Departamento de Electrónica
Universidad de Alcalá

Bioingeniería

Práctica 4

Filtrado de señales ECG

Grado en Ingeniería de Computadores

Universidad de Alcalá

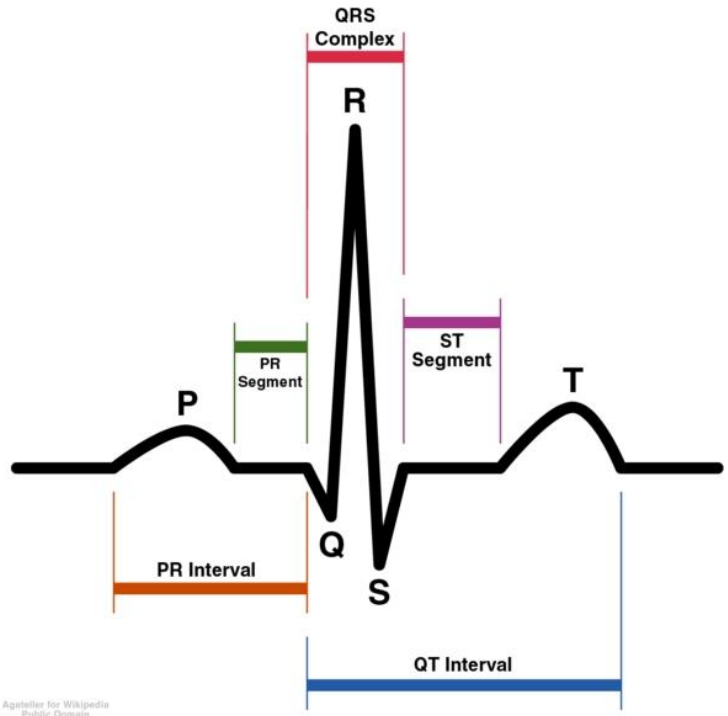
Curso 4º

Índice de contenidos

Introducción

Un trazo normal de un ciclo cardíaco consta de una onda P, un complejo QRS y una onda T. Además, en más de un 50% de los casos también es posible que sea visible una pequeña onda U. El voltaje de la línea base del ECG también es conocido como línea isoeléctrica o línea basal. Normalmente, la línea isoeléctrica es la porción de trazo que sigue a la onda T y precede a la siguiente onda P.

En la práctica, la señal electrocardiográfica capturada es tan débil que se encuentra muy contaminada con un zumbido de 50 Hz acoplado como interferencia y procedente de la red eléctrica, no resultando totalmente eliminado por el amplificador diferencial de instrumentación. Además, la señal resulta mezclada con diversos artefactos de baja frecuencia generados como consecuencia de la respiración del paciente y otras causas, que suelen estar comprendidos entre DC y 0.5 Hz. Por último, aparece también una cierta cantidad de ruido blanco que se genera en el propio sistema amplificador.



Objetivos

Se expone un ejemplo de filtrado completo de una señal ECG, que incluye un filtro *notch* para eliminar la interferencia de red de 50 Hz, un filtro paso alto que atenúa las variaciones de la línea base provocadas por los artefactos de muy baja frecuencia, seguido por un filtrado paso bajo que suaviza la señal al atenuar el ruido de alta frecuencia.

Como ampliación se propone diseñar otro tipo y técnicas de filtrado para contrastar resultados con los aquí propuestos.

Proceso de trabajo

Lectura y visualización del registro original

El registro de la señal electrocardiográfica se encuentra en el fichero llamado *ecg1ms.dat*, que contiene 33707 muestras *float* convertidas a ASCII y que podemos abrir con el bloc de notas para observar su contenido. Contiene dos columnas: la primera indica el número de muestra y la segunda el voltaje. La velocidad de muestreo con la que se obtuvo la señal fue de 1000 muestras por segundo.

Una vez en Matlab y en el directorio donde se encuentran todos los ficheros de esta práctica, podemos cargar el registro de la señal ECG mediante el siguiente comando:

```
>> load ecg1ms.dat;
```

Antes de filtrar esta señal conviene que echemos un vistazo a la misma mediante un *ploteo*. Podemos observar lo fuertemente contaminada con ruido, especialmente zumbido de 50 Hz, que se encuentra la señal.

```
>>figure(1)
>>plot(ecg1ms(:,2))
>>title('Señal ECG capturada');
```

La señal anterior fue convertida con un ADC de 12 bits de resolución y preamplificada con una ganancia de K=1000. La resolución del ADC es dada por:

$$ADC_{res} = (V_{high} - V_{low}) / 2^{12} = 0.002 \text{ volts}$$

con V_{low} = -4.096 volts y V_{high}=4.096 volts.

Por lo tanto, para reconstruir la señal original la ecuación es:

$$ECG_{sal}(n) = \frac{ECG_{ent}(n) \cdot ADC_{res} + V_{low}}{K}$$

El código Matlab sería:

```
figure(2)
Fs=1000; %Frecuencia de muestreo 1000Hz
Ts=1/Fs;
% Suponemos que el ADC convierte a enteros sin signo
digitos=12;
```

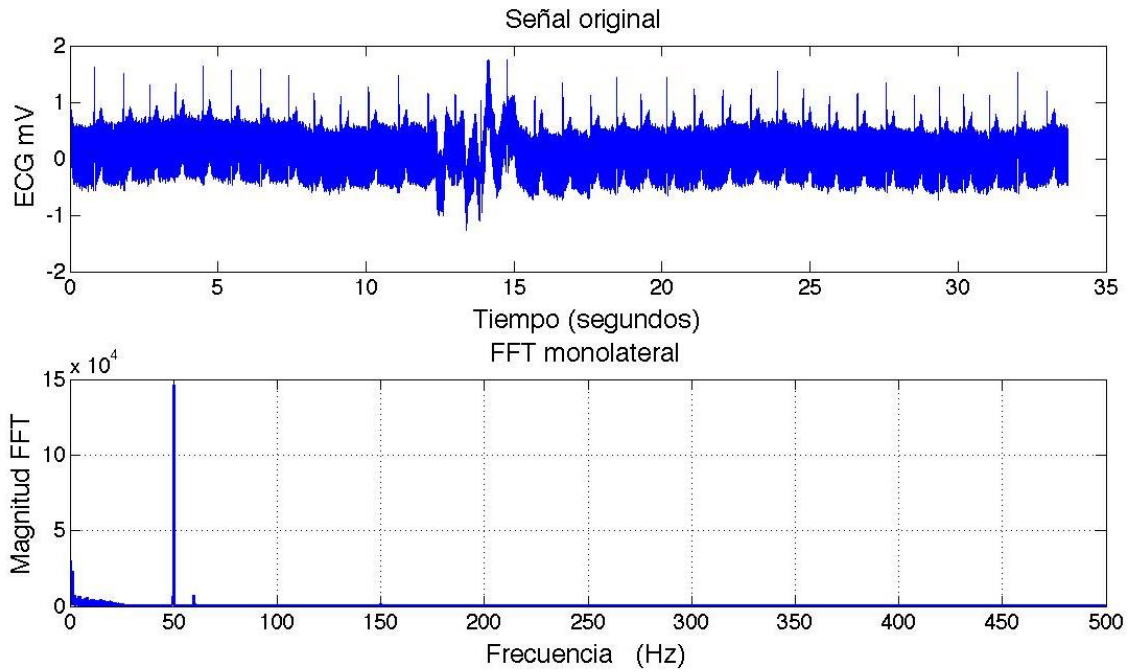
```
Vlow=-4.096;
Vhigh=+4.096;
ADCres=(Vhigh-Vlow)/(2^digitos);

fprintf('La resolución del ADC es %8.2f \n', ADCres);
K=1000; %Ganancia del preamplificador
% Reconstruimos la señal original
ecg=( ecg1ms(:,2).*ADCres+Vlow)./K;
% Las muestras están espaciadas por una cantidad Ts en
segundos
tmax=( ecg1ms(end,1)- ecg1ms(1,1))*Ts
tsec=0:Ts:tmax;

figure(3)
subplot(2,1,1)
% Señal formateada
ecgformat=ecg(1:length(tsec));
ecgformat=ecg.*1e3;
grid on
plot(tsec,ecgformat);
xlabel('Tiempo (segundos)');
ylabel(' ECG mV');
title('Señal original')
subplot(2,1,2)
fftPlot(tsec,ecg);
grid on;

title('FFT monolateral');
```

La función que calcula la FFT monolateral toma como parámetros el tramo de tiempo y la señal a transformar. Para la FFT monolateral sólo son necesarios la mitad de los coeficientes (son simétricos) y hay que escalarlos por un factor de *(número de muestras)/2*. Como resultado, plotea la magnitud de los coeficientes FFT.

```
function [fss,halfASpect]=fftPlot(tsec,ecg)
% Calcula la FFT de la señal muestreada
fftSpect=fft(ecg);
% el número de muestras es:
nsamp=length(tsec);
% Como los coeficientes de la FFT son simétricos, tomamos sólo
% la mitad de ellos
hindex=ceil(nsamp/2);
halfSpect=zeros(1,hindex);
halfSpect(1:hindex)=(nsamp/2)*fftSpect(1:hindex);
% Los intervalos de frecuencia están espaciados por deltaF
% La FFT considera la señal como periódica. En nuestro caso la
% señal dura tsec(end), aproximadamente 32 segundos
deltaFs=1/tsec(end);
fss=0:deltaFs:(hindex-1)*deltaFs;
halfASpect=abs(halfSpect);
plot(fss,halfASpect);
```

```
xlabel('Frecuencia (Hz)');
ylabel('Magnitud FFT');
end
```

El resultado obtenido lo vemos a continuación:



Observando el espectro podemos apreciar que predomina una componente continua y un zumbido de 50 Hz que, a pesar de ser atenuado por el amplificador de instrumentación, no pudo ser completamente eliminado. Existen, asimismo, otra serie de interferencias de origen diverso. Resulta importante mencionar las componentes próximas a 0.05 Hz, y que proceden del efecto capacitivo entre el electrodo y la piel.

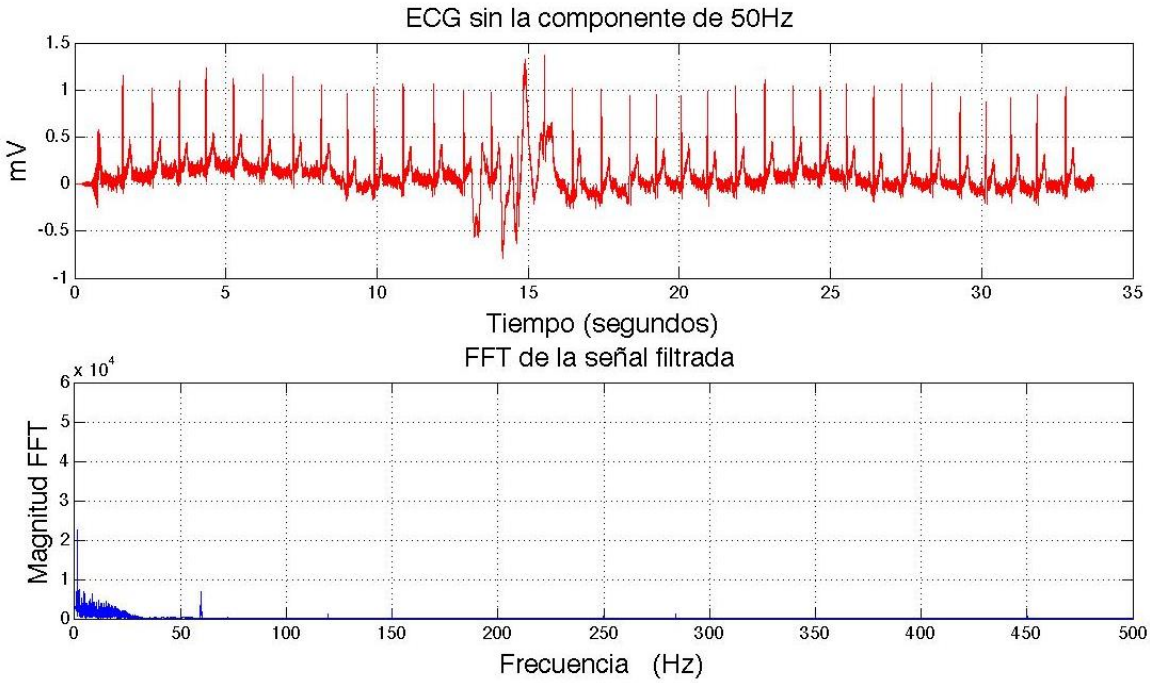
Diseño del filtro FIR para eliminar la DC y el zumbido de 50 Hz

Primero procesaremos la señal mediante un filtro *notch* para eliminar la componente de 50Hz. Consultar en Matlab el funcionamiento de las funciones *fir1* y *filter*, y luego comentar el significado de los parámetros aquí usados. El código Matlab correspondiente es:

```
clc
figure(4)
% Diseñamos el filtro notch de 50 Hz y filtramos la señal
% Frecuencia de Nyquist
fsHalf=Fs/2;
wl=48/fsHalf;
wh=51/fsHalf;
stop50=fir1(1536,[wl wh],'stop');
ecgno50=filter(stop50,1,ecg);
subplot(2,1,1)
```

```
plot(tsec,ecgno50.*1e3,'r');
title('ECG sin zumbido de 50 Hz');
xlabel('Tiempo (segundos)');
ylabel('mV');
grid on
subplot(2,1,2)
fftPlot(tsec,ecgno50);
title('FFT de la señal filtrada')
grid on;
fprintf('La componente de 50 Hz es atenuada %f veces
\n',122.9);
```

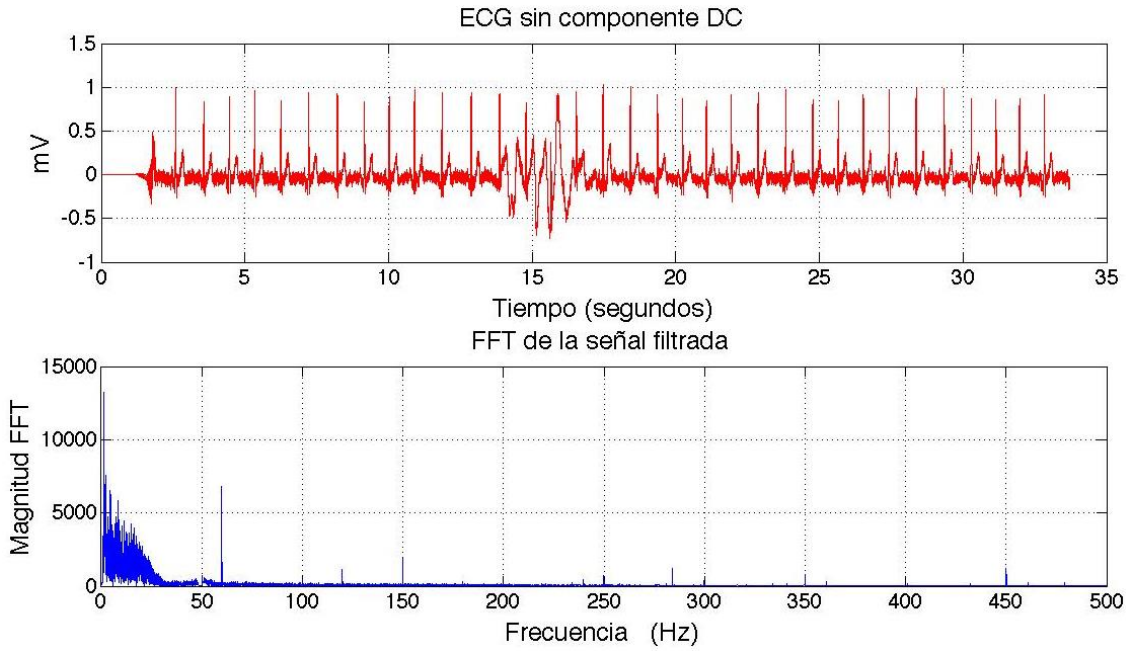
En la siguiente figura vemos la señal y su espectro sin la componente de 50 Hz. El orden del filtro se eligió por prueba y error, comenzando por un orden bajo de 64 y subiendo hasta un orden de 1536 donde la componente de 50 Hz en el espectro FFT se redujo suficientemente (atenuada hasta 122.9 veces). Si el orden subía de 2000, aparte del coste computacional, la respuesta de fase se hacía no lineal, distorsionando la forma de onda de salida. Se pide que el alumno obtenga con Matlab las respuestas (amplitud y fase) del filtro *notch* aquí empleado y el de otro de orden 2048. Para ello, usar el comando *freqz*.



Para eliminar la componente DC y las variaciones de la línea base, necesitamos un filtro paso alto. Se eligió una frecuencia de corte de 1 Hz y un orden de 2000, consiguiendo una atenuación de 271.88 veces en la componente continua. El procedimiento es similar al descrito para el filtro *notch*:

```
clc
figure(5)
```

```
% Diseño del filtro paso bajo y filtrado de la señal
% Frecuencia de Nyquist
fsHalf=Fs/2;
wcutt=1/fsHalf;
highpass=fir1(2000,wcutt,'high');
ecgnoDC=filter(highpass,1,ecgno50);
subplot(2,1,1)
plot(tsec,ecgnoDC.*1e3,'r');
title('ECG sin componente DC');
xlabel('Tiempo (segundos)');
ylabel('mV');
grid on
subplot(2,1,2)
[fss,coeff]=fftPlot(tsec,ecgnoDC);
title('FFT de la señal filtrada')
grid on;
fprintf('La componente DC es atenuada %f veces \n',coeff(1));
```



Resultados adicionales para seguir trabajando

1. Aplicar un zoom a la señal anterior y estudiar detenidamente la forma de onda de un ciclo P-QRS-T. Verificar que la morfología de la señal no ha sido distorsionada.
2. Obtener con Matlab, igual que en el filtro anterior, la respuesta (amplitud y fase) del filtro aquí empleado (usar *freqz(highpass)*).
3. Filtrar la señal con un filtro paso bajo adecuado para reducir el ruido restante sin distorsionar demasiado la señal útil. Fijar una frecuencia de corte igual o superior a 100 Hz.

4. Finalmente, realice el filtrado aquí descrito, pero haciendo uso de filtros IIR para reducir drásticamente el orden de los filtros empleados. La respuesta de fase obtenida podría ser no lineal y distorsionar la forma de onda en algún sentido.

Obtener las gráficas correspondientes y comentar conclusiones.

También se puede optar por otro enfoque diferente, borrando los coeficientes no deseados de la FFT de la señal original y reconstruyéndola mediante una IFFT. **Busque información en Internet y proponga un ejemplo sencillo en Matlab que ilustre el proceso.**

Promediado coherente de señales

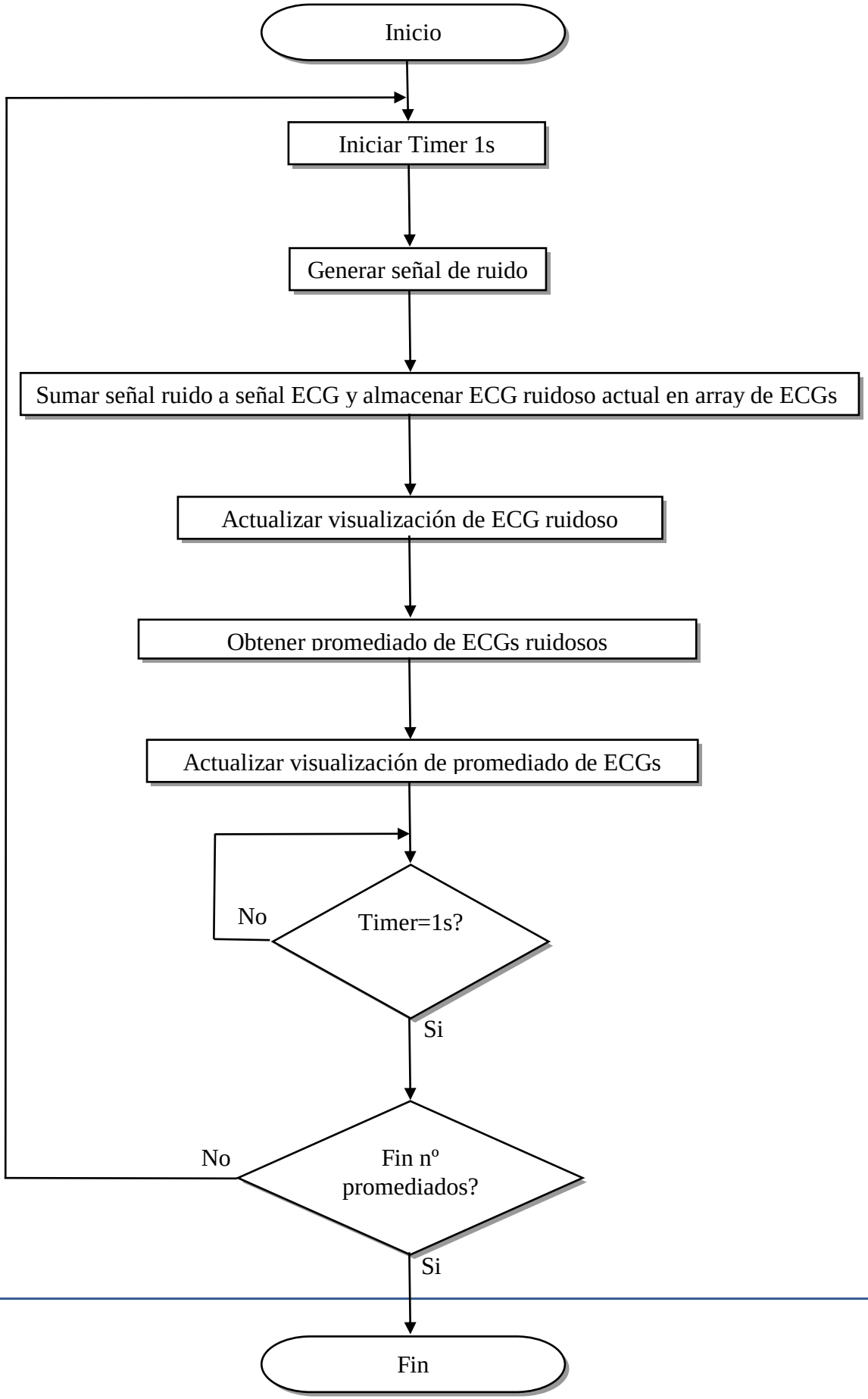
A continuación, se debe completar el listado de código que aparece más abajo y cuyo objetivo es ilustrar la utilidad del filtrado coherente en la reducción de ruido. En este tipo de filtrado es esencial que los registros sean coherentes o correlacionados, como suele ser habitual cuando se trabaja con potenciales evocados. Con objeto de simplificar el proceso y debido a la dificultad de trabajar con potenciales evocados en el laboratorio, utilizaremos registros ECG sintetizados con Matlab. Para ello, cada pulso ECG debe estar perfectamente alineado, de modo que su suma refuerce el ECG ideal resultante y atenúe el ruido aleatorio. Esto resulta ser así, ya que cada pulso ECG idealmente es idéntico (máxima correlación entre ellos), mientras que el ruido que se suma a cada pulso ECG está totalmente incorrelado de un pulso a otro al tratarse de un ruido aleatorio.

Suponiendo que los pulsos ECG originales contaminados con ruido tengan una relación señal/ruido $SNR_{original}$ y promediemos un número N de ellos, entonces la señal resultante de este promedio tendrá una relación señal/ruido SNR_{promed} calculada con la siguiente fórmula:

$$SNR_{promed} = N^{1/2} \cdot SNR_{original}$$

Para ilustrar este proceso se propone completar el siguiente programa en Matlab, de modo que se muestre el pulso ECG actual contaminado con ruido en la ventana de arriba, y la ventana inferior muestre el resultado del promediado de los *n* pulsos actuales. Un posible diagrama de flujo que explica la idea general se muestra a continuación, y le sigue un listado que el alumno debe completar. Si el alumno tiene otra idea más innovadora o diferente de hacer el diseño de este programa, se le anima a hacerlo a su gusto y se le valorará por ello.

Diagrama de flujo orientativo del ejercicio propuesto: Algoritmo para ejemplo que demuestre el promediado coherente de señales ECG



Script main_promediado.m:

```
clear all;
clc;
global ecg_promediado; %global, para que pueda ser vista desde la
%funcion_call_back (otra alternativa sería pasársela como parámetro,
%tal y como se ha hecho con latido_ecg más adelante).
ecg_promediado=zeros(2700,1);
global numero_promedios;
numero_promedios = uint16(0);

X = ____________________; %Generamos un latido QRS con 2700 muestras de
%resolución y amplitud 3.5.
latido_ecg = ________________; %suaviza la señal X usando un filtro de
%suavizado Savitzky-Golay y con un data frame de 61 muestras

%% Iniciamos un Timer en Matlab. Como necesitamos pasar el argumento
%latido_ecg a la función funcion_call_back, especificamos tal
%condición en un cell array con llaves
t = timer('TimerFcn',{@funcion_call_back, latido_ecg}, 'Period', 1);
set(t,'ExecutionMode','fixedRate'); %Imprescindible fijar esta propiedad
%para que el Timer se ejecute periódicamente. Si no, lo hará sólo una vez
start(t); %OJO!! para parar el Timer, tendremos que ejecutar más tarde
%desde la línea de comandos: delete(t), o bien stop(t) y luego delete(t)

disp('********RECUERDA TECLEAR delete(t) EN LA LINEA DE COMANDOS PARA
TERMINAR*******');
```

Función función_call_back:

```
%Función callback que llamará el Timer
function funcion_call_back(obj, event, latido_ecg)
  global ecg_promediado;
  global numero_promedios;
  numero_promedios = numero_promedios + 1;
  SNR=1;
  %Añadimos ruido blanco a la señal latido_ecg y lo almacenamos en
  %ecg_ruidoso, usando la función awgn. SNR es la proporción de señal
  %a ruido por muestra. El parámetro 'measured' hace que se calcule la
  %potencia de la señal latido_ecg antes de añadir ruido. El parámetro
  %'linear' especifica que la cifra de SNR se considere como lineal
  %(no en decibelios)
  ecg_ruidoso = ____________________________________________________;

  %Acumulamos ecg_ruidoso en ecg_promediado:
  ecg_promediado = ___________________________________;

  %Borramos plot actual y a continuación ploteamos ecg_ruidoso
  %(arriba) y ecg_promediado (abajo). En el título de la gráfica de
  %ecg_promediado actualizamos el número de promedios actuales.
  ________________; %Borra el plot actual
  ________________;
  ________________;
  ____________________________________________; %Título ploteo superior
  ________________;
  ________________;
  ____________________________________________; %Título ploteo inferior
```

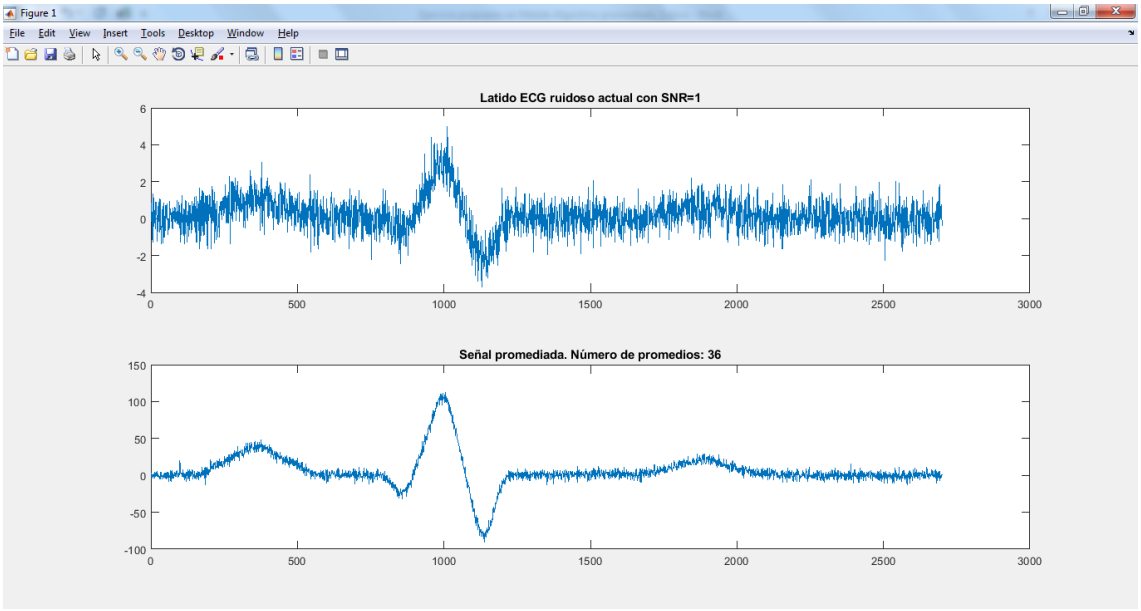


Figure 1
File Edit View Insert Tools Desktop Window Help
Latido ECG ruidoso actual con SNR=1
Señal promediada. Número de promedios: 36