicació i resolució de l'error del RLS al capítol 2

Com expliquem al capítol 2, el RLS falla si utilitzem el codi que fins ara havíem utilitzat. Si mirem la figura A.8 podem veure que fins aproximadament abans de la mostra 40000 funciona correctament però a partir d'aquesta mostra passa a tenir un valor exageradament alt. Una solució possible a això era augmentar el valor de M, ja que quan més alt era més retardàvem aquesta situació, i podíem arribar a situar aquest punt més enllà de la llargada del vector d'entrada; però aquesta solució no era ni molt menys la idònia.

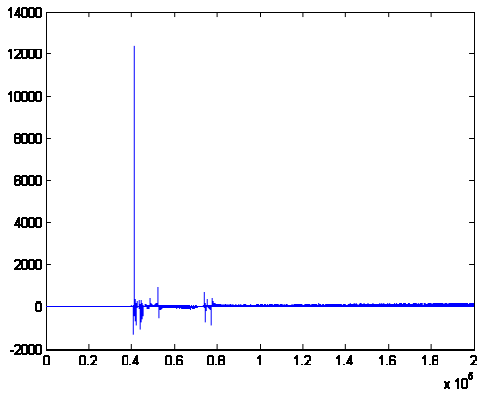


Fig. A.8 Error de l'algoritme RLS quan succeeix el problema de l'acumulació d'errors degut a les constants de l'algoritme

Després de diverses proves, hem arribat a la conclusió que el problema era l'acumulació d'errors que provocava que, quan aquest error era prou important, el càlcul de la inversa de R fos a cada iteració a poc a poc més mal calculada, fins que desembocava en la situació de la mostra 40000. Per esmenar això, hem tractat d'ajuntar en una mateixa línia de codi totes les constants, per d'aquesta manera, poder tenir una única constant simplificada (degut a que hi havia algunes constants que després de l'acumulació estaven en el numerador i denominador a la vegada), tractant així de poder controlar millor aquesta acumulació d'error. Fent això hem comprovat com el problema desapareixia.

A.5. Explicació de possible escenari pel capítol 5

En un principi, enlloc dels salts de freqüència, s'havia pensat en canvis de freqüència provocats per un *chirp* per a l'escenari del capítol 5. A diferència del salt de freqüències, el canvi és continu al llarg d'una regió freqüencial. El problema que teníem era que les zones que no estaven representades per una delta en el tren de deltes no les podia eliminar. Vam pensar en possibles solucions i una d'elles va ser la posar tantes deltes com fos possible, però llavors el problema evident és que el filtre no pot ser prou selectiu com per escollir una d'aquestes deltes sense deixar passar cap de les altres. Una altra possible solució va ser la de posar menys deltes i utilitzar un filtre no lineal. Aquests filtres tenen la particularitat de crear a la sortida, a partir del senyal de

l'entrada, senyals que no es tenen a l'entrada. Ens vam informar sobre les sèries de Volterra (veure [4]), però aquest mètode tampoc va tenir gaire èxit per la mateixa raó d'abans, és difícil crear a partir d'unes quantes deltes tots les freqüències possibles entre dos punts. A la figura A.9 podem veure el *chirp* i a la A.10 el resultat amb l'algoritme LMS de 40 coeficients.

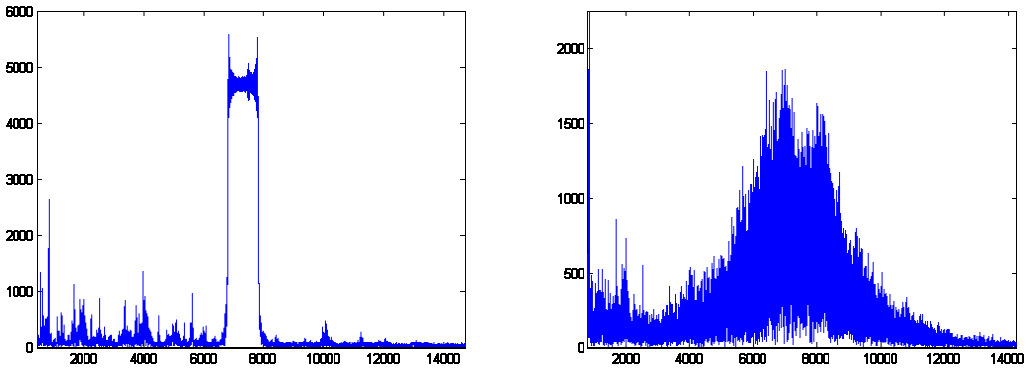


Fig. A.9 i A.10 A l'esquerra un *chirp* de 6 a 8 kHz sobre una cançó i a la dreta el resultat obtingut a l'error