Laboratori de Comunicacions II

PRÀCTICA 7:

CANCEL·LACIÓ d'ECO ACÚSTIC

Marta Casar i López

CANCEL·LACIÓ d'ECO ACÚSTIC

Introducció

El disseny d'un cancel·lador de l'eco acústic creat en un recinte tancat, tal com una sala o un cotxe, no disposa generalment del senyal sense eco per a fer-lo servir com a senyal de referència. Així doncs, s'han de desenvolupar altres estratègies que utilitzin el propi senyal rebut amb eco per a cancel·lar l'efecte no desitjat.

En aquesta pràctica s'implementarà un cancel·lador adaptatiu d'eco acústic amb el DSP TMS320C6173. El senyal acústic amb eco estarà sintetitzat per un altre DSP. L'objectiu final de la pràctica es pot esquematitzar amb el següent esquema de blocs que consta clarament de dues parts: la generació de l'eco i la seva cancel·lació.

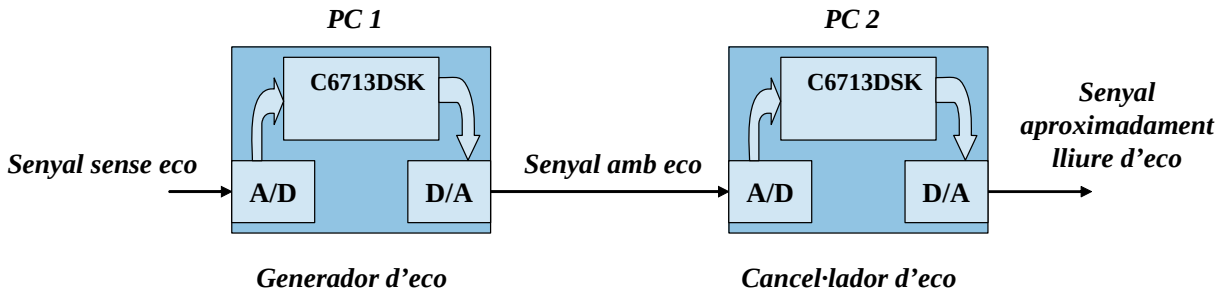

Figura 1. Esquema final de la pràctica

1. Generador d'eco

1.1. Estudi previ: Generador d'eco

L'efecte del recinte que produeix l'eco es modelarà mitjançant un filtre AR tal i com el de la següent figura que consta de *La* coeficients **a** = [*a*(0), *a*(1),…, *a*(*La*-1)].

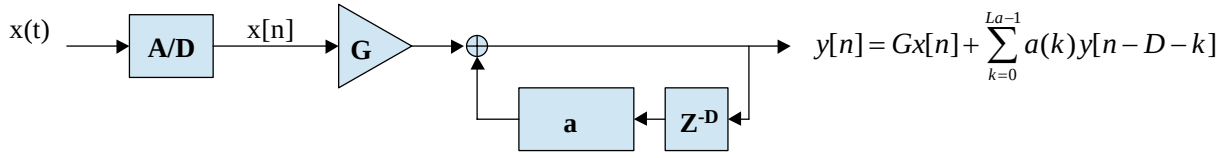

Figura 2. Model del generador d'eco.

Per tal de que l'eco sigui audible cal que el retard del filtre AR sigui aproximadament major que 0.3s. Tenint en compte que la freqüència de mostratge és de 8KHz, trobar el valor mínim que ha de tenir *D*.

Suposant que *La* = 11, proposar un filtre AR que sigui estable de manera que només el coeficient *a*(*j*) sigui diferent de zero, on *j* pren un valor entre 0 i *La*-1. Notar que modificant l'amplitud del coeficient *j* es poden crear diferents efectes d'eco. Obtenir el valor del guany *G* per a normalitzar a 1 la resposta freqüencial del generador d'eco.

1.2. Al laboratori: Generador d'eco.

L'objectiu d'aquest primera part és doble. Per una banda, aprendre a generar un projecte nou del Code Composer Studio (CCS). El segon consisteix en sintetitzar un eco com el de la Figura (2) a partir del senyal que entra des de la ràdio i escoltar-lo amb els auriculars.

Li suggerim que desenvolupi el projecte nou a partir d'algun ja existent que funcioni correctament i que tingui una estructura semblant a la que cal implementar. Com a indicació, li facilitem el fitxer *eco1.c* a l'Annex I que li pot servir de patró per a generar el fitxer principal del seu projecte. És important notar que al fitxer *eco1.c* cal afegir totes les instruccions que consideri oportunes per a generar un projecte nou que funcioni amb el CCS. A l'Annex II trobarà les indicacions que cal seguir per a crear un projecte nou amb el CCS

Pel fet que $D>>N$, on N és la longitud dels blocs d'entrada i sortida, en el fitxer *eco1.c* es defineix un buffer auxiliar *y_gen*[*M*] suficientment gran com per a emmagatzemar les mostres del senyal de sortida del generador d'eco necessàries per a implementar el filtre AR de la Figura (2). Raonar quant ha de valer M en funció de N, La i D.

2. Cancel·lador d'eco

2.1. Estudi previ: Cancel·lador d'eco

La figura següent representa l'esquema proposat del cancel·lador, en el que es suposa que D és conegut.

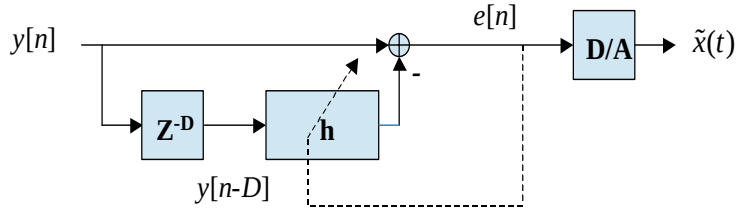

Figura 3. Model del cancel·lador d'eco.

A partir de les Figures (2) i (3), trobar la resposta impulsional teòrica del filtre cancel·lador d'eco, denotat pel conjunt de coeficients **h** = [$h(0)$, $h(1)$,…, $h(Lh-1)$], en funció dels coeficients **a** = [$a(0)$, $a(1)$,…, $a(La-1)$] del generador d'eco. Particularitzar aquest resultat pel cas d'un sol coeficient $a(j)$ diferent de zero tal i com s'ha proposat a l'apartat (EP.1) en funció de l'amplitud del coeficient i el retard associat $D + j$.
Per a aquest cas, demostrar que la solució de Wiener, obtinguda de minimitzar la potència instantània de la sortida del cancel·lador $e[n]$ respecte del coeficient $h(k)$, coincideix amb la solució teòrica sota la hipòtesi de que $x[n]$ està incorrelada amb $y[n-D]$. A continuació, proposar un disseny adaptatiu dels coeficients $h(k)$ del filtre que cancel·la l'eco utilitzant el LMS. Justifiqui el rang teòric de la constant de pas μ de l'algorisme per assegurar la convergència de l'algorisme en mitja.

2.2. Al laboratori: Cancel·lador d'eco adaptatiu

En aquesta part es tracta de desenvolupar un nou projecte a partir del que ha dissenyat per a generar l'eco que generi i cancel·li l'eco, tot en un mateix DSP. Per a això es disposa dels programes *eco2.c* i *eco2.h* de l'Annex III com a pauta per a generar el seu projecte.

La subrutina *can_eco( )* cal que implementi un cancel·lador d'eco adaptatiu de *Lh* coeficients amb l'algorisme adaptatiu NLMS (tingui en compte les conclusions a les que ha arribat a la secció 2.1). Notar que aquest cancel·lador d'eco ha de cancel·lar l'eco que conté el senyal d'entrada al buffer *io_block* i escriure el resultat a les mateixes posicions. (*io_block* també és un buffer d'entrada i sortida per aquesta subrutina).

L'algorisme adaptatiu NLMS es diferencia del LMS en que la constant de pas μ es normalitza per una estimació de la potència del senyal d'entrada obtinguda de la següent manera:

$$\mu_{NLMS} = \frac{\mu}{p[n] + p\_\min}; \; p[n] = \beta\, p[n-1] + (1-\beta)\|\mathbf{u}\|^2$$

on $\mathbf{u} = \left[y[n-D], y[n-D-1], \cdots, y[n-D-Lh-1] \right]$ representa el vector format les mostres d'entrada que es convolucionen amb els coeficients del filtre cancel·lador, *p_min* és un llindar mínim de potència per evitar divisions per valors massa petits i *n* denota el temps.

Per a fer les proves, considerar el generador d'eco d'un sol coeficient diferent de zero proposat a la secció 1.1. En aquesta situació provar un filtre cancel·lador adaptatiu. Escoltar la cancel·lació amb els auriculars i comprovar amb el depurador la convergència del vector *h* cap al valor teòric esperat.

A continuació, i un cop ha mostrat al professor el funcionament del cancel·lador, considerar un cancel·lador amb més coeficients (aquest cas correspondria a una situació més real en la que se sap el retard inicial de l'eco però dins un cert marge). Provar-ho per exemple amb La=Lh=100 coeficients, augmentant aquest nombre fins al límit màxim de funcionament en temps real.

3. Cancel·lador d'eco amb 2 DSPs

3.1. Cancel·lació de l'eco generat en un altre DSP

Crear dos executables: un que només generi eco i un altre que només cancel·li eco. Es tracta d'intentar cancel·lar l'eco generat per uns companys amb un altre DSP amb el cancel·lador adaptatiu dissenyat seguint la connexió de la Figura 1. Per un funcionament correcte és important que la sortida del DSP que genera l'eco no estigui saturada però que intenti aprofitar tot el rang dinàmic que admet el conversor A/D del l'altre DSP. Comprovar amb el depurador l'evolució dels coeficients i amb els auriculars tant l'eco generat pels companys com la cancel·lació. És capaç d'endevinar el retard i amplitud amb què s'està generant l'eco?

A continuació, els companys que generen l'eco poden canviar amb el depurador el valor dels coeficients del filtre generador d'eco. Comprovar com el cancel·lador d'eco adaptatiu reacciona en front de la nova situació. Finalment provar la situació inversa.

Annex I

```
......

#include "eco1.h"

.....

float io1[N], io2[N];        /* Reserva espacio para los buffers
float *io_block,*io_buffer;  /* Puntero del bloque a procesar
int index=0;

float y_gen[M];          /* Buffer de memoria interna del generador de eco */
float a[La];             /* Buffer dels coeficients del generador d'eco*/

.....

void wait_buffer(void)
{
    float *ptmp;                /* Puntero auxiliar                     */
    while(index < N);           /* Espera a que se llene el buffer      */
    ptmp = io_buffer;           /* Intercambia punteros:                */
    io_buffer = io_block;       /*     nuevo buffer de entrada          */
    io_block = ptmp;            /*     nuevo bloque de datos de entrada */
    index = 0;                  /* Inicializa index                     */
}

void init_proc_eco(void)
{
   /* Inicialización coeficientes a */
   /* Inicialización y_gen */
}

void eco_gen(float * io_block, float * y_gen, float * a)
{
...
}


void main(void)
{
   /* Inicialización */

    while (1)
    {
      wait_buffer();
      eco_gen(io_block, y_gen, a);
    }
}
```

ECO1.C

```
#define La 11
#define N 200
#define D 2500
#define M (N+D+La-1)

/* Completar definición de las nuevas funciones */
void init_proc_eco...
void gen_eco...
```

ECO1.H

Annex II

Com generar un projecte nou pel Code Composer Studio (CCS)

- Obri el CCS

- Generi un Projecte Nou obrint “Project/New...” amb el nom que vulgui i ubicat a l’escriptori. Esculli al camp “Target” el *TMS320C67XX*. El CCS genera automàticament una carpeta amb el nom que ha indicat.

- Fora del CCS; copiï en aquesta carpeta els següents fitxers agafant-los d’alguna pràctica anterior:
    - *vectors_intr.asm* (Fitxer amb les adreces de les interrupcions).
    - *c6713dskinit.c* i *c6713dskinit.h* (Fitxers d’inicialització i funcionament general de la placa).
    - *C6713dsk* (Fitxer .cmd).
    - *dsk6713bsl.lib* , *dsk6713_aic23.h* , *dsk6713.h* (Llibreries d’ús general).
    - *eco1.c* i *eco1.h* (Fitxers que cal editar seguint les pautes de l’Annex I i inspirant-se en algun projecte ja existent).

- Dins del CCS, afegeixi al seu projecte els següents fitxers:
    - A la carpeta *Source* cal afegir *eco1.c, c6713dskinit.c* i *vectors_intr.asm.*
    - A la carpeta *Documents* cal afegir *eco1.h*.
    - A la carpeta *Libraries* cal afegir *dsk6713bsl.lib.*
    - A la carpeta principal del projecte cal afegir *C6713dsk*

- Executi “Project/Scan All File Dependencies”.

- Per últim, modifiqui totes les opcions del compilador i del “linker” com les que s’indiquen a continuació que trobarà a “Project/Build Options...”

- Un cop fet això el fitxer ja es pot compilar i “linkar” amb l’opció “Project Build”

Annex III

```
...
#include "eco2.h"

.....

float io1[N], io2[N];        /* Reserva espacio para los buffers */
float *io_block,*io_buffer;  /* Puntero del bloque a procesar */
int index=0;

float y_gen[M];          /* Buffer de memoria interna del generador de eco */
float y_can[M];          /* Buffer de memoria interna del cancelador de eco*/
float a[La];             /* Buffer de coeficientes del generador de eco*/
float h[Lh];             /* Buffer de coeficientes del cancelador de eco*/

.....

void wait_buffer(void)
{
    float *ptmp;               /* Puntero auxiliar                     */
    while(index < N);          /* Espera a que se llene el buffer      */
    ptmp = io_buffer;          /* Intercambia punteros:                */
    io_buffer = io_block;      /*     nuevo buffer de entrada          */
    io_block = ptmp;           /*     nuevo bloque de datos de entrada */
    index = 0;                 /* Inicializa index                     */
}

void init_proc_eco(void)
{
   /* Inicialización coeficientes a i h*/
   /* Inicialización y_gen , y_can */
}

void eco_gen(float * io_block, float * y_gen, float * a)
{
/* Copiar eco_gen de ECO1.c */
}
void can_eco(float * io_block, float * y_can, float * h)
{
/* Completar código */
}

void main(void)
{
   /* Inicialización */

    while (1)
    {
      wait_buffer();
      eco_gen(io_block, y_gen, a);
      can_eco(io_block, y_can, h);
    }
}
```

ECO2.C

```
#define La 11
#define Lh La
#define N 200
#define D 2500
#define M (N+D+La-1)

#define PMIN 10000
#define BETA 0.95
#define MU 0.01

/* Completar definición de las nuevas funciones */
void init_proc_eco...
void eco_gen...
void can_eco...
```

ECO2.H