Exemples bàsics de Processament Digital de Senyal amb Matlab

Matlab és el software de càlcul científic per excel.elència. Entre d'altres moltes utilitats, conté eines per al tractament de senyals i imatges (representació gràfica, càlcul de transformades de Fourier i moltes més). El major 'inconvenient" de Matlab és el seu preu. Matlab està instal.lat als ordinadors del Laboratori de Telemàtica i farem les pràctiques amb ell.

Si algú vol fer pràctiques pel seu compte i treballar a nivell domèstic una alternativa és utilitzar Scilab, un 'clon' gratuit (software lliure) de Matlab i que té eines similars per al tractament dels senyals. Per a més informació sobre Scilab i la seva instal.lació: `http://www.scilab.org/`.

En aquest document mostram alguns exemples de processament bàsic de senyals de veu amb Matlab. Els senyals de veu (o música) utilitzats es troben en format .wav i s'han obtingut d'Internet (`http://www.thefreesite.com/Free_Sounds/Free_WAVs/` i `http://www.moviewavs.com/`). També és possible crear fitxers .wav amb la gravadora de sons de Windows (al menú Todos los programas/Accesorios/Entretenimiento) o descarregar fitxers d'audio dels diferents podcasts que es troben en l'actualitat. El format habitual dels fitxers d'audio dels podcasts és .mp3. En aquest enllaç podeu trobar un conversor on-line gratuit de .mp3 a .wav: http://media.io/

Començam per executar Matlab. S'obre una finestra amb vàries seccions: explorador del directori actual, finestra de comandes, espai de treball, història de comandes. És convenient establir el directori de treball (el que conté tots els fitxers que haurem d'obrir i on es guardaran els nous fitxers) com a directori actual, en la part de dalt de la finestra de Matlab.

1 Operacions bàsiques

Obrim un senyal de veu emmagatzemat a l'ordinador:

```
[x, Fs]=wavread('gomaespuma1.wav');
```

Els valors del senyal queden emmagatzemats en l'array x. $x[1]$ conté la primera mostra del senyal, $x[2]$ la segona i així successivament. El nombre total de mostres (longitud de l'array) es pot calcular fent

```
Nt=length(x)

 ans =

     8640000
```

F_s és la freqüència de mostreig del senyal (Hz). $F_s = 1/T$, on T és el periode de mostreig:

```
Fs

 Fs  =

    44100
```

La durada (en segons) del senyal és per tant:

```
Nt/Fs

 ans =

  195.9184
```

El senyal es pot escoltar (només en PC, no en Mac) per comprovar que s'ha carregat correctament:

```
wavplay(x, Fs);
```

(Nota: en Mac, pot utilitzar Quicktime player per reproduir els sons. En Windows es pot utilitzar la instrucció 'wavplay' o el reproductor de Windows).

Si executam la comanda 'size' veurem que l'audio està format per dos canals (stereo). Ens quedarem amb només un d'ells i visualitzarem el senyal:

```
size(x)

 ans =

     8640000           2

xmono=x(:,1);

figure, plot(xmono);
```

A continuació seleccionam una part del senyal original, de durada 5.2 segons, que correspon amb el principi de l'audio, la representam gràficament i la guardam en disc:

```
cut=xmono(1:round(5.2*Fs));

figure, plot(cut);

wavwrite(cut, Fs, 'gme1.wav');
```

Podem provar què passa si guardam el senyal amb una frqüència de mostreig diferent de l'original:

```
wavwrite(cut, Fs*2, 'gme2.wav');
wavwrite(cut, Fs/2, 'gme3.wav');
wavwrite(cut, Fs*0.75, 'gme4.wav');
```

Suposem que volem afegir una música de fons al senyal de veu. Per fer això obrim un nou fitxer .wav que contengui la música desitjada i seleccionam un bocí de la mateixa duració que el seleccionat abans (en aquest cas començan a seleccionar a partir del segon 8.8):

```
[stw, Fs2]=wavread('imperial.wav');

cut2=stw(round(round(8.8*Fs2):round((8.8+5.2)*Fs2)));

wavwrite(cut2, Fs2, 'stw.wav');
```

Podem esborrar algunes mostres del començament del só per crear un silenci al començament del só. En aquest cas suprimim aproximadament el primer mig segon de l'audio (5000/Fs2=0.4535 segons):

```
cut2(1:5000)=0;

wavwrite(cut2, Fs2, 'stw.wav');
```

Abans de mesclar les dues seqüències ens hem d'assegurar que les freqüencies de mostreig són iguals, però:

```
Fs

 Fs =

       44100

Fs2

 Fs2 =

       11025
```

Per poder mesclar els senyals les distàncies entre mostres (període de mostreig) han d'ésser les mateixes. Haurem de reduir per 4 el periode de mostreig del primer senyal i llevar una mostra del segon senyal perquè ambdues seqüències tenguin la mateixa longitud;

```
cut1=cut(1:4:end);

wavwrite(cut1, Fs2, 'gme1b.wav');

cut2=cut2(1:length(cut));
```

Ara ja podem sumar els dos senyals, ponderant el pes de la música de fons fins que quedi al nostre gust:

```
suma=cut1+cut2;

wavwrite(suma, Fs2, 'out1.wav');

suma=cut1+0.5*cut2;

wavwrite(suma, Fs2, 'out1.wav');

suma=cut1+0.1*cut2;

wavwrite(suma, Fs2, 'out1.wav');
```

2 Anàlisi en freqüència

Una eina molt útil per a l'anàlisi de senyals és la FFT (Fast Fourier Transform), una implementació ràpida de la transformada de Fourier discreta (DFT) i que ha contribuït de manera decisiva al desenvolupament del Processament Digital de Senyals dels darrers 40 anys. En tots cas, a nivell teòric no hi ha cap diferència entre la FFT i la DFT. Recordem algunes propietats de la transformada de Fourier de senyals discrets que ens permetran entendre els resultats que obtindrem:

- La DFT d'un senyal discret (periòdic) $x[n]$ de N mostres (periode N) és un nou senyal discret $X[k]$ de N mostres.
- $X[k]$ és un senyal periòdic de periode N.
- La transformada de Fourier (en temps discret) d'un senyal discret (no periòdic) de N mostres és un senyal continu $X(\omega)$.
- $X(\omega)$ és un senyal periòdic de periode 2π.
- La relació entre $X[k]$ i $X(\omega)$ és: $X[k] = \frac{1}{N}X(\frac{2\pi}{N}k)$, per a $k = 0, 1, \cdots, N-1$. És a dir, $X[k]$ s'obté per discretització de $X(\omega)$, agafant N mostres del periode principal.
- El senyal discret $x[n]$ es considera ben mostrejat si el periode de mostreig T cumpleix la condició de Nyquist: $1/T \geq 2B$, on B és la freqüència màxima del senyal continu original $x_a(t)$. Això equival a dir $B \leq F_s/2$, és a dir, que la freqüència màxima del senyal continu és inferior a la meitat de la freqüència de mostreig ($F_s = 1/T$). En aquest cas no hi ha aliasing i el senyal original es pot recuperar a partir de les mostres.
- Per a un senyal ben mostrejat $X_a(\Omega) = TX(\omega)$ entre $\omega = -\pi$ i $\omega = \pi$.
- La relació entre les freqüències discretes ω i les freqüències contínues Ω és $\Omega T = \omega$, on T és el periode de mostreig. Això significa que $NTX[k]$ és el valor de la component espectral de freqüència angular $\Omega = \frac{2\pi}{NT}k = \frac{2\pi F_s}{N}k$ del senyal continu original (on F_s és la freqüència de mostreig). En freqüència (no angular): $F = \frac{k}{NT} = k\frac{F_s}{N}$.
- La transformada de Fourier $X(\omega)$ d'un senyal $x[n]$ real té les següents propietats de simetria: $|X(-\omega)| = |X(\omega)|$, fase$(X(-\omega)) = -$fase$(X(\omega))$.

A continació calculam i representam la FFT del senyal de veu:

```
cutfft=fft(cut);

figure, plot(abs(cutfft));
```

Observem la simetria de l'espectre, deguda a que el senyal de veu pren valors reals. A més, veim que el senyal está ben mostrejat perquè no s'observa aliasing a la part central de l'espectre.

Podem comprovar com ha variat l'espectre del senyal quan s'ha submostrejat per poder-lo mesclar amb la música de fons:

```
cut1fft=fft(cut1);

figure, plot(abs(cut1fft));
```

Observam un petit aliàsing, gairebé sense importància ja que la qualitat del só és semblant a l'original (i el tamany del fitxer s'ha reduit per 4).

Si seguim submostrejant observam com l'efecte de l'aliasing es fa més evident i la qualitat del só empitjora (però el tamany del fitxer d'audio també s'ha reduit, la qual cosa implica un estalvi d'espai de disc). Observem que per guardar el fitxer s'ha d'utilitzar una freqüència de mostreig meitat de l'original, ja que s'ha reduit per dos el nombre de mostres.

```
cut1b=cut1(1:2:end);

cut1bfft=fft(cut1b);

figure, plot(abs(cut1bfft));

wavwrite(cut1b, Fs2/2, 'gme1b2.wav');
```

Ara podem fer alguns tests modificant (eliminant, atenuant o amplificant) selectivament algunes de les components espectrals dels senyals d'entrada.

Per exemple, eliminarem les components espectrals per damunt de la freqüència 2Khz. Per a això, primer hem de calcular quines mostres de la fft es veuran afectades (hem de tenir en compte la simetria de la fft i el fet que la numeració dels arrays en Matlab comença a partir de 1):

```
cut2fft=fft(cut2);

figure, plot(abs(cut2fft));

fcont=2000;

N=length(cut2);

k1=N*fcont/Fs2+1;

k2=N+2-k1;

cut2fftb=cut2fft;

cut2fftb(k1:k2)=0;

figure, plot(abs(cut2fftb));

cut2b=real(ifft(cut2fftb));

wavwrite(cut2b, Fs2, 'stw2.wav');
```

Notes:

- Aquest tipus de filtratge sembla que correspon a un filtratge ideal del senyal d'entrada. No obstant el filtratge ideal és irrealitzable. El que en realitat passa és que es convoluciona el senyal d'entrada amb una versió amb aliasing temporal del filtre ideal. D'una altra banda, el resultat obtingut en aplicar la ifft no es correspon amb la convolució del senyal discretitzat original amb la resposta impulsional del filtre, ja que la ifft d'un producte de senyals no és igual a la convolució habitual, sino a la convolució circular dels senyals.
- Una manera d'evitar l'aliasing és, abans del submostreig del senyals, filtrar-los amb un filtre passa-baix amb la freqüència de tall ajustada a Fs/k. D'aquesta manera no es produeix aliasing al submostrejar per un factor k (si bé l'eliminació de freqüències altes produeix un altre efecte anomenat **efecte de Gibbs**).

En la secció 'Filtratge de renou' es donen més detalls relatius al filtratge en freqüència.

Seguint un procés similar a l'anterior podem atenuar o amplificar (en l'exemple, per un factor 5) algunes freqüències:

```
cut2fftb=cut2fft;

cut2fftb(k1:k2)=5*cut2fftb(k1:k2);

figure, plot(abs(cut2fftb));

cut2b=real(ifft(cut2fftb));

wavwrite(cut2b, Fs2, 'stw3.wav');
```

Finalment, anam a veure l'efecte d'afegir a un senyal d'entrada amb un to pur (un senyal sinusoidal). Per exemple, afegirem un cosinus amb freqüència 5Khz:

```
fcont=5000;

N=length(cut2);

k1=N*fcont/Fs2+1;

k2=N+2-k1;

cut2fftb=cut2fft;

cut2fftb(k1)=1000;

cut2fftb(k2)=1000;

figure, plot(abs(cut2fftb));

cut2b=real(ifft(cut2fftb));

wavwrite(cut2b, Fs2, 'stw4.wav');
```

3 Algunes aplicacions del Processament Digital de Senyals

3.1 Filtratge de renou

Obrim un senyal contaminat amb renou, el reproduim, el visualitzam i visualitzam el seu espectre:

```
[r, Fs3]=wavread('exemplerenou.wav');

figure, plot(r);

rfft=fft(r);

figure, plot(abs(rfft))
```

Intentam eliminar renou amb un filtre de mitjana, provam amb filtres de longitud 3, 5, 11 i 31. Observam (i escoltam) els resultats en temps i freqüència.

```
h=[1/3 1/3 1/3];

r1=conv(r, h);

wavwrite(r1, Fs3, 'r1.wav');

figure, plot(r1), axis([0 300000 -0.8 0.8]);

r1fft=fft(r1);

figure, plot(abs(r1fft))

h=[1/5 1/5 1/5 1/5 1/5];

r2=conv(r, h);

wavwrite(r2, Fs3, 'r2.wav');

figure, plot(r2), axis([0 300000 -0.8 0.8]);

r2fft=fft(r2);

figure, plot(abs(r2fft))

h=[1/11 1/11 1/11 1/11 1/11 1/11 1/11 1/11 1/11 1/11 1/11];

r3=conv(r, h);

wavwrite(r3, Fs3, 'r3.wav');

figure, plot(r3), axis([0 300000 -0.8 0.8]);

r3fft=fft(r3);

figure, plot(abs(r3fft))

h=1/31* ones(31,1);

r4=conv(r, h);

wavwrite(r4, Fs3, 'r4.wav');

figure, plot(r4), axis([0 300000 -0.8 0.8]);

r4fft=fft(r4);

figure, plot(abs(r4fft))
```

Ara eliminarem el renou amb un filtre de mediana i compararem els resultats amb els anteriors. Ho farem amb filtres del mateix tamany que els anteriors.

```
r1m=medfilt1(r, 3);

wavwrite(r1m, Fs3, 'r1m.wav');

figure, plot(r1m), axis([0 300000 -0.8 0.8]);

r1mfft=fft(r1m);

figure, plot(abs(r1mfft))

r2m=medfilt1(r, 5);

wavwrite(r2m, Fs3, 'r2m.wav');

figure, plot(r2m), axis([0 300000 -0.8 0.8]);

r2mfft=fft(r2m);

figure, plot(abs(r2mfft))

r3m=medfilt1(r, 11);

wavwrite(r3m, Fs3, 'r3m.wav');

figure, plot(r3m), axis([0 300000 -0.8 0.8]);

r3mfft=fft(r3m);

figure, plot(abs(r3mfft))

r4m=medfilt1(r, 31);

wavwrite(r4m, Fs3, 'r4m.wav');

figure, plot(r4m), axis([0 300000 -0.8 0.8]);

r4mfft=fft(r4m);

figure, plot(abs(r4mfft))
```

Intentam també fer el filtratge en el domini freqüencial. Partint de la suposició que el renou és més evident en les freqüències altes, eliminam totes les components freqüencials per damunt de l'índex 12000 (aprox. 1250 Hz). Notem que, com la veu humama conté un rang de freqüències entre 300 i 3400 Hz, s'han perdut els tons més aguts de la veu amb el filtratge.

```
N=length(r);

k1=12000;

fcont=k1*Fs3/N

k2=N+2-k1;

rfftb=rfft;

rfftb(k1:k2)=0;
```

```
figure, plot(abs(rfftb));

rb=real(ifft(rfftb));

figure, plot(rb), axis([0 300000 -0.8 0.8]);

wavwrite(rb, Fs3, 'rb.wav');
```

Hem de tenir en compte que aquest filtratge "ideal" en realitat no ho és i que, a nivell temporal, no és exactament equivalent a la convolució del senyal d'entrada per una funció 'sinc'. Anam a desenvolupar una mica aquests conceptes.

En primer lloc anam a definir el filtre 'ìdeal" que hem utilitzat:

```
idealfft=ones(N, 1);

idealfft(k1:k2)=0;

figure, plot(abs(idealfft))

ideal=real(ifft(idealfft));

figure, plot(ideal);
```

La primera figura mostra la dft del filtre i la segona la seva resposta impulsional. Podem fer les següents observacions:

- Si prenim una porció de la resposta impulsional (*figure, plot(ideal(1:2000))*) observam que la forma és similar a la funció 'sinc'. En realitat és una funcuó de la forma 'sin/sin', resultat de la DFT inversa de la resposta en freqüència especificada (tot 1's a la banda de pas, 0 a la resta).
- La resposta impulsional és pràctizament zero llevat de la part inicial i la part final (indexos inferiors a 3000 i superiors a N-3000).
- Per propietats de la DFT i la seva inversa, el que observam és el periode principal d'una funció periòdica de periode N. Pel mateix motiu, el senyal d'entrada (el senyal amb renou) també s'ha de considerar periòdic, de periode N.
- La resposta impulsional és simètrica respecte a l'índex 0. Per la periodicitat del senyal, els valors pròxims a N corresponen a indexos negatius (valors negatius de temps). Es tracta per tant d'un filtre 'irrealitzable'.
- El senyal de sortida del filtre passa baix és (a nivell freqüencial): $rfftb = rfft \cdot idealfft$. A nivell temporal:

  $$rb = IDFT\{rfftb\} = IDFT\{rfft \cdot idealfft\} = r * ideal$$

  Es tracta per tant de la convolució dels senyals, però com els senyals són periòdics el resultat és equivalent a una **convolució circular** dels senyals originals (no perioditzats). Si el filtre té una resposta impulsional 'curta' (els valors s'anul.len a partir d'un índex petit en relació a N), com és el nostre cas, l'única diferència entre la convolució 'habitual' i la 'circular' és que les mostres del principi i del final del senyal d'entrada es veuen afectades, respectivament, pels valors del final i del principi del mateix senyal. Una manera de solucionar el problema és afegir mostres de valor zero al final i el principi del senyal d'entrada. En el nostre cas, com l'efecte de la convolució circular afecta només als primers i als darrers 0.10 segons (3000/Fs3=0.09 segons), no el tenim en compte.
- Per observar amb més detall la resposta freqüencial del filtre podem fer el següent:

  ```
  ideal2=zeros(4*N, 1);

  ideal2(1:N/2)=ideal(1:N/2);
  ```

```
ideal2(4*N-N/2:4*N)=ideal(N-N/2:N);

figure, plot(ideal2)

ideal2fft=fft(ideal2);

figure, plot(abs(ideal2fft));

figure, plot(real(ideal2fft(47500:48500)));
```

En la darrera figura veim el detall de la resposta en freqüència 'real' del filtre que hem creat. Observam com el comportament és lluny del del filtre ideal. De fet, les freqüències pròximes a la de tall queden molt amplificades. Això és degut a l'efecte de Gibbs.

Anam ara a obtenir una versió del filtre passa baix que resolgui els problemes apuntats més apunt: la 'irrealitzabilitat' i l'amplificació de freqüències.

Per obtenir una versió realitzable del filtre ideal partim de l'expressió del filtre ideal amb la freqüència de tall desitjada:

```
omegac=2*pi*k1/N;

sincreal=zeros(N, 1);


sincreal(1)=omegac/pi;

k=2:N;

sincreal(k)=sin(omegac*(k-1))./(pi*(k-1));

figure, plot(sincreal);

figure, plot(sincreal(1:1000));
```

Com que a partir de la mostra 3000 la resposta és pràcticament 0, obtenim ara una versió de la sinc centrada en 3000 i posam a zero els valor a partir de la posició 6000.

```
omegac=2*pi*k1/N;

sincreal=zeros(N, 1);

k0=3000;

k=1:N;

sincreal(k)=sin(omegac*(k-k0-1))./(pi*(k-k0-1));

sincreal(k0+1)=omegac/pi;

sincreal(2*k0+1:end)=0;

figure, plot(sincreal);

figure, plot(sincreal(1:3*k0));
```

Com hem truncat la funció sinc ideal, la resposta en freqüència ja no és la corresponent al filtre ideal.

```
sincrealfft=fft(sincreal);

figure, plot(abs(sincrealfft))
```

Les oscil.lacions que s'observen són degudes al truncat brusc dels valors originals. Per aconseguir un truncat més suau es multiplica la funció sinc obtinguda per una funció de suavitzat ('finestra').

```
w=window(@blackman, 2*k0+1);

windowreal=zeros(N,1);

windowreal(1:2*k0+1)=w;

figure, plot(w);

wsincreal=sincreal.*windowreal;

figure, plot(wsincreal);

wsincrealfft=fft(wsincreal);

figure, plot(abs(wsincrealfft))
```

Ara aplicam el filtre al senyal d'entrada:

```
rcfft=rfft.*wsincrealfft;

figure, plot(abs(rcfft))

rc=real(ifft(rcfft));

figure, plot(rc), axis([0 300000 -0.8 0.8]);

wavwrite(rc, Fs3, 'rc.wav');
```

3.2 Multiplexació en freqüència

Una aplicació directa de la IFFT en comunicacions és la multiplexació en freqüencia dels senyals. Anam a il.lustrar-ho amb un exemple senzill.

Imaginem que volemn transmetre 3 símbols ('1', '2' i '3') per un canal de comunicacions. Una opció és transmetre tres tons purs (sinusoides), una a continuació de l'altra, amb amplituds respectives '1', '2' i '3'. Aquest tipus de codificació s'anomena 'modulació en amplitud' (AM).

```
close all

clear all

N=10000;

n=1:N;

F=1000;

Fs=5000;

x1=0.1*cos(2*pi*F*n/Fs);

x2=0.2*cos(2*pi*F*n/Fs);

x3=0.3*cos(2*pi*F*n/Fs);

x(1:N)=x1;
```

```
x(N+1:2*N)=x2;

x(2*N+1:3*N)=x3;

figure, plot(x)

wavwrite(x, Fs, 'cos123.wav');
```

De manera similar podriem haver enviat tres sinusoides amb tres freqüències diferents, una per a cada símbol, una a continuació de l'altra ('modulació en freqüència' o FM).

```
N=10000;

n=1:N;

F1=1000;

F2=500;

F3=1500;

Fs=5000;

x1=0.5*cos(2*pi*F1*n/Fs);

x2=0.5*cos(2*pi*F2*n/Fs);

x3=0.5*cos(2*pi*F3*n/Fs);

x(1:N)=x1;

x(N+1:2*N)=x2;

x(2*N+1:3*N)=x3;

figure, plot(x)

wavwrite(x, Fs, 'cos123F.wav');
```

Però és més eficient (transmissió més ràpida i més robusta davant alteracions del canal) transmetre els tres símbols simultàniament, cadascun amb una freqüència diferent. Aquesta tècnica rep el nom de **multiplexat freqüencial**. Una opció és sumar les tres sinusoides.

```
N=10000;

n=1:N;

F1=1000;

F2=500;

F3=1500;

Fs=5000;

x1=0.5*cos(2*pi*F1*n/Fs);

x2=0.5*cos(2*pi*F2*n/Fs);
```

```
x3=0.5*cos(2*pi*F3*n/Fs);

clear x;

x=(x1+x2+x3)/3;

figure, plot(x)

wavwrite(x, Fs, 'cos123Fmix.wav');

figure, plot(abs(fft(x))
```

Una manera més senzilla d'aconseguir el multiplexat en freqüència és utilitzant la IFFT. A més, en aquest cas les sinusoides verifiquen la propietat d''ortogonalitat' i serà molt senzill separar unes de les altres en el receptor. Una tècnica de multiplexat en freqüència basada en IFFT anomenada OFDMA (Orthogonal frequency-division multiplexing multiple access) s'utilitza en els sistemes actuals de transmissions mòbils (4G).

```
N=10000;

xfft=zeros(N,1);

xfft(1001)=1;

xfft(N-1000+1)=1;

xfft(2001)=1;

xfft(N-2000+1)=1;

xfft(3001)=1;

xfft(N-3000+1)=1;

x=N*real(ifft(xfft));

figure, plot(x)

wavwrite(x, Fs, 'cos123FFTmix.wav');
```

3.3 Processament d'imatges

Farem un exemple senzill que ilustra l'efecte de l'aliasing en imatges digitals.

Obrim una imatge.

```
clear all

close all

im=imread('camisa.png');

imshow(im);
```

La submostrejam amb primer amb factor 2 i després 4 i observam l'efecte de l'aliasing.

```
im2=im(1:2:end, 1:2:end, 1);

imshow(im2);
```

```
im4=im(1:4:end, 1:4:end, 1);

imshow(im4);
```

Podem observar també els espectres (bidimensionals) de les imatges originals i submostrejades:

```
imshow(abs(fft2(im(:, 1))),[])

imshow(abs(fft2(im2)),[])

imshow(abs(fft2(im4)),[])
```