Laboratorio de Tratamiento Digital de Señales Curso 2020-2021. Sesión 2

Manejo de los sistemas LTI racionales con MATLAB

Un sistema LTI racional (también supondremos causal y estable durante esta práctica) tiene una función de sistema de la forma $H(z)=\frac{B(z)}{A(z)}$, es decir, una función racional cociente de dos polinomios en z. Más precisamente, $H(z)$ es de la forma:

$$H(z)=\frac{\sum_{i=0}^{i=M} b_i z^{-i}}{\sum_{i=0}^{i=N} a_i z^{-i}}; \qquad ROC=\{z/|z|>|p_{\max}|\}$$

Llamamos p_i a las raíces del polinomio del denominador, es decir, p_i son los polos del sistema y $p_{\max}$ el polo de mayor radio ($|p_{\max}|$ <1 ya que el sistema se supone estable). Este tipo de sistemas son los filtros digitales, de gran utilidad en el procesado digital de señales.

Una primera clasificación de los filtros digitales tiene en cuenta la duración de la respuesta impulsiva:

- Filtros IIR (duración infinita de la respuesta impulsiva). $a_0 \neq 0$ y al menos un $a_i \neq 0$ para $i \geq 1$. En este caso el grado del polinomio $A(z)$ es ≥ 1.
- Filtros FIR (respuesta impulsiva de duración finita). $A(z)=1$ y

$$h(n)=\begin{cases} b_n, n=0,1,\ldots,M \\ 0, \text{ resto} \end{cases}$$

MATLAB dispone de un amplio repertorio de funciones para el diseño, representación y análisis de los filtros digitales. Estas funciones siguen la nomenclatura de la referencia principal de la asignatura[1], haciendo uso extensivo de los polinomios $B(z)$ y $A(z)$, representados por dos vectores fila. Téngase en cuenta que en el caso FIR $A=1$.

[1] Alan V. Oppenheim, Ronald W. Schafer: "Discrete-Time Signal Processing", (3rd Edition), Ed. Prentice Hall

Ejercicios de la sesión 2, primera parte

1.1 Diseño de un filtro digital de tipo elíptico

En la sesión 4 trataremos el diseño de filtros digitales en el entorno MATLAB. Para el desarrollo de esta práctica nos limitaremos a diseñar un filtro IIR de tipo elíptico y un filtro FIR de tipo 1. El comando MATLAB

```
[B,A]=ellip(6,0.5,50,1/4);
```

diseña un filtro elíptico paso bajo de orden 6, 0.5 dB de rizado en la banda de paso, 50 dB de atenuación en la banda rechazada y frecuencia de corte $\pi/4$. Nótese que la frecuencia de corte se especifica normalizada, con 1 correspondiente a π. El resultado es dos vectores fila de longitud 7. Los comandos

```
disp(['B =     ' num2str(B)])
disp(['A =     ' num2str(A)])
```

presentan por pantalla los coeficientes. Los coeficientes del numerador son simétricos, que se corresponde, como se verá a continuación, con los ceros del sistema sobre la circunferencia unidad.

1.2 Diagrama polos-ceros

El comando

```
zplane(B,A)
```

dibuja el diagrama polos-ceros del filtro de coeficientes **(B,A)**. Obsérvese que:

- Los polos y ceros aparecen en pares complejos conjugados, como corresponde con un sistema real (coeficientes **B** y **A** reales).
- Los ceros aparecen sobre la circunferencia unidad y en la banda rechazada.
- Los polos aparecen dentro del círculo unidad y en la banda de paso.
- Aparece un polo dominante de radio próximo a 1 y ángulo $\pi/4$ (frecuencia de corte).

1.3 Respuesta impulsiva

El comando

```
[h,n]=impz(B,A,30)
```

Calcula la respuesta impulsiva del filtro de coeficientes **(B,A)** en el margen de índices $0 \le n < 30$. Calcule y dibuje (con `stem(. . .)`) la respuesta impulsiva del filtro. Calcule la frecuencia de oscilación de $h[n]$ con `ginput` y compárela con $\pi/4$.

La respuesta impulsiva de un filtro IIR causal tiene la forma general

$$h[n] = \sum_{i=1}^{i=N} A_i \left(p_i\right)^n u[n]$$

Siendo p_i el i-ésimo polo de $H(z)$. En nuestro caso hay un polo (bastante) dominante en $z = 0.9696\,e^{j0.7909}$, de aquí que $h[n]$ tenga la forma aproximada de un coseno amortiguado de frecuencia aproximada 0.7909 rad y factor de amortiguación aproximado 0.9696^n. Obsérvese los valores en magnitud y fase de los polos del filtro con los comandos `abs(roots(A))` y `angle(roots(A))`.

1.4 Respuesta en frecuencia $H(\omega)$.

Teniendo en cuenta que $H(\omega)$ es compleja y hermítica (amplitud par y fase impar), haremos una representación en magnitud y fase en el intervalo de frecuencias discretas $[0, \pi]$. El comando

```
[H w]=freqz(B,A,N)
```

calcula $H(\omega)$ en el intervalo $[0, \pi]$ y en una retícula de puntos

$$\omega = \left[0 \quad \frac{\pi}{N} \quad \frac{\pi}{N}2 \quad . \quad . \quad . \quad \frac{\pi}{N}(N-2) \quad \frac{\pi}{N}(N-1) \right]$$

La respuesta en amplitud (¡y en dBs!) se calcula con

```
20*log10(abs(H))
```

y la fase (en radianes/muestra) con

```
angle(H)
```

Calcule y dibuje 1000 puntos de la respuesta en amplitud y fase del filtro $H(z) = \frac{B(z)}{A(z)}$.

Para el dibujo introducimos el concepto de *subplots*, que permite dividir la ventana gráfica en varias gráficas. El siguiente código MATLAB divide la ventana gráfica en dos, dibujando en la parte superior la respuesta en amplitud y en la inferior la respuesta en fase:

```
subplot(2,1,1)
plot(w,20*log10(abs(H)))
subplot(2,1,2)
plot(w,angle(H))
subplot
```

El último comando anula la división de la ventana gráfica para su uso posterior. Tiene que añadir el etiquetado adecuado. Conviene alinear las dos figuras con el comando **`axis tight`** después de cada **`plot(. . .)`**. Una vez alineadas las figuras se puede comprobar que los saltos de valor π en la fase se corresponden con los ceros del sistema sobre la circunferencia unidad. El salto de fase de valor 2π se debe a que la función MATLAB **`atan(. . .)`** calcula los ángulos en el intervalo $[-\pi, \pi]$. Para suprimir estos saltos existe la función **`unwrap(. . .)`** de MATLAB. Por tanto, **`angle(H)`** se corresponde con $ARG\left[H\left(e^{j\omega}\right) \right]$ (valor principal de la fase) y

`unwrap(angle(H))` se corresponde con $\arg\left[H\left(e^{j\omega}\right)\right]$ (fase continua). Ver páginas 275-277 de la referencia mencionada.

1.5 Filtrado

El comando MATLAB

```
y=filter(B,A,a)
```

filtra la señal `a` y almacena la señal filtrada en `y`. El vector `y` tiene la misma longitud que el vector `a`. Para analizar el efecto del filtrado es conveniente utilizar un **plot múltiple**, con la siguiente sintaxis:

```
plot(n,a,'g',n,y,'r')
```

Este comando dibuja el vector `a` (en color verde) y el vector `y` (en rojo) en la misma gráfica, siendo así más fácil la comparación de las señales. Se observa que:

- La señal de salida está retrasada unas 5-6 muestras, retraso relacionado con el orden del filtro.
- La señal de entrada tiene un ancho de banda de 4 KHz. La frecuencia de muestreo es F_s = 8 KHz y el filtro paso bajo tiene una frecuencia de corte equivalente[2] F_c = 1 KHz, por lo que:
  - Las variaciones de la forma de onda de la salida son más suaves
  - La energía de la señal de salida es menor

En el caso de los plot múltiples, además de un adecuado etiquetado, conviene utilizar (en este caso) el comando

```
legend('Entrada','Salida','Location','NorthEast')
```

resultando un cuadro de texto en la esquina superior derecha de la figura que aclara qué señal es la entrada y qué señal es la salida.

1.6 Cálculo de la respuesta impulsiva con `filter( . . .)`.

Repita el apartado 1.3 pero mediante el procedimiento de filtrar el impulso unidad.

---

[2] Como la frecuencia de corte del filtro es π/4 y está normalizada a π, resulta $\frac{\pi/4}{\pi} = \frac{F_c}{F_s/2} \Rightarrow F_c = 1\text{kHz}$

Procesado local de señales

Una señal discreta $x[n]$ es una secuencia de números, en general complejos, definida en el rango de índices $-\infty < n < \infty$.

Para que esta señal, de duración ilimitada, se pueda tratar mediante métodos numéricos se tiene que "trocear" en segmentos de duración finita. Este troceado se formula mediante la señal localizada.

El fichero `tds.mat` contiene la señal correspondiente a la frase "tratamiento digital de señales" (la señal se ha obtenido mediante un proceso de muestreo con una $F_s = 8000\,Hz$ y resolución $N = 16\,bits$). Puede comprobarlo cargando el fichero con `load tds`, dibujando la señal con `plot(. . .)` y oyéndola con los siguientes comandos de MATLAB.

```
player = audioplayer(int16(tds), 8000);
play(player);
```

Como puede observar en la gráfica, la señal `tds` no es estacionaria, y por tanto sus características varían a lo largo del tiempo. Se puede hablar de características "locales" en el tiempo, como la energía, la potencia, el espectro, etc.

Se define la señal localizada de la siguiente manera:

$$x_m[n] = x[n + mL_v]\,w[n],\ m \in \mathbb{Z}$$

donde $w[n]$ es una secuencia de duración finita L_v, que se llama "ventana temporal". Así, la señal localizada $x_m[n]$ consiste en la señal $x[n]$ vista a través de la ventana $w[n]$. Por ejemplo, si la ventana es

$$w[n] = \begin{cases} 1,\ 0 \le n \le L_v - 1 \\ 0,\ resto \end{cases}$$

tendríamos los siguientes términos de la señal localizada:

$$x_{-1}[n] = \{x[-L_v], x[-L_v + 1], \cdots, x[-1]\}$$

$$x_0[n] = \{x[0], x[1], \cdots, x[L_v - 1]\}$$

$$x_1[n] = \{x[L_v], x[L_v + 1], \cdots, x[2L_v - 1]\}$$

$$x_2[n] = \{x[2L_v], x[2L_v + 1], \cdots, x[3L_v - 1]\}$$

Con respecto a la caracterización de la señal `tds` mediante el parámetro "potencia", podemos considerar la potencia global:

$$P_x = \frac{1}{N}\sum_{n=0}^{N-1} x^2[n],\ N = \text{duración}\{x[n]\}$$

o bien, dada la naturaleza no estacionaria de la señal, la potencia localizada:

$$P_{xloc}[m] = \frac{1}{L_v}\sum_{n=0}^{L_v-1} x_m^2[n]$$

Nótese que la potencia global es un número pero la potencia localizada es una secuencia, y por tanto otra señal.

En muchos casos prácticos conviene que los segmentos de señal localizada estén parcialmente solapados. Así, se define una señal localizada "con solapamiento" de la siguiente forma:

$$x_m[n] = x\left[n + m\left(L_v - L_{sol}\right)\right] w[n] = x[n + mL_d]\, w[n]\ ,\ m \in \mathbb{Z}\ ,\ 0 \le L_{sol} < L_v$$

siendo $w[n]$ de nuevo una ventana rectangular de duración L_v; $L_d = L_v - L_{sol}$ es el desplazamiento de la ventana y L_{sol} es el número de muestras solapadas entre dos ventanas consecutivas.

Suponiendo una ventana rectangular con $L_v = 20$ y un solapamiento del 50% $L_{sol} = L_v/2 \Rightarrow L_d = L_v - L_{sol} = 10$ tendríamos:

$$x_0[n] = \{x[0], x[1], \cdots, x[19]\}$$

$$x_1[n] = \{x[10], x[11], \cdots, x[29]\}$$

$$x_2[n] = \{x[20], x[21], \cdots, x[39]\}$$

Ajustando los valores de L_v y el porcentaje de solapamiento se puede controlar el grado de "suavidad" con que se mide una característica localizada de una señal.

Ejercicios de la sesión 2, segunda parte

2.1 Cálculo de la potencia localizada.

Calcule la potencia localizada de la señal `tds`, empleando una ventana de duración $60\ \text{ms}$ y solapamiento del 50%. El periodo de muestreo es $T_s = 1/(8\ \text{kHz}) = 125\ \mu\text{s}$.

La duración típica de una sílaba con dos fonemas es de unos $100\ \text{ms}$, por lo que la potencia localizada marca, aproximadamente, el ritmo silábico.

Ayuda para la programación del procesado local

La exploración de una señal $x[n]$ de duración L para la extracción de algún parámetro local es una operación frecuente y que por tanto conviene disponer de un procedimiento operativo simple y efectivo. A tal efecto se suministra la función `slocal(. . .)`, que dispone de la siguiente ayuda (comando `help slocal`):

```
% Tratamiento digital de la señal
%
% Exploración local de una señal
%
% xl=slocal(x,Lv,Ld,start)
% Entradas:
%    x,  señal que se explora
```

```
%    Lv, duración de la señal local
%    Ld, desplazamiento de la señal local (¡NO solapamiento!)
%    start=0 en una llamada inicial antes de comenzar la exploración
% Salidas
%    xl, señal local
%        si xl=[], se ha alcanzado el final de la señal
%
% Ejemplo de uso
%    Lv=100;
%    Ld=50;
%    xl=slocal(x,Lv,Ld,0);
%    while 1
%        xl=slocal(x,Lv,Ld,1);
%        if isempty(xl)
%            break
%        end
%         . . . EL PROCESADO
%         . . . LOCAL AQUÍ
%    end
%
```

Con la ayuda de la función `slocal(. . .)` una solución para este apartado sería:

```
%% Apartado 2.1

% Datos del ejercicio
load tds
Lv= ; % Calcúlese en función del periodo de muestreo
Ld= ; % Calcúlese en función del periodo de muestreo
Ploc=[];

% Inicialización de la función slocal
xl=slocal(tds,Lv,Ld,0);

% Bucle de cálculo de la señal localizada
while 1
    xl=slocal(tds,Lv,Ld,1);
    if isempty(xl)
        break,   % Se alcanzó el final de la señal tds
    end
    % Procesado local
    Ploc(end+1)=1/Lv*(xl'*xl); % Potencia local obtenida como prod. escal.
    % Ploc(end+1)=1/Lv*sum(xl.^2); % Alternativamente

end

% Dibujo
n=0:length(tds)-1; % Eje temporal para tds
m=0:length(Ploc)-1; % Eje temporal para Ploc
subplot(2,1,1), plot(n,tds)
title('Señal tds'), xlabel('n'), ylabel('tds[n]'), grid
axis tight
subplot(2,1,2), plot(m,Ploc)
title('Señal potencia localizada'),
xlabel('m'), ylabel('Ploc[m]'), grid
axis tight
subplot
```

En la solución de este apartado hemos introducido las siguientes novedades MATLAB:

- Los comandos

  ```
  Ploc=[];
  Ploc(end+1)= . . .
  ```

  permiten crear un vector vacío e ir añadiéndole elementos al final.
- La estructura de repetición

  ```
  while <condición lógica>
        "cálculos"
  end
  ```

  permite ejecutar una serie de cálculos mientras se satisfaga una condición lógica. En este caso el bucle sería infinito ya que la condición es 1 (=True).
- La estructura de selección

  ```
  if <condición lógica>
  "acción"
  end
  ```

  permite acabar el bucle infinito cuando en la exploración localizada se llega al final de la señal `tds` (segmento local `xl` vacío).

Corte y pegue la solución que se proporciona de este apartado. Varíe la longitud de la ventana y observe cómo varía la potencia localizada. Fíjese cómo al aumentar la longitud de la ventana se suaviza la señal potencia localizada (equivalente a perder resolución temporal).

2.2 Relación potencia de la señal a potencia del ruido (SNR)

Genere la señal

$$x[n] = A\cos\left(\frac{2\pi}{20}n\right);\ A=5;\ n=0,1,\ldots,N-1;\ N=100$$

Calcule y dibuje las señales $y[n] = x[n] + \mathrm{g}\, r[n]$, donde $r[n]$ es un ruido blanco gaussiano, que puede obtener con la función MATLAB randn(. . .), y g un factor de amplificación del ruido. Las señales $y[n]$ deben tener una SNR entre -10 dB y 20 dB, con saltos de 10 dB.
El factor de amplitud del ruido viene dado por la fórmula (¿por qué?):

$$g = \sqrt{\frac{\sum_{n=0}^{N-1} x^2[n]}{10^{\frac{SNR}{10}} \sum_{n=0}^{N-1} r^2[n]}}$$

Sugerencia: dibuje las cuatro señales en una única ventana gráfica con `subplot(2,2,x)`, con x=1, 2, 3, y 4, correspondientes a las diferentes relaciones señal-ruido.

2.3 Relación señal a ruido segmental

La señal $y[n] = x[n] + r[n]$, tiene una $SNR = -2.4928\ dB$ global, siendo $x[n]$ un coseno con los mismos parámetros que en el apartado anterior, pero con diferente duración, y $r[n]$ un ruido aditivo. Afortunadamente, disponemos de las señales $x[n]$ y $r[n]$, por lo que podemos aplicar las ideas expuestas acerca del procesado localizado a la "característica" SNR de la señal $y[n]$. Estas señales se encuentran en los ficheros `x.mat` y `r.mat`, que debe copiar a su directorio local y cargarlas con el comando `load`.

Desarrolle el apartado 2.1 de nuevo, pero ahora con respecto a la SNR, calculando la SNR segmental. Elija una longitud para la ventana $L_v = 100$ y solapamiento 50%.

Se sugiere que dibuje, utilizando subplots, la señal $x[n]$, el ruido $r[n]$ y la SNR segmental.

Sugerencia: la función `slocal(. . .)` no puede explorar dos señales simultáneamente. Una solución para este ejercicio puede ser calcular por separado las potencias localizadas de la señal (`Plocx`) y del ruido (`Plocr`) con ayuda de `slocal(. . .)`, y a continuación calcular la SNR segmental con el comando

```
SNRseg=10*log10(Plocx./Plocr)
```

Ejercicios adicionales de la segunda sesión de laboratorio

3.1 Repita los apartados desde el 1.2 al 1.6 para un filtro FIR.

Diseñe un filtro FIR de tipo 1 con el siguiente comando:

```
h=fir1(28,1/4);
```

El orden del filtro es 28 y la frecuencia de corte $\frac{\pi}{4}$.

Repita los apartados 1.2 al 1.6 con `A=1` y `B=h`.

3.2 Calcule la señal localizada "número de cruces por cero" de la señal `tds`.

Emplee una ventana de longitud 20 ms y solapamiento 50%. Recuerde que $T_s = 125\,\mu s$.

Ayuda: el cálculo del número de cruces por cero de los valores de un vector `v` se puede calcular con MATLAB con la siguiente sentencia:

```
sum(abs(diff(v>0)))
```

(intente entender esta sentencia)