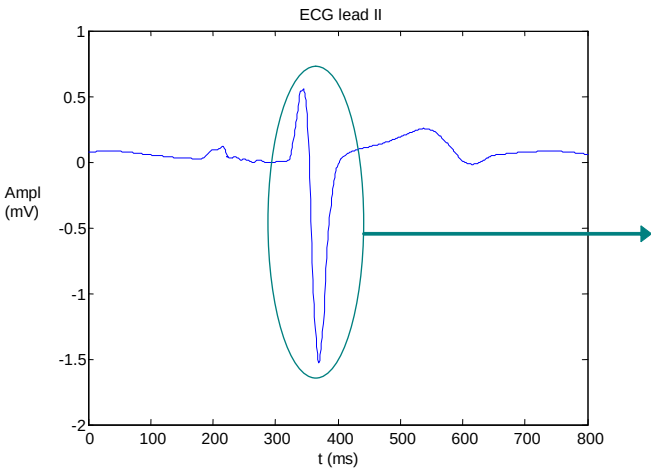
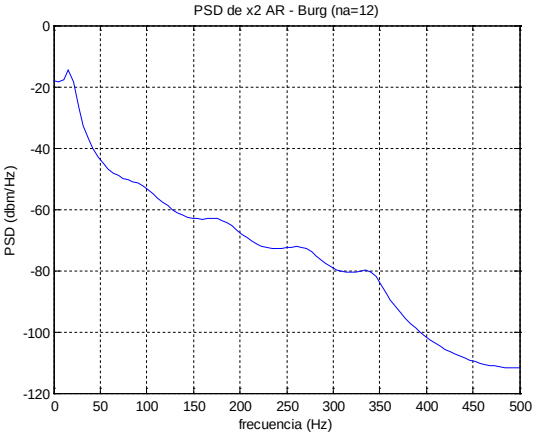
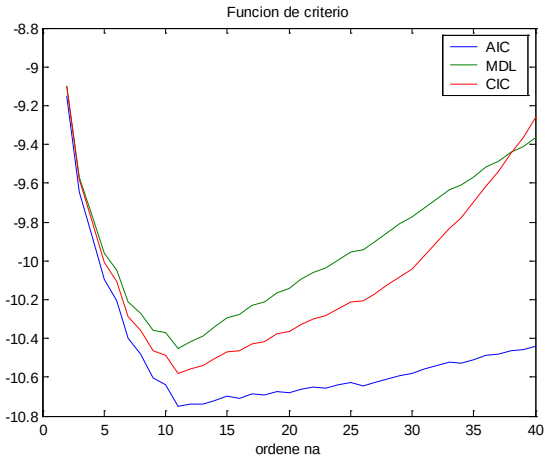
ESTIMACIÓN ESPECTRAL DE SEÑALES BIOMÉDICAS MÉTODOS CLÁSICOS (FFT) Y PARAMÉTRICOS: APLICACIONES PRÁCTICAS CON MATLAB.

TUTORIAL

Pedro Gomis







$$P_{ARMA}(q) = \sigma_e^2 \left. \frac{|B(q)|^2}{|A(q)|^2} \right|_{q=e^{j\omega}} \rightarrow P_{ARMA}(f) = T_s \sigma_e^2 \frac{\left| \sum_{k=0}^{nb} b_k e^{-j2\pi kf / f_s} \right|^2}{\left| 1 + \sum_{k=1}^{na} a_k e^{-j2\pi kf / f_s} \right|^2}$$

Estimación Espectral de Señales Biomédicas
Métodos Clásicos (FFT)y Paramétricos:
Aplicaciones Prácticas con Matlab.
Tutorial

Versión: Febrero, 2010

ISBN: 978-84-695-3841-8

Impreso en Barcelona, España

Última versión en:
http://dl.dropbox.com/u/12996917/Tutorial_Estimacion_Espectral.pdf

TABLA DE CONTENIDOS

OBJETIVOS.

Analizar y caracterizar señales biomédicas discretas de duración finita a través de su espectro estimado de frecuencias, considerando la energía o potencia de la señal distribuida a través de su espectro de frecuencias.

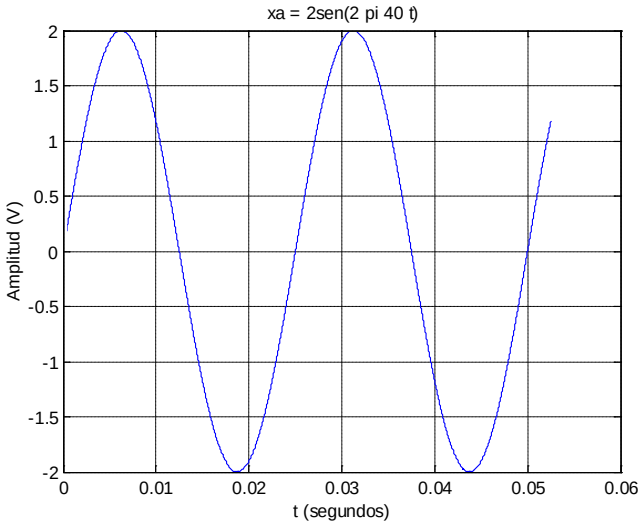
1. SEÑALES EN TIEMPO CONTINUO, DE VALORES CONTINUOS (ANALÓGICA)

Las señales de origen fisiológico son, en su gran mayoría, de naturaleza analógica, es decir, están definidas en todo momento de tiempo y su valor de amplitud es continuo en su rango dinámico. Las señales analógicas pueden ser periódicas o no-periódicas, que incluyen a las limitadas entre dos tiempos determinados. En principio, utilizaremos una señal periodica sinusoidal pura de amplitud A, como el voltaje a través de un resistor R de valor 1 Ω.

$$x_a(t) = A\sin(\omega t) = A\sin(2\pi F_0 t) \text{ [unidades :Volts]},$$

donde $F_0=1/T$ en ciclos/s [Hz] y T es el periodo de la sinusoide [s].

Tomemos, por ejemplo, la sinusoide con $A = 2$ y $F_0 = 40$ (T=0.025), graficamos un par de ciclos de la señal $x_a(t)$



La energía de esta señal continua es

$$E = \int_{-\infty}^{\infty} |x_a(t)|^2 dt \quad \text{[Volts}^2 \times \text{s = Watts} \times \text{s=Joules]} \qquad (1.1)$$

La potencia instantánea será $p(t)=x_a^2(t)/R$ o $p(t)=x_a^2(t)$ (normalizado respecto al Resistor) y la energía instantánea en un diferencial de tiempo será $x_a^2(t)dt$. En el estudio de señales la potencia instantánea no es tan útil como la idea de potencia promedio, media o *average*. La potencia media sobre un periodo T es

$$P_m = \frac{1}{T}\int_0^T |x_a(t)|^2 dt \qquad (1.2)$$

Para la señal de nuestro ejemplo, tenemos que la potencia media es

$$P_m = \frac{1}{0.025}\int_0^{0.025} |A\sin(2\pi 40t)|^2 dt = \frac{A^2}{2} = 2 \quad \text{[Volts}^2\text{] [Watts]}$$

Si se dispone del toolbox de matemática simbólica en MATLAB® (Symbolic Math), la integral de (1.2) se puede resolver:

```
>> syms x t A Pm T
>> x=A*sin(2*pi/T*t)
         x = A*sin(2*pi/T*t)
>> Pm=1/T*int(x^2,0,T)
       Pm = 1/2*A^2
```

La energía de un ciclo de señal será $E=A^2T/2 = 0.05$ Watts×segundo, de dos ciclos será E = 0.1 W×s. En una hora de señal habrá 144000 ciclos y E= 7200 W×s = 2 W×hora.

En el caso de señales no-periodicas (que pudieran incluir una señal periodica limitada en el un tiempo *Tc*), tenemos que la potencia media es

$$P_m = \lim_{Tc\to\infty} \frac{1}{Tc} \int_{Tc} \left| x_a(t) \right|^2 dt \tag{1.3}$$

La raiz-media-cuadrática (*root-mean-square*) o valor RMS se define como el valor de una señal (voltaje, en nuestro ejemplo) constante que produciría la misma potencia media P_m de la señal $x_a(t)$. Es decir:

$$x_{RMS}^2 / R = x_{RMS}^2 \Big|_{R=1} = P_m \tag{1.4}$$

por lo tanto

$$x_{RMS} = \sqrt{\frac{1}{T} \int_T \left| x_a(t) \right|^2 dt} \tag{1.5}$$

1.1) Análisis de Fourier de la señal en tiempo continuo

La transformada de Fourier es la transformación más común de una señal dependiente del tiempo para ser estudiada en el dominio de las frecuencias, debido a que las bases de transformación son funciones senos y cosenos, que caracterizan a la señal en el dominio de la frecuencia.

Si la señal $x_a(t)$ es periódica, entonces podemos representarla en términos de sumatorias de senos y cosenos armónicos (valores discretos en múltiplos enteros de la fundamental), que se relacionan con sus contenidos de frecuencia a través de **series de Fourier**, como

$$x_a(t) = \sum_{k=-\infty}^{\infty} c_k e^{j2\pi k F_0 t}, \qquad \text{donde } c_k = \frac{1}{T} \int_T x_a(t) e^{-j2\pi k F_0 t} dt \tag{1.6}$$

Sin embargo si la señal continua es de duración finita, estamos en el caso de una señal **no-periodica**. En estos casos de señales de energía finita, su descomposición espectral se realiza con la **transformada de Fourier**, que está definida como:

$$X_a(F) = \int_{-\infty}^{\infty} x_a(t) e^{-j2\pi F t} dt \qquad \text{[Volts × s]} \tag{1.7}$$

1.2) Potencia y energía en el dominio de Fourier

La energía de la señal se puede expresar en el dominio de la frecuencia como

® The Mathworks

$$E = \int_{-\infty}^{\infty} |x_a(t)|^2 dt = \int_{-\infty}^{\infty} |X_a(F)|^2 dF = \frac{1}{2\pi} \int_{-\infty}^{\infty} |X_a(\omega)|^2 d\omega \quad (1.8)$$

A la igualdad de la ecuación anterior se conoce como el **Teorema de Parseval**. El integrando de E, $|X_a(F)|^2$, es la distribución de la energía de la señal en función de la frecuencia, $S_{xx}(F)$, llamada *densidad de energía espectral* de la señal $x_a(t)$.

$$S_{xx} = |X_a(F)|^2 \qquad [\text{Volts}^2\times\text{s}^2 = \text{Volts}^2\times\text{s/Hz} = \text{Watts}\times\text{s/Hz} = \text{Joules/Hz}] \quad (1.9)$$

La *densidad de potencia spectral* (o spectrum), PSD, es la distribución de la potencia de la señal en función de la frecuencia *F*.

Cuando la señal continua es no-periodica, como sería el caso de nuestro ejemplo si se acotase la señal sinusoidal en el tiempo, digamos a 10 ciclos, la señal estaría definida en un intervalo *Tc* desde $0 \le t \le 0.25$, entonces la transformada de esta señal truncada sería

$$X_{aTc}(F) = \int_0^{Tc} x_{aTc}(t) e^{-j2\pi Ft} dt \quad (1.10)$$

donde X_{aTc} es la transformada de Fourier de $x_a(t)$ sobre el intervalo de tiempo *Tc*, $x_{aTc}(t)$, y la energía finita de la señal es

$$E_{Tc} = \int_{-\infty}^{\infty} |X_{aTc}(F)|^2 dF \quad (1.11)$$

Si $Tc \to \infty$, entonces en el límite $x_{aTc}(t) \to x_a(t)$. La potencia media de la señal se puede expresar como [Carl68]

$$P_m = \lim_{Tc\to\infty} \frac{E_{Tc}}{Tc} = \lim_{Tc\to\infty} \frac{1}{Tc} \int_{-\infty}^{\infty} |X_{aTc}(F)|^2 dF \quad (1.12)$$

La densidad espectral de potencia (PSD), que denotaremos P_{xx}, se halla de [Carl68]

$$\int_{-\infty}^{\infty} P_{xx}(F) dF = P_m \quad (1.13)$$

Y si comparamos la ecuación (1.13) con (1.12), intercambiando el orden del límite y la integral, tenemos $P_m = \int_{-\infty}^{\infty} \left[\lim_{Tc\to\infty} \frac{1}{Tc} |X_{aTc}(F)|^2 \right] dF = \int_{-\infty}^{\infty} P_{xx}(F) dF$, por lo que la PSD de la señal no periodica se puede expresar como

$$P_{xx} = \lim_{Tc\to\infty} \frac{1}{Tc} |X_{aTc}(F)|^2 \qquad [\text{Volts}^2/\text{Hz} = \text{Watts/Hz}] \quad (1.14)$$

Aunque, en general, el intercambio del límite y la integral que nos condujo a la expresión (1.14) no es estrictamente correcto, en un rango de frecuencias finitas, digamos f_1 y f_2, sí lo es; es decir,

$$\int_{f_2}^{f_2} P_{xx}(F) dF = \lim_{Tc\to\infty} \int_{f_1}^{f_2} \frac{1}{Tc} |X_{aTc}(F)|^2 dF \quad (1.15)$$

2. SEÑALES EN TIEMPO DISCRETO DE VALORES DISCRETOS

La mayoría de las señales de origen fisiológico son de naturaleza analógica, en tiempo continuo, aunque para su análisis por computador han de ser digitalizadas y convertidas a señales en tiempo discreto de valores cuantizados o discretos. Si queremos digitalizar la señal analógica $x_a(t)$ para ser procesada y analizada en computador debemos tomar muestras representativas de la señal continua a través del muestreo de la señal. Muestrear una señal continua (analógica) consiste en reemplazar la señal por sus valores en un conjunto de puntos discretos. Comúnmente, estos instantes de muestreo se distribuyen en intervalos regulares de tiempo, llamado *muestreo periódico.* Esto se describe por

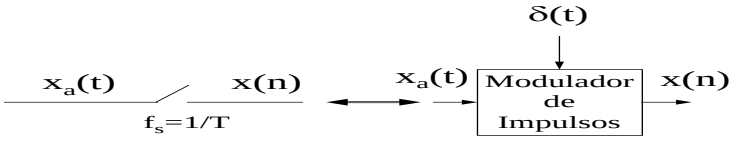
$$x(n) = x_a(nT_s), \qquad -\infty < n < \infty \tag{2.1}$$

donde $x(n)$ es la señal en tiempo discreto obtenida "tomando muestras" de la señal analógica $x_a(t)$ cada T_s segundos. La señal $x_a(t)$ muestreada, $x_a(nT_s)$, se denota también $x_a^*(t)$. El intervalo de tiempo T_s entre muestras sucesivas se llama *período de muestro* o *intervalo de muestreo* y su inverso: $1/T_s = f_s$ se denomina *frecuencia de muestreo* (en muestras por segundo o Hz). La conversión analógico-digital (A/D) de la señal $x_a(t)$ produce la secuencia, señal digital o serie de tiempo, $x(n)$, que puede considerarse como la señal de salida de un *muestreador ideal.* De esta forma, $x(n)$ se puede expresar en función de una suma de impulsos escalados, como el producto de la señal $x_a(t)$ por un tren de impulsos unitario

$$p(t)=\ldots+\delta(0)+\ \delta(t-T_s)+\ \delta(t-2T_s)+\ldots = \sum_{n=-\infty}^{\infty}\delta(t-nT_s) \tag{2.2}$$

de forma:

$$x(n) = x_a(nT_s) = \cdots + x_a(0)\delta(t) + x_a(T_s)\delta(t-T_s) + x_a(2T_s)\delta(t-2T_s) + \cdots \tag{2.3}$$



Representación de un Muestreador Ideal

2.1) Potencia y Energía en el dominio del tiempo

Como ejemplo de señal discreta utilizaremos la señal analógica $x_a(t) = A\ sin(2\pi F_0\ t)$, con $F_0 = 40$, A =2, muestreada a $f_s = 1000$ ($T_s = 0.001$). En MATLAB:

```
>>Fo=40;
>>To=1/Fo ;
>>fs=1000;
>>Ts=1/fs
```

Tomemos un ciclo de la señal muestreada.

```
>>t= Ts:Ts:To ;
>>x=2*sin(2*pi*Fo*t) ;
```

La Energía en un ciclo de señal, aproximando la integral de (1.1), se puede hallar como

```
>>Ener=trapz(t,x.^2)
```

Ener = 0.0499 [W-segundo]

La energía, en caso de muestreo normalizado = 1, es:

$$E = \sum_{n=1}^{N} |x(n)|^2 \tag{2.4}$$

Sin embargo, tomando en cuenta que las muestras están espaciadas en T_s=0.001, podemos calcular la energía como

$$E = T_s \sum_{n=1}^{N} |x(n)|^2 = \frac{1}{f_s} \sum_{n=1}^{N} |x(n)|^2 \qquad \text{[W-s]} \tag{2.5}$$

```
>> Ener=Ts*sum(x.^2) %
```
Ener = 0.0500 [W-s].

La potencia media de la señal discreta puede hallarse, a partir de (1.2) y (1.3), como:

$$P_m = \frac{1}{N} \sum_{n=1}^{N} |x(n)|^2 \qquad \text{(señal periodica de periodo N)} \tag{2.6}$$

$$P_m = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} |x(n)|^2 \quad \text{(señal no-periodica de duración 2N+1)} \tag{2.7}$$

Si se considera que la duración *N* es mucho mayor que el periodo de la componente mayor de frecuencia de la señal no-periodica a analizar, podemos utilzar la ecuación (2.6), donde *N* es igual al número de muestras de la señal *x(n)*.

El valor RMS de *x(n)* se halla de

$$x_{RMS} = \sqrt{\frac{1}{N} \sum_{n=1}^{N} |x(n)|^2} \tag{2.8}$$

Para hallar la potencia media de la señal considerando 10 ciclos de la sinusoide:

```
>>t= 0:Ts:10*To ;
>>x=2*sin(2*pi*Fo*t) ;
>> Ener=Ts*sum(x.^2) ;    %   Ener =   0.5000 [W-s]
>> Ener=trapz(t,x.^2) ;      %  Ener =    0.5000 [W-s]
>> Pm=trapz(t,x.^2)/(t(end)-t(1)) % trapz toma la integral de x por el
                                  % método trapezoidal, respecto a t
```
P = 2 [Watts]
```
>> Pm=1/length(x)*sum(x.^2)   % utilizando la ecuación (2.6)
```
Pm = 1.992 [W]

Si tomamos 3 segundos de señal (120 ciclos):

```
>> t=0:Ts:3;
>> x=2*sin(2*pi*Fo*t);
>> N=length(x); % Número de muestras de la señal (N=3001)
>> Ener=Ts*sum(x.^2)  %
```
Ener = 6.0000 [W-s]

La potencia media es

```
>> Pm=1/length(x)*sum(x.^2)
```
Pm = 1.9993

Si t=Ts:Ts:3, entonces Pm = 2.000 [W]

2.2) Análisis de Fourier de la señal en tiempo discreto

La ***transformada de Fourier de una señal en tiempo discreto*** (DTFT), *x(n)* es

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \tag{2.9}$$

Para señales discretas, *x(n)*, de duración finita: $0 \le n \le N$-1 (N muestras), como suelen ser las señales biomédicas adquiridas en la práctica, se calculan N valores de la transformada de Fourier entre los puntos $0 \le k \le N-1$. Estos coeficientes se hallan muestreando $X(e^{j\omega})$ sobre el eje ω (normalizado), entre $0 \le \omega \le 2\pi$ (muestreo en el dominio de la frecuencia) para valores $\omega = 2\pi k/N$, $0 \le k \le N-1$. Se halla así la ***transformada discreta de Fourier*** (DFT) de la señal *x(n)*

$$X(k) = X(e^{j\omega})\Big|_{\omega=2\pi k/N} = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}, 0 \le k \le N-1 \tag{2.10}$$

En el caso de la señal analógica $x_a(t)$, muestreada y expresada por el tren de impulsos, tenemos que su transformada de Fourier es (por la propiedad de traslación en el tiempo)

$$\begin{aligned} X(e^{j\omega}) &= X^*(F) = \cdots + x_a(0) + x_a(T_s)e^{-j\omega T_s} + x_a(2T_s)e^{-j\omega 2T_s} + \cdots \\ X(e^{j\omega}) &= X^*(F) = \sum_{n=-\infty}^{\infty} x_a(nT)e^{-j\omega nT_s} = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega nT_s} \end{aligned} \tag{2.11}$$

donde $x(n) \equiv x_a(nT)$ y $X^*(F)$ denota el espectro de la señal $x_a(t)$ muestreada, $x_a^*(t)=x_a(nT_s)$.

Si usamos la representación en serie de Fourier de un tren de impulsos unitarios

$$p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT_s) = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} e^{j2\pi kt/T_s} \tag{2.12}$$

la señal muestrada se puede expresar

$$x_a(nT_s) = \left(\frac{1}{T_s} \sum_{n=-\infty}^{\infty} e^{j2\pi kt/T_s} \right) x_a(t) \tag{2.13}$$

Así, el espectro de esta representación de $x_a(nT)$ es

$$X(e^{j\omega}) = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} X(F - kf_s) \tag{2.14}$$

El espectro de la señal $x_a(nT)$ alrededor de 0, se repite indefinidamente alrededor de k veces la frecuencia de muestreo f_s, escalado por f_s $(1/T_s)$.

La DFT de la señal *x(n)*[1](ecuación (2.10)) se expresar, de acuerdo con los índices en MATLAB, como

$$X(k) = \sum_{n=1}^{N} x(n)e^{-j2\pi(k-1)(n-1)/N}, \text{ para } 1 \le k \le N \tag{2.15}$$

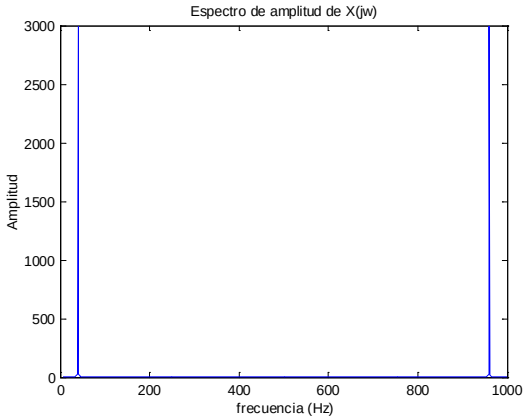
```
>>X=fft(x); % equivalente a X=fft(x,N);
```

[1] El índice de la primera muestra de la señal *x(n)* es 0 y la última muestra es N-1 para la fórmula de Fourier. En Matlab, la primera muestra tiene índice 1 y la última N

El cálculo de la DFT, que se realiza en MATLAB con la FFT, es más rápido si se hace con un número de puntos, *Nfft*, potencia de 2. Si utilizamos un número de coeficientes *Nfft=4096*, se hallaría: **X=fft(x,Nfft)=fft(x,4096)**; En este caso el vector **X** es de *4096* elementos. Sigamos este ejemplo con las 3001 muestras de *x* y **X=fft(x)**. Utilizaremos *Nfft* para denotar el número de puntos de cálculo de la **fft** y *N* el tamaño del vector *x(n)*.

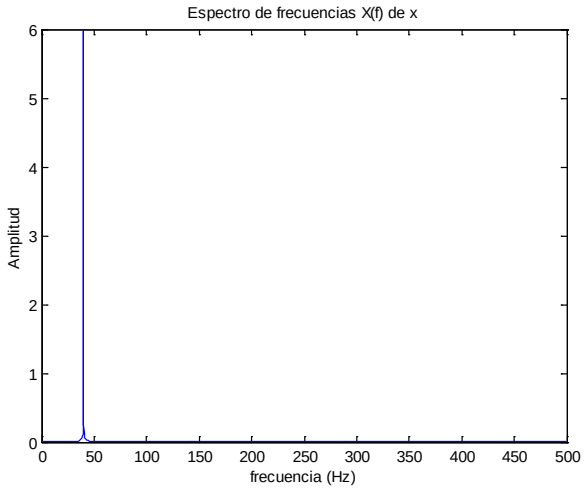
Como la señal *x* (de 3 s de duración) fue muestreada a *fs = 1000* Hz, entonces los índices que representan los *3001* valores de Fourier *k* de *X(k)* deben escalarse para llevarlos a escala de frecuencia, *f*, como *f=k fs/Nfft*. Así *X(k)* → *X(f)*. Trabajemos con *Nfft = N*;

```
>> f=(0:N-1)*fs/N;
>> plot(f,abs(X))
```



Se puede observar que la amplitud está escalada por fs=1000. Es decir, si dividimos abs(X) entre fs, de acuerdo a la ecuación (2.14), obtenemos los valores de amplitud del espectro equivalentes a la señal analógica muestreada. Es conveniente, además, expresar el espectro de frecuencias de esta señal (originalmente analógica) en el ancho de banda de 0 a la mitad de la frecuencia de muestreo, es decir, hasta la *frecuencia de Nyquist*.

```
>>if rem(N,2), % se evalua la paridad de N
>>      select = 1:(N+1)/2; % si N es impar
>>else
>>      select = 1:N/2+1; % si N es par
>> end
>>% select en este caso es de 1501 elementos
>>plot(f(select),Ts*2*abs(X(select)))
```



2.3) Potencia y Energía en el dominio de Fourier

La energía total de la señal, en el dominio de Fourier está dada por (muestreo normalizado)

$$E = \frac{1}{N}\sum_{k=0}^{N-1}\left|X(f)\right|^2 \tag{2.16}$$

Tomando en cuenta que las muestras están espaciadas en T_s (en el ejemplo T_s=0.001), podemos calcular la energía como (tomaremos los índices como en MATLAB $1 \le n \le N$, en lugar de $0 \le k \le N\text{-}1$)

$$E = \frac{T_s}{N}\sum_{n=1}^{N}\left|X(f)\right|^2 = \frac{1}{N\,f_s}\sum_{n=1}^{N}\left|X(f)\right|^2 \qquad \text{[W-s]} \tag{2.17}$$

```
>> Ener=(Ts/N)*sum(abs(X).^2)  % equivalente a Ener =1/(fs*N)*sum(abs(X).^2)
 Ener =  6.0000  [W-s]
```

Por lo cual, la Energía puede ser hallada de acuerdo al teorema de Parseval de conservación de la energía en ambos dominios, considerando la frecuencia de muestreo, como:

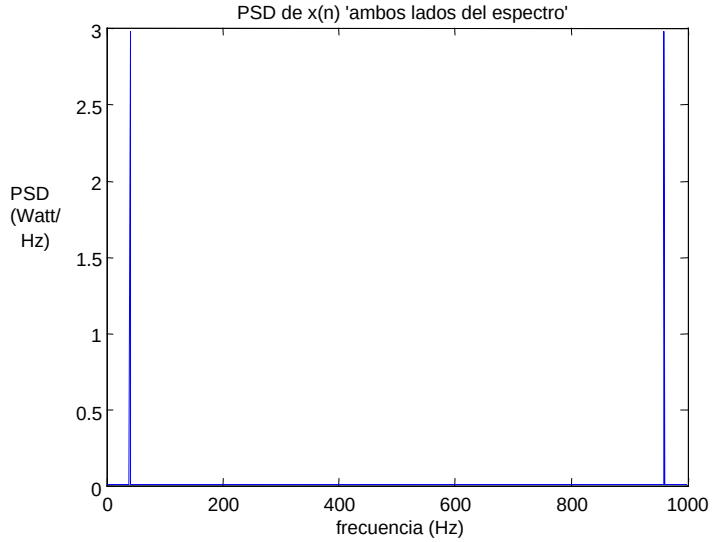
$$E = \frac{1}{f_s}\sum_{n=1}^{N}\left|x(n)\right|^2 = \frac{1}{f_s N}\sum_{n=1}^{N}\left|X(f)\right|^2 \tag{2.18}$$

La *densidad de potencia espectral* (PSD)[2], P_{xx}, está dada, por la expresión

$$P_{xx}(f) = \frac{1}{f_s N}\left|X(f)\right|^2 \qquad \text{[Watt/Hz]} \tag{2.19}$$

Hallemos la PSD de la señal $x_a(t)$ de 3 s, discretizada, con el espectro completo (los dos lados simétricos del espectro),

```
>> Pxx=(abs(X).^2)/(N*fs);
>> plot(f,Pxx)
```



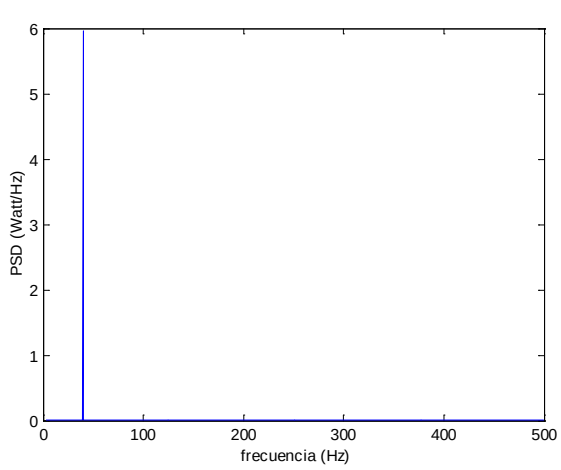
[2] del inglés *power spectral density*

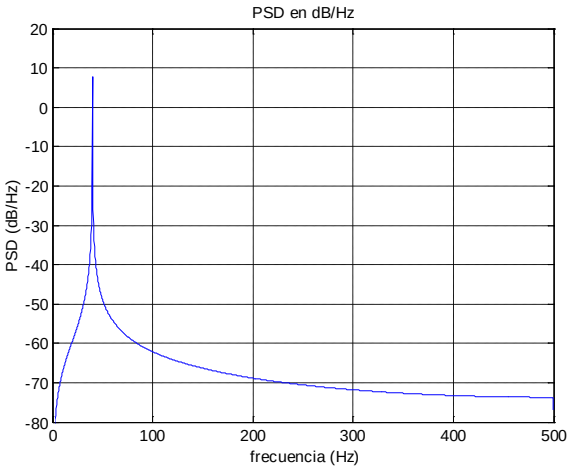
La energía total de la señal coincide con la hallada anteriormente tanto en el tiempo como en la frecuencia. Para hallar el espectro de "un solo lado", representando la densidad de potencia espectral desde 0 hasta la frecuencia de Nyquist o mitad de la frecuencia de muestreo:

```
>> Pxx=Pxx'; % vector columna
>> Pxx_unlado = Pxx(select); % se toman solo la mitad de las frecuencias o sea hasta fs/2
>> Pxx_u = [Pxx_unlado(1); 2*Pxx_unlado(2:end-1); Pxx_unlado(end)];
>> f_u = f(select);
>>plot(f_u, Pxx_u),
```

Se presenta también, a la derecha, la PSD en dB/Hz,

```
>>plot(f_u,10*log10(Pxx_u)), grid
>> axis([0 500 -80 20])
```





La potencia media de esta señal se puede aproximar, utilzando las ecuaciones (2.6), (2.7), (2.16) y (2.19), a partir del valor Pxx como:

$$P_m = \frac{1}{N^2}\sum_{n=1}^{N}\left|X(f)\right|^2 = \frac{fs}{N}\sum_{k=1}^{N} Pxx(k) \qquad (2.19)$$

Este cálculo sería equivalente al realizado por la ecuación (1.13):

$$P_m\Big|_{\text{entre } f_1 \text{ y } f_2} = \int_{f_2} P_{xx}(F)dF = \lim_{Tc\to\infty}\int_{f_1}^{f_2}\frac{1}{Tc}\left|X_{aTc}(F)\right|^2 dF \qquad (2.20)$$

cuando el tiempo de duración de la señal truncada es mucho mayor que los periodos de los componentes de mayor frecuencia. Probemos las dos ecuaciones para hallar la potencia media de este segmento de señal sinusoidal

```
>> Pxx=(abs(X).^2)/(N*fs);
>> Pm=(fs/N)*sum(Pxx) % igual a Pm=(fs/N)*sum(Pxx_u)
```
Pm = 1.9993 [Watts]
```
>> Pm=trapz(f_u, Pxx_u) % En este caso las frecuencias se integran entre f1 = 0 Hz y f2 = fs/2 Hz
```
Pm = 1.9993 [W]

El valor de Pm es el mismo que el obtenido en el dominio del tiempo. Si se utiliza t=Ts:Ts:3, entonces Pm = 2.000 W. El valor medio teórico de la potencia para la señal continua es $A^2/2 = 2$ W.

3. ANÁLISIS ESPECTRAL CON TÉCNICAS NO PARAMÉTRICAS (FOURIER)

3.1) Introducción

Las técnicas actuales de análisis espectral tienen su origen en la transformada de Fourier de señales en tiempo continuo. Remontándonos a los primeros trabajos de análisis espectral, podemos partir de Isaac Newton, que introdujo el término *spectrum* para describir las bandas de múltiples colores en que se descompone la luz al pasar por un prisma [Mar87][3]. La propia palabra *spectrum* o *espectro* se origina del latín imagen o fantasma. Bernoulli, Euler y Fourier hicieron grandes aportaciones al estudio de ecuaciones de ondas aplicadas al análisis espectral de vibración de cuerdas musicales y teoría analítica del calor, introduciendo los términos de *análisis armónico*. A finales del siglo XIX y principios del XX, Schuster introdujo el término *periodograma* para aplicar el estudio del análisis armónico a varios campos, entre ellos la periodicidad obtenida en el estudio de series temporales de "manchas solares".

3.2) Resolución espectral y dispersión espectral (*spectral lekeage*)

Los métodos de estimación espectral con la transformada de Fourier se basan en secuencias infinitas. Esto crea algunos problemas cuando tomamos secuencias finitas o segmentos de señal para analizar. Por un lado, un intervalo finito de observación T (segmento $T = N*T_s$) limita la ***resolución espectral***, Δf_n, a un valor "fundamental" $\Delta f_n = 1/T$. Por ejemplo, en un segmento corto de señal de 120 ms de duración, la resolución espectral fundamental es de 8,33 Hz. Con las técnicas de Fourier esta resolución no puede mejorarse incrementando la frecuencia de muestreo, ni tampoco agregando ceros para aumentar el tamaño del segmento. Por otro lado, este truncamiento en el dominio del tiempo, equivalente a multiplicar la secuencia por una ventana rectangular, produce discontinuidades en los bordes del segmento, empeorando la resolución espectral debido al efecto de dispersión espectral[4] y distorsionando el espectro a estimar. Por las propiedades de la transformada de Fourier se tiene que la multiplicación de dos funciones en el dominio del tiempo es equivalente a la convolución en el dominio frecuencial de los espectros de esas funciones. Por ejemplo, una sinusoide continua pura de duración infinita, de frecuencia f_0, tendrá como espectro un impulso en f_0. En la práctica, si se toma un segmento de la sinusoide con una ventana dada, el espectro obtenido es el de la ventana, centrado en f_0. Como resultado de esto, el espectro del segmento a analizar es afectado por el tipo de ventana que se use en la selección de la porción de la señal biomédica a analizar. El fenómeno de *esparcimiento* del impulso, que representa al espectro de la sinusoide pura, a la forma del espectro de la ventana (de tipo lóbulo principal con lóbulos laterales) se denomina **dispersión espectral** e implica que la resolución espectral, Δf_n sea $\geq 1/T$. Esta relación se llama también compromiso del producto tiempo − ancho de banda (resolución espectral):

$$\Delta f_n T \geq 1 \tag{3.1}$$

La dispersión espectral se puede reducir, para mejorar la distorsión, con el uso de ventanas de tipo cosenoidales que suavizan los extremos del segmento. Una definición práctica de resolución espectral, con el uso de ventanas, es el "ancho de banda efectivo" del espectro de la ventana [Har78], considerado como el intervalo de frecuencia que contiene la energía

[3] Un análisis mucho más completo, con una amplia introducción histórica, de estimación espectral se puede obtener en el libro de S.L. Marple Jr. [Mar87]

[4] En inglés *spectral leakage*

significativa del espectro. La resolución espectral "efectiva", Δf_e, se puede expresar como

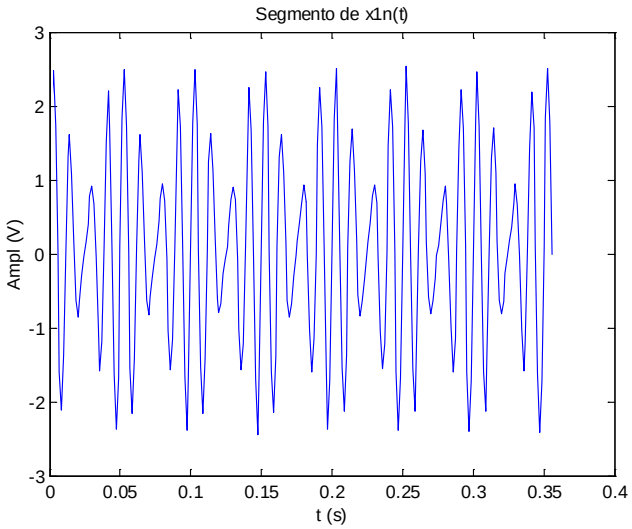
$$\Delta f_e = K_w / T \tag{3.2}$$

donde K_w es una constante que refleja el incremento del ancho de banda para cada tipo de ventana. Se suelen usar varios criterios para definir la energía significativa: *(i)* el 95 % de la energía total (el más conservador); *(ii)* la energía del espectro cuando éste cae 6 dB (sugerido por Harris, 1978); *(iii)* la energía del espectro cuando éste cae 3 dB (el menos conservador). Para ventanas de tipo cosenoidales, típicamente usadas en señales biomédicas, como las de Blackman-Harris, Hamming, Hanning o Keisser-Bessel, K_w está en el rango de 3,6 a 4 (usando el criterio más conservador). Esto implica que para un segmento de 120 ms la resolución espectral efectiva es alrededor de 32 Hz. Ventanas como la de Blackman-Harris de 4 términos producen una reducida dispersión espectral, pues presenta lóbulos laterales muy atenuados. En esta ventana K_w = 2,44, utilizando el criterio de -6 dB, lo que produce una resolución espectral de 20 Hz para el segmento de 120 ms.

La estimación de la potencia espectral es realizada a partir de la señal misma. Este análisis se realizará utilizando el *toolbox* "*signal processing*" de MATLAB. A manera de ejemplo utilizaremos una primera señal formada por dos sinusoides de frecuencias cercanas (80 y 100 Hz), de 2 segundos de duración, muestreadas a 700 muestras por segundo con ruido añadido. Posteriormente utilizaremos el QRS de un electrocardiograma (ECG) adquirido a *fs* = 1000 muestras por segundo y señales RR de variabilidad de ritmo cardiaco.

Señal 1 : 1,5 sen(2π 80 t) + sen(2π 100 t) + ruido. En MATLAB:

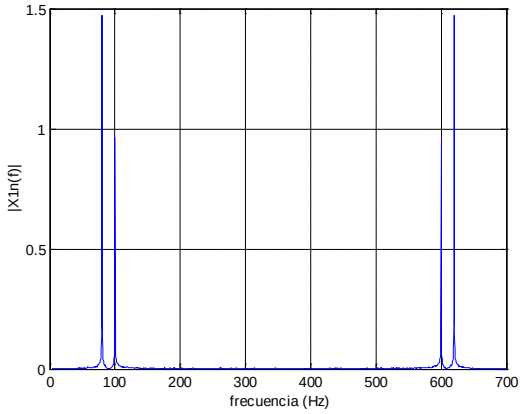
```
>>fs=700 ;
>>t = (0 :1/fs :2)' ;
>>x1 = 1.5*sin(2*pi*80*t) + sin(2*pi*100*t) ;
>>ruido= 0.1*rand(size(t)) ;
>>x1n = x1 + ruido ;
>>N=length(x1n)   % N =  1401
```



La energía de esta señal se puede calcular en ambos dominios

```
>> Ener=trapz(t,x1n.^2)    % Area: Integral sobre el segmento de señal
>> Ener=sum(x1n.^2)/fs     % Como señal discreta
>> X1n=fft(x1n);
```

```
>> Ener =1/(fs*N)*sum(abs(X1n).^2)  % En el dominio de Fourier
  Ener = 3.2557 [W-s]
>> f=(0:N-1)' *fs/N;
>> plot(f,abs(X1n)/fs) %Amplitud escalada equivalente al espectro de la señal analógica
```



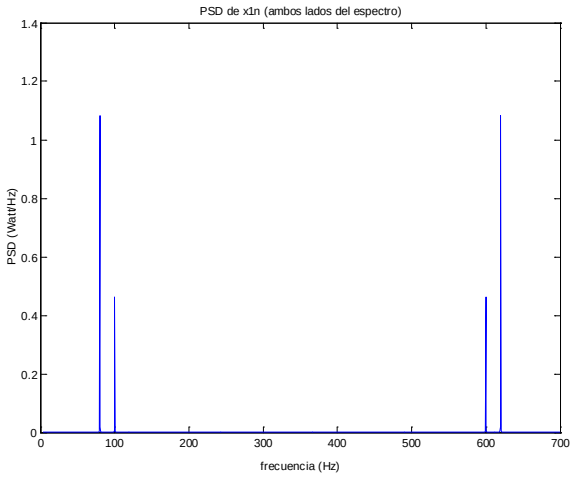
La gráfica muestra la magnitud de la transformada de Fourier, escalada, considerando la frecuencia en que fue muestreada la señal original.

3.3) Densidad de Potencia Espectral (PSD) con el periodograma estándar.

El periodograma estándar de una señal discreta está dado por la PSD hallada en la sección anterior, $P_{xx}(f) = \frac{1}{f_s N}\left|X(f)\right|^2$, aplicado a un segmento de señal con una ventana rectangular.

De la misma forma como fue hallado previamente la PSD con una sinusoide pura, el periodograma de la señal 1 puede encontrarse a través de:

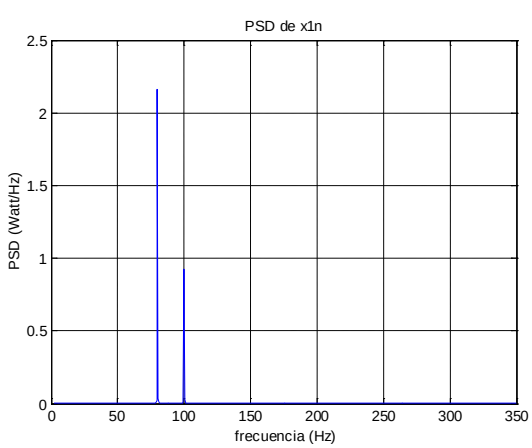
```
>> X1n=fft(x1n);
>> Pxx=(abs(X1n).^2)/(N*fs);
>> f=(0:N-1)' *fs/N;
```

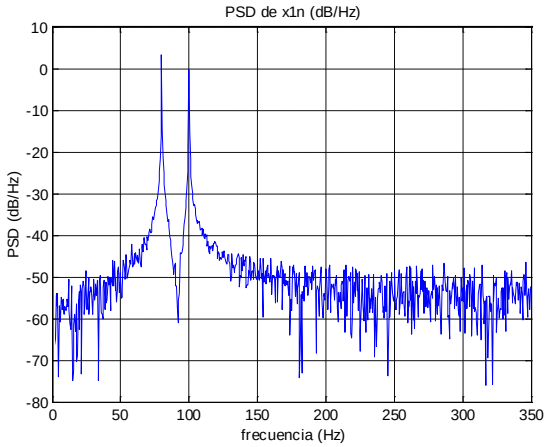


La PSD desde 0 hasta la mitad de la frecuencia de muestreo:

```
>>if rem(N,2),  % se evalua la paridad de N
>>      select = 1:(N+1)/2; % si N es impar
>>else
>>      select = 1:N/2+1; % si N es par
>> end
```

```
>> % select en este caso es de 701 elementos
>> Pxx_unlado = Pxx(select); % se toman solo la mitad de las frecuencias o sea hasta fs/2
>> Pxx_u = [Pxx_unlado(1); 2*Pxx_unlado(2:end-1); Pxx_unlado(end)];
>> plot(f(select),Pxx_u), grid
>> plot(f(select),10*log10(Pxx_u)), grid % PSD en db/Hz
```





En MATLAB podemos obtener la misma gráfica de la PSD en dB o los valores de *Pxx* y *f* con

```
>> periodogram(x1n,[],N,fs);   % Se obtiene la gráfica directamente
>> [Pxx,f]=periodogram(x1n,[],N,fs);
>> plot(f,10*log10(Pxx)), grid
```

La energía total de la señal se halla también de Pxx, como

$$E = \sum_{k=1}^{N} Pxx(k) \tag{3.3}$$

```
>> Ener=sum(Pxx)
   Ener =    3.2557  [W-s] [V^2-s]
```

La potencia media Pm de la señal sería:

```
>> Pm=fs*sum(Pxx_u)/N
    Pm = 1.6267
```

La potencia media de cada componente sinusoidal es $A^2/2$. Para la componente de 80 Hz, con amplitud A=1,5 V, P_{m1}= 1,125 W, y para la componente de 100 Hz, de Amplitud 1 V, P_{m2} = 0,5 W. La suma es Pm = P_{m1} + P_{m2} = 1,625 W.

Si quisiéramos hallar la energía de la banda de frecuencia de 75 a 85 Hz, utilizamos:

```
>> ind75_85=find((f >= 75) & (f <= 85));
>> Ener75_85=sum(Pxx(ind75_85))
   Ener75_85 =    2.2543  [W-s] [V^2-s]
```

La Potencia media en esta banda es:

```
>> Pm75_85=fs*sum(Pxx_u(ind75_85))/N
   Pm75_85 =   1.1254   [W]   [V^2]
```

La energía de la banda de frecuencia de 95 a 105 Hz, utilizamos:

```
>> ind95_105=find(f >= 95 & f <= 105);
```

```
>> Ener95_105=sum(Pxx_u(ind95_105))
  Ener95_105 = 0.9864  [W-s] [V^2-s]
```

La Potencia media en esta banda es:

```
>> Pm95_105=fs*sum(Pxx_u(ind95_105))/N
  Pm85_105 = 0.4929 [V^2]
```

Si utilizamos 4096 puntos para calcular el periodograma (Nfft = 4096)

```
>> [P,f]=periodogram(x1n,[],Nfft,fs);  % Nfft = 4096;
```

Obtenemos la misma gráfica PSD (o similar). Sin embargo, la energía total se halla:

$$E = \frac{N}{N_{fft}} \sum_{k=1}^{N_{fft}} Pxx(k) \tag{3.4}$$

```
>> Ener=sum(P)*N/Nfft  % donde N = length(x1n) = 1401
```

Ener = 3.2557 % el mismo valor que el hallado con el periodograma usando N puntos para hallar la fft

En este caso la energía en la banda de frecuencia de 75 a 85 Hz sería:

```
>> ind75_85=find(f >= 75 & f <= 85);
>> Ener75_85=sum(P(ind75_85))*N/Nfft
        Ener75_85 = 2.2262 [W-s] [V^2-s]
```

La potencia media, utilizando Nfft puntos se halla

$$Pm = \frac{fs}{N_{fft}} \sum_{k=1}^{N_{fft}} Pxx(k) \tag{3.5}$$

```
>> Pm=fs*sum(P(ind75_85))/Nfft
      Pm  =   1.1124  [V^2-s]
```

La función **periodogram** de MATLAB permite incluir una ventana a la señal de entrada con la cual se calcula la PSD. El valor por defecto (2da entrada en el comando utilizado previamente = [ ]) es la ausencia de ventana, que equivale a una ventana rectangular o boxcar(N).

3.4) Periodograma modificado (señal enventanada)

El uso de ventanas que suavicen el inicio y final de las ondas es ampliamente recomendado para el análisis de segmentos de señales. Si queremos enventanar previamente la señal *x1n(n)*, con una ventana, *w(n)*, definimos el tipo de ventana en el comando **periodogram**. Por ejemplo, en MATLAB, una ventana de Hamming se puede definir:

```
>>w=hamming(length(x1n));  % equivale a w=hamming(N);
```

Para enventanar *x1n*:

```
>>xw=x1n.*w;
>>Xw=fft(xw);
```

La Potencia espectral, de la señal enventanada *xw* queda

$$Pxw(f) = \frac{|Xw(f)|^2}{f_s NU}, \tag{3.6}$$

donde U es la constante de normalización de la ventana

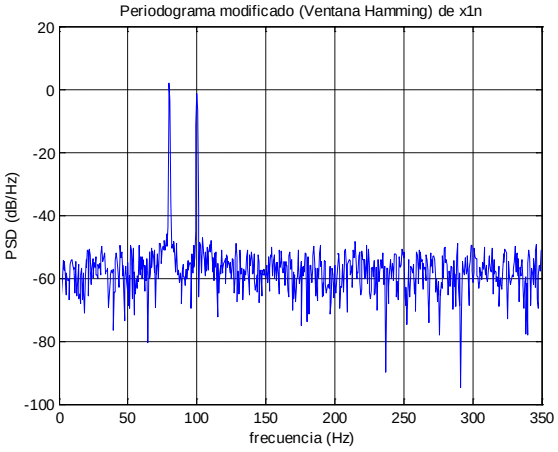
$$U = \sum_{n=0}^{N_{fft}-1} |w(n)|^2 / N \qquad (3.7)$$

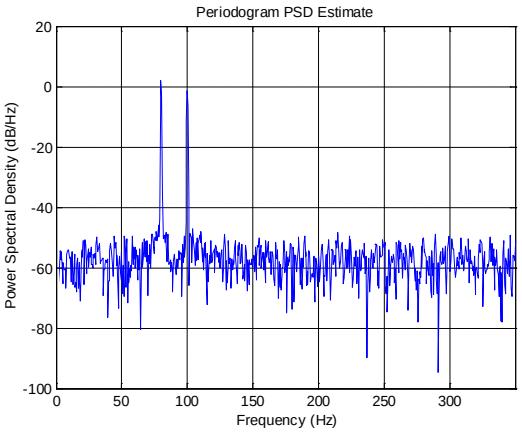
En MATLAB, podemos calcular el periodograma enventanado utilizando las ecuaciones previas o directamente con la función **periodogram**. Veamos ambos métodos:

```
>> U=sum(w.^2)/N;
>> Pxxw=(abs(Xw).^2)/(N*fs*U);
>> Pxx_unlado = Pxxw(select);
>> Pxxw_u = [Pxx_unlado(1); 2*Pxx_unlado(2:end-1); Pxx_unlado(end)];
>> ind75_85=find(f >= 75 & f <= 85);
```

La energía en la banda de frecuencia de 75 a 85 Hz es

```
>> Ener75_85=sum(Pxxw_u(ind75_85))
  Ener75_85 = 2.2520  [W-s] [V^2-s]
>> plot(f(select),10*log10(Pxxw_u)), grid
>> periodogram(x1n,hamming(N),N,fs);
```



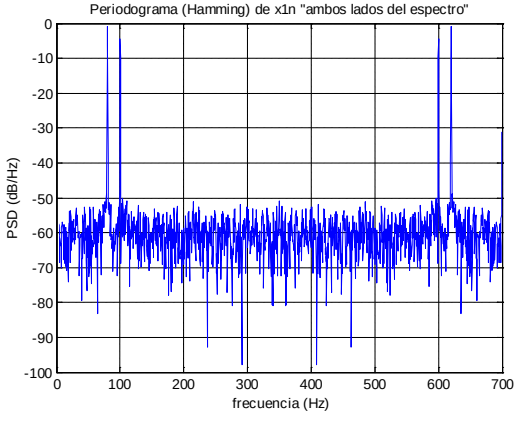


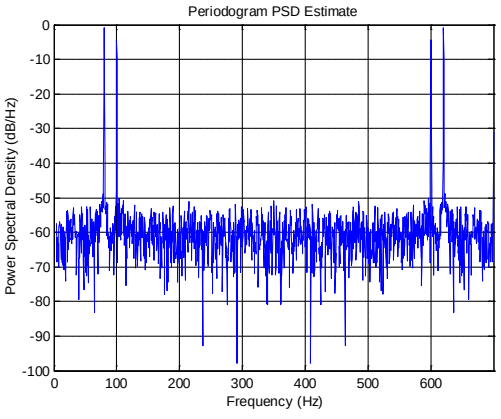
La ausencia del uso de una ventana en el segmento de señal usada se interpreta como una ventana rectangular. La ventana rectangular en el comando **periodogram,** es el valor por defecto del tipo de ventana: se coloca en la 2da entrada [ ] o boxcar(length(x1n)). Los tipos de ventanas se describen en la página. 4-13 del tutorial de Signal Processing.

Los espectros “de ambos lados” se pueden hallar como:

```
>>plot(f,10*log10(Pxxw)),grid
```

```
>> periodogram(x1n,hamming(N),N,fs,'twosided');
```





3.5) Método de Welch (periodograma enventanado, segmentado y promediado)

El método propuesto por Welch divide la señal temporal en segmentos (que pueden solaparse), se halla el periodograma enventanado a cada segmento y se promedian las PSD estimadas. En MATLAB, la función **pwelch**, en sus valores por defecto, divide la señal en 8 segmentos con 50% de solapamiento y cada segmento es enventanado por una ventana de hamming y se promedian los 8 periodogramas modificados:

```
>>[Pxx,f] = pwelch(x,window,Noverlap,NFFT,Fs)
                % MATLAB version 6 y 7 (Signal Processing toolbox 5.x, 6.x)
```

Versión previa: >>[Pxx,f]=pwelch(x,NFFT,Fs,window,Noverlap) % MATLAB version 5 (Signal 4.2)

window: si es un vector, se divide x en segmentos solapados del tamaño de window;
si es un entero, se divide x en ese número de segmento solapados.

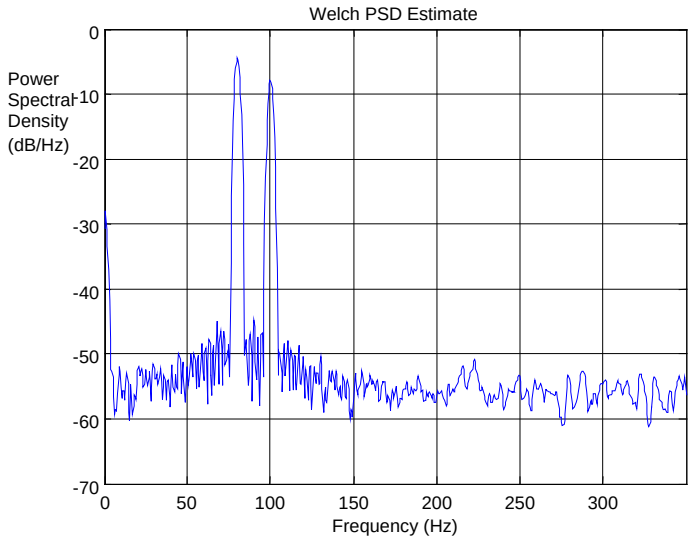
Noverlap, número de puntos de solapamiento

NFFT: número de puntos del cálculo de Fourier

Fs: Frecuencia de muestreo

Al igual que las demás funciones que estiman PSD en MATLAB, la ausencia de argumentos de salida de pwelch produce el espectro directamente en la ventana de figuras

```
>> pwelch(x1n,[],[],N,fs);
```



4. ANÁLISIS ESPECTRAL CON MÉTODOS PARAMÉTRICOS

Los modelos paramétricos de las señales son técnicas alternativas a la transformada de Fourier para el análisis espectral de estas series temporales. Cuando las señales son cortas o ruidosas, los métodos paramétricos pueden proveer mejor resolución que los métodos no-paramétricos para estimar la densidad de potencia espectral. Además no necesitan de un enventanado previo.

4.1) Modelado de señales

Los métodos paramétricos *modelan* la señal como la respuesta de un sistema lineal con ruido blanco, e intentan estimar los parámetros de ese sistema lineal.

Un modelo paramétrico genérico de un sistema (o de una señal, como caso particular), lineal o no lineal, puede expresarse como

$$y(n) = f(y(n-1), y(n-2), \cdots, u(n), u(n-1), \cdots) + v(n) \tag{4.1}$$

donde $y(n)$ es la señal de salida, $u(n)$, la de entrada y $v(n)$ es el término aditivo no considerado o "explicado" por el modelo, que se considera el error o ruido. La función $f(\cdot)$ puede ser parametrizada con un vector de parámetros θ, de dimensión finita, y expresada en términos de un vestor de regresión $\varphi(n)$. De esta forma, el sistema en tiempo discreto de la ecuación (4.1) representa una estructura general de modelo de regresión:

$$y(n) = f(\varphi(n), \theta) + v(n) \tag{4.2}$$

siendo $\varphi(n)$ es el vector de regresión de dimensión finita

$$\varphi(n) = \varphi(y(n-1), \cdots, y(n-na), u(n), u(n-1), \cdots, u(n-nb)) \tag{4.3}$$

donde na y nb son los retardos máximos considerados para las muestras de las señales salida y entrada, respectivamente. Los parámetros se seleccionan de $\theta = \hat{\theta}_N$; es decir, son ajustados a partir de los valores de entrada y salida, minimizando una función de pérdida $V_N(\theta$, definida en términos del error entre la señal real y la modelada:

$$\hat{\theta}_N = \arg\min_{\theta} V_N(\theta) \tag{4.4}$$

(arg min significa el argumento que minimiza), siendo

$$V_N(\theta) = \frac{1}{N}\sum_{n=1}^{N} (y(n) - f(\varphi(n), \theta))^2 \tag{4.5}$$

El problema del modelado paramétrico de sistemas o señales se centra en: (i) escoger una estructura de modelo $f(\cdot)$ apropiada, (ii) con un número de parámetros adecuado y (iii) utilizar métodos de optimización o estimación paramétricas cónsonos al modelo.

Diversas nomenclaturas se han presentado en la literatura de sistemas de identificación y control (ver Ljung [Lju87]) y de procesado de señales (ver Therrien [The92], Proakis [Pro92]) para el modelado de sistemas y señales. Consideremos, del modelo paramétrico genérico, una estructura simple lineal donde la señal de salida dependa de sus valores pasados y de los valores de la entrada con un error o ruido añadido $v(n)$:

$$y(n) = -a_1 y(n-1) - \cdots - a_{na} y(n-na) + b_0 u(n) + \cdots + b_{nb} u(n-nb) + v(n) \tag{4.6}$$

Esta estructura se conoce como modelo ARX (AutoRegresiva con componente eXógena) en la literatura de identificación de sistemas. El modelo ARX se refiere también como enfoque ARMA (AutoRegresivo de promedio móvil –*Moving Average*-) indirecto del modelado de la señal en la literatura de procesado de señales (ver Therrien [The92]). Si tomamos la transformada z de la estructura ARX despreciando el término de error, queda

$$\frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + \cdots b_{nb} z^{-nb}}{1 + a_1 z^{-1} + \cdots a_{na} z^{-na}} = \frac{B(z)}{A(z)} \tag{4.7}$$

Si en lugar de la transformada *z* utilizamos el operador de desplazamiento hacia delante *q* (tal que $q^{-1}y(n)=y(n-1)$), entonces *A(q)* y *B(q)* representarán los operadores en el dominio del tiempo de *A(z)* y *B(z)*, respectivamente, quedando el modelo ARX

$$A(q)y(n) = B(q)u(n) + \nu(n) \tag{4.8}$$

donde

$$A(q) = 1 + a_1 q^{-1} + \cdots + a_{na} q^{-na} \tag{4.9}$$

$$B(q) = b_0 + b_1 q^{-1} + \cdots + b_{nb} q^{-nb} \tag{4.10}$$

En forma general, la estructura de un sistema lineal en tiempo discreto se puede representar

$$y(n) = G(q)u(n) + H(q)e(n) \tag{4.11}$$

donde *G(q)* sería la función de transferencia del sistema, que evaluada en el círculo unitario ($q=e^{j\omega}$) sería una función de frecuencia $G(e^{j\omega})$. El término de error o perturbación *ν(n)* está expresado en función de *e(n)*: ruido blanco de varianza σ_e^2, filtrado por *H(q)*. Las propiedades de *ν(n)* se pueden expresar en términos de su espectro de potencia como

$$\Phi_\nu(\omega) = \sigma_e^2 \left| H(e^{j\omega}) \right|^2 \tag{4.12}$$

En forma paramétrica, la familia de modelos sería

$$A(q)y(n) = \frac{B(q)}{F(q)}u(n) + \frac{C(q)}{D(q)}e(n) \tag{4.13}$$

El modelado de señales puede considerarse un caso particular de esta familia de modelos cuando la entrada *u(n)* no está presente. Considerando *D(q)* = 1, tenemos la estructura de modelo **ARMA** (Auto Regresivo de promedio (*Average*) Móvil)

$$A(q)y(n) = C(q)e(n) \tag{4.14}$$

Esta estructura es conocida en la literatura de procesado de señales con *B(q)=C(q)*, y expresada en términos de la transformada z, como: *A(z)y(n)=B(z)e(n)*. El modelo ARMA es autoregesivo de orden *na* y de promedio móvil de orden *nb*. La estructura **AR** depende solamente de los valores pasados de *y(n)* y es una auto-regresión de orden *na*,

$$A(q)y(n) = e(n) \tag{4.15}$$

Esta ecuación se puede describir como

$$y(n) = -a_1 y(n-1) - \cdots - a_{na} y(n-na) + e(n) = -\sum_{i=1}^{na} a_i y(n-i) + e(n) \tag{4.16}$$

La estructura AR puede expresarse en función de un vector de parámetros θ y un vector de regresión ϕ como

$$y(n) = \varphi^T \theta + e(n) \tag{4.17}$$

donde $\theta = [a_1 \cdots a_{na}]^T$ y $\varphi^T = [-y(n-1) \cdots - y(n-na)]$.

La estimación de los parámetros de los polinomios AR, ARMA, ARX, etc, se puede realizar a través de distintos algoritmos, los cuales buscan minimizar una función de pérdida o error. En MATLAB, se pueden hallar los modelos paramétricos AR o ARMA con el *toolbox* de *Signal Procesing* o con el de *System Identification*.

4.1.1) Estimación de Modelos AR

De la ecuación (4.15) podemos expresar un modelo AR como

$$F(q) = \frac{\sqrt{e}}{A(q)} = \frac{\sigma_e}{1 + a_1 q^{-1} + \cdots + a_{na} q^{-na}} \tag{4.18}$$

Los métodos incluidos en el *Signal Procesing* para el modelado AR son:

a) Método de Burg. Minimiza por mínimos cuadrados la media aritmética del error de predicción hacia delante y hacia atrás, satisfaciendo el recursivo de Levinson-Durbin [Bur67]. La función es **arburg**:

```
>> [A,e]=arburg(y,na);
```

donde A es el vector de parámetros θ estimados de orden *na*, *e* es la varianza del error (σ_e^2), *y* es la señal de salida del sistema AR o señal a modelar.

b) Método de Yule-Walker, de autocorrelación o eneventanado. Minimiza el error de predición hacia delante por mínimos cuadrados. Se forman las ecuaciones de Yule-Walker que se resuelven por el recursivo de Levinson-Durbin. La función es **aryule**.

```
>> [A,e]=aryule(y,na);
```

c) Método de la Covarianza. Halla el valor de los parámetros, minimizando el error de predicción hacia delante por mínimos cuadrados. La función es **arcov**.

```
>> [A,e]=arcov(y,na);
```

d) Método de la Covarianza modificado. Halla el valor de los parámetros, minimizando el error de predicción hacia delante y hacia atrás por mínimos cuadrados. La función es **armcov**.

```
>> [A,e]=armcov(y,na);
```

Los modelos AR en *System Identification toolbox* se pueden hallar con la función **ar**, escogiendo el método de estimación paramétrica como un argumento de entrada de la función.

e) Rutina **ar**

```
>> [model]=ar(y,na,approach);
```

- approach = 'fb': Hacia adelante y hacia atrás *-forward-backward-*. Es el método por defecto. Se minimiza la suma del criterio de mínimos cuadrados hacia adelante y el criterio análogo hacia atrás en el tiempo. Equivale a **armcov**.

- approach = 'ls': Método de mínimos cuadrados -*least-squares approach*-. Se minimiza la suma de los cuadrados de los errores de predicción hacia delante. Equivale a **arcov**.
- approach = 'yw': Método de Yule-Walker. Se resuelven las ecuaciones de The Yule-Walker. Equivale a **aryule**.
- approach = 'burg': Método de Burg. Minimiza por mínimos cuadrados la media aritmética del error de predicción hacia delante y hacia atrás, satisfaciendo el recursivo de Levinson-Durbin. Equivale a **arburg**.
- approach = 'gl': Método como el de Burg, pero se usa la media geométrica.

4.1.2) Estimación de Modelos ARMA

Para modelos ARMA { $A(q)y(n) = C(q)e(n)$ } se utilizan en *Signal processing Toolbox* las funciones **prony** (Método de Prony) y **stmcb** (Método de Steigltz y McBride). Aunque la literatura de procesado de señales el modelado ARMA suele denotar: $A(z)y(n) = B(z)e(n)$, las técnicas de Prony y Steigltz-McBride asumen una entrada externa impulsional y arbitraria (que puede ser un impulso), respectivamente en cada técnica, utilizando una estructura ARX en el método de Prony y OE (*output error*) en Steigltz-McBride.

a) Método de Prony. Calcula un filtro de respuesta al impulso infinita (IIR) *B(q)/A(q)* cuya respuesta impulsional es la señal a modelar *y*. Se basa en una estructura ARX $A(q)y(n) = B(q)u(n) + e(n)$. Esta estructura se origina en un método propuesto por el Barón de Prony en 1795 para modelar la expansión de varios gases con la suma de exponeciales amortiguadas. Este método se actualizó para ajustar modelos con exponenciales amortiguadas o sinusoides puras a una señal [Par87],[The92] y es llamado también "enfoque ARMA indirecto de modelado de la señal" [The92]. En MATLAB:

```
>> [B,A] = prony(y,nb,na);
```

donde B y A son los vectores de parámetros de orden *nb* y *na*, respectivamente. Incorporamos nuestra función

```
>>  [B,A,e] = prony_e(y,nb,na);
```

que incluye en la salida a *e* como la varianza del error (σ_ε^2)

b) Método de Steiglitz-McBride. Calcula un filtro IIR *B(q)/A(q)* cuya respuesta impulsional es la señal a modelar *y* [Ste65]. Es llamado también "enfoque ARMA directo de modelado de la señal" [The92]. Opcionalmente, este método permite hallar el modelo de un sistema cuyas entradas y salidas son *u(n)* e *y(n)* respectivamente. Se basa en una estructura *output error* (OE) que se obtiene considerando una relación lineal entre la salida de un sistema no perturbado $y_{np}(n)$ y la entrada *u(n)*. La salida observada será la suma de $y_{np}(n)$ más el error:

$$A(q)y_{np}(n) = B(q)u(n) \qquad (4.19)$$

$$y(n) = y_{np}(n) + e(n) \qquad (4.20)$$

donde *A(q)* y *B(q)* están descritas por (4.9) y (4.10). Si se expresa la estructura como un modelo de función de transferencia, queda

$$y(n) = \frac{B(q)}{F(q)}u(n) + e(n) \quad \text{(donde } F(q)=A(q)\text{)} \qquad (4.21)$$

Esta estructura representa un caso especial del modelo lineal genérico de la ecuación (4.1) y (4.13), donde la secuencia no predecible $e(t)=v(t)$ representa la diferencia

(error) entre la salida observada y la no perturbada. Con la señal a modelar de tamaño *N* y modelos de orden *na* y *nb*, se pueden definir un error de predicción (ε) para cada valor de $n > p$, donde *p* es el valor mayor entre *na* y *nb*. La estimación de los parámetros se realiza optimizando la función de pérdida

$$V_{OE}(\theta) = \sum_{n=p+1}^{N} \varepsilon_{OE}^2(n) = \sum_{n=p+1}^{N} \left[y(n) - \frac{\hat{B}(q)}{\hat{A}(q)} u(n) \right]^2 \qquad (4.22)$$

Esta función de pérdida, al igual que la de la estructura ARX, utiliza el criterio de mínimos cuadrados; sin embargo, mientras en el modelo ARX hay una relación lineal entre la función de error y los parámetros a estimar, en la estructura OE la relación es no-lineal. Esto lleva a buscar soluciones no analíticas, como técnicas de optimización iterativas, para el problema de la minimización de $V_{OE}(\theta)$. Una de las técnicas de estimación paramétrica utilizada originalmente, con esta estructura, es la de prefiltrado iterativo de Steiglitz-McBride [Ste65]. Consiste en *(i)* estimar por mínimos cuadrados un modelo ARX: $A(q)\,y(n) = B(q)\,u(n) + e(n)$ con el método de Prony, *(ii)* utilizar los parámetros estimados de $\hat{A}(q)$ para filtrar las señales *y(n)* y *u(n)*, de la forma

$$y_F(n) = \frac{1}{\hat{A}(q)} y(n), \qquad u_F(n) = \frac{1}{\hat{A}(q)} u(n) \qquad (4.23)$$

y *(iii)* utilizar las señales filtradas, para estimar nuevamente *A(q)* y *B(q)* por mínimos cuadrados. El proceso se repite intentando minimizar el residuo, y los parámetros convergen a sus valores reales (mínimo global) si la perturbación es en realidad ruido blanco [Sod8] o si el modelo no es muy grande. Con *Signal Processing Toolbox* de MATLAB tenemos:

```
>> [B,A] = stmcb(y,nb,na);
>> [B,A] = stmcb(y,u,nb,na);  % Para identificación de sistemas de salida y, entrada u
```

Este algoritmo, por defecto, inicia la búsqueda de los parámetros con un modelo de Prony y realiza 5 iteraciones. Incluimos nuestra function

```
>> [B,A,e] = stmcb_e(y,nb,na);
```

que contiene a *e* en la salida como la varianza del ruido o error (σ_ε^2).

Los modelos ARMA en el *System Identification toolbox* se pueden hallar con los comandos **arx** y **oe** (*output error*), considerando que la entrada del sistema es un impulso unitario, y con la función **armax**.

c) La función **arx** produce el mismo resultado que **prony**, al ser la misma familia de modelo. Si en el comando **arx** se coloca como salida del sistema la señal a modelar *y*, y como entrada un impulso unitario, se puede obtener un modelo de sistema igual al obtenido por **prony**.

$$A(q)\,y(n) = B(q)\,u(n) + e(n) \qquad (4.24)$$

Para esto hay que usar un pre-enventanando de tantos ceros como sea el orden máximo, *p*, de *na* y *nb* en las señales *y* (a modelar) e impulso de entrada. El retardo *nk* se coloca en 0. Hay que hacer notar que el orden *nb* en **prony** significa el orden del polinomio *B(q)* (*nb*+1 parámetros), sin embargo en **arx**, la entrada **nb** significa el número de parámetros de *B(q)*, es decir, su orden es *nb*–1. La función **arx** utiliza el método de mínimos cuadrados. Si utilizamos **arx** con la señal *y* (sin pre-enveantando) como serie temporal, nos dará el mismo resultado que la función **ar** con el *approach* = 'ls' o **arcov** de *signal processing*.

En el modelo ARX se pueden predecir los valores de la salida $\hat{y}(n)$ como una regresión lineal (similar a la ecuació 4.17):

$$\hat{y}(n) = \varphi^T(n)\theta \tag{4.25}$$

donde $\theta = \left[a_1 \cdots a_{na} \;\; b_0 \; b_1 \cdots b_{nb}\right]^T$ (vector de los parámetros a estimar) y $\varphi^T(n) = \left[-y(n-1)\cdots -y(n-na)\, u(n)\, u(n-1)\cdots u(n-nb)\right]$. Con la señal *y* de tamaño *N* y modelos de orden *na* y *nb*, se pueden definir un error de predicción (ε) para cada muestra $n > p$, donde *p* es el valor mayor entre *na* y *nb*:

$$\varepsilon(n) = y(n) - \hat{y}(n) \tag{4.26}$$

Si agrupamos (4.25) y (4.26) en (4.5), los parámetros se hallan minimizando:

$$V_{ARX}(\theta) = \sum_{n=p+1}^{N} \varepsilon^2(n) = \sum_{n=p+1}^{N} \left[y(n) - \varphi^T(n)\theta\right]^2 \tag{4.27}$$

Como el segmento de señal de *N* muestras incluye un pre-enventanado de *p* ceros, la sumatoria en la ecuación $V_{ARX}(\theta)$ comienza en la primera muestra de la señal *y* a modelar. La solución se presenta más sencillamente en forma matricial:

$$V_{ARX}(\theta) = (\mathbf{y} - \Phi\theta)^T(\mathbf{y} - \Phi\theta) \tag{4.28}$$

donde

$$\mathbf{y} = \begin{bmatrix} y(p+1) & y(p+2) & \cdots & y(N) \end{bmatrix} \tag{4.29}$$

y Φ la matriz de los regresión (o de datos) dada por:

$$\begin{aligned} \Phi &= \begin{bmatrix} \phi^T(p+1) \\ \phi^T(p+2) \\ \vdots \\ \phi^T(N) \end{bmatrix} \\ &= \begin{bmatrix} -y(p) & \cdots & -y(p-na+1) & u(p+1) & \cdots & u(p-nb+1) \\ -y(p+1) & \cdots & -y(p-na+2) & u(p+2) & \cdots & u(p-nb+2) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -y(N+1) & \cdots & -y(N-na) & u(N) & \cdots & u(N-nb) \end{bmatrix} \end{aligned} \tag{4.30}$$

Para minimizar la función de pérdida descrita por (4.28), se expande:

$$V_{ARX}(\theta) = \mathbf{y}^T\mathbf{y} - \theta^T\Phi^T\mathbf{y} - \mathbf{y}^T\Phi\theta + \theta^T\Phi^T\Phi\theta \tag{4.31}$$

e igualando a cero su gradiente, se obtiene:

$$\left.\frac{\partial V}{\partial \theta}\right|_{\theta=\hat{\theta}_N} = -2\Phi^T\mathbf{y} + 2\Phi^T\Phi\hat{\theta}_N = 0 \tag{4.32}$$

los parámetros se estiman como:

$$\hat{\theta}_N = (\Phi^T\Phi)^{-1}\Phi^T\mathbf{y} \tag{4.33}$$

si la Matriz $\Phi^T\Phi$ es no-singular (Ecuación de mínimos cuadrados de Wiener-Hopf).

El modelado ARMA con la función **arx** la implementamos con nuestra función

```
>> [B,A,e]=arx_arma(y,nb,na);
```

obteniendo los mismos vectores de parámetros B y A que con la función **prony**. *na* y *nb* siguen siendo los órdens de *A(q)* y *B(q)* respectivamente.

d) La función OE,

$$y(n) = \frac{B(q)}{F(q)}u(n) + e(n) \qquad \text{(donde } F(q)=A(q)\text{)} \tag{4.34}$$

obtiene un modelo ARMA al ser utilizado de la misma forma que **arx**, es decir, con la señal a modelar *y* como respuesta al impulso, y pre-enventanando de tantos ceros *p* como sea el orden máximo de *na* y *nb*. Los valores de los coeficientes de *B(q)* y *A(q)* son diferentes a los obtenidos por **stmcb**, ya que usan distintos algoritmos iterativos. La rutina **oe** utiliza el método de Gauss-Newton [Den83],[Lju87] para otimizar la función $V_{OE}(\theta)$ de la ecuación (4.22). En forma sucinta, el método de Gauss-Newton consiste en buscar los parámetros $\hat{\theta}_N$ que minimicen una función de criterio iterativamente, de acuerdo con

$$\hat{\theta}_N^{(i+1)} = \hat{\theta}_N^{(i)} + \mu g n^{(i)} \tag{4.35}$$

donde $\hat{\theta}_N^{(i)}$ son los parámetros estimados en la iteración número *i*; μ es una constante positiva, con el fin de hallar una reducción adecuada de $V_{OE}(\theta)$ (ecuación (4.22)) y $gn^{(i)}$ es la dirección de búsqueda de *Gauss-Newton*, basada en la función $V_{OE}(\theta)$ obtenida de las iteraciones previas:

$$gn^{(i)} = \left[V''_{OE}(\hat{\theta}_N^{(i)})\right]^{-1} V'_{OE}(\hat{\theta}_N^{(i)}) \tag{4.36}$$

donde el gradiente de $V_{OE}(\theta)$ es

$$V'_{OE}(\hat{\theta}_N^{(i)}) = \sum_{n=p+1}^{N} \psi(n,\theta)\varepsilon_{OE}(n,\theta) \tag{4.37}$$

siendo $\psi(n,\theta)$ el gradiente del error de salida ε_{OE}. El método de Gauss-Newton aproxima $V''_{OE}(\hat{\theta}_N^{(i)})$, en la vecindad del mínimo, a

$$V''_{OE}(\hat{\theta}_N^{(i)}) = \sum_{n=p+1}^{N} \psi(n,\theta)\psi^T(n,\theta) \tag{4.38}$$

Nuestra función

```
>> [B,A,e]=oe_arma(y,nb,na);
```

obtiene un modelo ARMA utilizando la función **oe**.

e) La función **armax** calcula los parámetros del modelo $A(q)y(n) = C(q)e(n)$, cuando trabaja con una sola señal de entrada como serie temporal. Para el cálculo se utiliza un algoritmo iterativo de Gauss-Newton [Den83],[Lju87], como el descrito previamente para la función **oe**, para la minimización de la función de error. Sin embargo, a diferencia de las funciones **prony**, **stmcb,** **arx_arma** y **oe_arma** donde $B(q) = b_0 + b_1 q^{-1} + \cdots + b_{nb} q^{-nb}$, la función **armax** para series temporales utiliza el vector de parámetros (nótese que $c_0 = 1$)

$$C(q) = 1 + c_1 q^{-1} + \cdots + c_{nc} q^{-nc} \tag{4.39}$$

Con nuestra rutina

```
>>  [C,A,e]=armax_arma(y,nc,na);
```

obtenemos un modelo ARMA a partir de la función **armax** de *System Identification*.

f) ARMA con el método de Durbin-Broersen de ARMASA [BroMat]. Broersen [Bro00a], [Bro02] perfeccionó el método de Durbin [Dur60] para poder trabajar con órdenes de modelo suficientemente grandes. El método de Durbin, mejorado por Broersen, consiste en el uso de los errores estimados $\hat{\varepsilon}(n)$ y de las observaciones previas de *y(n)* como regresores en un ajuste por mínimos cuadrados. En el modelo ARMA

$$A(q)y(n) = C(q)e(n) \tag{4.40}$$

los residuos $\hat{\varepsilon}(n)$ son reconstruidos utilizando un modelo AR muy largo: $A^{largo}(q)$:

$$A^{largo}(q)y(n) = \hat{\varepsilon}(n) \tag{4.41}$$

El modelo de la ecuación (4.41) se puede estimar para un orden de $A^{largo}(q)$ que tienda a infinito cuando sus parámetros coinciden con los de la expansión en serie de *A(q)/C(q)*. En el método de Durbin-Broersen, se estiman una primera vez los parámetros $\hat{A}(q)$ y $\hat{C}(q)$, luego se usan los parámetros estimados $\hat{A}(q)$ como condición inicial para actualizar iterativamente $\hat{A}(q)$ y $\hat{C}(q)$ con el un nuevo método de Durbin adaptado por Broersen. Si sustituimos *y(n)* de la ecuación (4.40) en (4.41) y reemplazamos los valores reales de *e(n)* por los estimados, tenemos

$$\frac{A^{largo}(q)}{\hat{A}(q)}\hat{C}(q) \approx 1 \tag{4.42}$$

Para estimar $A^{largo}(q)$, Broersen [Bro00a],[Bro02] utiliza un orden 3 veces mayor que para *A(q)* más el número de parámetros que tienen que estimarse. *C(q)* se actualiza del cociente $\hat{A}(q)/A^{largo}(q)$ dado en (4.42). Luego $\hat{A}(q)$ se estima de nuevo con la expresión $\hat{A}(q) \approx A^{largo}(q)\,\hat{C}(q)$. La estimación final se busca de los parámetros $\hat{A}(q)$ que minimicen la diferencia entre los dos modelos estimados dados por $[\hat{C}(q)/\hat{A}(q),\ 1/A^{largo}(q)]$. En el toolbox ARMASA [BroMat] la estimación se realiza con la rutina

```
>> [A,B, sel] = sig2arma(y,na);
```

donde *y* es la señal a modelar, *na* es el orden de *A(q)*, *nb = na-1* es el orden de *C(q)*, y *sel* contiene información del modelo incluyendo la varianza del error (σ_ε^2).

4.2) Búsqueda del "mejor" orden de modelo

La escogencia del orden del modelo es uno de los aspectos más importante del modelado paramétrico. El ajuste del modelo a la señal o al sistema va mejorando (disminución del error) a medida que se incrementa el orden del modelo. Se busca, entonces, escoger el "mejor" orden de modelo a partir de ciertos criterios que penalicen el incremento del orden si el error no disminuye significativamente. En otros casos, si tuviéramos un sistema conocido o verdadero que se ajusta con un modelo de orden "verdadero", se busca que el orden no sub-ajuste o subestime ni sobre-ajuste o sobreestime este sistema o señal. Mencionaremos en primer lugar los métodos clásicos y luego las últimas técnicas desarrollas para la selección de los "mejores" órdenes de modelo. Comenzaremos por los modelos AR.

4.2.1) Búsqueda del "mejor" modelo con estructura AR

La varianza del error, σ_e^2, se denomina varianza del residuo, si el error es el hallado de la señal o serie temporal usada para estimar los parámetros del modelo. En una estructura AR de orden *na*, la varianza del residuo está dada por:

$$RES(na) = \sigma_e^2 = \frac{1}{N-na}\sum_{n=na+1}^{N}\left[x(n)-\hat{x}(n)\right]^2 = \frac{1}{N-na}\sum_{n=na+1}^{N}\left[x(n)+\sum_{i=1}^{na}a_i x(n-i)\right]^2 \tag{4.43}$$

donde *N* el número de muestras de la señal y $\hat{x}(n)$ es la señal estimada con el modelo.

Por otro lado, la varianza del error, σ_e^2, se denomina error de predicción (PE), si el error es el hallado de una señal distinta a la usada para estimar los parámetros; esto es, de otra realización del proceso estocástico o de un segmento de señal distinto si el proceso es ergódico (señal $x_2(n)$ de *M* muestras). La varianza del error de predicción en este caso es $\sigma_e^2 = PE(na)$:

$$PE(na) = \sigma_e^2 = \frac{1}{M}\sum_{n=n_0+1}^{n_0+M}\left[x_2(n)+\sum_{i=1}^{na}a_i x_2(n-i)\right]^2 \tag{4.44}$$

Una de las primeras técnicas utilizada para la búsqueda del "mejor" orden es el Error de Predicción Final de Akaike –*Akaike's Final Prediction Error*- (FPE) [Aka70]. Considerando que el orden "verdadero" del modelo es *K*, y la varianza del error de ese modelo es σ_{ev}^2, Akaike usó los resultados asintóticos de la división de los valores esperados de la varianza del residuo (*RES(na)*) y del error de predicción (*PE(na)*)

$$E\left[PE(na)\right] = \sigma_{ev}^2(1+na/N)\ \text{, para } na \geq K \tag{4.45}$$

$$E\left[RES(na)\right] = \sigma_{ev}^2(1-na/N)\text{, para } na \geq K \tag{4.46}$$

deduciendo el criterio FPE, como

$$FPE = \sigma_e^2\frac{1+na/N}{1-na/N} \tag{4.47}$$

Otro criterio de Akaike similar es el Criterio de Información Teórica de Akaike – *Akaike's Information Theoretic Criterion*- (AIC) [Aka74].

$$AIC = \ln\left(\sigma_e^2\right) + 2\frac{na}{N} \tag{4.48}$$

De acuerdo con estos criterios de Akaike, se escoje el modelo con el menor valor de FPE o AIC. Otro criterio ampliamente usado es el de Rissanen -*Rissanen's Minimum Description Length*- (MDL) [Ris78], el cual selecciona la estructura que permita la descripción más compacta de la data observada. Este modelo penaliza un mayor número de muestras de la señal relativas al orden del modelo y puede expresarse como

$$MDL = \ln\left(\sigma_e^2\right) + \frac{na}{N}\ln(N) \tag{4.49}$$

Estos criterios pueden escribirse como un criterio de información generalizado (GIC) [Bro93] como

$$GIC(na,\alpha) = \ln\sigma_e^2 + \alpha\frac{na}{N} \tag{4.50}$$

donde $\alpha = 2$ (AIC)
$\alpha = \ln(N)$ (MDL)
$\alpha = 2\ln\ln(N)$ Criterio de consistencia mínima [Han79]

Broersen [Bro93] introdujo expresiones que aproximan la varianza de los coeficientes de reflexión (definidos como el último parámetro a_{na} del vector de coeficientes θ) de los métodos

de Yule-Walker [Kay81], de Burg [Bur67] y de mínimos cuadrados, a través de coeficientes de varianza de muestras finitas de modelos AR:

$$v(i,met)$$

para órdenes de modelo i mayores o iguales al orden "verdadero" K y el método de estimación *met*. Si consideramos que σ_{ev}^2 es la varianza del orden "verdadero" K, entonces para señales de N muestras y los métodos de estimación paramétrica dados, tenemos las siguientes aproximaciones de *v(i,met)* [Bro93],[Bro00b]:

$$v(i,YW) = (N-i)/\big(N(N+2)\big) \tag{4.51}$$

$$v(i,Burg) = 1/(N+1-i) \tag{4.52}$$

$$v(i,lsfb) = 1/(N+1.5-1.5i) \tag{4.53}$$

$$v(i,lsf) = 1/(N+2-2i) \tag{4.54}$$

donde *YW*: método de Yule-Walker (MATLAB: **aryule** y **ar** (*approach*='yw'))
Burg: Método de Burg (MATLAB: **arburg** y **ar**(*approach*='bg'))
lsfb: Método de mínimos cuadrados (*ls*) hacia adelante (*f*) y hacia atrás (*b*) o de covarianza modificado (MATLAB: **armcov** y **ar** (*approach*='fb')
lsf: Método de mínimos cuadrados (*ls*) hacia adelante (*f*) o de la covarianza (MATLAB: **arcov** y **ar**(*approach*='ls'))

Cuando el orden del modelo *na* es igual o mayor al orden del modelo "verdadero" K, el valor esperado de la varianza del residuo *RES(na)* y del error de predicción *PE(na)* se pueden aproximar a [Bro93]:

$$E\big[RES(na)\big] \approx \sigma_{ev}^2 \prod_{i=1}^{na} (1 - v(i,met)) \text{ , para } na \geq K \tag{4.55}$$

$$E\big[PE(na)\big] \approx \sigma_{ev}^2 \prod_{i=1}^{na} (1 + v(i,met)) \text{ , para } na \geq K \tag{4.56}$$

Tomando el logaritmo a *RES(na)*, se puede expresar un criterio equivalente al GIC, llamado criterio de información de muestra finita de Broersen [Bro93]

$$FIC(na,\alpha) = \ln(\sigma_e^2) + \alpha \sum_{i=1}^{na} v(i,met) \tag{4.57}$$

Un buen compromiso entre la sobreestimación y subestimación de los parámetros está dado por un factor de penalización $\alpha = 3$.

El cociente de los valores esperados del error de predicción y de la varianza del residuo fue usado [Bro00b] para hallar el criterio de información de muestra finita (FISC) que selecciona el orden del modelo:

$$FSIC(na) = \ln(\sigma_e^2) + \prod_{i=0}^{na} \frac{1+v(i,met)}{1-v(i,met)} - 1 \tag{4.58}$$

De estos dos últimos criterios, se derivó el criterio de información combinada (CIC) de Broersen [Bro00b]. Este criterio toma en cuenta el balance entre sobre-estimación y sub-estimación del criterio FIC con una penalización α=3, para órdenes bajos, y el comportamiento del criterio FSIC para órdenes altos, cuando el tamaño de la señal lo requiera. Así el criterio *CIC(na)* se expresa

$$CIC(na) = \ln(\sigma_e^2) + \max\left[\prod_{i=0}^{na} \frac{1+v(i,met)}{1-v(i,met)} - 1, 3\sum_{i=1}^{na} v(i,met) \right] \tag{4.59}$$

Se selecciona el orden *na* con el menor valor CIC de los órdenes probados. Además del tamaño *N* de la señal y del orden aprobar, el criterio CIC depende del método de estimación paramétrica utilizado. Por ejemplo, si tomamos un modelo AR, estimado con el método de Burg, el criterio CIC quedaría

$$CIC(na) = \ln(\sigma_e^2) + \max\left[\prod_{i=0}^{na}\frac{1+\frac{1}{N+1-i}}{1-\frac{1}{N+1-i}} - 1, 3\sum_{i=1}^{na}\frac{1}{N+1-i}\right] \tag{4.60}$$

En la práctica se desea un modelo con un menor error de predicción, PE(na). Los errores de predicción de los modelos estimados pueden ser hallados con la señal dada a través de [Bro02]

$$PE(na) = \sigma_e^2 \prod_{i=1}^{na}\frac{1+v(i,met)}{1-v(i,met)} \tag{4.61}$$

Por ejemplo, para el método de Burg (utilizando ecuación 4.52):

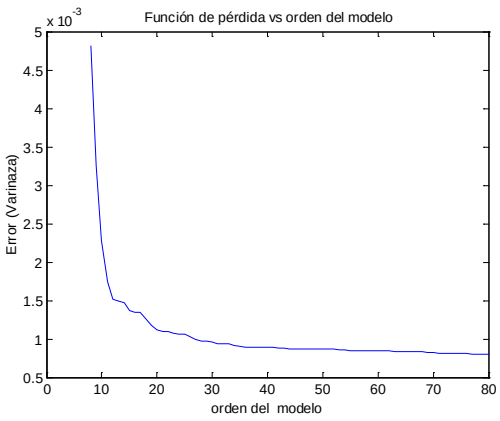
$$PE(na) = \sigma_e^2 \prod_{i=1}^{na}\frac{1+\frac{1}{N+1-i}}{1-\frac{1}{N+1-i}} \tag{4.62}$$

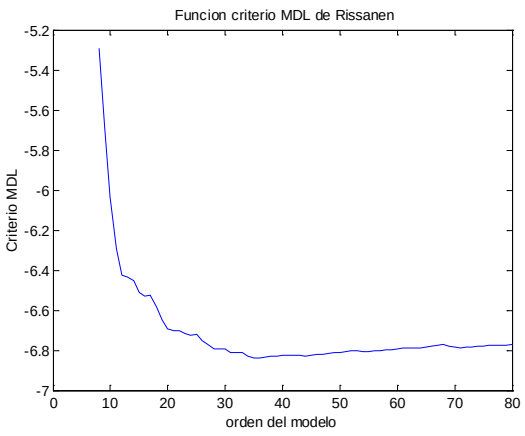
- Uso de MATLAB para hallar el "mejor" orden de modelo AR

En el *System Identification toolbox* existen las rutinas **arxstruc** y **selstruc** que utilizan los criterio AIC y MDL. **arxstruc** usa el comando **arx**, por lo tanto, utiliza el método de mínimos cuadrados (*ls*) para el cálculo del modelo.

Hallemos el "mejor" modelo para estructura AR de la señal del ejemplo 1:

```
>>V=arxstruc(x1n,x1n,(8 :80)') ;
  % V: La primera fila contiene las varianzas del error de las estructuras dadas, presentadas en
  % las filas siguientes. La última columna es el número de datos de la señal.
>>[na,Vn]=selstruc(V, 'aic'); % Criterio AIC en el vector Vn
nn=78
>> [na,Vn]=selstruc(V, 'mdl');  % Criterio MDL en el vector Vn
na=35
   >> plot(V(2,1:73),V(1,1:73))          >> plot(Vn(2,:),Vn(1,:))
```





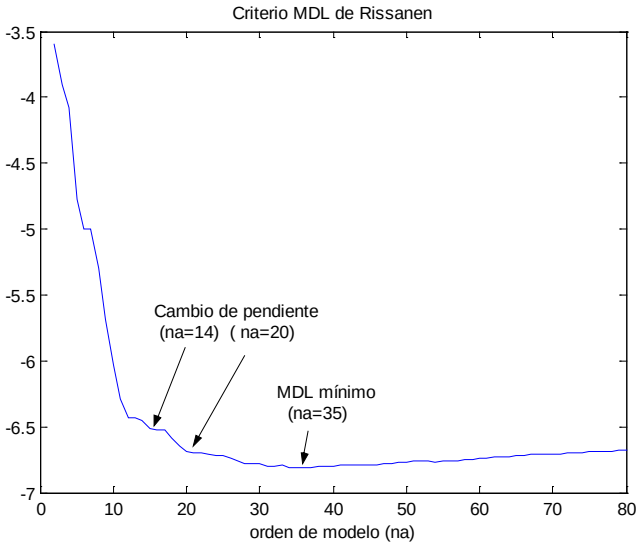
En la práctica, es recomendable graficar la curva de la función de criterio vs. la dimensión de parámetros del modelo. Observando el codo de la curva de la función de pérdida o la curva de la función criterio MDL se aprecia que el orden 35 es el mínimo dado por el criterio MDL, mientras que el criterio AIC propone un orden *na*=78.

Para modelos AR del *toolbox* de *Signal Processing*, utilizamos la rutina **selarstruc** (de nuestras funciones) para calcular el "mejor" orden del modelo AR dado por el método escogido, para un conjunto de ordenes de modelo. La función **selarstruc** permite escoger el método de estimación paramétrica del modelo AR: *met* = 'arburg', 'arcov', 'armcov' o 'aryule', y el *criterio* de búsqueda del mejor orden de modelo:

criterio 'aic': Criterio de información de Akaike
'mdl': Rissanen's *minimum description length*
'fpe': Error de predicción final de Akaike (valor por defecto)
'gic3': Criterio de informacion generalizado de Broersen con alfa=3
'cic': Criterio de informacion combinada de Broersen

Si utilizamos el criterio MDL para la señal *x1n*, tenemos

```
>> [na,V,Vn]=selarstruc(x1n,'arcov',8:80, 'mdl');
        % na es el orden que produce el valor mínimo del criterio
>> plot(V(2,:),V(1,:),Vn(2,:),Vn(1,.))   % na = 35
```



El "mejor" orden según el criterio MDL de Risanen es *na* = 35. Sin embargo la curva de la función de pérdida o error vs. el orden del modelo presenta cambios de pendiente que tienden a una reducción suave del error para *na* = 14 y también para *na* = 20. Estos valores pueden ser tomados en la práctica, si se quiere trabajar con modelos de estructuras reducidas.

Una excelente referencia reciente sobre métodos de escogencia del "mejor" orden de modelo de estructuras ARMA se encuentra en el trabajo de Broersen [Bro02] y en su toolbox de MATLAB ARMASA [BroMat], donde se propone calcular automáticamente el mejor modelo de estructura AR, MA y ARMA.

En el toolbox ARMASA, el mejor orden de modelo, así como, los parámetros estimados para ese orden, la varianza del error de todos los órdenes probados y los respectivos valores CIC, se busca automáticamente con:

```
>> [A,sel] = sig2ar(x1n);
>> sig_e = sel.pe_est;  % varianza del error para cada na probado
```

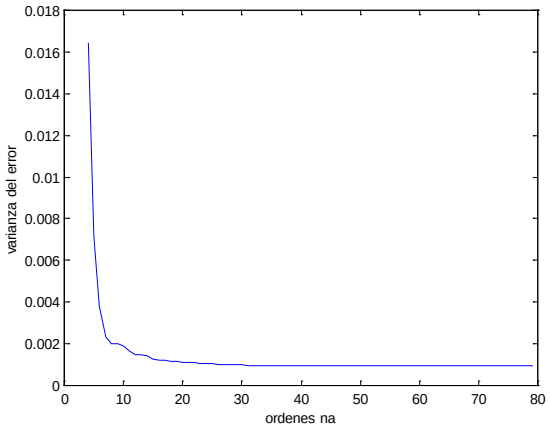
```
>> cic = sel.cic;  % criterio CIC para cada na probado
>> na_candidato = sel.cand_order;  % valores na probados
>> na = length(A) -1 % orden del modelo escogido, es decir, con length(A) parámetros
   na =  40
```

En ARMASA, por omisión, se remueve el valor medio de la señal a modelar. Si se quiere dejar la señal original se ejecuta:

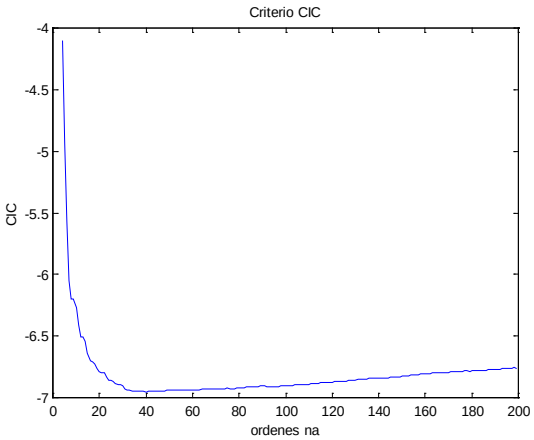
```
>> ASAglob_subtr_mean = 0;
```

El orden de A es *na*=40 con el criterio CIC. La función **sig2ar** utiliza el método de Burg [Bur67] para estimar el modelo AR. Equivale a **arburg** y **ar** con el *approach* = 'burg'. En la variable estructurada *sel* está la información de la varianza del error y del valor CIC para cada orden de modelo probado, que fue de 630 órdenes en este caso.

```
>> plot(na_candidato(5:80),sig_e(5:80))
```




```
>> plot(na_candidato(5:200),cic(5:200))
```



Este mismo orden de modelo (na=40) se encontró con la función **selarstruc**, con el método 'arburg' y el criterio 'cic', con la señal x1n removiéndole su valor medio: x1n – **mean**(x1n).

4.2.2) Búsqueda del "mejor" modelo ARMA

Para modelos ARMA y otras estructuras de modelado de señales y sistemas, los criterios FPE, AIC, MDL y GIC pueden expresarse utilizando la dimensión del vector de parámetros *d* en lugar del factor *na*, del modelado AR [Lju87]. Por ejemplo, el criterio GIC quedaría:

$$GIC(na,\alpha) = \ln \sigma_e^2 + \alpha \frac{d}{N} \tag{4.63}$$

El criterio FPE de Akaike [Aka70] para modelos ARMA se puede expresar:

$$FPE = \sigma_e^2 \frac{1 + d/N}{1d/N} \tag{4.64}$$

El criterio propuesto por Broersen [Bro00b] para escoger el orden en los modelos ARMA se basa en métodos de estimación de parámetros de Durbin [Dub60] adaptado descrito previamente, utilizando el criterio GIC con α=3. Para modelos ARMA de órdenes (*na*,*nb*), es decir de dimensión *d*=*na*+*nb* el orden estará dado por el valor mínimo del criterio

$$GIC(d,3) = \ln(\sigma_e^2) + 3\frac{d}{N} \tag{4.65}$$

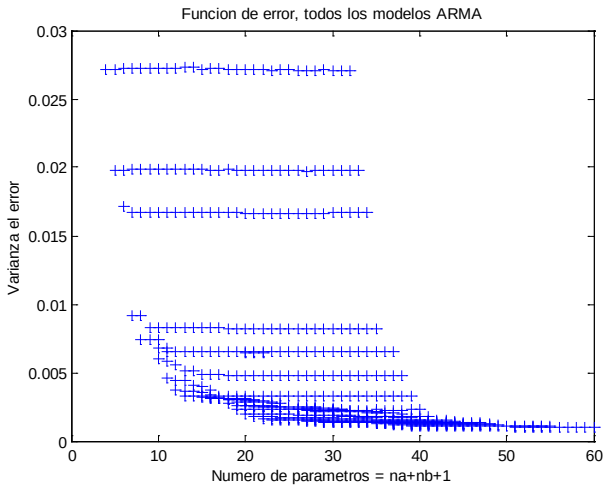
Los errores de predicción de los modelos estimados pueden ser hallados con la señal dada a través de [Bro02]

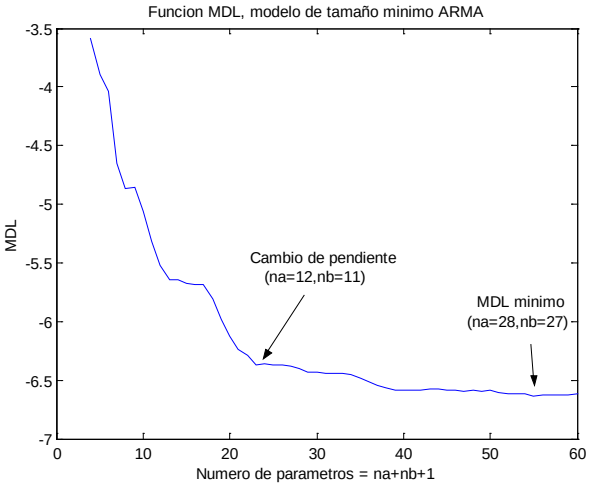
$$PE(d) = \sigma_e^2 \frac{1 + d/N}{1 - d/N} \tag{4.66}$$

- Uso de MATLAB para hallar el "mejor" orden de modelo ARMA

La búsqueda del mejor modelo con una estructura ARMA la realizamos con nuestra función **selarmastruc** y graficamos luego la varianza del error o el Criterio con los valores de la función **min_arma**. Para ello utilizamos nuestras funciones **prony_e**, **stmcb_e**, **arx_arma**, **oe_arma y armax_arma** como métodos para estimar los parámetros de la estructura ARMA. Y, al igual que en la estructura AR, los criterios FPE, AIC, MDL y GIC con $\alpha = 3$ (GIC(d,3)).

```
>>[nn,V,Vn]=selarmastruc(x1n,'arx_arma',2:30,2:30,'mdl');
              % Prueba todas las combinaciones de órdenes de
              % modelo desde na = 2 hasta 30 y nb = 2 hasta 30
>>[Vmin,Vnmin]=min_arma(V,Vn);
>> plot(Vn(4,:),Vn(1,:),'+') % incluye todos los modelos probados
                                        >> plot(Vmin(4,:),Vmin(1,:))
```

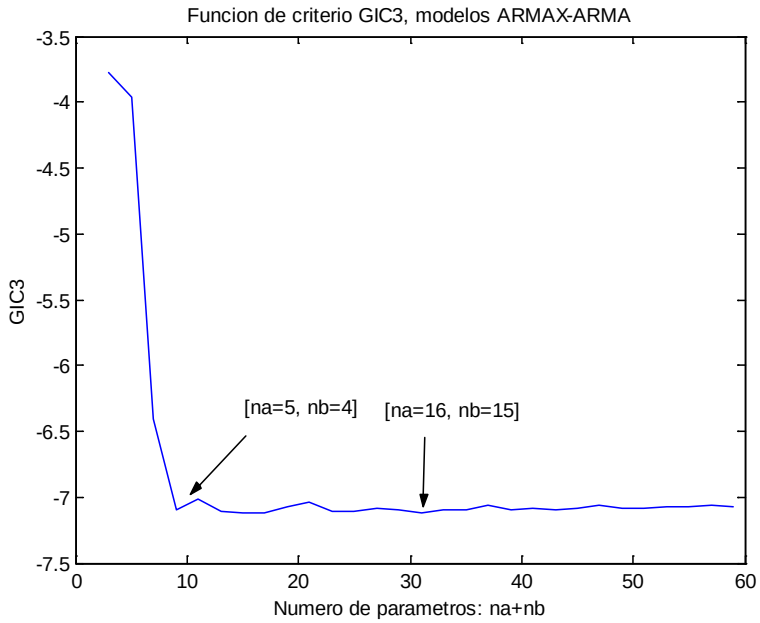




Según estos resultados, un modelo óptimo de acuerdo al criterio MDL puede ser un orden na=28 y nb=27. Sin embargo, los cambios de pendiente de la función de error o pérdida proponen tamaños de estructura de 13 o 23 parámetros, que corresponde a órdenes de modelo [na=11, nb=2] y [na=12, nb=11], respectivamente.

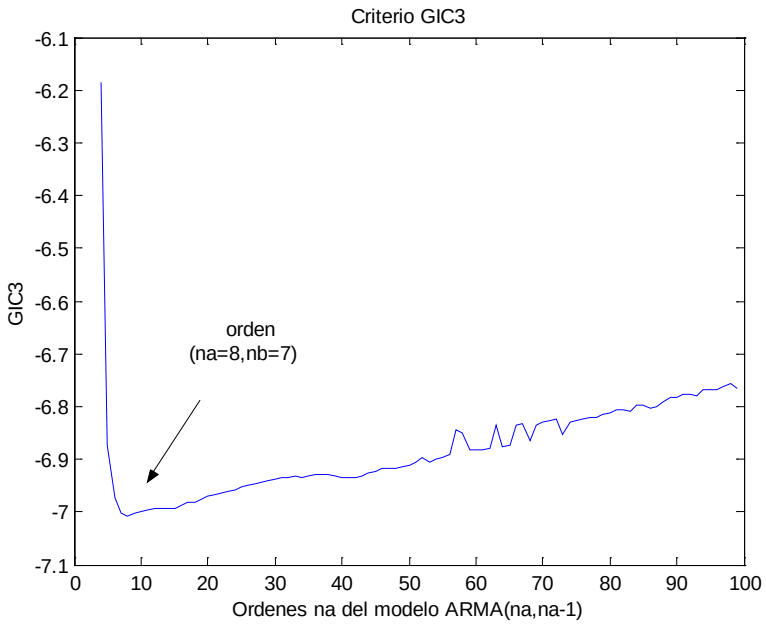
Una versión simplificada de la función **selarmastruc** es **selarmastruc2**, la cual solamente prueba órdenes de modelo [*na*, *nb=na-1*].

```
>> [nn,V,Vn]=selarmastruc2(x1n,'arx_arma',2:30,'mdl'); % nn = [na=28, nb=27]
>> [nn,V,Vn]=selarmastruc2(x1n-mean(x1n),'armax_arma',2:30,'gic3');
          % nn = [na=16, nb=15]
          % Se le ha removido el valor medio a la señal x1n
>> plot(V(4,:),V(1,:))
```



Utilizando el toolbox ARMASA, el orden de modelo ARMA y sus parámetros estimados se hallan:

```
>> [A,B,sel]=sig2arma(x1n);    % por defecto se remueve el valor medio de la señal x1n
>>sig_e=sel.pe_est;
>>gic3=sel.gic3;
>>na_candidato=sel.cand_ar_order;
>> na=length(A) -1
    na =   8
>>nb=length(B) -1
    nb =   7
>> plot(na_candidato(5:100),gic3(5:100))
```



4.3) Resolución espectral con modelos paramétricos de señales

Las técnicas de estimación espectral que utilizan directamente una señal con su transformada de Fourier presentan una resolución espectral, Δf_n, acotada a un valor "fundamental" $\Delta f_n = 1/T$, donde $T=T_s*N$ es la longitud del segmento de señal equivalente al periodo de muestreo T_s por el número de muestras N. Se pueden lograr mejores resoluciones que este valor fundamental con el modelado paramétrico de las señales porque extrapolan "efectivamente" la señal analizada más allá de su intervalo original T [Mar77]. La PSD obtenida con modelos AR (ec. 4.68) equivale al espectro obtenido por una secuencia de autocorrelación extrapolada y amortiguada hasta el infinito [Kay81, Mar87]. Tiempos "efectivos", T_e, más largos producen una menor Δf_n y una mejor resolución espectral. Por eso, a estas técnicas paramétricas se les llama también de "alta-resolución" espectral [Mar87].

En el caso de modelado AR el nivel del incremento de la resolución espectral depende de la relación señal-ruido y del orden *na* del modelo. Una buena aproximación empírica de resolución espectral con modelado AR (método de Burg) se puede expresar por [Mar82], [Mar87]

$$\Delta f_n = \frac{1.03}{T_s N\left(SNR(na+1)\right)^{0.31}} \tag{4.67}$$

donde na es el orden del modelo AR, T_s*N es el periodo de muestreo por el número de puntos (segmento de señal, T) y SNR es la relacion señal-ruido expresada en unidades lineales (Watts) en lugar de dB. Un orden de modelo lo suficientemente alto mejorará la resolución espectral pero un orden excesivo produciría picos espurios en el espectro frecuencial [Kay81].

4.4) Densidad de Potencia Espectral (PSD) con modelado paramétrico

La densidad de potencia espectral, PSD, de la señal se halla de la respuesta frecuencial de estos sistemas lineales estimados. Una vez hallados los parámetros de los modelos AR que ajusten la señal a modelar, se puede estimar la PSD a través de la expresión

$$P_{AR}(q) = \left.\frac{\sigma_e^{\,2}}{|A(q)|^2}\right|_{q=e^{j\omega}} = \frac{\sigma_e^{\,2}}{\left|1+\sum_{k=1}^{na} a_k e^{-j2\pi kf}\right|^2} \tag{4.68}$$

Si consideramos que la frecuencia de muestreo es $f_s=1/T_s$ y el retardo k corresponde a un retardo $kT_s=k/f_s$. la PSD se expresa

$$P_{AR}(f) = \frac{T_s\sigma_e^{\,2}}{\left|1+\sum_{k=1}^{na} a_k e^{-j2\pi kf/f_s}\right|^2} \tag{4.69}$$

Del mismo modo, la PSD estimada de una señal con modelado ARMA es

$$P_{ARMA}(q) = \sigma_e^{\,2}\left.\frac{|B(q)|^2}{|A(q)|^2}\right|_{q=e^{j\omega}} \rightarrow P_{ARMA}(f) = T_s\sigma_e^{\,2}\,\frac{\left|\sum_{k=0}^{nb} b_k e^{-j2\pi kf/f_s}\right|^2}{\left|1+\sum_{k=1}^{na} a_k e^{-j2\pi kf/f_s}\right|^2} \tag{4.70}$$

En el *toolbox* de *Signal Processing* se pueden hallar directamente la PSD, utilizando los métodos de estimación paramétrica de Burg, Yule-Walker, Covarianza y Covarianza modificado con los comandos **pburg**, **pyulear**, **pcov** y **pmcov** respectivamente.

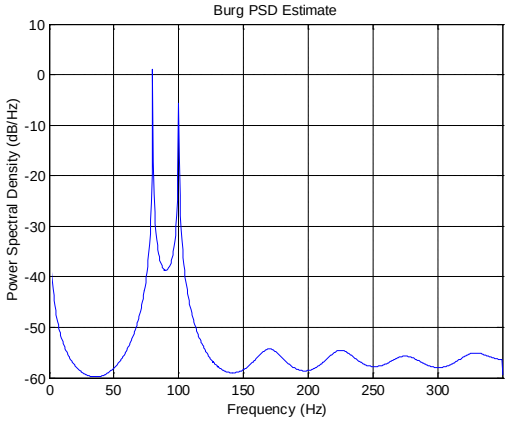
4.4.1) Densidad de Potencia Espectral con modelos AR

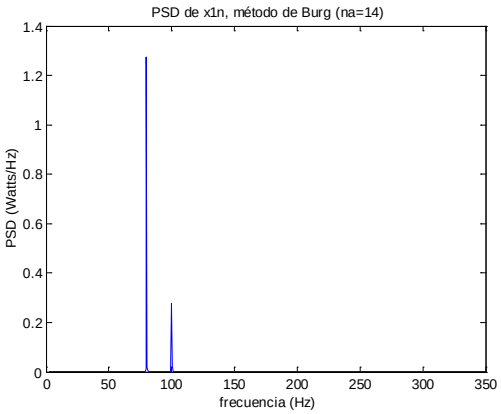
Vamos a estimar la PSD de la señal del ejemplo 1 con modelos AR de orden 14 (el "codo" e la fución de error) y 40 ("mejor" orden según criterio CIC) utilizando varios métodos de estimación paramétrica:

a) Método de Burg.

Señal 1 con un orden *na* =14:

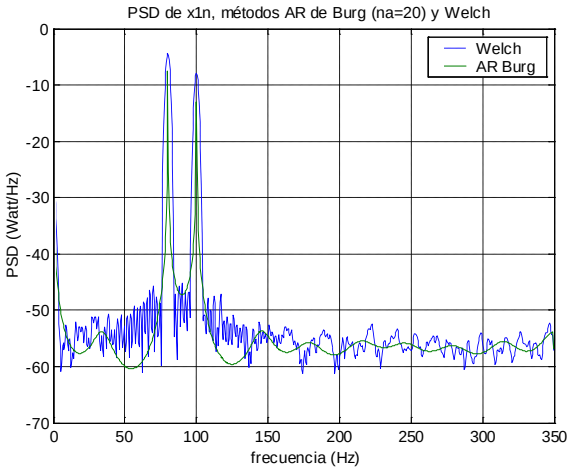
```
>> N=length(x1n); N=1401
>> pburg(x1n,14,N,fs) % equivale a
>> [Pb,fb]=pburg(x1n,14,N,fs);
>>plot(fb, 10*log10(Pb))                    >> plot(fb,Pb)
```





Utilicemos el orden *na* = 20 y se compara el PSD por el método de Welch:

```
>> [Pb,fb]=pburg(x1n,20,N,fs);
>> [P1,f1]=pwelch(x1n,[],[],N,fs);
>> plot(f1,10*log10(P1),':',fb,10*log10(Pb)), grid
>> legend('Welch','AR Burg')
```



A continuación vamos a realizar el modelado de la señal *x1n(n)* con el método de Burg y *na*=14 y hallar paso a paso la PSD estimada:

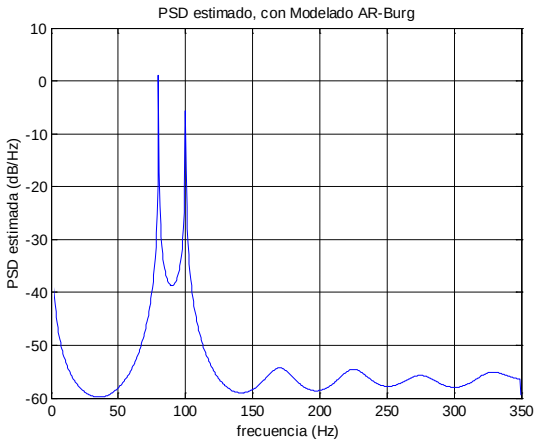
```
>> [Ab,eb]=arburg(x1n,14);
```

Ab= [1.0000 -0.4590 0.0582 0.3916 0.3042 0.0852 -0.2077 -0.2041 -0.2016
-0.1170 -0.1128 -0.1874 -0.3006 -0.1674 0.1470]

eb = 0.001466968 % varianza estimada del ruido blanco (Final Prediction Error)

```
>> [H,f]=freqz(1,Ab,N,'whole',fs);
>> Pxb=eb*(abs(H).^2)/fs;   PSD (de ambos lados del espectro)
>>if rem(N,2), % se evalua la paridad de N
>>      select = 1:(N+1)/2; % si N es impar
```

```
>>else
>>      select = 1:N/2+1; % si N es par
>> end
>>  % select en este caso es de 701 elementos
>> Pxb_unlado=Pxb(select);
>> Pxb_u=[Pxb_unlado(1); 2*Pxb_unlado(2:end-1);Pxb_unlado(end)];
>> fb=f(select);
>> plot(fb,10*log10(Pxb_u))
```

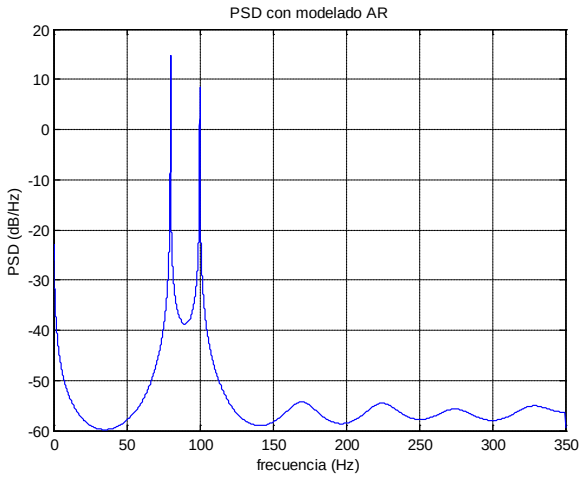


La función **armaspectra**, realiza los pasos anteriores para calcular y, opcionalmente, graficar la PSD del modelo AR hallado:

```
>> armaspectra(1,Ab,eb,N,fs);  % grafica la misma PSD anterior
>> [Px,f]=  armaspectra(1,Ab,eb,N,fs); % devuelve los valores PSD en Px y frecuencia en f
```

Para mejorar la resolución para el cálculo de la respuesta frecuencia del modelo paramétrico, se puede usar un vector de frecuencias entre 0 y fs, correspondientes al círculo unitario sobre el plano 'z'. Con **armaspectra**, se introduce el vector de frecuencias en lugar de N. Por ejemplo:

```
>> [Pxx,f]=armaspectra(1,Ab,eb,0:0.01:fs,fs);
>> plot(f,10*log10(Pxx)),grid,
>> xlabel('frecuencia (Hz)'),ylabel ('PSD (dB/Hz)')
```



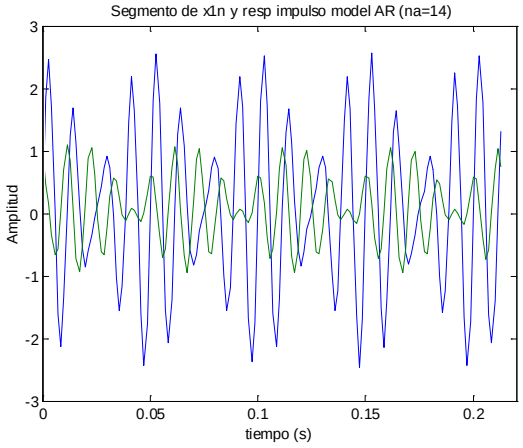

```
>> E=sum(Pxx)*N/length(0:0.01:fs)
  E =  3.2511
>> Pm=fs/length(0:0.01:fs)*sum(Pxx)
  Pm =  1.6244
>> ind75_85=find(f>=75 & f<85);
>> Pm80=fs/length(0:0.01:fs)*sum(Pxx(ind75_85))
```

```
  Pm80 = 1.1205
>> ind95_105=find(f>=95 & f<105);
>> Pm100=fs/length(0:0.01:fs)*sum(Pxx(ind95_105))
  Pm100 =  0.4985
```

Estos valores de Potencia promedio coinciden con los hallados con el método no-paramétrico y los valores teóricos de Pm = 1,625 W, Pm80 = 1,125 W y Pm100 = 0,5 W.

Vamos a representar la señal x1n(n) como la respuesta al impulso del modelo AR obtenido.

```
>>impulso=[1;zeros(N-1,1)];
>> y=filter(1,Ab,impulso);
       % es equivalente a
>> y=impz(1,Ab,N,fs);
>> plot(t(1:150),x1n(1:150), t(1:150),y(1:150))
```



Para hallar también los modelos paramétricos con el *System Identification toolbox*:

```
>> Abi=ar(x1n,14,'burg',[],[],1/fs)
   Discrete-time IDPOLY model: A(q)y(t) = e(t)
   A(q) = 1 - 0.459 q^-1 + 0.05819 q^-2 + 0.3916 q^-3 + 0.3042 q^-4
           + 0.08516 q^-5 - 0.2077 q^-6 - 0.2041 q^-7 - 0.2016 q^-8
           - 0.117 q^-9 - 0.1128 q^-10 - 0.1874 q^-11 - 0.3006 q^-12
                                              - 0.1674 q^-13 + 0.147 q^-14
   Estimated using AR
   Loss function 0.00145991 and FPE 0.00148938
   Sampling interval: 0.00142857
```

Para extraer los parámetros A en un vector de parámetros:

```
>>A = Abi.a  % equivalente a: A=get(Abi,'a');
  A=[1.0000   -0.4590  0.0582  0.3916  0.3042 0.0852   -0.2077
     -0.2041 -0.2016 -0.1170 -0.1128  -0.1874  -0.3006  -0.1674  0.1470]
```

Son los mismos valores que los obtenidos con la función **arburg** de *Signal Processing.*
Para extraer la varianza del ruido o Función de Pérdida:

```
>>VarNoise = Abi.NoiseVariance   % VarNoise= 0.00145991
>>LossFcn = Abi.EstimationInfo.LossFcn
```

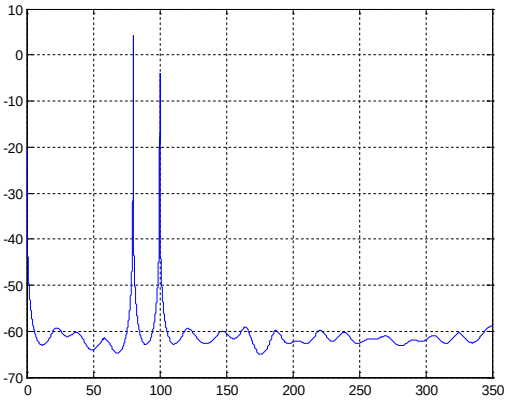
Para extraer el criterio de Akaike "Error de Predicción Final" FPE

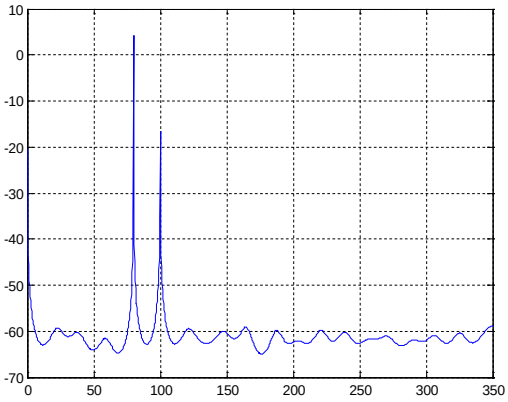
```
>>FPE = Abi.EstimationInfo.FPE  % = 0.00148938
```

Con el toolbox ARMASA calculamos la PSD (ASAglob_subtr_mean = 0)

```
>> [Aa,sel]=sig2ar(x1n);
   % no se remueve el valor medio de x1n:
>> ASAglob_subtr_mean = 0;
```

```
    >> na=length(Aa)-1    % na =  55  Criterio CIC
>> [Pxx,f]=arma2psd(Aa,1,0:0.01:350,1/fs); >> [Pxx,f]=arma2psd(Aa,1,N,1/fs);
>> plot(f,10*log10(Pxx)), drid              >> plot(f,10*log10(Pxx)), grid
```





b) Método de la covarianza

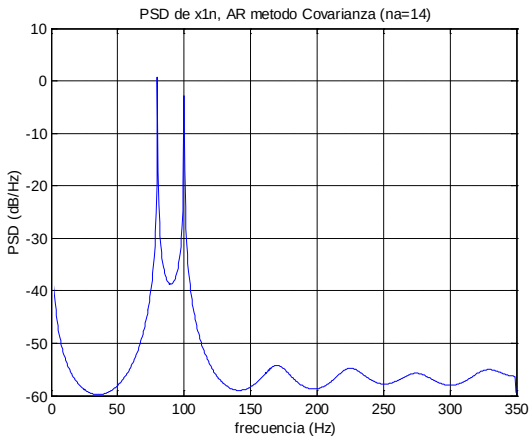
Se resume el procedimiento de la sección anterior con un orden *na* = 14:

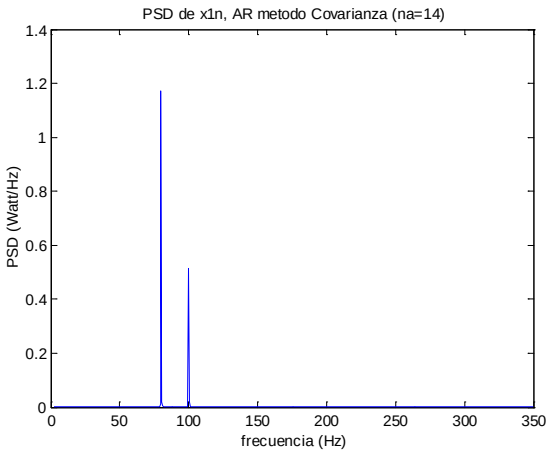
```
    >> [Ac,ec]=arcov(x1n,14);
```
Ac = [1.0000 -0.4623 0.0557 0.3983 0.2996 0.0799 -0.2044 -0.1990 -0.2083
-0.1146 -0.1093 -0.1897 -0.2984 -0.1669 0.1469]
ec= 0.00145914 % Varianza del error o estimada del ruido
% Equivalente en System identification a
```
    >> Aci=ar(x1n,14,'ls');
    Ac=Aci.a;
    >> e=Aci.NoiseVariance
    e =  0.00145914
```

Se halla directamente la PSD con

```
    >> [Pc,fc]=pcov(x1n,14,N,fs);
    >> plot(fc,10*log10(Pc))
```

```
>>plot(fc,Pc)
```





c) Método de la Covarianza modificado

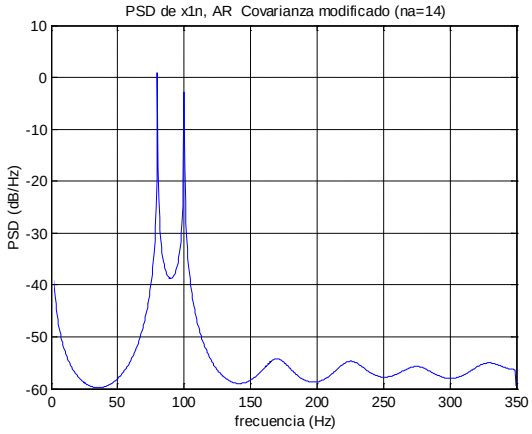
```
    >> [Am,em]=armcov(x1n,14);
```
Am = [1.0000 -0.4608 0.0555 0.3970 0.3018 0.0805 -0.2056 -0.1991 -0.2068
-0.1164 -0.1096 -0.1882 -0.2997 -0.1673 0.1472]
ec = 0.00145914

% Equivalente en System identification a

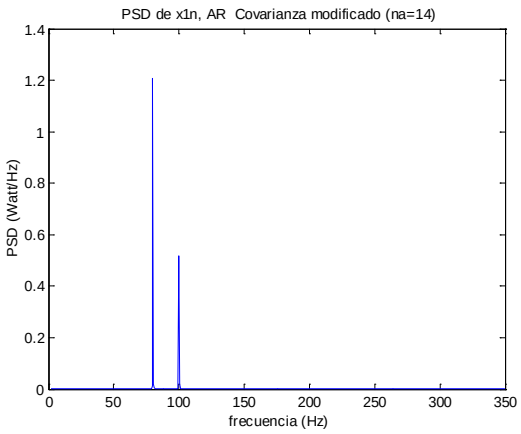
```
>> Ami=ar(x1n,14,'fb');
```

% equivale a Aci=ar(x1n,14); ya que 'fb' es el método por defecto en ar

```
>> Am=Ami.a;
>> e=Ami.NoiseVariance
e =   0.001459159
>> [Pm,fm]=pmcov(x1n,14,N,fs);
>> plot(fm,10*log10(Pm))
```




```
>> plot(fm,Pm)
```



d) Método de Yule-Walker

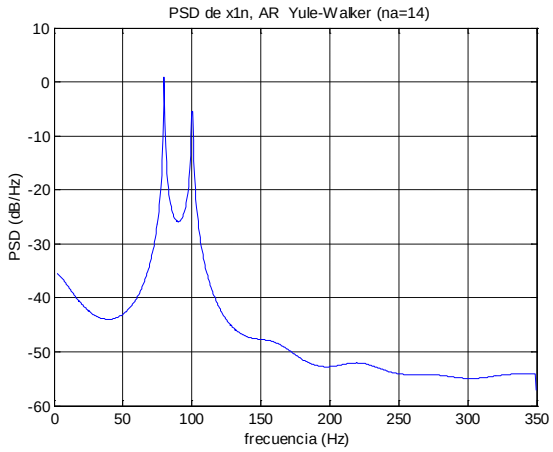
```
>> [Ay,ey]=aryule(x1n,14);
```

Ay = [1.0000 -1.2691 0.6934 0.3015 -0.0631 -0.0779 -0.1055 0.0306 -0.0628
-0.0592 -0.0419 -0.0505 -0.1047 0.0102 0.0924]

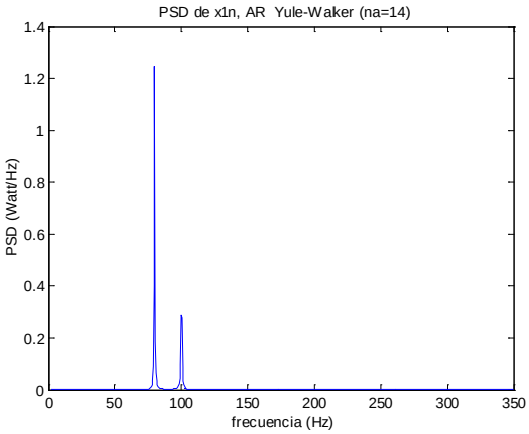
ey = 0.008705831

% Equivalente en System identification a

```
>> Ayi=ar(x1n,14,'yw');
Ay=Ayi.a;
>> e=Ayi.NoiseVariance
e = 0.002832392
>> [Py,fy]=pyulear(x1n,14,N,fs);
>> plot(fy,10*log10(Py))
```




```
>>plot(fy,Py)
```



4.4.2) Densidad de Potencia Espectral con modelos ARMA

Aplicaremos nuestras rutinas para obtener los modelos ARMA y la varianza del error, con el fin de estimar la PSD con estos modelos paramétricos. Se utilizará el orden de modelo

[na=12, nb=11], escogido del cambio de pendiente o "codo" de la curva de la función de pérdida y el orden dado por el criterio GIC(d,3) de ARMASA: [na=8, nb=7]

a) Método de Prony

Usando la función **prony_e**, tenemos:

```
>>na=12;
>>nb=11;
>>[Bp,Ap,ep]=prony_e(x1n,nb,na);
```

Bp=[0.0649 1.8210 1.5740 0.7090 0.2022 0.1294 0.2968 -0.0735 -0.8202
-1.4406 -1.2395 -0.6184]

Ap=[1.0000 -0.4828 0.0781 0.4197 0.3169 0.0831 -0.2000 -0.1955 -0.1696
-0.0988 -0.1176 -0.2418 -0.3649]

ep=0.00149817

Utilizando la función **arx_arma** hay sólo una ligera diferencia en la varianza del error, $e=\sigma^2$. En este caso, σ^2 se obtiene directamente del modelo que proporciona la función **arx**. En **prony_e**, σ^2 se calcula con la función **pe**.

```
>>[Ba,Aa,ea]=arx_arma(x1n,nb,na);
```

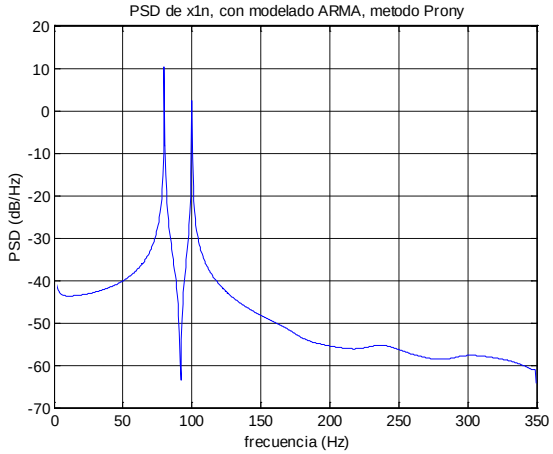
Ba=[0.0649 1.8210 1.5740 0.7090 0.2022 0.1294 0.2968 -0.0735 -0.8202
-1.4406 -1.2395 -0.6184]

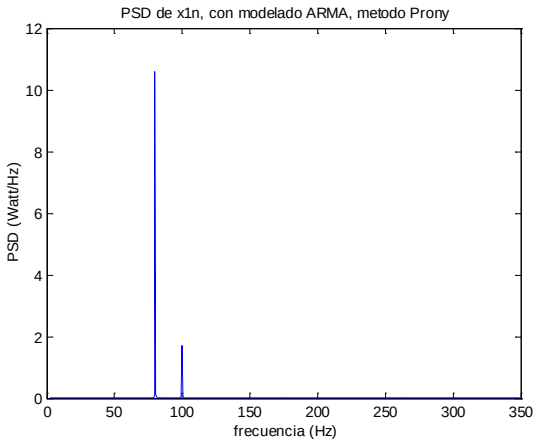
Aa=[1.0000 -0.4828 0.0781 0.4197 0.3169 0.0831 -0.2000 -0.1955 -0.1696
-0.0988 -0.1176 -0.2418 -0.3649]

ea= 0.00152428

Calculemos la PSD con los parámetros Bp, Ap y la varianza del error ea

```
>> [H,f]=freqz(Bp,Ap,N,'whole',fs);
>> Pxp=ea*(abs(H).^2)/fs;  % PSD (de ambos lados del espectro)
>> if rem(N,2), % se evalúa la paridad de N
>> select = 1:(N+1)/2; % si N es impar
>> else
>> select = 1:N/2+1; % si N es par
>> end
>> % select en este caso es de 701 elementos
>> Pxp_unlado=Pxp(select);
>> Pxp_u=[Pxp_unlado(1); 2*Pxp_unlado(2:end-1);Pxp_unlado(end)];
>> fp=f(select);
  >> plot(fp,10*log10(Pxp_u))              >> plot(fp,Pxp_u)
```

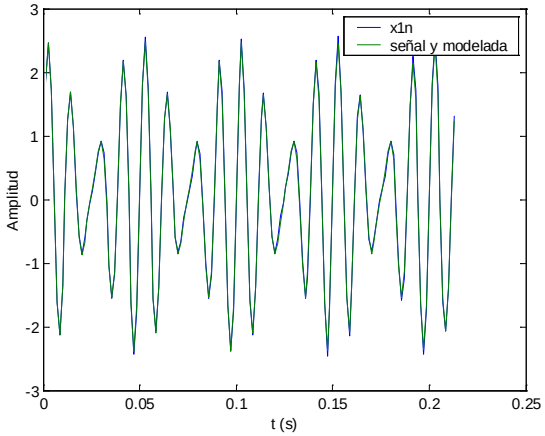




La función **armaspectra**, realiza los pasos anteriores para calcular y, opcionalmente, graficar la PSD del modelo ARMA hallado. Usando un vector de frecuencias entre 0 y fs, tenemos:

Vamos a representar la señal x1n(n) como la respuesta al impulso del modelo AR obtenido.

```
>> impulso=[1;zeros(N-1,1)];
>> y=filter(Bp,Ap,impulso);
>> plot(t(1:150),x1n(1:150),t(1:150),y(1:150))
```



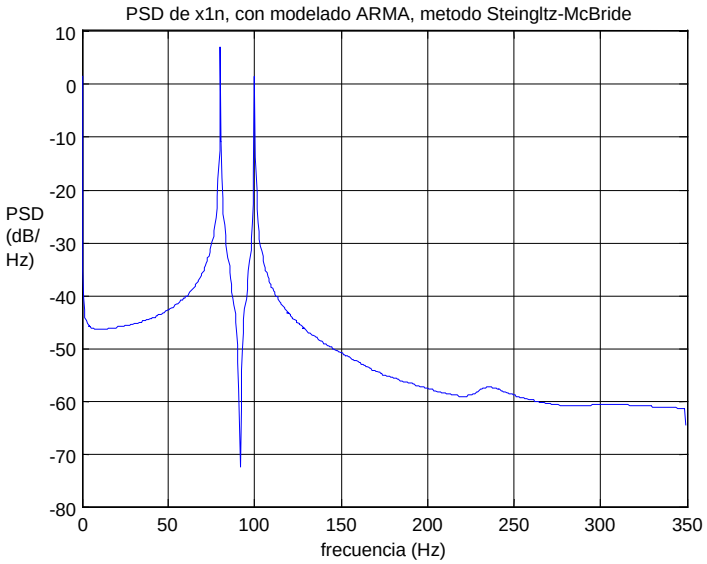
Este método no produce tan buenos resultados para estimar las cantidades correctas de Energía y potencia media:

```
>> [Pxx,f]=armaspectra(Bp,Ap,ep,N,fs);
>> ind75_85=find(f>=75 & f<85);
>> ind95_105=find(f>=95 & f<105);
>> Pm=fs/N*sum(Pxx);              %  Pm =    6.3830
>> Pm=fs/N*sum(Pxx(ind75_85));  % Pm =    5.4320

>> Pm=fs/N*sum(Pxx(ind95_105));   % Pm =  0.9320
```

b) Método de Steingltz-McBride

```
>> [Bs, As, es]=stmcb_e(x1n, nb, na);
   Bs = [0.0397   1.8303   1.4761   0.5890   0.3106   0.1995   0.3386  -0.2674  -0.7788
         -1.2259  -1.2413  -0.7643]
   As = [1.0000  -0.5362   0.0804   0.5382   0.1843   0.1288  -0.3212   0.0142  -0.2074
         -0.2192  -0.1011  -0.1215  -0.4397]
   es = 0.00079935
>> armaspectra(Bs,As,es,N,fs);
>> title('PSD de x1n, con modelado ARMA, metodo Steingltz-McBride')
```



c) Método ARMA, utilzando la función **oe** de *System Identification*

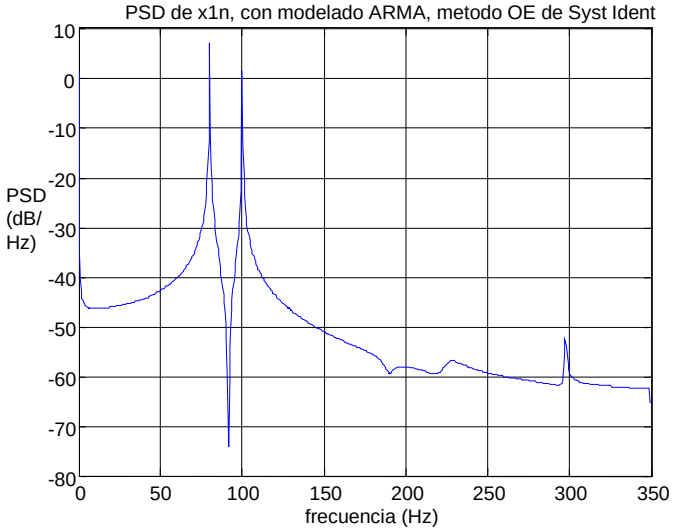
```
>> [Bo,Ao,eo]=oe_arma(x1n,nb,na);
```

Bo= [0.0639 1.8365 2.1106 1.6461 0.3486 -0.1883 0.1777 0.4851 -0.9863 -1.7883 -2.0268 -1.0001]

Ao= [1.0000 -0.1983 0.2150 0.0396 0.5071 0.3552 0.0288 -0.8270 0.0814 -0.2891 0.0361 -0.3807 -0.5686]

eo= 0.00080584

```
>> armaspectra(Bo,Ao,eo,N,fs), title('PSD de x1n, con modelado ARMA, metodo OE de Syst Ident')
```



d) Método ARMA, utilzando la función **armax** de *System Identification*

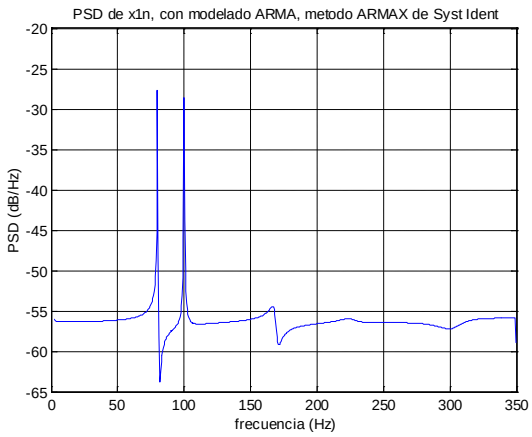
```
>> [Bam,Aam,eam]=armax_arma(x1n,nb,na);
```

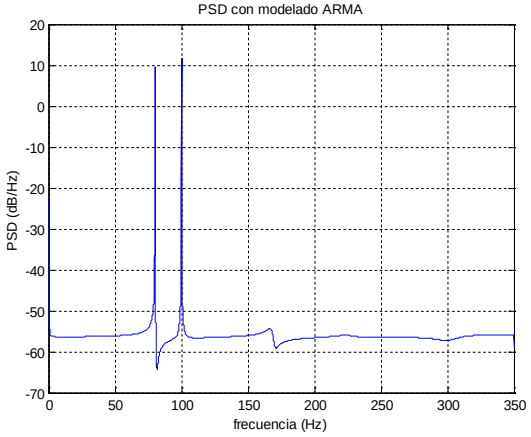
Bam = [1.0000 -1.3420 1.4481 -0.8115 0.6621 -0.2105 0.1134 -0.4657 0.2824 -0.6486 0.5883 -0.6512]

Aam = [1.0000 -1.3695 1.4782 -0.8018 0.6519 -0.1702 0.0618 -0.3701 0.2205 -0.5974 0.5641 -0.6346 -0.0335]

eam = 0.00083017

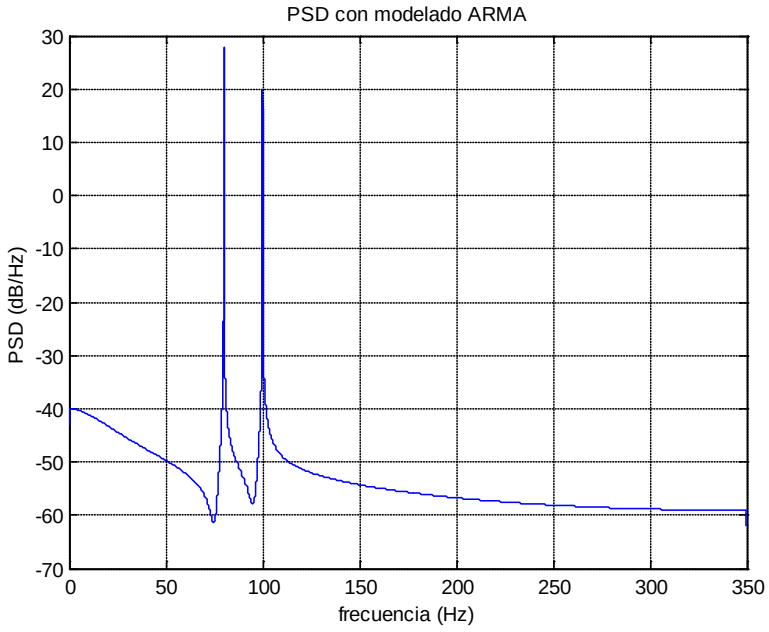
```
>> armaspectra(Bam,Aam,eam,N,fs); >> armaspectra(Bam,Aam,eam,0:0.02:700,fs);
   >> title('PSD de x1n, con modelado ARMA, metodo ARMAX de Syst Ident')
```





Se puede apreciar los diferentes valores de escalado al variar el número de puntos Nfft de cálculo del espectro. Probemos el orden [na=5, nb=4], con el cual obtenemos razonables valores de Potencia media

```
>> [Bp,Ap,ep]=armax_arma(x1n,4,5);
>> [Pxx,f]=armaspectra(Bp,Ap,ep,300000,fs);
>> ind75_85=find(f>=75 & f<85);
>> ind95_105=find(f>=95 & f<105);
>> Pm=fs/300000*sum(Pxx);                % Pm =    2.0587
>> Pm=fs/300000*sum(Pxx(ind75_85));      % Pm =    1.5926
>> Pm=fs/300000*sum(Pxx(ind95_105));     % Pm =    0.4629
>> armaspectra(Bp,Ap,ep,300000,fs);
```



e) Método ARMA, utilizando el método de Durbin-Broersen con ARMASA

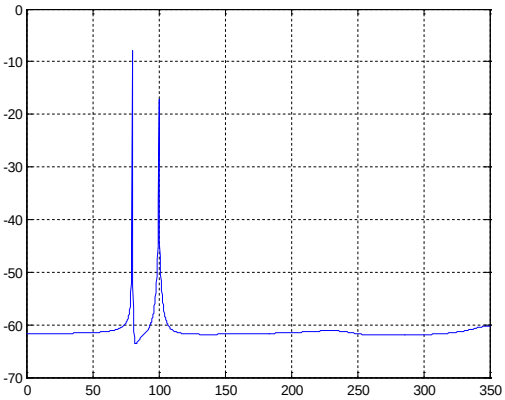
```
>> [A,B,sel]=sig2arma(x1n,8);  % na=8,    nb=na-1= 7
>>sel
```

funct_name: 'sig2arma'
funct_version: [2001 1 7 12 0 0]
date_time: 'Mon 31-Mar-2003 21:36:55'
comp_time: 0.6000
ar: [1.0000 -0.9699 0.4674 0.6529 0.2133 0.0255 -0.1674 0.5950 0.0073]
ma: [1 -0.9098 0.4084 0.5971 0.2519 -0.0048 -0.1271 0.4970]

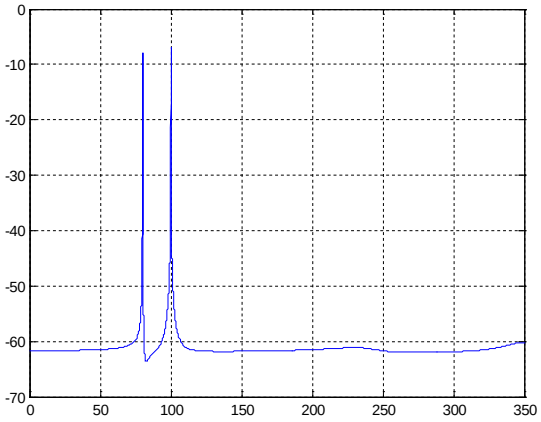
```
        ar_sel: [1x41 double]
        mean_adj: 1     %  se sustrajo la media de la señal
        cand_ar_order: 8
        arma_order_diff: 1
        gic3: -7.0076
        pe_est: 8.9473e-004
>> e=sel.pe_est;  % varianza del error
```

Cálculo de la PSD con la función **arma2psd** de ARMASA:

```
>> [P,f]=arma2psd(A,B,N,1/fs);
>> plot(f,10*log10(P)), grid
```

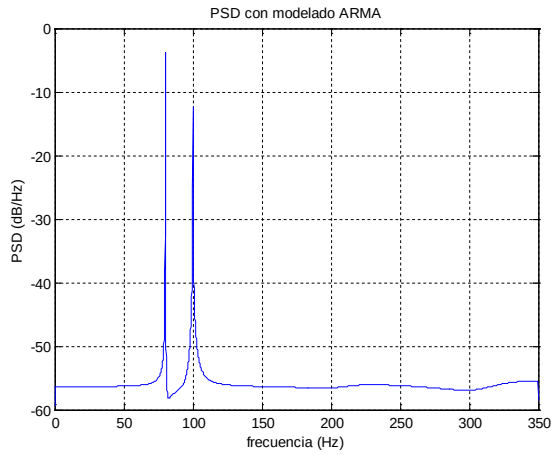



```
>> [P,f]=arma2psd(A,B,0:0.02:350,1/fs);
>> plot(f,10*log10(P)), grid
```

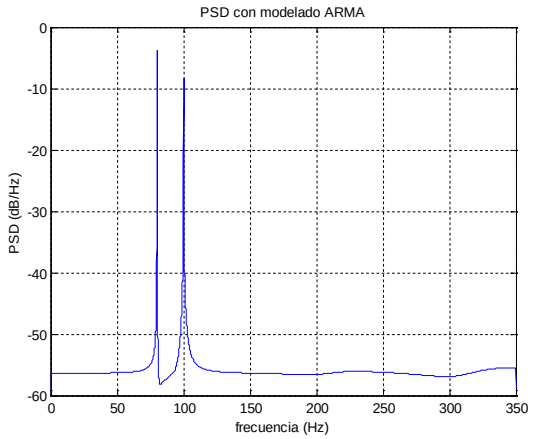


Cálculo de PSD con la función **armaspectra**:

```
>> armaspectra(B,A,e,0:0.1:700,fs);
```

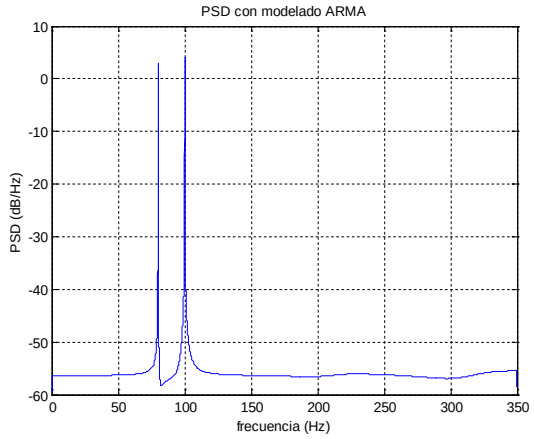


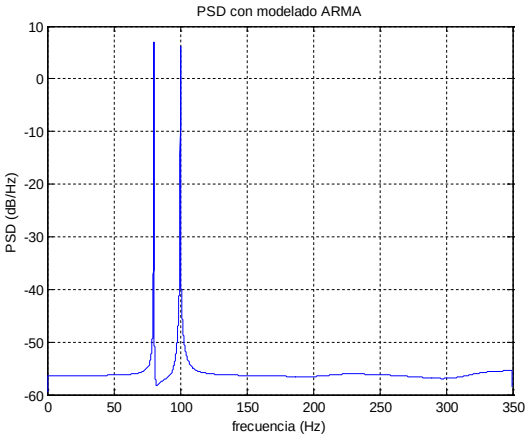

```
>> armaspectra(B,A,e,0:0.05:700,fs);
```




```
>> armaspectra(B,A,e,0:0.015:700,fs);
```

```
>> armaspectra(B,A,e,4096*10,fs);
```



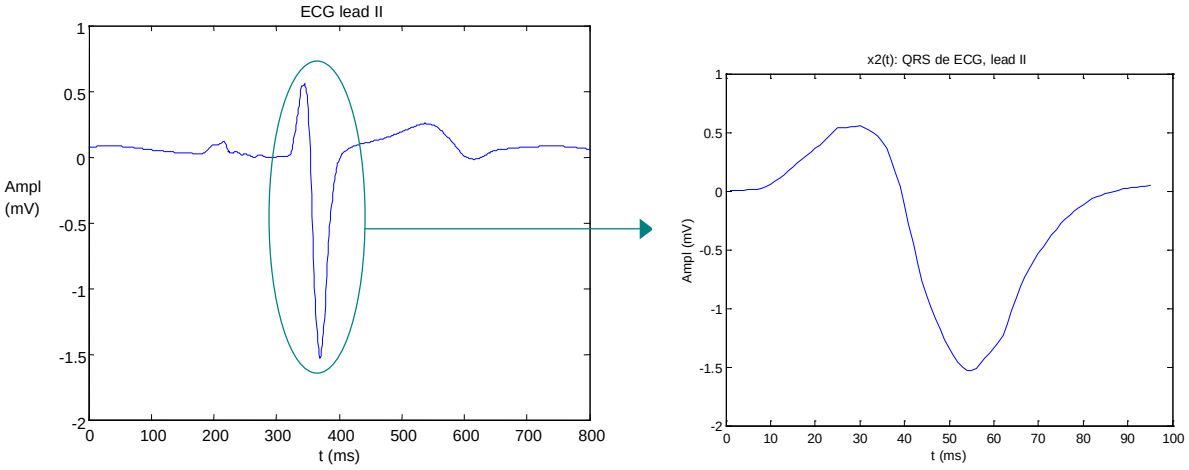


Las gráficas PSD previas del mismo modelo ARMA utilizando distintos número de puntos, para evaluar la respuesta frecuencial de la función de transferencia discreta B(q)/A(q), muestran la posible sensibilidad de la respuesta frecuencial y la PSD al cálculo de los puntos frecuenciales sobre el círculo unitario. Hay que notar que la señal a modelar se compone de dos sinusoides puras de 80 y 100 Hz con ruido añadido, por lo que la evaluación de los puntos frecuenciales alrededor o sobre las frecuencias 80 y 100 Hz es crítica.

5. ESTIMACIÓN ESPECTRAL. APLICACIONES A SEÑALES BIOMÉDICAS

5.1) Energía espectral del complejo QRS de un ECG.

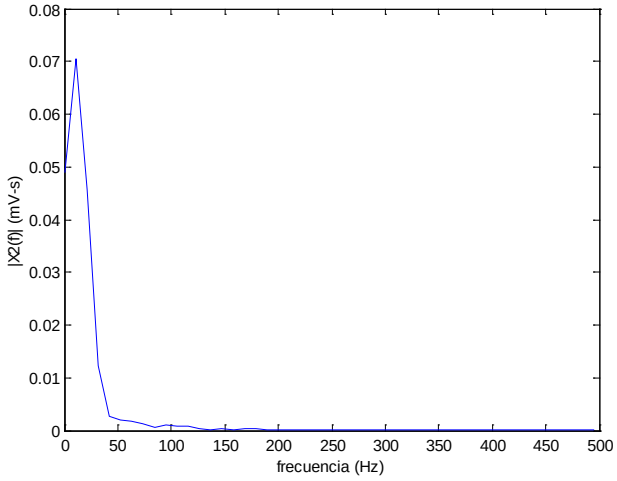
Aplicaremos las técnicas de estimación de la PSD para hallar la energía espectral del QRS de un electrocardiograma (ECG). Tomemos la señal x2(t) como el QRS de la señal ECG, lead II del paciente N° 9, en pre-PTCA de la base de datos de Staff3:



x2(t) está muestrado a fs=1000 Hz y es de N = 95 muestras. Hallaremos la energía de la señal:

```
>> load x2
>> fs=1000;
>> N=length(x2)
```
N = 95
```
>> t=(1:N)'/fs; % base de tiempo en segundos de la señal x2
>> Ener=trapz(t,x2.^2)  % Area: Integral sobre el segmento de señal
>> Ener= sum(x2.^2)/fs  %Cálculo equivalente
```
Ener = 0.0444 % Unidades: (mV)^2-s
```
>> X2 = fft(x2);
>> E=sum(abs(X2).^2)/(fs*N)
```
% Cálculo de la Energía en el dominio de Fourier

E = 0.0444 % Unidades (mV)^2-s
```
>> f=(0:N-1)' *fs/N;
>>    select = 1:(N+1)/2; %  N es impar
>> plot(f(select),2*abs(X2(select))/fs)
```
%Amplitud escalada equivalente al espectro de la señal analógica

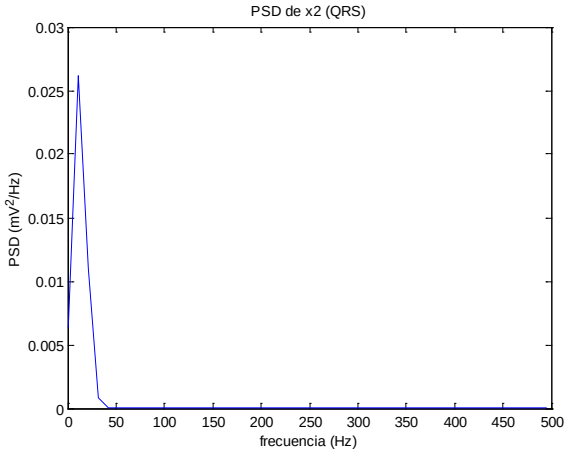


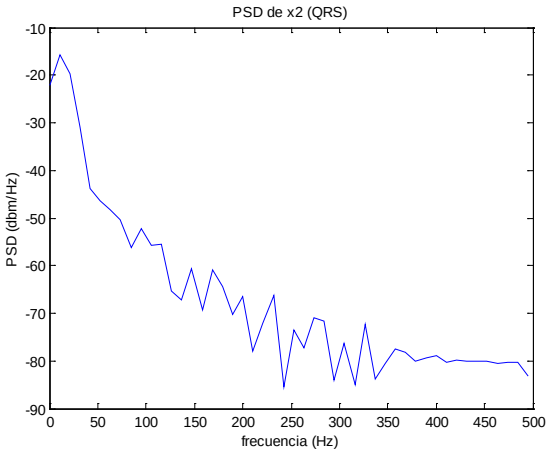
5.1.1) PSD con técnicas no-paramétricas

Utilizando el periodograma estándar, tenemos:

```
>> [Px,f]=periodogram(x2,[],N,fs);
>> plot(f,Px)
>> xlabel('frecuencia (Hz)')
>> ylabel('PSD (mV^2/Hz)')
>> title('PSD de x2 (QRS)')
```

>> plot(f,10*log10(Px))





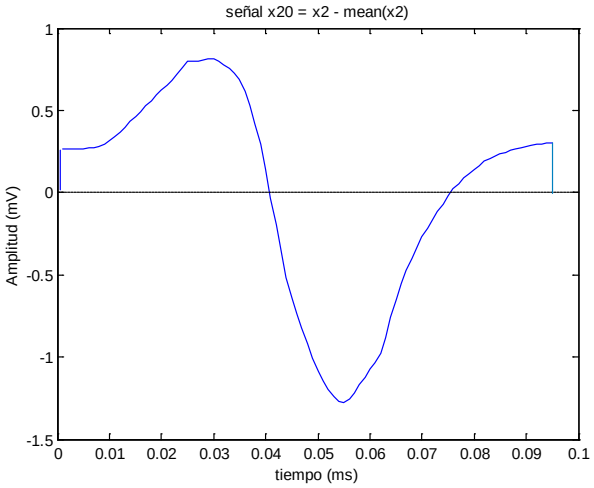
Vamos a hallar la energía y potencia media total y en las bandas de [0 20], [20 40], [40 80], [80 150] y [150 250] Hz. Probaremos evaluar el círculo unitario del plano *z* con *N=length(x2)* =95, 256 y 4096 puntos. El incoveniente de usar el periodograma estándar (sin envantanar) es que el inicio y final del segmento a analizar puede contener un valor diferente de cero que, en la práctica, equivale a funciones escalón que altera el contenido de altas frecuencias al aplicar la FFT. El QRS de la señal x2 a analizar comienza y termina en valores cercanos al eje isoeléctrico de potencial 0.

```
>> [Px,f]=periodogram(x2,[],N,fs);
>> E=sum(Px);              % E =  0.0444  mV^2-s   es la energía total
>> Pm=fs/N*sum(Px); % Pm =  0.4671  mV^2 es la potencia media (average) total
>> Pm2= trapz(f, Px);
>> ind0_20=find(f>=0 & f<20);
>> ind20_40 =find(f>=20 & f<40);
>> ind40_80=find(f>=40 & f<80);
>> ind80_150=find(f>=80 & f<150);
>> ind150_250=find(f>=150 & f<=250);
```

	N= length(x2)=95		Nfft = 256			Nfft = 4096		
Bandas (Hz)	Energía E=sum(Px) (mV^2 s)	Pot med Pm=fs/N* sum(Px) (mV^2)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)
Total	0.0444	0.4671	0.0444	0.4671	0,4547	0,0444	0,4671	0,4663
[0 20]	0.0325	0.3421	0.0403	0.4246	0,3846	0,0385	0,4048	0,4024
[20 40]	0.0118	0.1239	0.0040	0.0416	0,0282	0,0058	0,0614	0,0598
[40 80]	8.85e-005	9.31e-004	5.83e-005	6.13e-004	5,47E-04	6,65E-05	7,00E-04	6,90E-04
[80 150]	1.54e-005	1.62e-004	1.60e-005	1.68e-004	1,64E-04	1,60E-05	1,69E-04	1,69E-04
[150 250]	1.96e-006	2.06e-005	2.16e-006	2.27e-005	2,20E-05	2,17E-06	2,29E-05	2,28E-05

Tomaremos ahora el QRS de la señal x2, removiéndole el valor medio. Se verá en la gráfica, que ahora la señal presenta un ligero escalón al comienzo y final que suele aumentar los componentes de alta frecuencia.

```
>> x20=x2-mean(x2);
>> plot(t,x20)
```



Hallaremos otra vez la energía y potencia media total y en las bandas de [0 20], [20 40], [40 80], [80 150] y [150 250] Hz.

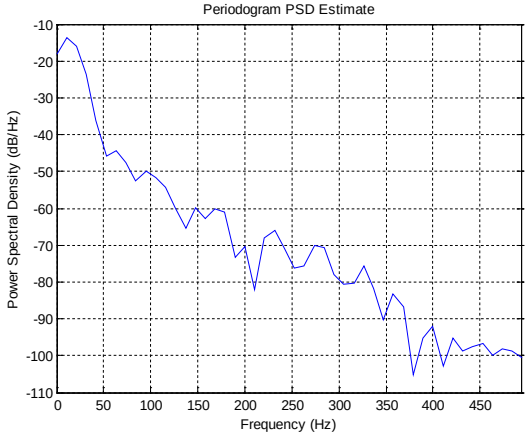
	N= length(x2)=95		Nfft = 256			Nfft = 4096		
Bandas (Hz)	Energía E=sum(Px) (mV^2 s)	Pot med Pm=fs/N* sum(Px) (mV^2)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)
Total	0,0380	0,4004	0,0380	0,4004	0,4004	0,0380	0,4004	0,4004
[0 20]	0,0262	0,2754	0,0344	0,3616	0,3314	0,0324	0,3410	0,3392
[20 40]	0,0118	0,1239	0,0034	0,0362	0,0244	0,0054	0,0567	0,0549
[40 80]	8,85E-05	9,31E-04	1,22E-04	1,29E-03	1,20E-03	1,40E-04	1,48E-03	1,45E-03
[80 150]	1,54E-05	1,62E-04	6,72E-05	7,07E-04	6,96E-04	6,80E-05	7,16E-04	7,13E-04
[150 250]	1,96E-06	2,06E-05	2,69E-05	2,84E-04	2,70E-04	2,67E-05	2,82E-04	2,81E-04

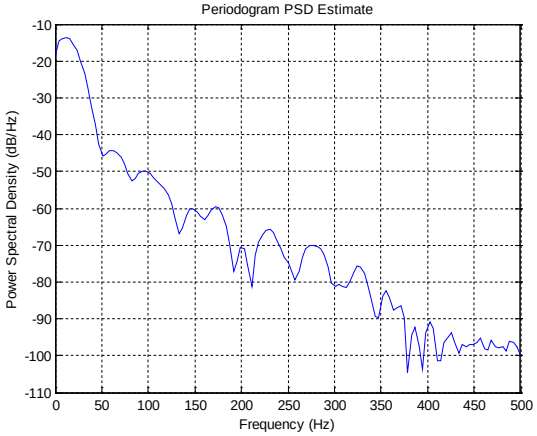
Se nota la elevación de los componentes de alta frecuencia respecto a la señal x2, por los efectos del inicio y final del segmento que distorsionan el espectro en las de altas frecuencias.

Al utilizar una ventana de Hamming con el periodograma enventanado:

```
>> periodogram(x2,hamming(N),N,fs);
```

```
>> periodogram(x2,hamming(N),256,fs);
   % se usan 256 puntos parea la DFT
```





Hallemos a continuación la energía y potencia media total y en las mismas bandas de frecuencia, con la señal x2 enventanada con la ventana de Hanning y luego la señal x20, donde se le removió el valor medio, con la misma ventana.

```
>> [Px,f]=periodogram(x2,hann(N),N,fs); % con N=length(x20), N= 256 y N=4096
```

Señal x2 original

	N= length(x2)=95		Nfft = 256			Nfft = 4096		
Bandas (Hz)	Energía E=sum(Px) (mV^2 s)	Pot med Pm=fs/N* sum(Px) (mV^2)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)
Total	0,0918	0,9666	0,0918	0,9666	0,9344	0,0918	0,9666	0,9646
[0 20]	0,0600	0,6313	0,0772	0,8124	0,7197	0,0728	0,7666	0,7609
[20 40]	0,0315	0,3311	0,0144	0,1519	0,1113	0,0187	0,1971	0,1935
[40 80]	3,63E-04	3,82E-03	1,83E-04	1,93E-03	1,49E-03	2,39E-04	2,51E-03	2,45E-03
[80 150]	3,03E-05	3,19E-04	2,97E-05	3,13E-04	2,98E-04	2,98E-05	3,14E-04	3,13E-04
[150 250]	2,99E-06	3,15E-05	3,22E-06	3,39E-05	3,22E-05	3,25E-06	3,43E-05	3,41E-05

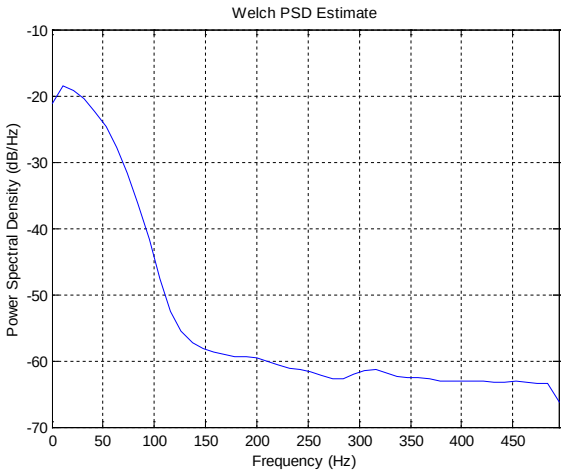
Señal x20 = x2 – mean(x2)

	N= length(x2)=95		Nfft = 256			Nfft = 4096		
Bandas (Hz)	Energía E=sum(Px) (mV^2 s)	Pot med Pm=fs/N* sum(Px) (mV^2)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)
Total	0,0663	0,6983	0,0663	0,6983	0,6904	0,0663	0,6983	0,6978
[0 20]	0,0346	0,3637	0,0515	0,5416	0,4747	0,0472	0,4963	0,4922
[20 40]	0,0314	0,3305	0,0147	0,1544	0,1130	0,0189	0,1991	0,1955
[40 80]	3,62E-04	3,81E-03	1,86E-04	1,96E-03	1,52E-03	2,40E-04	2,53E-03	2,46E-03
[80 150]	3,04E-05	3,20E-04	2,97E-05	3,12E-04	2,98E-04	2,98E-05	3,14E-04	3,13E-04
[150 250]	3,00E-06	3,16E-05	3,23E-06	3,40E-05	3,22E-05	3,26E-06	3,43E-05	3,42E-05

Se puede observar de las tablas que la remoción del valor medio de la señal x2 afecta la estimación de los componentes frecuenciales en la banda de [0 20] Hz, que incluye el nivel de continua o frecuencia 0. El uso de 4096 aunque no mejora la resolución para evaluar los componentes de las distintas bandas de frecuencia, mejora la exactitud al aproximar mejor el espectro continuo de una hipotética señal de duración infinita, interpolando mayor número de puntos en el dominio frecuencial.

Apliquemos el método de Welch para estimar la PSD

```
>> pwelch(x2,[],[],128,fs);
```



Vamos a hallar la energía y potencia media total y en las bandas de [0 20], [20 40], [40 80], [80 150] y [150 250] Hz. Probaremos evaluar el círculo unitario del plano *z* con *Nfft*= 256 y 4096 puntos. Utilizaremos las señales x2 y x20 (x2 con la media removida).

Señal x2 original

	Nfft = 256			Nfft = 4096		
Bandas (Hz)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)
Total	0,0548	0,5770	0,5622	0,0548	0,5770	0,5760
[0 20]	0,0287	0,3016	0,2625	0,0268	0,2822	0,2798
[20 40]	0,0168	0,1768	0,1414	0,0180	0,1890	0,1868
[40 80]	0,0090	0,0951	0,0832	0,0097	0,1022	0,1013
[80 150]	3,07E-04	3,23E-03	2,64E-03	3,14E-04	3,30E-03	3,26E-03
[150 250]	9,98E-06	1,05E-04	1,01E-04	9,91E-06	1,04E-04	1,04E-04

Señal x20 = x2 – mean(x2)

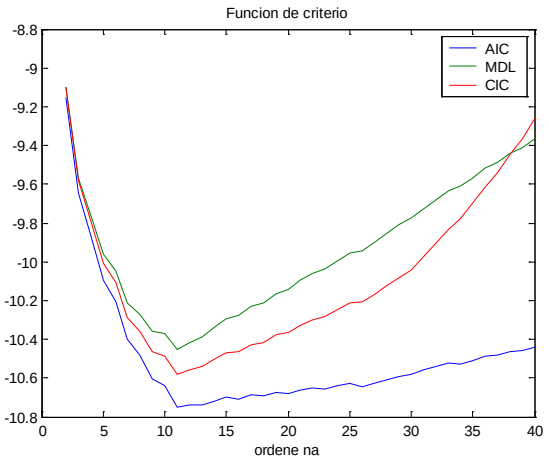
	Nfft = 256			Nfft = 4096		
Bandas (Hz)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)
Total	0,0453	0,4774	0,4654	0,0453	0,4774	0,4766
[0 20]	0,0232	0,2443	0,2126	0,0217	0,2284	0,2265
[20 40]	0,0139	0,1467	0,1174	0,0149	0,1565	0,1546
[40 80]	0,0079	0,0831	0,0729	0,0085	0,0891	0,0883
[80 150]	2,95E-04	3,10E-03	2,55E-03	3,01E-04	3,17E-03	3,12E-03
[150 250]	8,96E-06	9,43E-05	9,03E-05	8,91E-06	9,38E-05	9,36E-05

5.1.2) PSD con métodos paramétricos

5.1.2.1) Estructuras AR

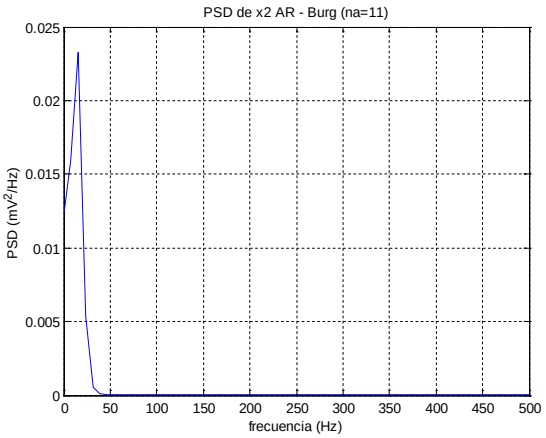
En primer lugar buscaremos el mejor orden de modelo con estructuras AR.

```
>> [nn,V,Vn]=selarstruc(x2,'arburg',2:40,'aic');% se repite para criterio = MDL, FPE y CIC
nn = 11   % el mismo orden mínimo se halló con los criterios MDL, FPE y CIC
>> plot(V(2,:),V(1,:))
```

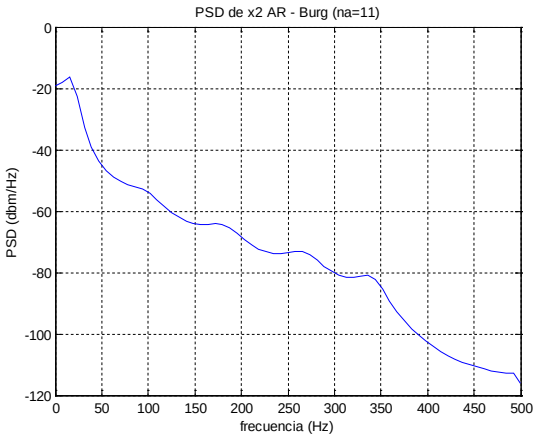


Se halla el modelo AR de orden na = 11:

```
>> [Ab,eb]=arburg(x2,11);
>> [Pb,fb]=armaspectra(1,Ab,eb,128,fs);
>> plot(fb,Pb), grid
```

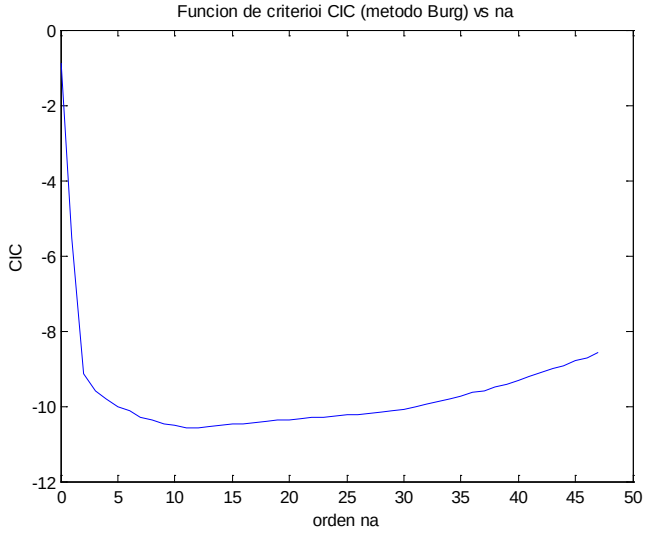



```
>> plot(fb,10*log10(Pb)), grid
```



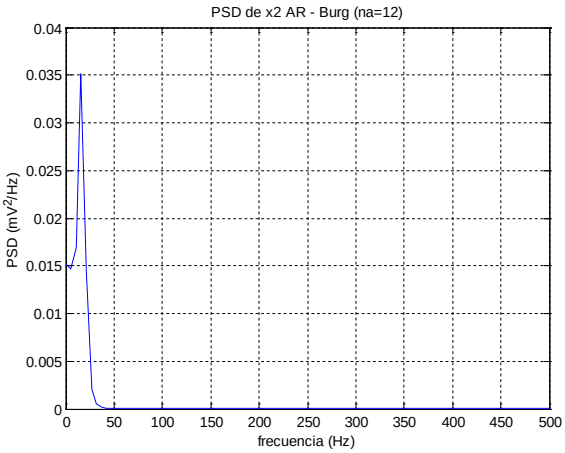
Con el *toolbox* ARMASA, se usa el criterio CIC con el método de Burg para hallar el "mejor" orden de modelo. Por defecto, el toolbox de ARMASA sustrae el valor medio de la señal. Equivale a método de Burg de *Signal Processing*, tomando en cuenta esta sustracción. Para deshabilitar la sustracción del valor medio de la señal se utiliza: ASAglob_subtr_mean = 0;

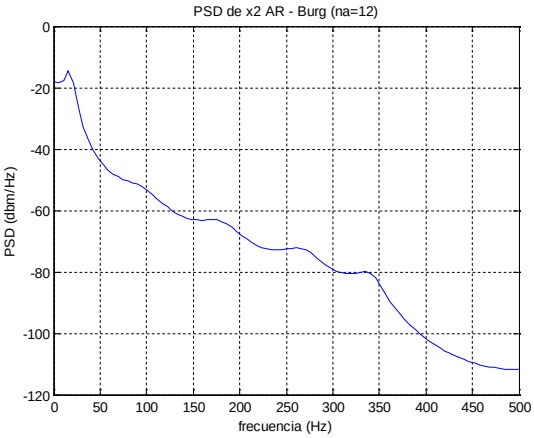
```
>> [Ab2,sel]=sig2ar(x2);
  na=length(Ab2)-1 ;  % na = 11
>>sig_e=sel.pe_est;  % varianza del error para cada na probado
>>cic=sel.cic;  % criterio CIC para cada na probado
>>na_candidato=sel.cand_order;  % valores na probados
```




```
>> [P,f]=arma2psd(A,1,128,1/fs);  %   rutina de ARMASA para cálculo de PSD
>> plot(f,P), grid
```

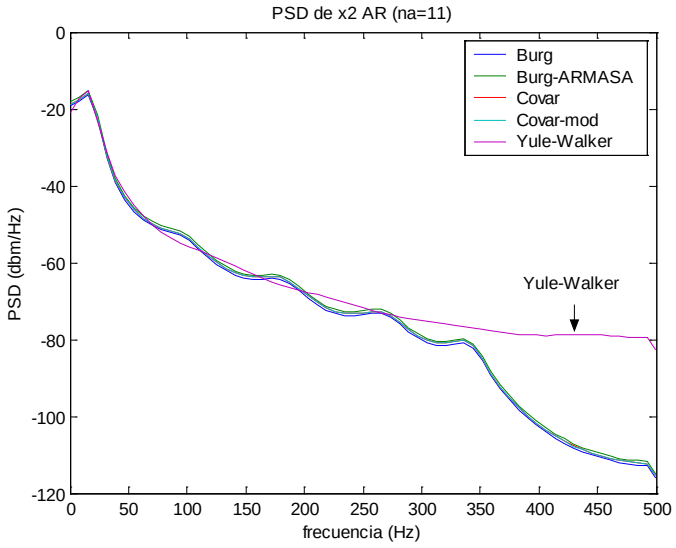
```
>> plot(f,10*log10(P)), grid
```





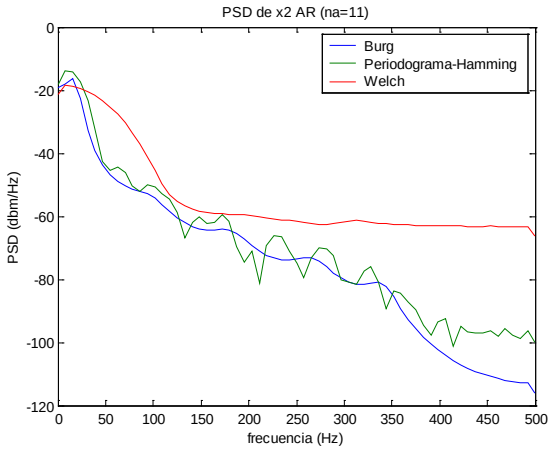
Comparemos la PSD con varios métodos de estructuras AR orden na = 11:

```
>>[Ab,eb]=arburg(x2,11);
>>[Ac,ec]=arcov(x2,11);
>>[Am,em]=armcov(x2,11);
>>[Ay,ey]=aryule(x2,11);
>> ASAglob_subtr_mean = 0;  % No se remueve el valor medio de x2
>> [Ab2,sel]=sig2ar(x2);
>> sig_e=sel.pe_est;
>> [m,i]=min(sel.cic);
>> na_c=sel.cand_order;
 na_c(i)  =  11
>> eb2=sig_e(i)
>> [Pb,fb]=armaspectra(1,Ab,eb,128,fs);
>> [Pb2,fb2]=armaspectra(1,Ab2,eb2,128,fs);
>> [Pc,fc]=armaspectra(1,Ac,ec,128,fs);
>> [Pm,fm]=armaspectra(1,Am,em,128,fs);
>> [Py,fy]=armaspectra(1,Ay,ey,128,fs);
>> plot(fb,10*log10(Pb),fb2,10*log10(Pb2),fc,10*log10(Pc),fm,10*log10(Pm),fy,10*log10(Py))
>> xlabel('frecuencia (Hz)')
>> ylabel('PSD (dbm/Hz)')
>> title('PSD de x2 AR (na=11)'), grid
```



En la gráfica siguiente se comparan la PSD con los métodos del periodograma con ventana de Hamming, de Welch y paramétrico AR-Burg de orden 11:

```
>> [Pp,fp]=periodogram(x2,hamming(N),128,fs);
>> [Pw,fw]=pwelch(x2,[],[],128,fs);
```



Calcularemos la energía y la potencia media media total y en las bandas de [0 20], [20 40], [40 80], [80 150] y [150 250] Hz. con modelado AR y el método de Burg (orden *na* = 11).

```
>> [A,e]=arburg(x2,11);
>> [Px,f]=armaspectra(1,A,e,Nfft,fs);  % Nfft = 256 y 4096
```

Señal x2 original

	Nfft = 256			Nfft = 4096		
Bandas (Hz)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)
Total	0,0443	0,4668	0,4425	0,0444	0,4671	0,4656
[0 20]	0,0414	0,4357	0,3655	0,0387	0,4073	0,4031
[20 40]	0,0028	0,0300	0,0192	0,0056	0,0586	0,0561
[40 80]	8,42E-05	8,86E-04	7,36E-04	9,50E-05	1,00E-03	9,87E-04
[80 150]	1,67E-05	1,76E-04	1,62E-04	1,68E-05	1,76E-04	1,76E-04
[150 250]	1,85E-06	1,94E-05	1,86E-05	1,85E-06	1,95E-05	1,95E-05

Señal x20 = x2 – mean(x2)

	Nfft = 256			Nfft = 4096		
Bandas (Hz)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)
Total	0,0378	0,3974	0,3855	0,0380	0,4004	0,3996
[0 20]	0,0351	0,3690	0,3118	0,0328	0,3455	0,3421
[20 40]	0,0026	0,0273	0,0177	0,0051	0,0537	0,0513
[40 80]	8,76E-05	9,22E-04	7,68E-04	9,86E-05	1,04E-03	1,02E-03
[80 150]	1,67E-05	1,76E-04	1,62E-04	1,68E-05	1,77E-04	1,76E-04
[150 250]	1,87E-06	1,96E-05	1,88E-05	1,87E-06	1,97E-05	1,97E-05

La remoción del valor medio de la señal x2 afecta la estimación de los componentes frecuenciales en la banda de [0 20] Hz, que incluye el nivel de continua o frecuencia 0. El uso de 4096 puntos ofrece una buena opción para evaluar los componentes de las distintas bandas de frecuencia sobre señales ECG, con modelos AR y el método de Burg.

5.1.2.2) Estructuras ARMA

Busquemos el mejor orden con estructuras ARMA:

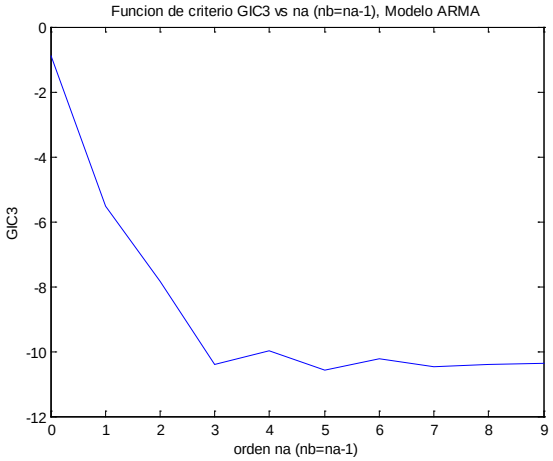
Con el toolbox de ARMASA, utilizando el método de Durbin-Broersen, tenemos

```
>> ASAglob_subtr_mean = 1;  % Se remueve el valor medio de x2
>>[Aarmasa,Barmasa,sel]=sig2arma(x2);
>> sel
```

funct_name: 'sig2arma'
funct_version: [2001 1 7 12 0 0]
date_time: 'Sat 05-Apr-2003 10:36:50'
comp_time: 0.8300
ar: [1 -2.6806 2.7966 -1.8267 1.0252 -0.3109]
ma: [1 1.1390 -0.1046 -0.3838 -0.0078]
ar_sel: [1x12 double]
mean_adj: 1
cand_ar_order: [0 1 2 3 4 5 6 7 8 9]
arma_order_diff: 1
gic3: [1x10 double]
pe_est: [1x10 double]

El mejor orden propuesto es na=5, nb=4

```
>> plot(sel.cand_ar_order,sel.gic3)
>> xlabel('orden na (nb=na-1)')
>> ylabel('GIC3')
```



Si omitimos la remoción del valor medio, tenemos:

```
>> ASAglob_subtr_mean = 0;
>> [Aarmasa0,Barmasa0,sel0]=sig2arma(x2);
```

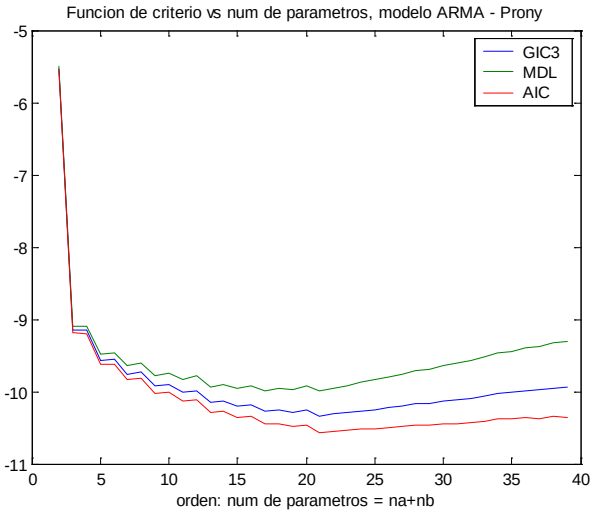
sel2 =
funct_name: 'sig2arma'
funct_version: [2001 1 7 12 0 0]
date_time: 'Sat 05-Apr-2003 13:08:46'
comp_time: 0.1100
ar: [1 -2.0772 1.2075 -0.1232]
ma: [1 1.7429 0.8205]
ar_sel: [1x12 double]
mean_adj: 0
cand_ar_order: [0 1 2 3 4 5 6 7 8 9]
arma_order_diff: 1
gic3: [1x10 double]
pe_est: [1x10 double]

El orden propuesto con la señal original, sin remover su valor medio es (na = 3, nb = 2).

Utilizando nuestra función **selarmastruc**, podemos escoger el método para estimar el modelo ARMA. Utilizaremos la señal x2 original y removiéndole su valor medio y probaremos el método de Prony y el armax_arma:

```
>> m=mean(x2);
>> [nn,Vg,Vng]=selarmastruc(x2-m,'prony_e',1:20,1:19,'gic3');
```

```
nn = [na nb]
   11
   10
>> [nn,Vm,Vnm]=selarmastruc(x2-m,'prony_e',1:20,1:19,'mdl');
nn =
    9
    8
>> [nn,Va,Vna]=selarmastruc(x2-m,'prony_e',1:20,1:19,'aic');
nn =
   11
   10
>> [Vgmin,Vngmin]=min_arma(Vg,Vng);
>> [Vmmin,Vnmmin]=min_arma(Vm,Vnm);
>> [Vamin,Vnamin]=min_arma(Va,Vna);
>> plot(Vgmin(4,:),Vgmin(1,:),Vmmin(4,:),Vmmin(1,:),Vamin(4,:),Vamin(1,:))
>> xlabel('orden: num de parametros = na+nb')
>> title('Funcion de criterio vs num de parametros, modelo ARMA')
>> legend('GIC3','MDL','AIC')
```



Con la señal x2 sin remover su valor medio, tenemos:

```
>> [nn,Vg,Vng]=selarmastruc(x2,'prony_e',1:20,1:19,'gic3');
>> [nn,Vm,Vnm]=selarmastruc(x2,'prony_e',1:20,1:19,'mdl');
>> [nn,Va,Vna]=selarmastruc(x2,'prony_e',1:20,1:19,'aic');
nn =    % (con los 3 criterios)
   11
    8
```

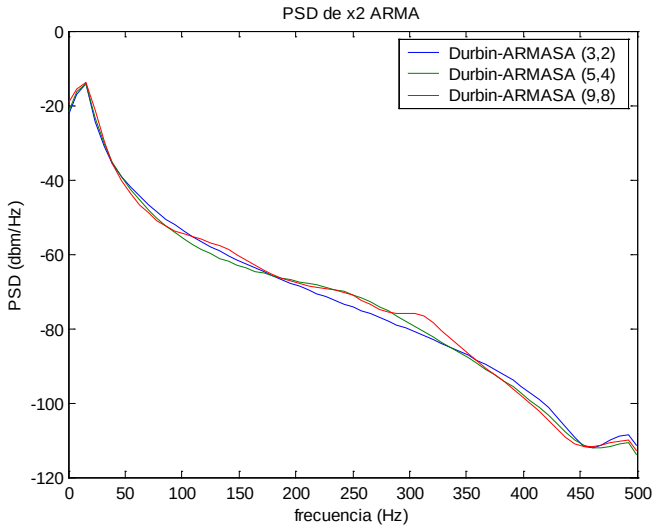
Con el método aramax_arma

```
    % Removiendo el valor medio de x2
>> [nn,Vg,Vng]=selarmastruc2(x2-m,'armax_arma',1:20,1:19,'gic3');
>> [nn,Vm,Vnm]=selarmastruc(x2-m,'armax_arma',1:20,1:19,'mdl');
>> [nn,Va,Vna]=selarmastruc(x2-m,'armax_arma',1:20,1:19,'aic');
  %  nn =n [6 5] con los 3 criterios mdl y gic3 y aic
    % Sin remover el valor medio
>> [nn,Vg,Vng]=selarmastruc2(x2,'armax_arma',1:20,1:19,'gic3');
>> [nn,Vm,Vnm]=selarmastruc2(x2,'armax_arma',1:20,1:19,'mdl');
    % el mínimo está en nn =[3 2]
   % con los criterios mdl y gic3  y luego nn =[8 7]
>> [nn,Va,Vna]=selarmastruc2(x2,'armax_arma',1:20,1:19,'aic');
    % mínimo nn = [10 9]
```

Para la estimación de la PSD utilizaremos los métodos de Durbin-Broersen (ARMASA), de Prony y armax_arma con órdenes (na=3, nb=2), (na=5, nb=4) y (na=9, nb=8) con la señal x2 original.

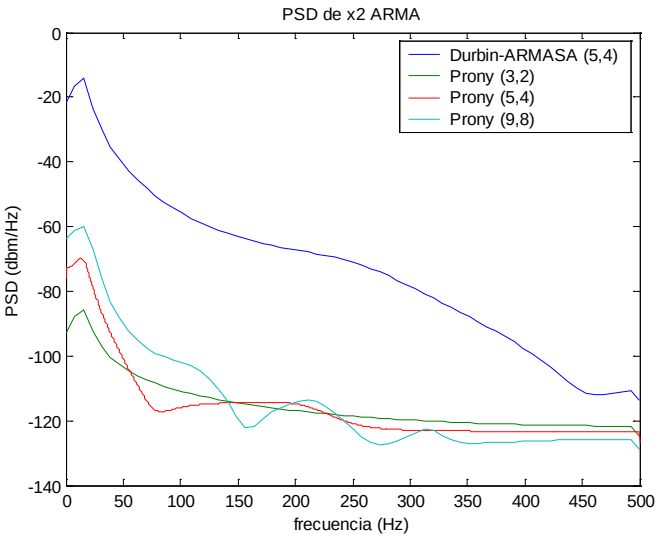
Con ARMASA:

```
>> ASAglob_subtr_mean =0;
>> [Aarmasa0,Barmasa0,sel0]=sig2arma(x2,3);  % orden (na=3,nb=2)
>> earmasa0=sel0.pe_est;
>> [Pa0,fa0]=armaspectra(Barmasa0,Aarmasa0,earmasa0,128,fs);
>> [Aarmasa05,Barmasa04,sel054]=sig2arma(x2,5);  % (na=5,nb=4)
>> earmasa054=sel054.pe_est;
>> [Pa054,fa054]=armaspectra(Barmasa04,Aarmasa05,earmasa054,128,fs);
>> [Aarmasa09,Barmasa08,sel098]=sig2arma(x2,9);  % (na=9,nb=8)
>> earmasa098=sel098.pe_est;
>> [Pa098,fa098]=armaspectra(Barmasa08,Aarmasa09,earmasa098,128,fs);
```



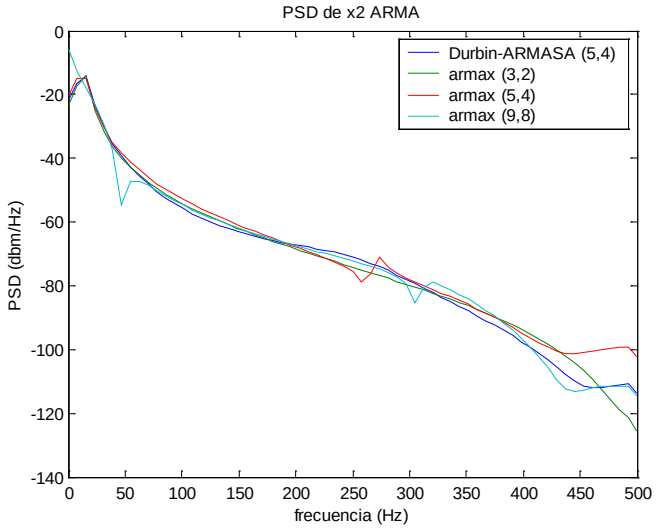
Con las funciones **prony_e** y **armax_arma**:

```
>> [Bp2,Ap3,ep32]=prony_e(x2,2,3);  % (na=3,nb=2)
>> [Bp4,Ap5,ep54]=prony_e(x2,4,5);  % (na=5,nb=4)
>> [Bp8,Ap9,ep98]=prony_e(x2,8,9);  % (na=9,nb=8)
>> [Ppr32,fpr32]=armaspectra(Bp2,Ap3,ep32,128,fs);
>> [Ppr54,fpr54]=armaspectra(Bp4,Ap5,ep54,128,fs);
>> [Ppr98,fpr98]=armaspectra(Bp8,Ap9,ep98,128,fs);
```



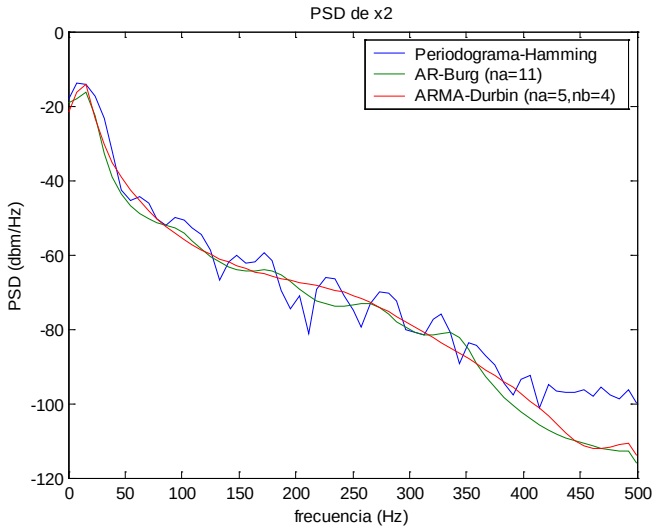

```
>> [Bamx2,Aamx3,eamx32]=armax_arma(x2,2,3);
```

```
>> [Pamx32,famx32]=armaspectra(Bamx2,Aamx3,eamx32,128,fs);
>> [Bamx4,Aamx5,eamx54]=armax_arma(x2,4,5);
>> [Pamx54,famx54]=armaspectra(Bamx4,Aamx5,eamx54,128,fs);
>> [Bamx8,Aamx9,eamx98]=armax_arma(x2,8,9);
>> [Pamx98,famx98]=armaspectra(Bamx8,Aamx9,eamx98,128,fs);
>>plot(fa054,10*log10(Pa054),famx32,10*log10(Pamx32),famx54,10*log10(Pa
mx54),famx98, ... 10*log10(Pamx98))
>> xlabel('frecuencia (Hz)')
>> ylabel('PSD (dbm/Hz)')
>> title('PSD de x2 ARMA')
>> legend('Durbin-ARMASA(5,4)','armax(3,2)','armax(5,4)','armax (9,8)')
```



Comparación de los métodos de periodograma enventanado de hamming, AR-Burg (na=11) y ARMA-Durbin-Broersen (na=5,nb=4)

```
>> plot(fp,10*log10(Pp),fb,10*log10(Pb),fa054,10*log10(Pa054))
```



Hallemos la energía y la potencia media media total y en las bandas de [0 20], [20 40], [40 80], [80 150] y [150 250] Hz. con modelado ARMA y los método de Durbin-Broersen de ARMASA (orden *na* = 5, *nb* = 4) y armax_arma (orden *na* = 8, *nb* = 7) .

Método de Durbin-Boersen de ARMASA:

```
% Se deja el valor medio a la señal: se usa x2 original
>> e=sel.pe_est; >> ASAglob_subtr_mean = 0;
>> [A,B,sel]=sig2arma(x2,5);
>> e=sel.pe_est;
>> [Px,f]=armaspectra(B,A,e,Nfft,fs);  % Nfft = 256 y 4096
```

Señal x2 original (Modelado ARMA-Durbin-Broersen)

	Nfft = 256			Nfft = 4096		
Bandas (Hz)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)
Total	0,0542	0,5706	0,5566	0,0543	0,5712	0,5703
[0 20]	0,0509	0,5359	0,4957	0,0494	0,5198	0,5174
[20 40]	0,0031	0,0324	0,0228	0,0046	0,0487	0,0472
[40 80]	2,07E-04	2,18E-03	1,80E-03	2,34E-04	2,47E-03	2,43E-03
[80 150]	1,49E-05	1,56E-04	1,41E-04	1,50E-05	1,57E-04	1,56E-04
[150 250]	2,05E-06	2,15E-05	2,05E-05	2,05E-06	2,16E-05	2,15E-05

```
>> ASAglob_subtr_mean = 1;   % Se remueve el valor medio a la señal: la función realiza x2-mean(x2)
>> [A,B,sel]=sig2arma(x2,5);
>> e=sel.pe_est;
```

Señal x20 = x2 – mean(x2) (Modelado ARMA-Durbin-Broersen)

	Nfft = 256			Nfft = 4096		
Bandas (Hz)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)
Total	0,0477	0,5023	0,4931	0,0474	0,4994	0,4989
[0 20]	0,0443	0,4665	0,4231	0,0421	0,4429	0,4404
[20 40]	0,0032	0,0338	0,0233	0,0051	0,0542	0,0524
[40 80]	1,80E-04	1,90E-03	1,57E-03	2,04E-04	2,15E-03	2,12E-03
[80 150]	1,43E-05	1,50E-04	1,37E-04	1,44E-05	1,51E-04	1,50E-04
[150 250]	2,21E-06	2,32E-05	2,21E-05	2,21E-06	2,33E-05	2,32E-05

Método armax_arma:

```
>>[B,A,e]=armax_arma(x2,7,8);
>> [Px,f]=armaspectra(B,A,e,Nfft,fs);  % Nfft = 256 y 4096
```

Señal x2 original (Modelado armax_ARMA)

	Nfft = 256			Nfft = 4096		
Bandas (Hz)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)
Total	0,0789	0,8300	0,7591	0,0789	0,8300	0,8256
[0 20]	0,0765	0,8056	0,7194	0,0755	0,7951	0,7897
[20 40]	0,0022	0,0235	0,0165	0,0032	0,0339	0,0331
[40 80]	7,08E-05	7,45E-04	6,04E-04	8,25E-05	8,68E-04	8,54E-04
[80 150]	1,27E-05	1,34E-04	1,24E-04	1,28E-05	1,34E-04	1,34E-04
[150 250]	1,76E-06	1,85E-05	1,75E-05	1,77E-06	1,87E-05	1,86E-05

```
>> [B,A,e]=armax_arma(x20,7,8);
>> [Px,f]=armaspectra(B,A,e,Nfft,fs);  % Nfft = 256 y 4096
```

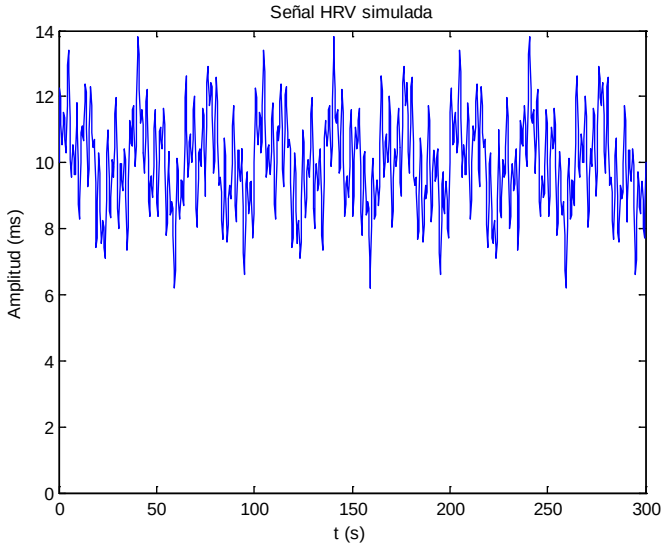
Señal x20 = x2 – mean(x2) (Modelado armax_ARMA)

	Nfft = 256			Nfft = 4096		
Bandas (Hz)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)	Energía E=sum(Px)* N/Nfft (mV^2 s)	Pot med Pm=fs/Nfft* sum(Px) (mV^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (mV^2)
Total	0,0333	0,3508	0,3489	0,0496	0,5225	0,5223
[0 20]	0,0311	0,3271	0,2521	0,0447	0,4707	0,4669
[20 40]	0,0021	0,0224	0,0148	0,0048	0,0503	0,0473
[40 80]	1,10E-04	1,16E-03	9,72E-04	1,23E-04	1,30E-03	1,28E-03
[80 150]	1,31E-05	1,38E-04	1,26E-04	1,32E-05	1,38E-04	1,38E-04
[150 250]	1,71E-06	1,80E-05	1,69E-05	1,72E-06	1,81E-05	1,81E-05

5.2) Señal de variabilidad del ritmo cardiaco (HRV) simulada.

Vamos a simular una señal de variabilidad de ritmo cardiaco (HRV) con componentes espectrales de 0,03 Hz, 0,08 Hz, 0,25 Hz y 0,45 Hz de amplitud 1 ms más una señal continua de amplitud 10 ms, muestreada a fs = 3 Hz, de 300 segundos de duración:

```
>> fs=3;
>> t=0:1/fs:300;
>> x=sin(2*pi*0.03*t)+sin(2*pi*0.08*t)+sin(2*pi*0.25*t)+sin(2*pi*0.45*t)+10;
>> N= length(x); %N =  901
>> plot(t, x)
```



Hallemos la energía (Ener) y potencia promedio (Pm) de la señal *x* y de la misma quitándole la continua de amplitud 10 ms.

```
>> Ener=trapz(t,x.^2)  % E =  3.0600e+004
>> Ener=sum(x.^2)/fs  % E = 3.0633e+004
>> Pm=1/(t(end)-t(1))*trapz(t,x.^2)
Pm =  102.0000
>> Pm=1/N*sum(x.^2)
Pm = 101.9978
>> sigma2=var(x)
sigma2 =   2.0000
```

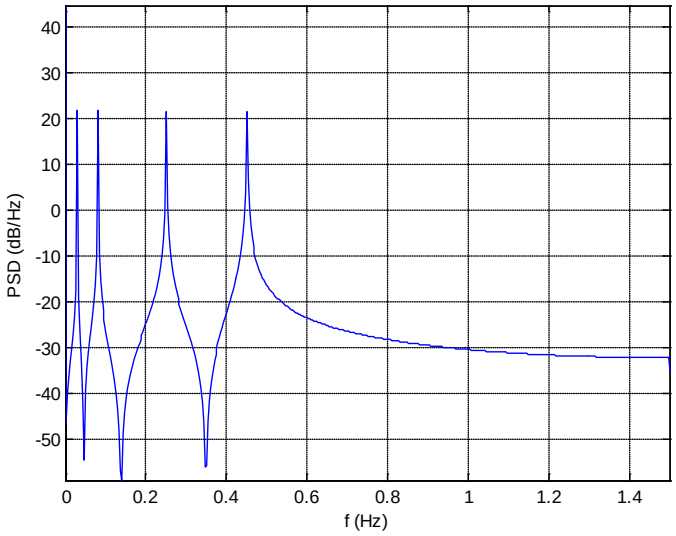
La potencia promedio es 102 ms^2, correspondiente a 0,5 ms^2 de cada una de las cuatro componente sinusoidales ($A^2/2$ cada componente) y 100 ms^2 de la continua (A^2). La varianza de la señal coincide con la potencia promedio de la señal al removerle la continua. Si removemos la componente continua: x0 = x – 10 o x0 = x – mean(x);

```
>> m=mean(x)
>> x0=x-m;
>> Pm=1/(t(end)-t(1))*trapz(t,x0.^2)
Pm =   2
>> Pm=1/N*sum(x0.^2)
Pm =  1.9978
>> sigma2=var(x0)
sigma2 =  2.0000
```

5.2.1) PSD y potencia promedio con técnicas no paramétricas

Con el periodograma sin utilizar ventana tenemos:

```
>> [Pxx,f]=periodogram(x,[],N,fs);
>> plot(f, 10*log10(Pxx)), grid
```



La energía total en el dominio frecuencial coincide con la hallada en el tiempo

```
>> Ener=sum(Pxx)
Ener = 3.0633e+004  ms^2-s (ms^2/Hz)
```

Esta es la energía correspondiente a la señal continua

```
>> Ener_dc=Pxx(1)
Ener_dc = 3.0033e+004
```

Mientras que la energía correspondiente a las sinusoides es

```
>> Ener_s=sum(Pxx(2:end))
Ener_s = 599.9997
```

Calculemos la Potencia promedio en el dominio frecuencial:

```
>> Pm=fs/N*sum(Pxx)
Pm = 101.9978
>> Pm=trapz(f(2:end),Pxx(2:end))
Pm =  1.9978
>> Pm= Pxx(1)*fs/N + trapz(f(2:end),Pxx(2:end))  % Pm_dc + Pm_sinusoides
Pm = 101.9978
```

De cada componente sinusoidal hallaremos la energía y la potencia media, evaluando distintas bandas de frecuencia donde se encuentran los componentes de este ejemplo.

Sinusoide de 0,03 Hz

```
>> ind_vlf=find(f>=0.001 & f <=0.04);
>> Ener_vlf=sum(Pxx(ind_vlf))
Ener_vlf =   150.40  (ms^2 · s)
>> Pm=trapz(f(ind_vlf),Pxx(ind_vlf))   % Potencia promedio (aprox. integral)
Pm = 0.5008
>> Pm=fs/N*sum(Pxx(ind_vlf))   % Potencia Promedio
Pm =  0.5008  (ms^2)
```

Sinusoide de 0,08 Hz

```
>> ind_lf=find(f>0.04 & f <=0.15);
>> Ener_lf=sum(Pxx(ind_lf))
```

```
Ener_lf  150.0164
>> Pm=trapz(f(ind_lf),Pxx(ind_lf))
Pm =  0.4995
>> Pm=fs/N*sum(Pxx(ind_lf))
Pm =  0.4995
```

Sinusoide de 0,25 Hz

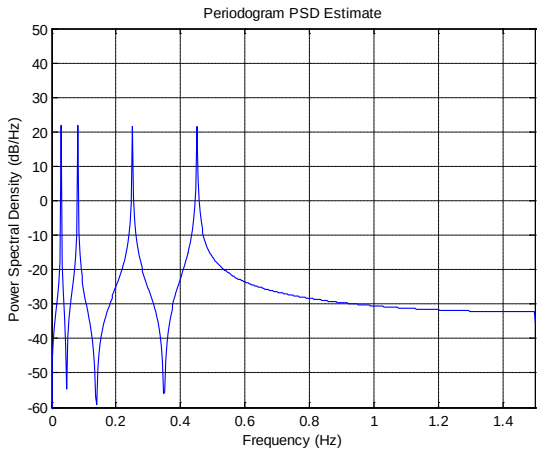
```
>> ind_hf=find(f>0.15 & f <=0.4);
>> Ener_hf=sum(Pxx(ind_hf))
Ener_hf = 150.0428
>> Pm=trapz(f(ind_hf),Pxx(ind_hf))
Pm = 0.4996
>> Pm=fs/N*sum(Pxx(ind_hf))
Pm =  0.4996
```

Sinusoide de 0,45 Hz

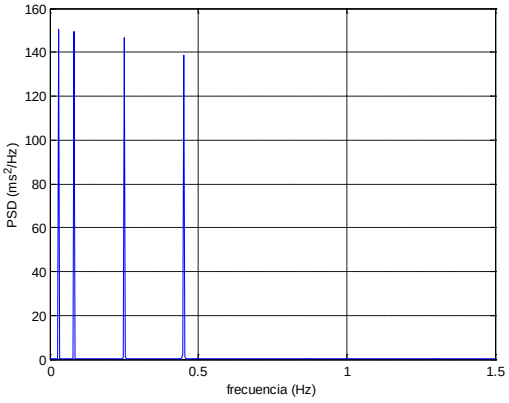
```
>> ind_vhf=find(f>0.4 & f <=1);
>> Ener_vhf=sum(Pxx(ind_vhf))
Ener_vhf = 149.4391
>> Pm=trapz(f(ind_vhf),Pxx(ind_vhf))
Pm =  0.4976
>> Pm=fs/N*sum(Pxx(ind_vhf))
Pm = 0.4976
```

Se puede observar que la potencia promedio de cada componte sinusoidal se aproxima a 0,5 ms^2 ($A^2/2$). Si trabajamos con la señal sin su valor medio (x0), tenemos

```
>> [Pxx,f]=periodogram(x0,[],N,fs);
>> periodogram(x0,[],N,fs);
```




```
>> plot(f, Pxx)
```



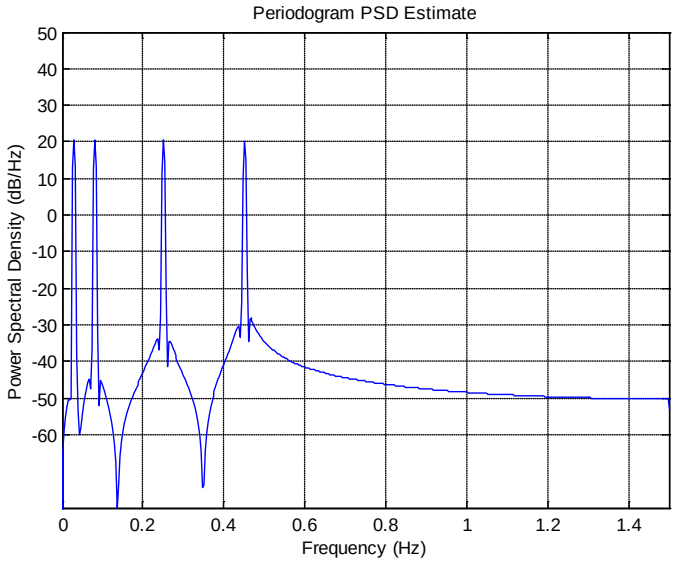
La energía y potencia promedio total coinciden con los de las componentes sinusoidales halladas anteriormente:

```
>> Ener=sum(Pxx)
Ener = 599.9997
>> Pm=trapz(f,Pxx)
Pm = 1.9978
>> Pm=fs/N*sum(Pxx)
Pm = 1.9978
```

Y la energía y potencia media de cada banda evaluada anteriormente produce también los mismos resultados.

Utilizando el periodograma donde la señal con valor medio 0 se enventana con una ventana de Hamming, tenemos

```
>> [Pxx,f]=periodogram(x0,hamming(N),N,fs);
>>  periodogram(x0,hamming(N),N,fs);
```




```
>> Ener=sum(Pxx)
Ener = 600.6559
>> Pm=fs/N*sum(Pxx)
Pm = 2.0000
```

La potencia promedio y energía de cada banda de frecuencias es:

Sinusoide de 0,03 Hz

```
>> ind_vlf=find(f>=0.001 & f <=0.04);
>> Ener=sum(Pxx(ind_vhf))
Ener = 150.1550
>> Pm=fs/N*sum(Pxx(ind_vlf))
Pm = 0.5000
```

Sinusoide de 0,08 Hz

```
>> ind_lf=find(f>0.04 & f <=0.15);
>> Ener_lf=sum(Pxx(ind_lf))
Ener_lf  150.1645
>> Pm=fs/N*sum(Pxx(ind_lf))
Pm =  0.5000
```

Sinusoide de 0,25 Hz

```
>> ind_hf=find(f>0.15 & f <=0.4);
>> Ener_hf=sum(Pxx(ind_hf))
Ener_hf = 150.1647
>> Pm=fs/N*sum(Pxx(ind_hf))
Pm =  0.5000
```

Sinusoide de 0,45 Hz

```
>> ind_vhf=find(f>0.4 & f <=1);
>> Ener_vhf=sum(Pxx(ind_vhf))
Ener_vhf = 150.1550
>> Pm=fs/N*sum(Pxx(ind_vhf))
Pm = 0.5000
```

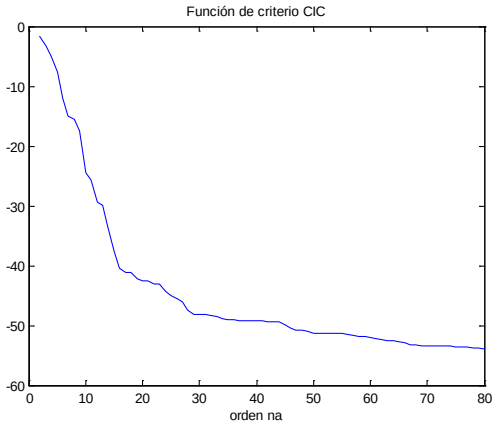
Como se puede observar, los valores de la potencia media son prácticamente igual a los teóricos aunque se usen las ventanas para suavizar el comienzo y final del segmento de señal a estimar su espectro. Lo mismo ocurre con ventanas de Hanning o Blackman-Harris.

5.2.2) PSD y potencia promedio con métodos paramétricos

a) Estructuras AR

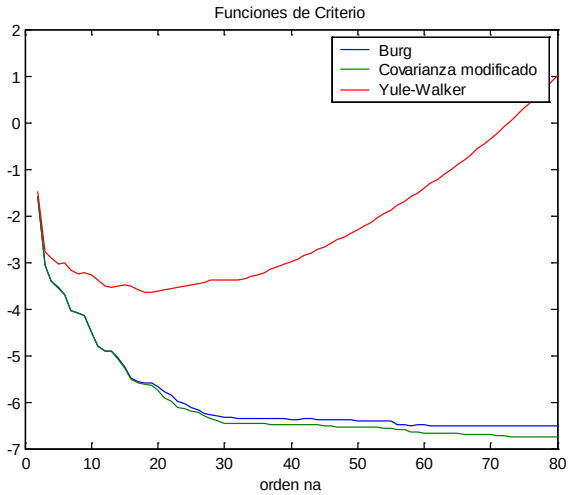
Busquemos el mejor orden de modelo con estructuras AR. Trabajaremos con la señal x0, es decir, con valor medio 0.

```
>> [na,V,Vn]=selarstruc(x0,'arburg',2:80,'cic');
>> plot(V(2,:),V(1,:))
```



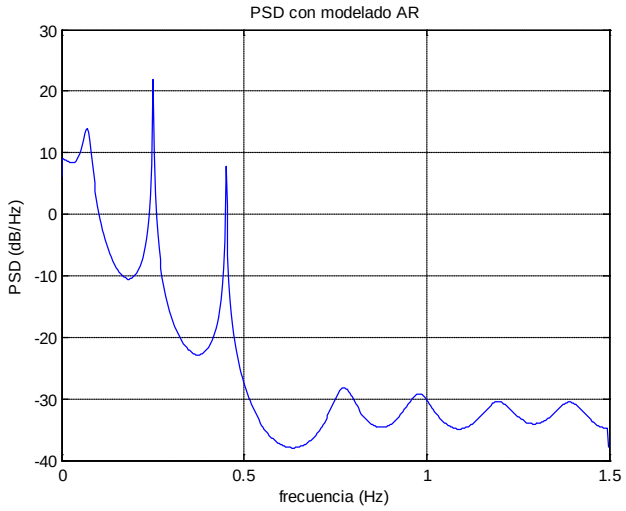
Le añadimos algo de ruido a la señal. Probaremos la función de criterio CIC con los métodos de Burg, covarianza modificado y Yule-Walker.

```
>> xd = x0 + 0.1*rand(size(t));
>> [nn,Vb,Vn]=selarstruc(xd,'arburg',2:80,'cic'); % nn =  63
>> [nn,Vc,Vn]=selarstruc(xd,'armcov',2:80,'cic'); % nn = 73
>> [nn,Vy,Vn]=selarstruc(xd,'aryule',2:80,'cic'); % nn = 19
>> plot(Vb(2,:),Vb(1,:),Vc(2,:),Vc(1,:),Vy(2,:),Vy(1,:))
>> legend('Burg','Covarianza modificado','Yule-Walker')
```



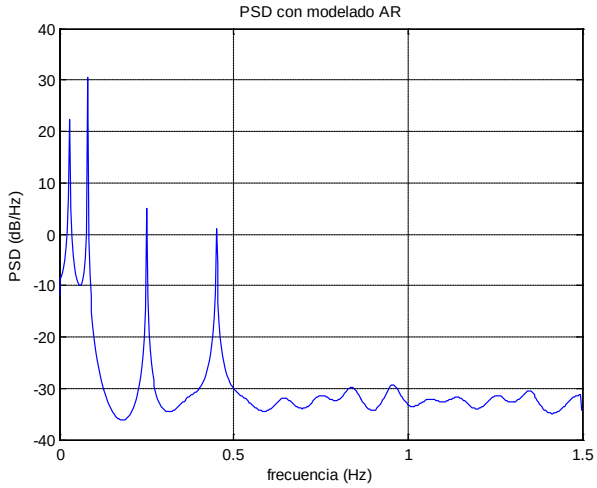
Con el método de estimación de parámetros de Burg escogemos primero un orden de modelo bajo, por ejemplo orden 15 si lo buscamos sobre el “codo” de la función de criterio CIC.

```
>> [A,e]=arburg(xd,15);
>> [Pxx,f]=armaspectra(1,A,e,N,fs);
>> plot(f,10*log10(Pxx)), grid % equivale a >> armaspectra(1,A,e,N,fs);
```



De la gráfica se observa que el modelo paramétrico de orden 15 de la señal no es capaz de discriminar bien los dos componentes sinusoidales de menor frecuencia, de 0,03 y 0,08 Hz. Probemos el orden de modelo 30, que corresponde al orden de la función de criterio donde cambia la pendiente y tiende a aplanarse.

```
>> [A,e]=arburg(xd,30);
>> [Pxx,f]=armaspectra(1,A,e,N,fs);
>> armaspectra(1,A,e,N,fs);
```



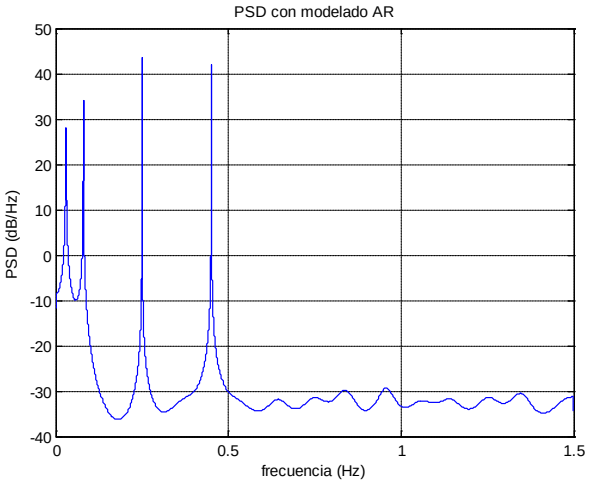
Al hallar la PSD con N puntos se consiguen los siguientes valores de potencia promedio:

```
>> Pm=fs/N*sum(Pxx)  %                  Pm = 4.4428
>> ind_vlf=find(f>=0.001 & f <= 0.04);
>> ind_lf=find(f>0.04 & f <= 0.15);
>> ind_hf=find(f>0.15 & f < 0.4);
>> ind_vhf=find(f>=0.4 & f < 1);
>> Pm=trapz(f(ind_vlf),Pxx(ind_vlf)); %   Pm =  0.6247
>> Pm=fs/N*sum(Pxx(ind_vlf))  %           Pm =  0.6247
>> Pm=fs/N*sum(Pxx(ind_lf))  %            Pm = 3.8004
>> Pm=fs/N*sum(Pxx(ind_hf))  %            Pm =  0.0110
>> Pm=fs/N*sum(Pxx(ind_vhf))  %           Pm = 0.0062
```

Si utilizamos un mayor número de puntos para hallar la respuesta frecuencial sobre el círculo unitario del plano z, tenemos

```
>> [Pxx,f]=armaspectra(1,A,e,300000,fs);
```

```
>> armaspectra(1,A,e,300000,fs);
```

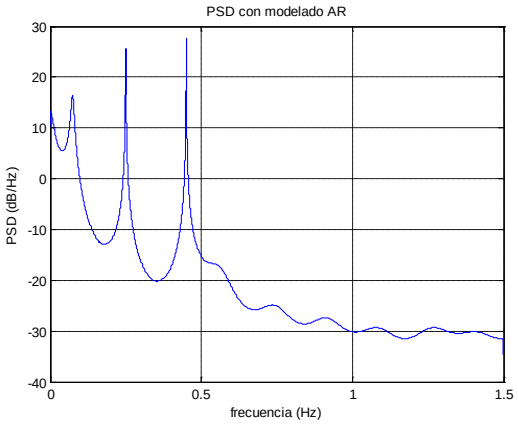


La potencia promedio de cada componente se aproxima mejor a los valores teóricos:

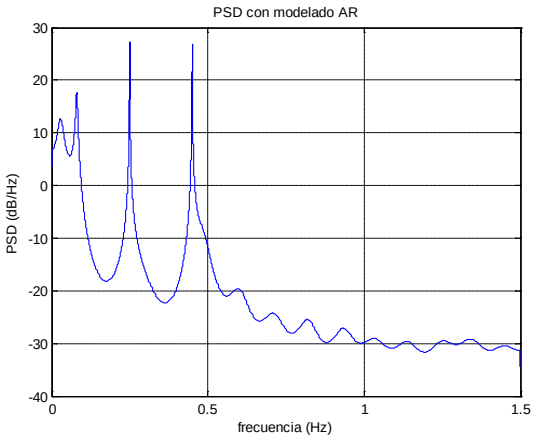
```
>> Pm=fs/300000*sum(Pxx)  %                Pm = 1.9148
>> ind_vlf=find(f>=0.001 & f <= 0.04);
>> ind_lf=find(f>0.04 & f <= 0.15);
>> ind_hf=find(f>0.15 & f < 0.4);
>> ind_vhf=find(f>=0.4 & f < 1);
>> Pm=fs/300000*sum(Pxx(ind_vlf))  %    Pm = 0.5066
>> Pm=fs/300000*sum(Pxx(ind_lf))   %    Pm = 0.4980
>> Pm=fs/300000*sum(Pxx(ind_hf))  %     Pm = 0.4133
>> Pm=fs/300000*sum(Pxx(ind_vhf)) %     Pm = 0.4965
```

Con los métodos de la covarianza y de la covarianza modificada se obtienen resultados similares al método de Burg. Mostraremos el método de Yule-Walker con el órden óptimo dado por CIC de na = 19 y luego con na =30.

```
>> [A,e]=aryule(xd,19);
>> [Pxx,f]=armaspectra(1,A,e,300000,fs);
>> armaspectra(1,A,e,300000,fs);
```




```
>> [A,e]=aryule(xd,30);
>> [Pxx2,f2]=armaspectra(1,A,e,300000,fs);
>> armaspectra(1,A,e,300000,fs);
```



Observamos que con un orden 19 no se puede discriminar los 2 componentes de baja frecuencia mientras que con un orden 30 se logran discriminar pero con mayor desparramamiento en comparación con el método de Burg. Los valores de Potencia promedio de los componentes individuales son:

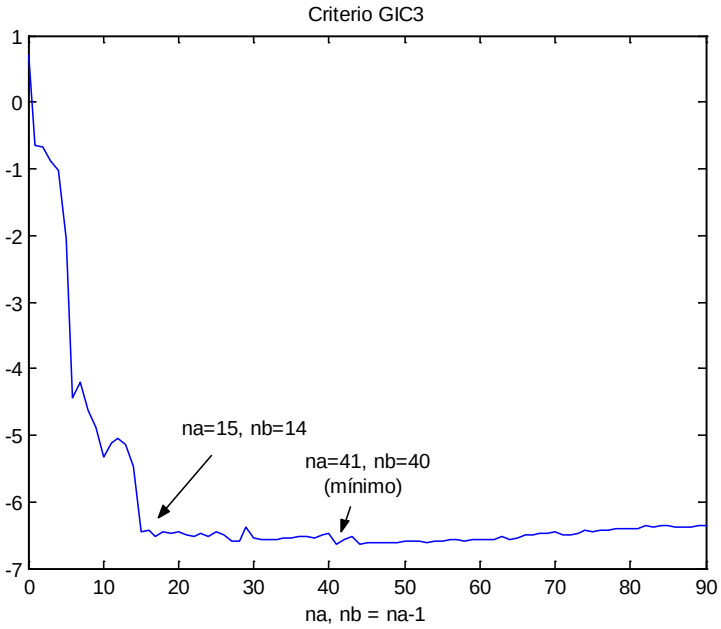
```
>> ind_vhf=find(f>=0.4 & f < 1);
>> ind_hf=find(f>0.15 & f < 0.4);
```

```
>> ind_lf=find(f>0.04 & f <= 0.15);
>> ind_vlf=find(f>=0.001 & f <= 0.04);
>> Pm=fs/300000*sum(Pxx(ind_vlf))  %          Pm = 0.3309
>> Pm=fs/300000*sum(Pxx(ind_lf))   %          Pm = 0.6538
>> Pm=fs/300000*sum(Pxx(ind_hf))   %          Pm =  0.4977
>> Pm=fs/300000*sum(Pxx(ind_vhf))  %          Pm =  0.4973
```

b) Estructuras ARMA

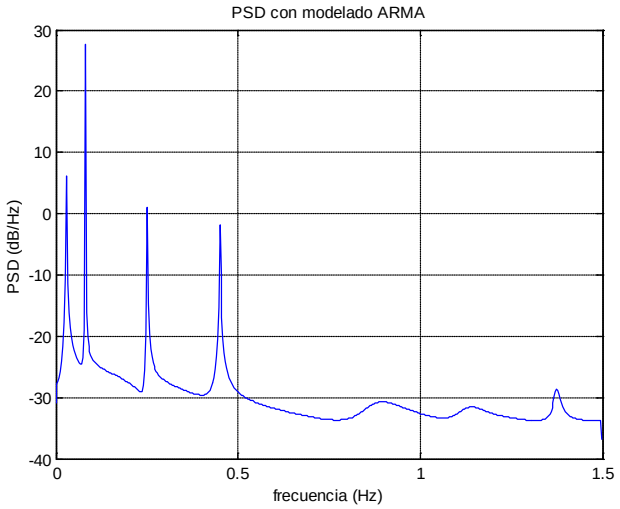
Con el toolbox ARMASA (método de Durbin-Broersen) buscaremos el mejor orden de modelo

```
>> [Aa,Ba,sel]=sig2arma(xd);  % Aa y Ba son de orden 41 y 40 respectivamente
>> plot(sel.cand_ar_order,sel.gic3)
```



Utilizaremos el orden na = 15, nb = 14

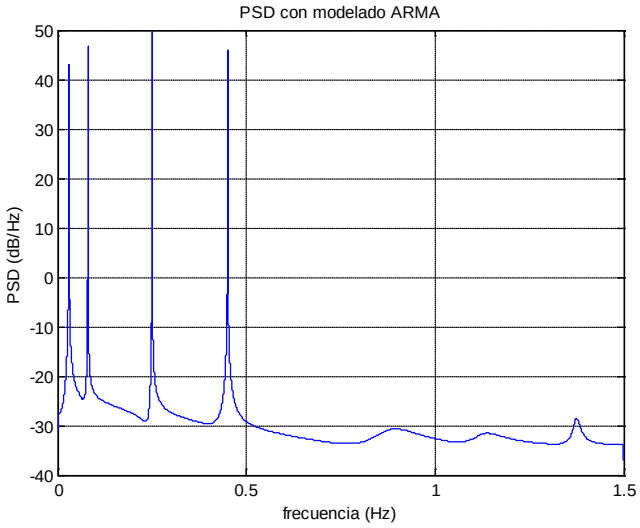
```
>> [Aa,Ba,sel]=sig2arma(xd,15);
>> e=sel.pe_est;
>> armaspectra(Ba,Aa,e,N,fs);
```



Hallando la PSD con mayor número de puntos sobre el círculo del plano z

```
>> [Pxx,f]=armaspectra(Ba,Aa,e,300000,fs);
```

```
>> armaspectra(Ba,Aa,e,300000,fs);
```

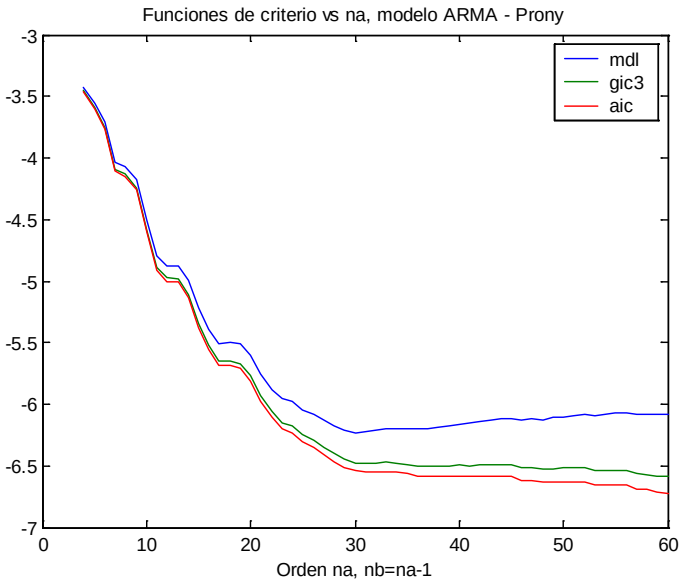


La potencia promedio estimada de cada componente es

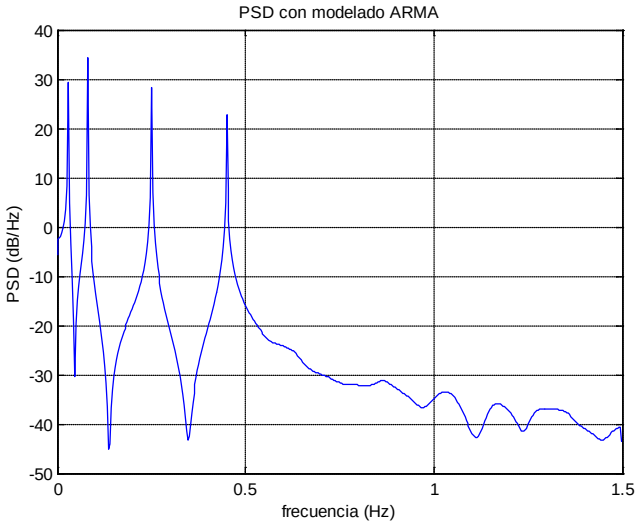
```
>> [Pxx,f]=armaspectra(Ba,Aa,e,300000,fs);
>> ind_vlf=find(f>=0.001 & f <= 0.04);
>> ind_lf=find(f>0.04 & f <= 0.15);
>> ind_hf=find(f>0.15 & f < 0.4);
>> ind_vhf=find(f>=0.4 & f < 1);
>> Pm=fs/300000*sum(Pxx)     %          Pm =  2.5926
>> Pm=fs/300000*sum(Pxx(ind_vlf)) %     Pm =  0.4961
>> Pm=fs/300000*sum(Pxx(ind_lf))  %     Pm =  0.5366
>> Pm=fs/300000*sum(Pxx(ind_hf))  %     Pm =  0.9795
>> Pm=fs/300000*sum(Pxx(ind_vhf)) %     Pm =  0.5801
```

Utilizando el método de Prony:

```
>> [nn,V,Vn] = selarmastruc2(xd,'prony_e',4:60,'mdl'); % nn= [30 29]
>> [nn,Vai,Vn] = selarmastruc2(xd,'prony_e',4:60,'aic'); % nn= [60 59]
>> [nn,Va,Vn] = selarmastruc2(xd,'prony_e',4:60,'gic3'); % nn = [60 59]
>> plot(V(2,:),V(1,:),Va(2,:),Va(1,:),Vai(2,:),Vai(1,:))
>> legend('mdl','gic3','aic')
>> xlabel('Orden na, nb=na-1')
>> title('Funciones de criterio vs na, modelo ARMA - Prony')
```

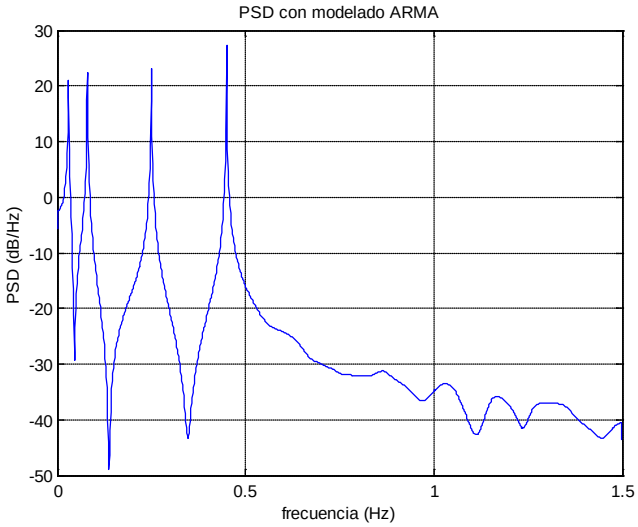


```
>> [B,A,e]=prony_e(xd,29,30);
>> [Pxx,f]=armaspectra(B,A,e,N,fs);
>>                     armaspectra(B,A,e,N,fs);
```




```
>> ind_vhf=find(f>=0.4 & f < 1);
>> ind_hf=find(f>0.15 & f < 0.4);
>> ind_lf=find(f>0.04 & f <= 0.15);
>> ind_vlf=find(f>=0.001 & f <= 0.04);
>> Pm=trapz(f,Pxx)  %                  Pm =  14.75
>> Pm=fs/N*sum(Pxx) %                  Pm =  14.75
>> Pm=fs/N*sum(Pxx(ind_vlf)) %         Pm =   2.90
>> Pm=fs/N*sum(Pxx(ind_lf))  %         Pm =   8.92
>> Pm=fs/N*sum(Pxx(ind_hf))  %         Pm =   2.25
>> Pm=fs/N*sum(Pxx(ind_vhf)) %         Pm =   0.68

>> [Pxx,f]=armaspectra(B,A,e,2048,fs);
>> armaspectra(B,A,e,2048,fs);
```

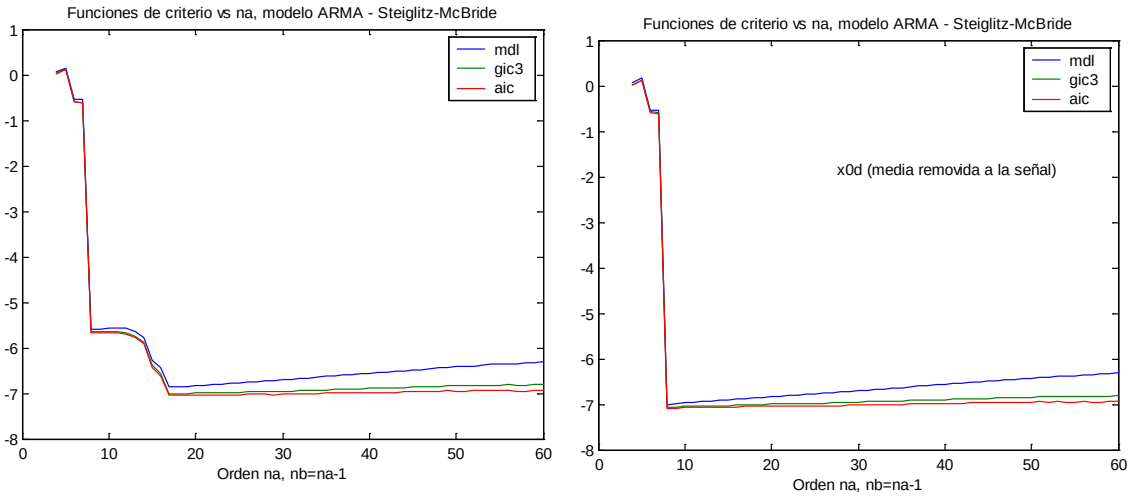



```
>> Pm=trapz(f,Pxx) %                   Pm =   2.10
>> ind_vlf=find(f>=0.001 & f <= 0.04);
>> ind_lf=find(f>0.04 & f <= 0.15);
>> ind_hf=find(f>0.15 & f < 0.4);
>> ind_vhf=find(f>=0.4 & f < 1);
>> Pm=trapz(f(ind_hf),Pxx(ind_hf)) %   Pm =   0.43
>> Pm=trapz(f(ind_lf),Pxx(ind_lf))  %  Pm =   0.40
```

```
>> Pm=trapz(f(ind_vlf),Pxx(ind_vlf)) %     Pm =   0.35
>> Pm=trapz(f(ind_vhf),Pxx(ind_vhf)) %    Pm =   0.92
```

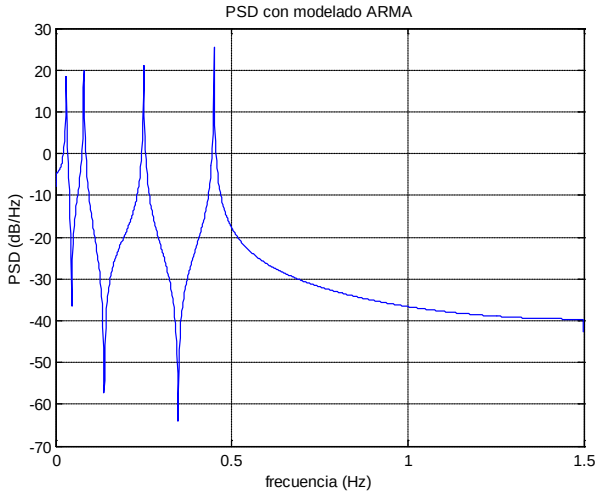
Probemos el método de Steiglitz-McBride con la señal xd (con algo de ruido) y la misma removiéndole el valor medio x0d =xd-mean(xd);

```
                    >> [nn,V,Vn] = selarmastruc2(x0d,'stmcb_e',4:60,'mdl'); % nn = [ 8  7]
>> [nn,V,Vn] = selarmastruc2(xd,'stmcb_e',4:60,'mdl');   % nn = [17 16]
                    >> [nn,Vai,Vn] = selarmastruc2(x0d,'stmcb_e',4:60,'aic'); % nn = [8  7]
>> [nn,Vai,Vn] = selarmastruc2(xd,'stmcb_e',4:60,'aic');  % nn = [19 18]
                    >> [nn,Va,Vn] = selarmastruc2(x0d,'stmcb_e',4:60,'gic3');  % nn = [8  7]
>> [nn,Va,Vn] = selarmastruc2(xd,'stmcb_e',4:60,'gic3'); % nn = [19 18]
                >> plot(V(2,:),V(1,:),Va(2,:),Va(1,:),Vai(2,:),Vai(1,:))
```



Usando la señal con media cero, x0d:

```
>> [B,A,e]=stmcb_e(x0d,7,8);
>> [Pxx,f]=armaspectra(B,A,e,2048,fs);
>> armaspectra(B,A,e,2048,fs);
```

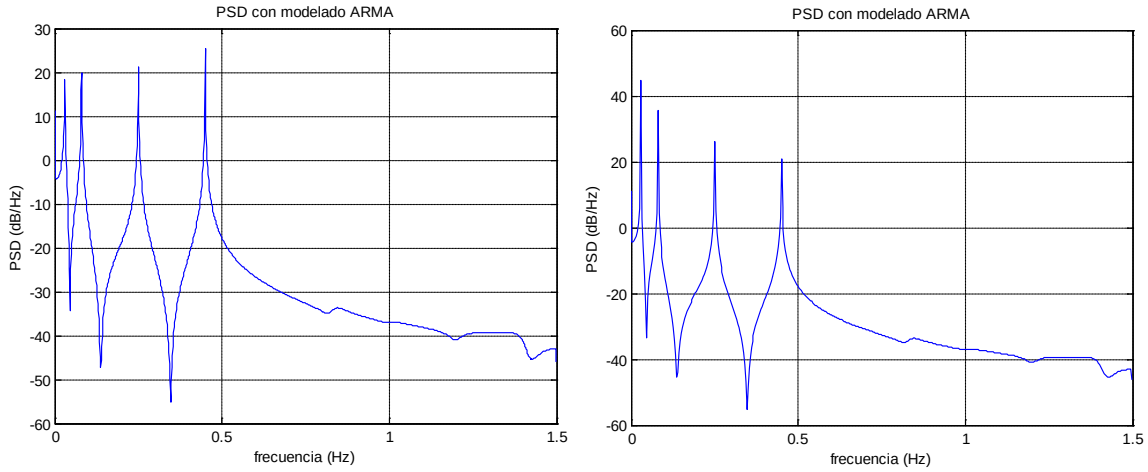



```
>> ind_vlf=find(f>=0.001 & f <= 0.04);
>> ind_lf=find(f>0.04 & f <= 0.15);
>> ind_hf=find(f>0.15 & f < 0.4);
>> ind_vhf=find(f>=0.4 & f < 1);
>> Pm=trapz(f(ind_hf),Pxx(ind_hf)) %     Pm =   0.21
>> Pm=trapz(f(ind_lf),Pxx(ind_lf))  %     Pm =   0.23
>> Pm=trapz(f(ind_vlf),Pxx(ind_vlf)) %     Pm =   0.27
```

>> Pm=trapz(f(ind_vhf),Pxx(ind_vhf)) % Pm = 0.59

Con la señal xd:

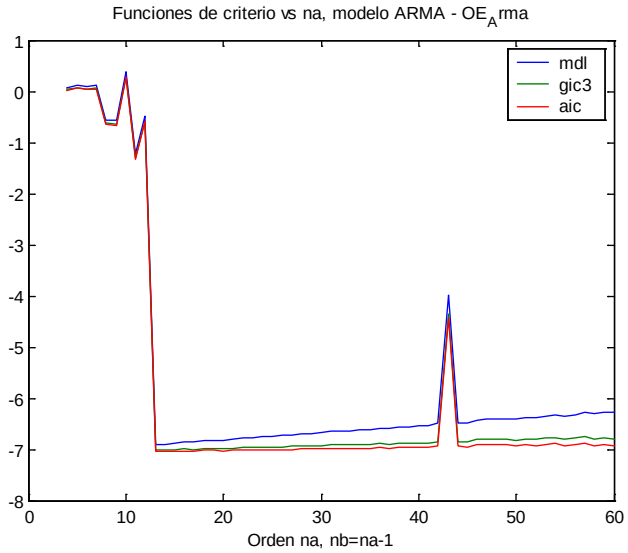
[B,A,e]=stmcb_e(xd,16,17);
>> [Pxx,f]=armaspectra(B,A,e,2048,fs);
>> armaspectra(B,A,e,2048,fs); armaspectra(B,A,e,N,fs);



>> Pm=trapz(f,Pxx) % Pm = 1.3276
>> Pm=fs/2048*sum(Pxx(ind_vhf)) % Pm = 0.5938
>> Pm=fs/2048*sum(Pxx(ind_hf)) % Pm = 0.2757
>> Pm=fs/2048*sum(Pxx(ind_lf)) % Pm = 0.2333
>> Pm=fs/2048*sum(Pxx(ind_vlf)) % Pm = 0.2152

Usaremos también los métodos output error arma y armax arma:

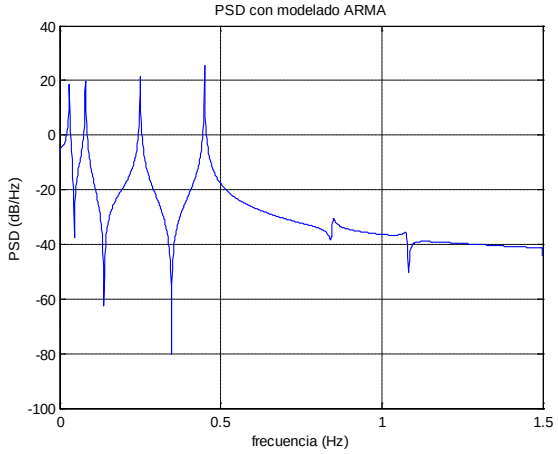
>>[nn,Vai0,Vn] = selarmastruc2(x0d,'oe_arma',4:60,'aic');% nn = [14 13] >>[nn,Vai,Vn] = selarmastruc2(xd,'oe_arma',4:60,'aic'); % nn=[20 19]
>>[nn,V0,Vn] = selarmastruc2(x0d,'oe_arma',4:60,'mdl'); % nn = [13 12] % mdl --> nn = [20 19]
>>[nn,Va0,Vn] = selarmastruc2(x0d,'oe_arma',4:60,'gic3'); % nn = [14 13] % gic3 --> nn = [20 19]
>> plot(V0(2,:),V0(1,:),Va0(2,:),Va0(1,:),Vai0(2,:),Vai0(1,:))



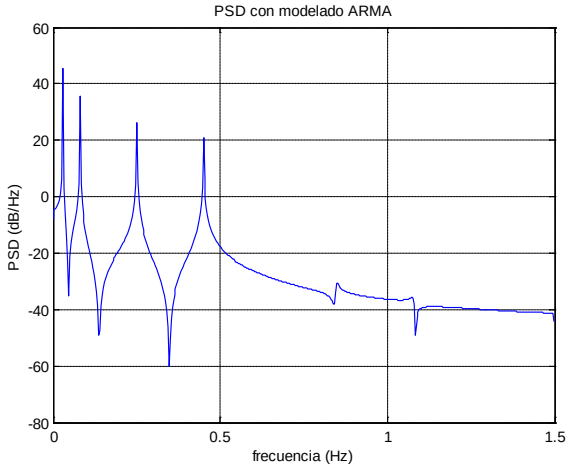
Con la señal de media 0:

>> [B,A,e]=oe_arma(x0d,12,13);
>>[Pxx,f]=armaspectra(B,A,e,2048,fs);

>>armaspectra(B,A,e,2048,fs);

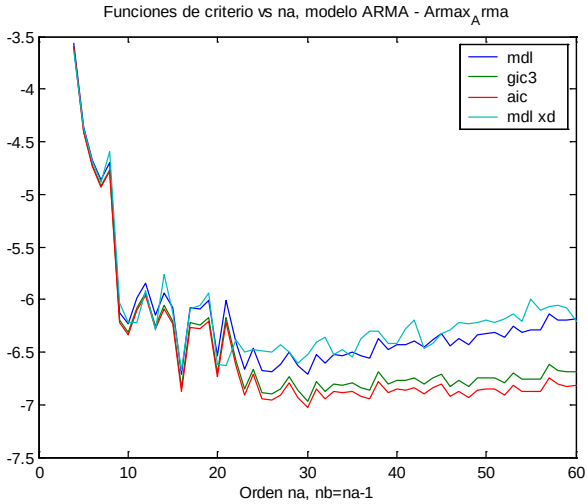


>>armaspectra(B,A,e,N,fs);

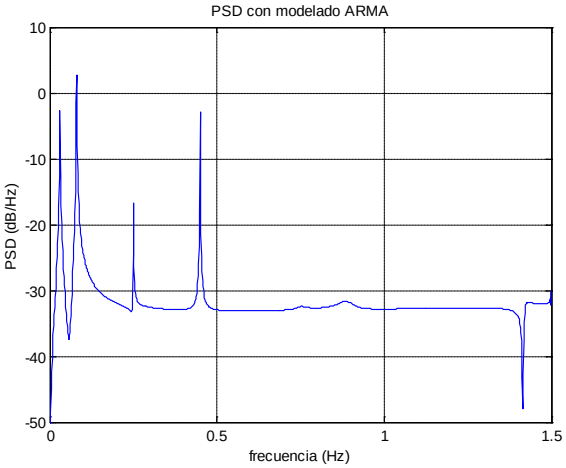


>> Pm=trapz(f,Pxx) % Pm = 1.346
>> Pm=fs/2048*sum(Pxx(ind_vhf)) % Pm = 0.218
>> Pm=fs/2048*sum(Pxx(ind_hf)) % Pm = 0.239
>> Pm=fs/2048*sum(Pxx(ind_lf)) % Pm = 0.282
>> Pm=fs/2048*sum(Pxx(ind_vlf)) % Pm = 0.607

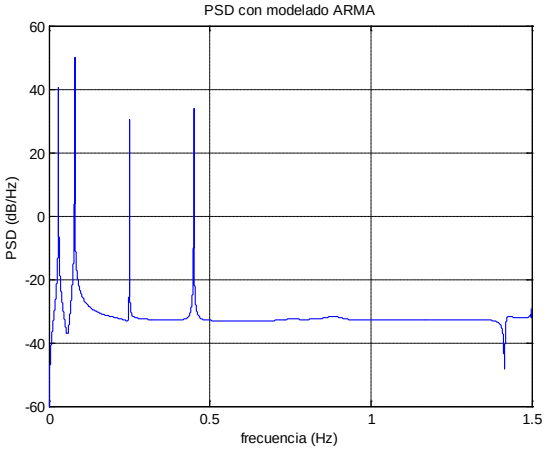
>>[nn,V2a0,Vn] = selarmastruc2(x0d,'armax_arma',4:60,'gic3'); %nn =[30 29]
>>[nn,V2ai0,Vn] = selarmastruc2(x0d,'armax_arma',4:60,'aic'); %nn = [30 29]
>>[nn,V20,Vn] = selarmastruc2(x0d,'armax_arma',4:60,'mdl'); %nn = [30 29]
>>[nn,V2,Vn] = selarmastruc2(xd,'armax_arma',4:60,'mdl'); % nn =[16 15]
>> plot(V20(2,:),V20(1,:),V2a0(2,:),V2a0(1,:),V2ai0(2,:),V2ai0(1,:))



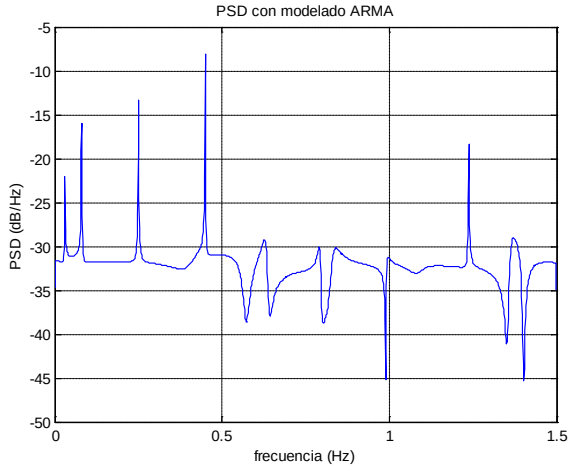
[B,A,e]=armax_arma(x0d,15,16);
>> armaspectra(B,A,e,2048,fs);



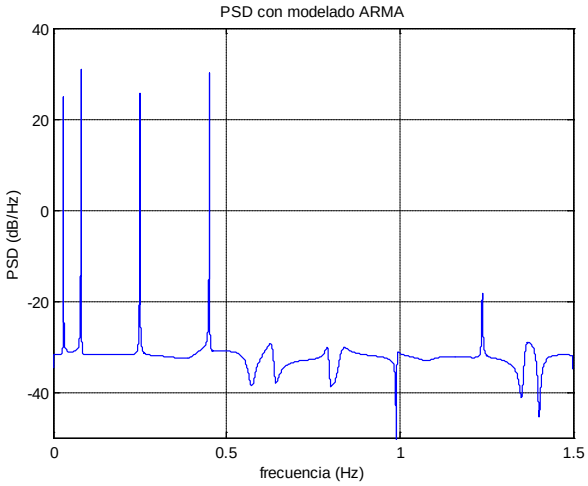
>> armaspectra(B,A,e,300000,fs);



```
>> [B,A,e]=armax_arma(x0d,29,30);
>> armaspectra(B,A,e,2048,fs);
```




```
>> armaspectra(B,A,e,300000,fs);
```

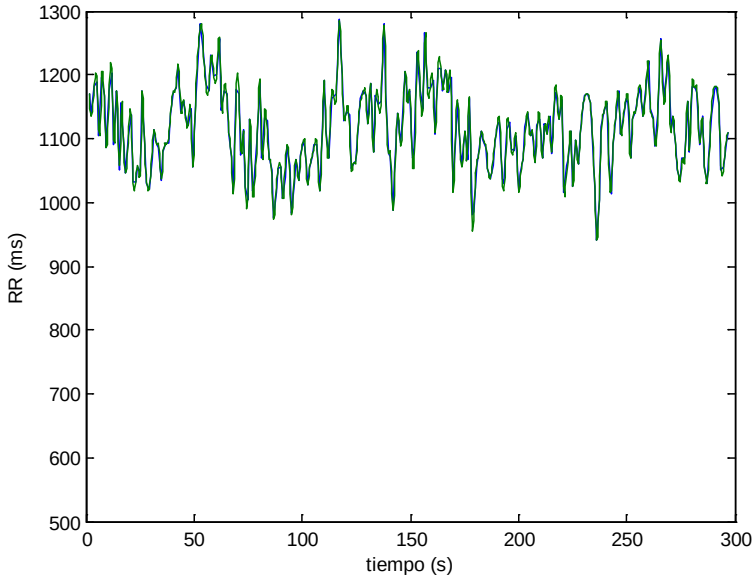



```
>> [Pxx,f]=armaspectra(B,A,e,300000,fs);
>> Pm=trapz(f,Pxx) %                    Pm =   0.0430
>> Pm=fs/300000*sum(Pxx(ind_vlf))  %    Pm =   0.0041
>> Pm=fs/300000*sum(Pxx(ind_lf))  %     Pm =   0.0153
>> Pm=fs/300000*sum(Pxx(ind_hf))  %     Pm =   0.0087
>> Pm=fs/300000*sum(Pxx(ind_vhf)) %     Pm =   0.0145
```

5.3) Señal de variabilidad del ritmo cardiaco (HRV) real remuestreada a 3 Hz.

Tenemos ahora una señal real RR de 300 segundos de duración que previamente interpolaremos con splines y re-muestrearemos a una frecuencia fs = 3Hz. Para ello usaremos una señal RR (RR66b) que contiene en la 1ra columna los valores de tiempo y en la 2da los RR en milisegundos (ms).

```
>> load RR66b
>> fs=3;
>> t=(RR66b(1,1):1/fs:RR66b(end,1))';
>> x=spline(RR66b(:,1),RR66b(:,2),t);
>> N=length(x) ;       %   N =  887
>> plot(RR66b(:,1),RR66b(:,2),t,x)
```



Calculamos la energía (Ener) y potencia promedio (Pm) de la señal *x* y de la misma quitándole la continua (*x0*)

```
>> Ener=trapz(t,x.^2) ;  %          Ener =  3.7010e+008  ms^2-s (ms^2/Hz)
>> Ener=sum(x.^2)/fs;  %            Ener =  3.7054e+008  ms^2-s (ms^2/Hz)
>> Pm=1/(t(end)-t(1))*trapz(t,x.^2) ;  %     Pm =  1.2532e+006  ms^2
>> Pm=1/N*sum(x.^2);  %                      Pm =  1.2532e+006  ms^2
>> sigma2=var(x);    %              sigma2 =  3831.6
```

Con *x0*, donde se ha removido la continua, tenemos

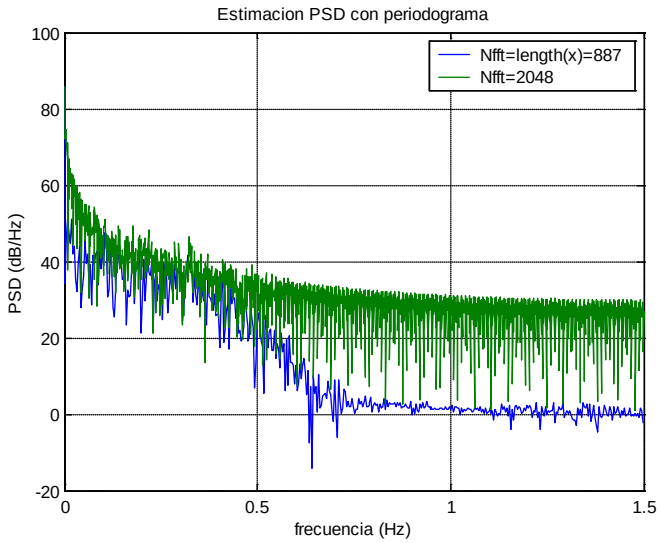
```
>> x0 = x – mean(x);
>> Pm=1/N*sum(x0.^2);   %                 Pm =  3827.2  ms^2
>> Pm=1/(t(end)-t(1))*trapz(t,x0.^2) ;  %    Pm =  3830.0  ms^2
```

5.3.1) PSD y potencia promedio con técnicas no paramétricas

Con el periodograma sin utilizar ventana tenemos:

```
>> [Pxx,f]=periodogram(x,[],N,fs);  [Pxx2,f2]=periodogram(x,[],2048,fs);
>> plot(f,10*log10(Pxx),f2,10*log10(Pxx2)),grid
>> xlabel('frecuencia (Hz)'),  ylabel('PSD (dB/Hz)'), title ('Estimacion PSD con periodograma')
```

```
>> legend('Nfft=length(x)=887','Nfft=2048')
```



La energía total en el dominio frecuencial coincide con la hallada en el tiempo

```
>> Ener=sum(Pxx);  %     Ener =  3.7054e+008  ms^2-s (ms^2/Hz)
```

Calculemos la Potencia promedio en el dominio frecuencial:

```
>> Pm_tot =fs/N*sum(Pxx);                              % Pm_tot =  1.2532e+006  ms^2
>> Pm_tot= Pxx(1)*fs/N + trapz(f(2:end),Pxx(2:end));   % Pm_tot =  1.2494e+006 + 3822.9 = 1.2532e+006
```

La potencia promedio de la continua correspondería a la media del periodo cardíaco (inverso de la frecuencia cardiaca, en segundos). Esta media y su potencia son:

```
>> m=mean(x);     %        m =  1117.8 ms
>> Pm_0= Pxx(1)*fs/N  %           Pm_0 =  1.2494e+006  ms^2 (equivale a m^2)
```

La potencia promedio relacionada al resto de la señal (distinto de la media) corresponde a:

```
>> Pm= trapz(f(2:end),Pxx(2:end));  %       Pm =  3822.9  ms^2
>> Pm=fs/N*sum(Pxx(2:end));     %          Pm =  3827.2  ms^2
```

Calcularemos a continuación la Potencia promedio en las bandas de frecuencia recomendadas [Tas96] para el análisis de la HRV. Usaremos: VLF_dc = [0 – 0,04] Hz, VLF = [0,003 – 0,04] Hz, LF = [0,04 – 0,15] Hz y HF = [0,15 – 0,4] Hz. Hallaremos también los índices: LFnorm = LF/(Pm_total-VLF_dc)*100, HF = HF/(Pm_total-VLF_dc)*100 y LF/HF. Le añadiremos el índice VHF = [0.4 – 1] Hz.

```
>> ind_vlf_dc=find(f <= 0.04);
>> ind_vlf=find(f>=0.003 & f <= 0.04);
>> ind_lf=find(f>0.04 & f <= 0.15);
>> ind_hf=find(f>0.15 & f < 0.4);
>> ind_vhf=find(f>=0.4 & f < 1);
>> Pm_vlf_dc=fs/N*sum(Pxx(ind_vlf_dc));  %       1.2509e+006  =  VLF_dc
>> Pm_vlf=fs/N*sum(Pxx(ind_vlf));  %             1484.9   = VLF
>> Pm=trapz(f(ind_vlf),Pxx(ind_vlf)); %          1463.9    (equivalente)
>> Pm_lf=fs/N*sum(Pxx(ind_lf));  %               1272.6    = LF
>> Pm=trapz(f(ind_lf),Pxx(ind_lf)); %            1222.8    (equivalente)
>> Pm_hf=fs/N*sum(Pxx(ind_hf)); %                995.48  = HF
>> Pm=trapz(f(ind_hf),Pxx(ind_hf)); %            987.85  (equivalente)
>> Pm_vhf=fs/N*sum(Pxx(ind_vhf))  %               73.60
```

```
>> LFnorm=Pm_lf/(Pm_tot-Pm_vlf_dc)*100 %       LFnorm =   54.33
>> HFnorm=Pm_hf/(Pm_tot-Pm_vlf_dc)*100 %       HFnorm =   42.50
>> LFdivHF=Pm_lf/Pm_hf     % equivale a LFnorm/HFnorm   LFdivHF =    1.28
```

Calculemos de nuevo la potencia media utilizando un mayor número de puntos *Nfft* = 2048. Esta señal con un valor continuo alto, sin envantanar, produce unos escalones que distorsionan las altas frecuencias. Se comparan los resultados con *Nfft = N = length(x)* = 887.

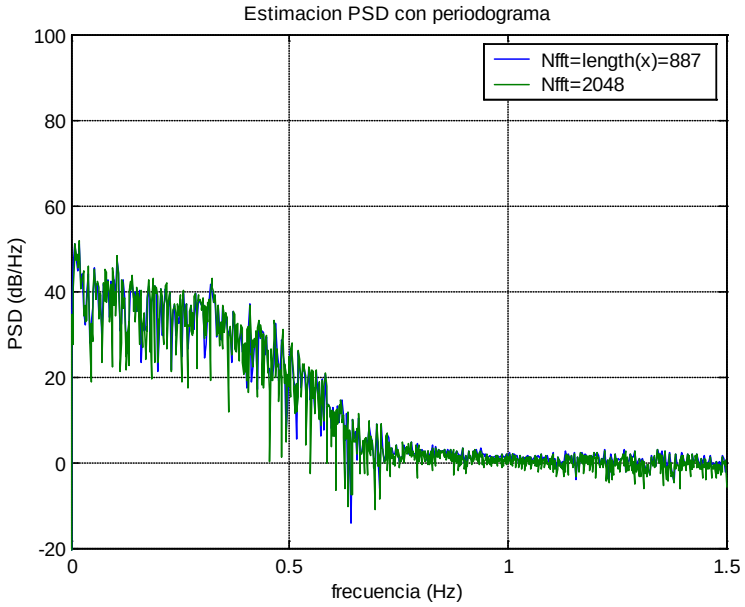
```
[Pxx,f]=periodogram(x,[],2048,fs);
```

Señal *x* sin enventanar

	N= length(x)=887		Nfft = 2048	
Bandas (Hz)	Pot med Pm=fs/N*sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft*sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)
Total	1,253E+06	6,285E+05	1,253E+06	9,827E+05
VLF_dc = [0 – 0,04]	1,251E+06	6,262E+05	1,241E+06	9,696E+05
VLF = [0,003 – 0,04	1484,92	1463,92	117766,44	96036,15
LF = [0,04 – 0,15]	1272,63	1222,84	8628,05	8602,44
HF = [0,15 – 0,4]]	995,48	987,85	2881,68	2863,70
VHF = [0.4 – 1]	73,60	73,51	859,10	855,62
LFnorm	54,33	51,84	68,25	65,64
HFnorm	42,50	41,88	22,80	21,85
LF/HF	1,278	1,238	2,994	3,004

Si trabajamos con la señal sin su valor medio (x0), los escalones de inicio y final son menores, disminuyendo los errores de altas frecuencias, sin embargo, la señal deberá luego ser enventanada:

```
>> [Pxx,f]=periodogram(x0,[],N,fs); [Pxx2,f2]=periodogram(x0,[],2048,fs);
>> plot(f,10*log10(Pxx),f2,10*log10(Pxx2)),grid,   axis([0 1.5 -20 100])
>> legend('Nfft=length(x)=887','Nfft=2048')
```



La potencia promedio total equivale a la hallada anteriormente en el dominio del tiempo.

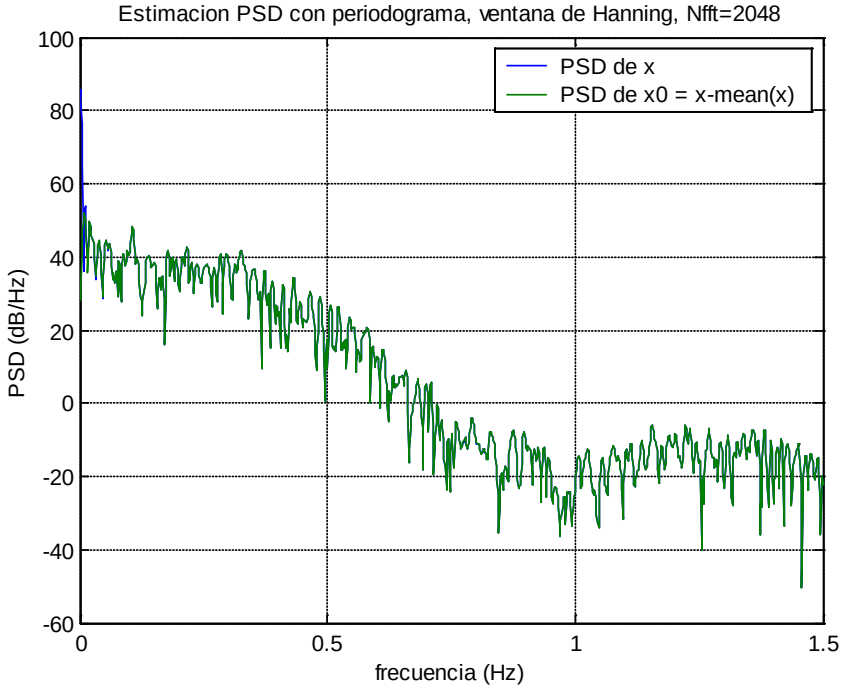
>> Pm_tot=fs/N*sum(Pxx); % Pm = 3827.2

Señal *x* sin enventanar, removida su media: *x*0

	N= length(x)=887		Nfft = 2048	
Bandas (Hz)	Pot med Pm=fs/N*sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft*sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)
Total	3827,24	3827,23	3827,24	3827,24
VLF_dc = [0 – 0,04]	1484,92	1468,31	1556,96	1529,70
VLF = [0,003 – 0,04	1484,92	1463,92	1551,41	1523,72
LF = [0,04 – 0,15]	1272,63	1222,84	1196,54	1177,82
HF = [0,15 – 0,4]]	995,48	987,85	1000,86	991,56
VHF = [0.4 – 1]	73,60	73,51	72,39	72,30
LFnorm	54,33	51,84	52,70	51,26
HFnorm	42,50	41,88	44,09	43,16
LF/HF	1,278	1,238	1,196	1,188

Utilizando el periodograma donde la señal *x* original y con valor medio *x*0 se enventana con una ventana de Hanning, tenemos

```
>> [Pxx,f]=periodogram(x,hann(N),2048,fs);   % Señal x original
>> [Pxx2,f2]=periodogram(x0,hann(N),2048,fs);  % Señal x removido su valor medio: x0
>> plot(f,10*log10(Pxx),f2,10*log10(Pxx2)),grid
>> xlabel('frecuencia (Hz)'),  ylabel('PSD (dB/Hz)')
>> legend('PSD de x','PSD de x0 = x-mean(x)')
>> title ('Estimacion PSD con periodograma, ventana de Hanning, Nfft=2048')
```



Señal *x* original enventanada con ventana de Hanning

	N= length(x)=887		Nfft = 2048		Nfft = 100000	
Bandas (Hz)	Pot med Pm=fs/N* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)
Total	1,254E+06	8,395E+05	1,254E+06	1,075E+06	1,254E+06	1,250E+06
VLF_dc = [0 – 0,04]	1,252E+06	8,370E+05	1,252E+06	1,072E+06	1,252E+06	1,248E+06
VLF = [0,003 – 0,04	4,224E+05	2,120E+05	6,797E+04	3,696E+04	1,656E+05	1,630E+05
LF = [0,04 – 0,15]	1274,17	1222,74	1233,53	1212,60	1240,01	1239,51
HF = [0,15 – 0,4]]	1111,27	1101,18	1114,14	1108,79	1114,96	1114,85
VHF = [0.4 – 1]	57,85	57,34	57,89	57,72	58,03	58,03
LFnorm	52,15	49,24	51,28	50,00	51,39	51,36
HFnorm	45,48	44,35	46,31	45,72	46,21	46,19
LF/HF	1,147	1,110	1,107	1,094	1,112	1,112

Podemos observar de la tabla anterior que al utilizar un número elevado de puntos para hallar la FFT (100000) los resultados utilizando la expresión de la sumatoria (Pm=fs/Nfft*sum(Px)) son similares a los hallados con *Nfft* = 2048 puntos, sin embargo los cálculos obtenidos con la aproximación de la integral se acercan más a los de la fórmulan discreta.

Si agregamos el índice VHF normalizado (*Nfft* = 2048):

```
>> VHFnorm=Pm_vhf/(Pm_tot-Pm_vlf_dc)*100  % VHFnorm =    2.406
```

Señal *x* removida su media: x0, enventanada con ventana de Hanning

	N= length(x)=887		Nfft = 2048		Nfft = 100000	
Bandas (Hz)	Pot med Pm=fs/N* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)
Total	4090,92	4089,80	4090,92	4090,44	4090,92	4090,91
VLF_dc = [0 – 0,04]	1647,47	1606,72	1686,87	1665,74	1679,14	1678,73
VLF = [0,003 – 0,04	1645,25	1601,02	1683,02	1653,67	1676,46	1676,04
LF = [0,04 – 0,15]	1274,30	1222,78	1232,00	1211,37	1238,77	1238,26
HF = [0,15 – 0,4]]	1111,26	1101,18	1114,14	1108,79	1114,96	1114,84
VHF = [0.4 – 1]	57,85	57,34	57,89	57,72	58,03	58,03
LFnorm	52,15	49,24	51,25	49,96	51,36	51,33
HFnorm	45,48	44,35	46,34	45,73	46,23	46,22
LF/HF	1,147	1,110	1,106	1,093	1,111	1,111

Analizando los resultados, podemos sugerir que se puede usar un número de puntos Nfft = 2048 para esta longitud de señal RR, con una ventana de Hanning. En el caso de los índices LF, HF, VHF y sus valores relacionados, es similar usar la señal original o removiéndole el valor medio. Vamos a calcular también la PSD y las potencias medias de la señal x eliminando su tendencia de lineal (primer orden), es decir, restándole la recta que la interpola: *xd*. El valor medio de *xd* es también cero. Añadiremos, además, el índice VHFnorm=Pm_vhf/(Pm_tot-Pm_vlf_dc)*100.

```
>> xd=detrend(x);
>> [Pxx,f]=periodogram(xd,hann(N),N,fs);
>> [Pxx2,f2]=periodogram(xd,hann(N),2048,fs);
>> [Pxx3,f3]=periodogram(xd,hann(N),100000,fs);
```

Señal *x* removida su tendencia lineal: xd, enventanada con ventana de Hanning

	N= length(x)=887		Nfft = 2048		Nfft = 100000	
Bandas (Hz)	Pot med Pm=fs/N* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)
Total	4089,32	4088,20	4089,32	4088,83	4089,32	4089,31
VLF_dc = [0 – 0,04]	1645,87	1605,11	1685,26	1664,13	1677,54	1677,12
VLF = [0,003 – 0,04	1643,64	1600,19	1682,34	1653,26	1675,47	1675,05
LF = [0,04 – 0,15]	1274,30	1222,78	1232,00	1211,37	1238,77	1238,26
HF = [0,15 – 0,4]]	1111,26	1101,18	1114,14	1108,79	1114,96	1114,84
VHF = [0.4 – 1]	57,85	57,34	57,89	57,72	58,03	58,03
LFnorm	52,15	49,24	51,25	49,96	51,36	51,33
HFnorm	45,48	44,35	46,34	45,73	46,23	46,22
VHFnorm	2,37	2,31	2,41	2,38	2,41	2,41
LF/HF	1,15	1,11	1,11	1,09	1,11	1,11

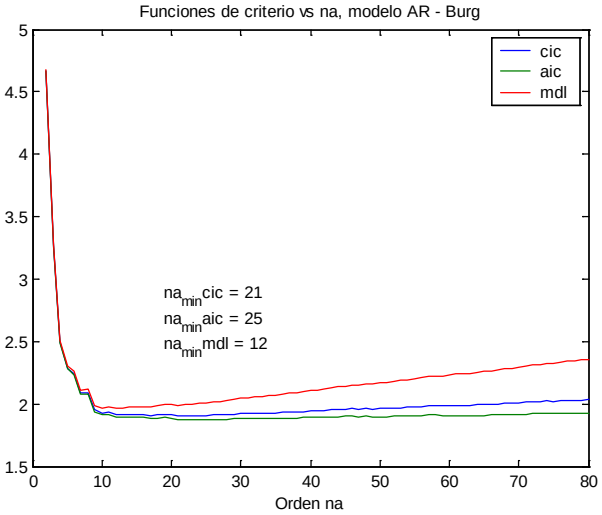
Es una buena oopción remover la tendencia lineal a la señal x para el análisis de las potencias medias relacionadas con los índices LF, HF y VHF.

5.2.2) PSD y potencia promedio con métodos paramétricos

a) Estructuras AR

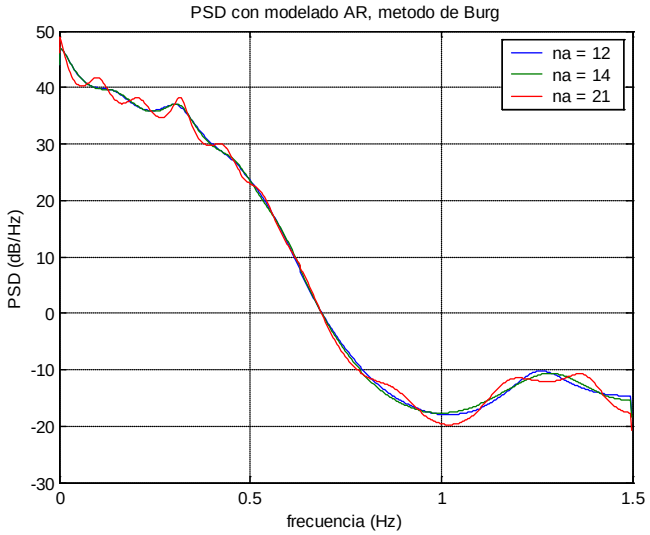
Busquemos el mejor orden de modelo con estructuras AR. Usaremos la señal x0, es decir, con valor medio 0.

```
>> [na,Vcic,Vn]=selarstruc(x0,'arburg',2:80,'cic');     %        na =   21
>> [na,Vaic,Vn]=selarstruc(x0,'arburg',2:80,'aic');na   %        na =   25
>> [na,Vmdl,Vn]=selarstruc(x0,'arburg',2:80,'mdl');na   %        na =   12
>> plot(Vcic(2,:),Vcic(1,:),Vaic(2,:),Vaic(1,:),Vmdl(2,:),Vmdl(1,:))
```



Realizando el mismo cálculo con la señal original, *x*, y con la señal removiéndole la tendencia lineal, *xd*, se obtienen los mismos resultados y una gráfica muy similar de las funciones de criterio *vs* el orden *na*. Con el método de la covarianza y covarianza modificada se consiguen valores muy similares de *na* para cada función de criterio. Utlizaremos los valores *na* = 12, *na* = 14 y *na* = 21 con el método de Burg.

```
>> [A,e]=arburg(x0,12);
>> [A14,e14]=arburg(x0,14);
>> [A21,e21]=arburg(x0,21);
>> [Pxx,f]=armaspectra(1,A,e,N,fs);
>> [Pxx14,f14]=armaspectra(1,A14,e14,N,fs);
>> [Pxx21,f21]=armaspectra(1,A21,e21,N,fs);
>> plot(f,10*log10(Pxx),f14,10*log10(Pxx14),f21,10*log10(Pxx21)),grid
>> xlabel('frecuencia (Hz)')
>> ylabel('PSD (dB/Hz)')
>> title ('PSD con modelado AR, metodo de Burg')
>> legend('na = 12', 'na = 14', 'na = 21')
```



Calculemos la potencia promedio con la señal *x0* en las diferentes bandas de frecuencia, utilizando los 3 órdenes de modelo *na*.

```
>> ind_vlf_dc=find(f <= 0.04);     % Para na = 12
>> ind_lf=find(f>0.04 & f <= 0.15); ind_hf=find(f>0.15 & f < 0.4); ind_vhf=find(f>=0.4 & f < 1);
>> Pm_tot=fs/N*sum(Pxx);                  %       Pm_tot =  3827.2
>> Pm_vlf_dc=fs/N*sum(Pxx(ind_vlf_dc)); %       Pm_vlf_dc =  1451.5
>> Pm_lf=fs/N*sum(Pxx(ind_lf));           %       Pm_lf =  1276.7
>> Pm_hf=fs/N*sum(Pxx(ind_hf));           %       Pm_hf =  1034.5
>> Pm_vhf=fs/N*sum(Pxx(ind_vhf));         %       Pm_vhf =   64.51
>> LFnorm=Pm_lf/(Pm_tot-Pm_vlf_dc)*100; %       LFnorm =   53.74
>> HFnorm=Pm_hf/(Pm_tot-Pm_vlf_dc)*100; %       HFnorm =   43.54
>> LFdivHF=Pm_lf/Pm_hf;                   %       LFdivHF =    1.23
```

Señal *x* removida su valor medio: x0, modelo AR-Burg, orden 12

	N= length(x)=887		Nfft = 2048		Nfft = 100000	
Bandas (Hz)	Pot med Pm=fs/N* sum(Px) (ms^2)	Pot med $P_m=\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft* sum(Px) (ms^2)	Pot med $P_m=\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft* sum(Px) (ms^2)	Pot med $P_m=\int_{f_1}^{f_2} P_x df$ (ms^2)
Total	3827,24	3783,57	3827,24	3808,32	3827,24	3826,85
VLF_dc = [0 – 0,04]	1451,47	1369,74	1480,64	1446,21	1474,78	1474,08
VLF = [0,003 – 0,04	1364,14	1239,79	1292,37	1239,79	1321,11	1320,03
LF = [0,04 – 0,15]	1276,73	1227,65	1244,55	1223,48	1249,30	1248,87
HF = [0,15 – 0,4]]	1034,51	1018,97	1037,36	1030,58	1037,82	1037,68
VHF = [0.4 – 1]	64,51	62,85	64,67	63,94	65,30	65,29
LFnorm	53,74	50,86	53,04	51,80	53,11	53,08
HFnorm	43,54	42,21	44,21	43,63	44,12	44,10
VHFnorm	2,72	2,60	2,76	2,71	2,78	2,77
LF/HF	1,23	1,20	1,20	1,19	1,20	1,20

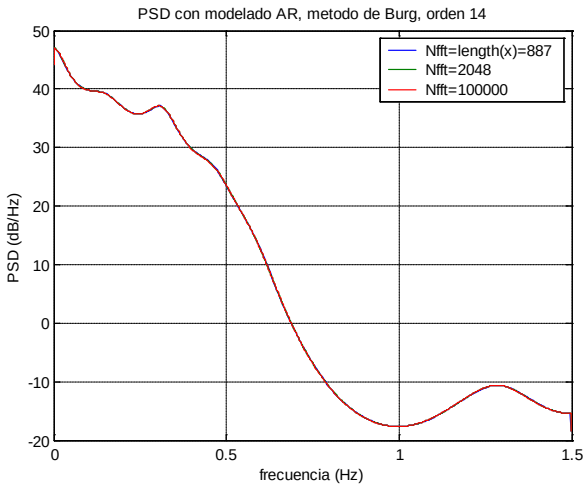
Señal *x* removida su valor medio: x0, modelo AR-Burg, orden 14

	N= length(x)=887		Nfft = 2048		Nfft = 100000	
Bandas (Hz)	Pot med Pm=fs/N* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)
Total	3827,24	3784,09	3827,24	3808,55	3827,24	3826,85
VLF_dc = [0 – 0,04]	1451,31	1369,56	1480,89	1446,46	1474,95	1474,25
VLF = [0,003 – 0,04	1365,01	1241,10	1294,82	1242,42	1323,09	1322,01
LF = [0,04 – 0,15]	1271,46	1221,74	1238,83	1217,48	1243,66	1243,21
HF = [0,15 – 0,4]]	1039,39	1023,75	1042,28	1035,47	1042,80	1042,66
VHF = [0.4 – 1]	65,06	63,51	65,21	64,53	65,80	65,79
LFnorm	53,51	50,60	52,80	51,54	52,87	52,84
HFnorm	43,75	42,40	44,42	43,84	44,33	44,32
VHFnorm	2,74	2,63	2,78	2,73	2,80	2,80
LF/HF	1,22	1,19	1,19	1,18	1,19	1,19

Señal *x* removida su valor medio: x0, modelo AR-Burg, orden 21

	N= length(x)=887		Nfft = 2048		Nfft = 100000	
Bandas (Hz)	Pot med Pm=fs/N* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)
Total	3827,24	3757,71	3827,24	3797,12	3827,24	3826,62
VLF_dc = [0 – 0,04]	1497,93	1404,28	1516,39	1476,45	1512,68	1511,87
VLF = [0,003 – 0,04	1358,87	1203,09	1221,38	1156,54	1270,71	1269,33
LF = [0,04 – 0,15]	1249,19	1217,33	1228,67	1215,00	1231,62	1231,34
HF = [0,15 – 0,4]]	1009,45	998,49	1011,37	1006,56	1011,53	1011,43
VHF = [0.4 – 1]	70,64	69,03	70,79	70,09	71,38	71,37
LFnorm	53,63	51,73	53,17	52,36	53,21	53,20
HFnorm	43,34	42,43	43,77	43,37	43,70	43,69
VHFnorm	3,03	2,93	3,06	3,02	3,08	3,08
LF/HF	1,24	1,22	1,21	1,21	1,22	1,22

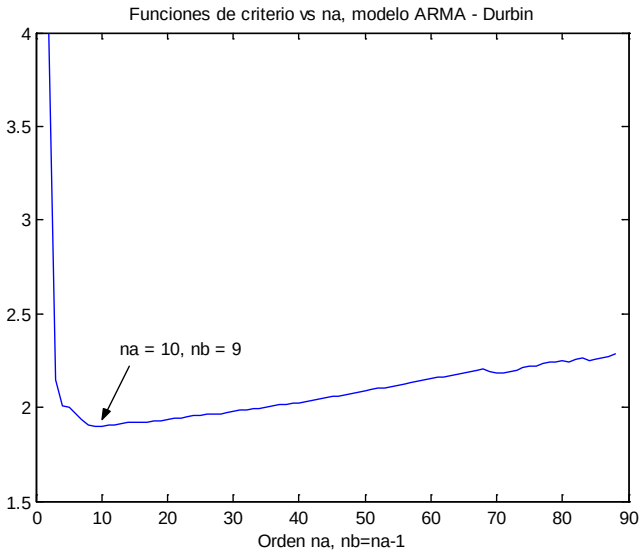
Un orden de modelo AR *na* = 14, con el método de Burg, es un buen compromiso para estimar la PSD en señales RR de corta duración. El número de puntos para hallar la PSD puede ser 2048. La PSD para este orden de modelo y usando Nfft = 887, 2048 y 100000 puntos es:



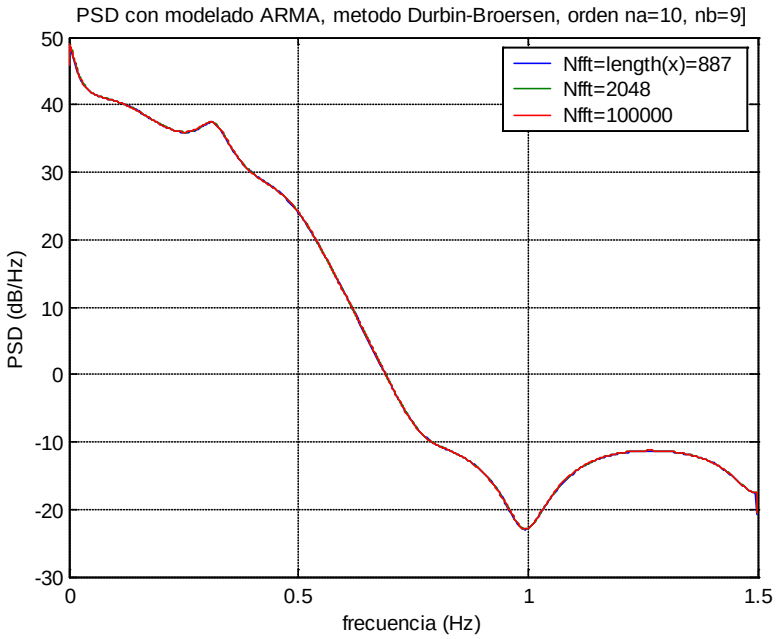
b) Estructuras ARMA

Usaremos en primer lugar el método de Durbin-Broersen [Dur60],[Bro00a],[Bro02] con el toolbox ARMASA [BroMat].

```
>> [Aa,Ba,sel]=sig2arma(x0);
>> plot(sel.cand_ar_order,sel.gic3)
```




```
>> [Aa,Ba,sel]=sig2arma(x0,10);
>> e=sel.pe_est
>> [Pxx,f]=armaspectra(Ba,Aa,e,N,fs);
>> [Pxx2,f2]=armaspectra(Ba,Aa,e,2048,fs);
>> [Pxx1,f1]=armaspectra(Ba,Aa,e,10000,fs);
>> plot(f,10*log10(Pxx),f2,10*log10(Pxx2),f1,10*log10(Pxx1)),grid
```



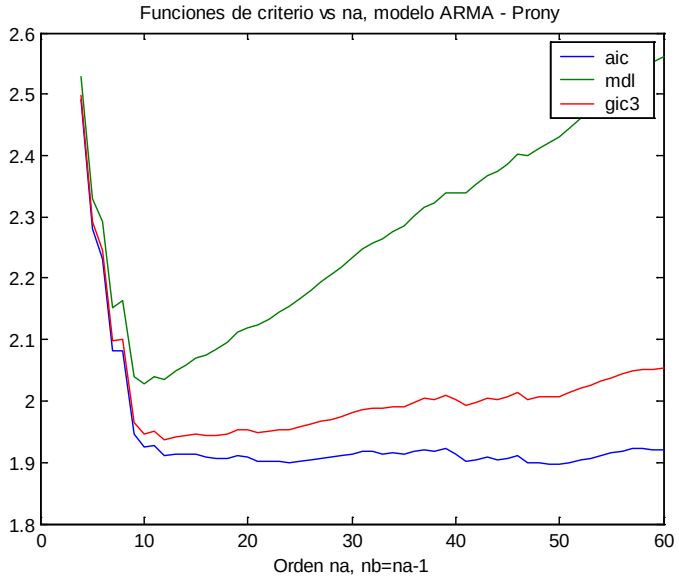
Calculemos la potencia promedio en las diferentes bandas de frecuencia utilizando los diferentes puntos sobre el eje *z*: *Nfft* = length(x0) = 887, *Nfft* = 2048 y *Nfft* = 100000.

Señal *x* removida su valor medio: x0, modelo ARMA-Durbin-Broersen, orden na=10, nb=9

	N= length(x)=887		Nfft = 2048		Nfft = 100000	
Bandas (Hz)	Pot med Pm=fs/N* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)
Total	3985,38	3918,50	3985,38	3956,42	3985,38	3984,79
VLF_dc = [0 – 0,04]	1517,23	1419,54	1541,37	1499,60	1536,53	1535,67
VLF = [0,003 – 0,04	1383,48	1226,49	1258,03	1192,60	1304,03	1302,64
LF = [0,04 – 0,15]	1332,84	1289,76	1305,73	1287,18	1309,48	1309,10
HF = [0,15 – 0,4]]	1068,46	1053,18	1071,29	1064,61	1071,75	1071,62
VHF = [0.4 – 1]	66,82	65,21	66,98	66,27	67,59	67,58
LFnorm	54,00	51,61	53,43	52,39	53,47	53,45
HFnorm	43,29	42,14	43,83	43,33	43,77	43,76
VHFnorm	2,71	2,61	2,74	2,70	2,76	2,76
LF/HF	1,25	1,22	1,22	1,21	1,22	1,22

Probaremos el método de Prony

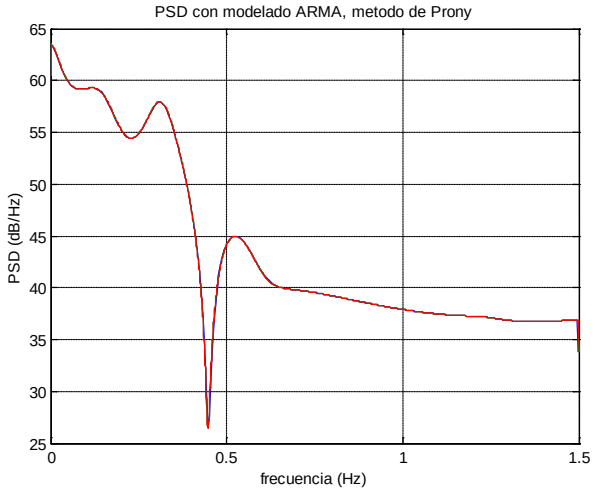
```
>> [nn,Vai,Vn] = selarmastruc2(x0,'prony_e',4:60,'aic');      %     nn =  [50 49]
>> [nn,Vm,Vn] = selarmastruc2(x0,'prony_e',4:60,'mdl');       %     nn =  [10   9]
>> [nn,Vg,Vn] = selarmastruc2(x0,'prony_e',4:60,'gic3');      %     nn =  [12  11]
>> plot(Vai(2,:),Vai(1,:),Vm(2,:),Vm(1,:),Vg(2,:),Vg(1,:))
>> legend('aic','mdl','gic3')
>> xlabel('Orden na, nb=na-1')
>> title ('PSD con modelado ARMA, metodo de Prony')
```



Con el orden [na, nb] = [12 11], tenemos

```
>> [B,A,e]=prony_e(x0,11,12);
>> [Pxx,f]=armaspectra(B,A,e,N,fs);
>> [Pxx2,f2]=armaspectra(B,A,e,2048,fs);
>> [Pxx1,f1]=armaspectra(B,A,e,10000,fs);
>> plot(f,10*log10(Pxx),f2,10*log10(Pxx2),f1,10*log10(Pxx1)),grid
```

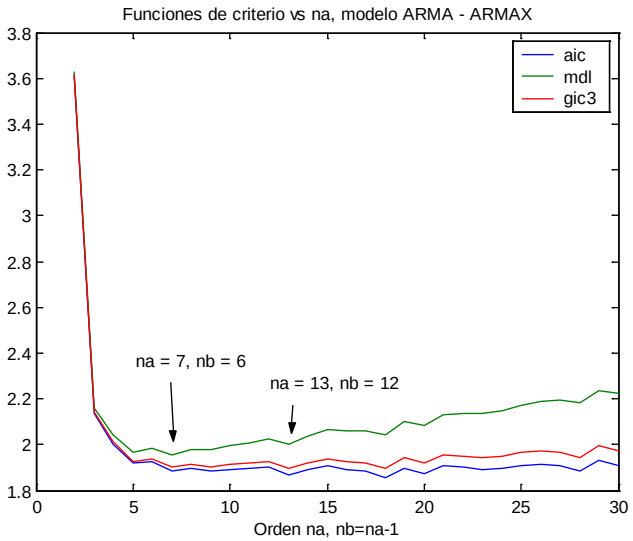
```
>> legend('aic','mdl','gic3'), xlabel('Orden na, nb=na-1')
>>  title('Funciones de criterio vs na, modelo ARMA - Prony')
```



N = length (x0) = 887	N = 2048	N = 10000
Pm_tot = 2.6578e+005	Pm_tot = 2.6579e+005	Pm_tot = 2.6579e+005
Pm_vlf_dc = 6.5741e+004	Pm_vlf_dc 6.7281e+004	Pm_vlf_dc = 6.7022e+004
Pm_lf = 9.4823e+004	Pm_lf = 9.3021e+004	Pm_lf = 9.3282e+004
Pm_hf = 9.5461e+004	Pm_hf = 9.5712e+004	Pm_hf = 9.5678e+004
Pm_vhf = 7200.6	Pm_vhf = 7.3093e+003	Pm_vhf = 7.3437e+003
LFnorm = 47.40	LFnorm = 46.86	LFnorm = 46.93
HFnorm = 47.72	HFnorm = 48.22	HFnorm = 48.14
LFdivHF = 0.993	LFdivHF = 0.972	LFdivHF = 0.975

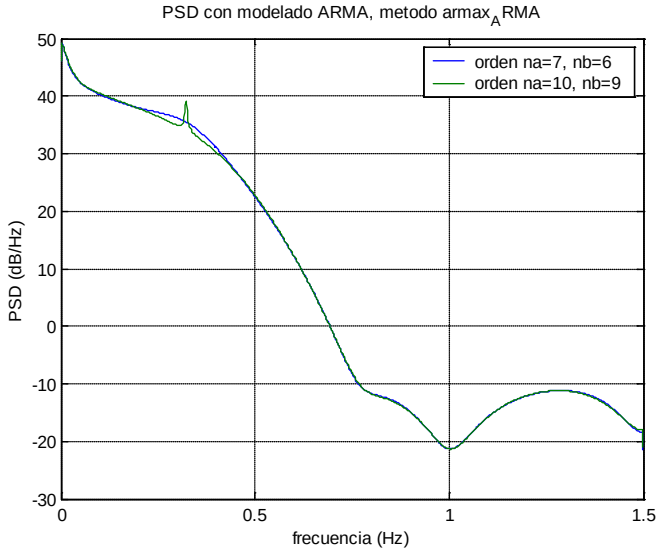
El modelado ARMA con el método Prony no produce tan buenos resultados como el de Durbin-Broersen. Probemos el método Armax-ARMA.

```
>> [nn,Vm,Vn] = selarmastruc2(x0,'armax_arma',2:30,'mdl');        % nn = [7  6]
>> [nn,Vg,Vn] = selarmastruc2(x0,'armax_arma',2:30,'gic3');       % nn = [18 17]
>> [nn,Vai,Vn] = selarmastruc2(x0,'armax_arma',2:30,'aic');       % nn = [18 17]
>> plot(Vai(2,:),Vai(1,:),Vm(2,:),Vm(1,:),Vg(2,:),Vg(1,:))
```




```
>> [B,A,e]=armax_arma(x0,6,7);
>> [Pxx,f]=armaspectra(B,A,e,N,fs);
>> [Pxx2,f2]=armaspectra(B,A,e,2048,fs);
```

```
>> [B,A,e]=armax_arma(x0,12,13);
>> [Pxx3,f3]=armaspectra(B,A,e,N,fs);
>> [Pxx,f4]=armaspectra(B,A,e,2048,fs);
```



Señal *x* removida su valor medio: x0, modelo armax_ARMA, orden na=7, nb=6

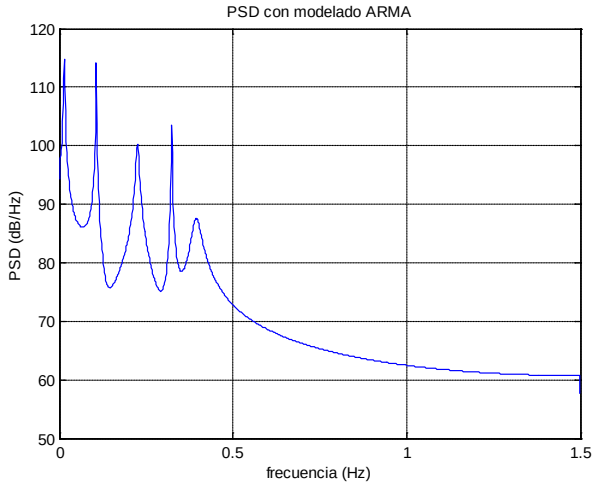
	N= length(x)=887		Nfft = 2048		Nfft = 100000	
Bandas (Hz)	Pot med Pm=fs/N* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)
Total	4270,97	4200,07	4270,97	4240,26	4270,97	4270,34
VLF_dc = [0 – 0,04]	1728,39	1621,21	1756,58	1710,90	1750,93	1749,99
VLF = [0,003 – 0,04	1586,60	1414,95	1454,86	1383,00	1503,68	1502,17
LF = [0,04 – 0,15]	1336,83	1289,55	1305,78	1285,44	1310,40	1309,98
HF = [0,15 – 0,4]]	1141,56	1126,21	1144,20	1137,53	1144,44	1144,30
VHF = [0.4 – 1]	64,17	62,11	64,38	63,46	65,18	65,16
LFnorm	52,58	50,00	51,93	50,82	52,00	51,98
HFnorm	44,90	43,67	45,51	44,97	45,41	45,40
VHFnorm	2,52	2,41	2,56	2,51	2,59	2,59
LF/HF	1,17	1,15	1,14	1,13	1,15	1,14

Señal *x* removida su valor medio: x0, modelo armax_ARMA, orden na=10, nb=9

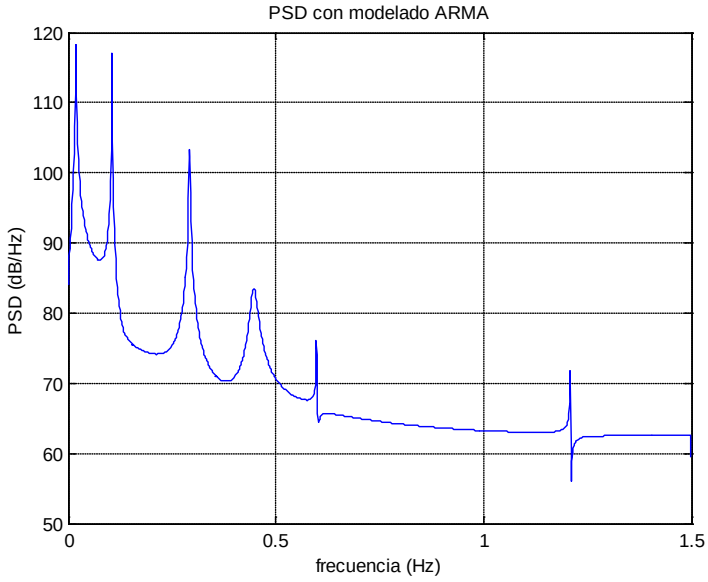
	N= length(x)=887		Nfft = 2048		Nfft = 100000	
Bandas (Hz)	Pot med Pm=fs/N* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)	Pot med Pm=fs/Nfft* sum(Px) (ms^2)	Pot med $P_{m=}\int_{f_1}^{f_2} P_x df$ (ms^2)
Total	4252,21	4182,40	4252,16	4221,92	4252,16	4251,54
VLF_dc = [0 – 0,04]	1750,94	1645,04	1778,83	1733,77	1773,24	1772,32
VLF = [0,003 – 0,04	1611,31	1441,01	1480,93	1409,47	1529,31	1527,81
LF = [0,04 – 0,15]	1376,02	1327,78	1344,98	1324,24	1349,44	1349,01
HF = [0,15 – 0,4]]	1062,19	1045,85	1065,10	1057,99	1065,55	1065,41
VHF = [0.4 – 1]	63,04	61,24	63,22	62,42	63,91	63,89
LFnorm	55,01	52,33	54,38	53,22	54,44	54,41
HFnorm	42,47	41,22	43,06	42,52	42,98	42,97
VHFnorm	2,52	2,41	2,56	2,51	2,58	2,58
LF/HF	1,30	1,27	1,26	1,25	1,27	1,27

Con los métodos Output error-ARMA y Steiglitz-McBride no se consiguen resultados satisfactorios con este tipo de señal.

```
>> [B,A,e]=stmcb_e(x0,9,10);
>> armaspectra(B,A,e,2048,fs);
```




```
>> [B,A,e]=oe_arma(x0,12,13);
>> armaspectra(B,A,e,2048,fs);
```



Entre las estructuras ARMA analizadas sugerimos el uso de los métodos de Durbin-Broersen y armax_ARMA para estimar los componentes frecuenciales de este tipo de señal. Los órdens na = 10, nb = 9, propuesto por el criterio GIC en Durbin-Broersen y na = 7, nb = 6 en armax_ARMA pueden ser una buena aproximación para señales RR re-muestreadas a fs = 3 Hz de 3 minutos de duración. Un número de puntos Nfft = 2048 es suficiente con esta señal analizada. Dependiendo de la longitud del segmento HRV a evaluar, y el número de muestras luego del remuestreo, se pueden escoger órdenes y número de puntos diferentes.

REFERENCIAS

[Aka70] Akaike H. Statistical predictor identification. Ann Inst Stat Math, vol 22, pp. 173-220, 1970.

[Aka74] Akaike H. A new look at the statistical model identification. IEEE Trans Autom Control, vol 19, pp. 716-723, 1974.

[Bro93] Broersen PMT, Wensink HE. On finite sample theory autoregressive model order selection. IEEE Trans Signal Processing, vol. 41 (1), pp. 194-204, 1993.

[Bro00a] Broersen PMT. Autoregressive Model Order for Durbin's MA and ARMA Estrimator. IEEE Trans Signal Processing, vol. 48 (8), pp. 2454-2457, 2000.

[Bro00b] Broersen PMT. Finite sample criteria for autoregressive order selection. IEEE Trans Signal Processing, vol. 48 (12), pp. 3550-3558, 2000.

[Bro02] Broersen, PMT. Automatic spectral analysis with time series models. IEEE Trans Instrum Meas, vol 51 (2), pp 211-216, 2002.

[BroMat] Broersen, PMT. Matlab Toolbox ARMASA [oOnline]. Available: http://www.tn.tudelft.nl/mmr.

[Bur67] Burg JP, Maximun likehood spectral analysis. In Proc 37th Meet Soc Exploration Geophysicist., Oklahoma City, OK, p. 1-6, 1967.

[Car68] Carlson AB, Communication Systems: An Introduction to Signals and Noise in Electrical Communication, McGraw-Hill, Tokyo, 1968.

[Den83] Dennis J, Schnabel R, "Numerical methods for unconstrained optimization and and nonlinear equation", Prentice Hall, Englewood Cliffs, New Jersey. 1983.

[Dur60] Durbin J. The fitting of time series models. Rev Inst Int Stat, vol 28, pp. 233-243, 1960.

[Han79] Hannan EJ, Quinn BG. The determination of the order of an autoregression. J R Statist Soc. Ser B, vol B-41, pp. 190-195, 1979.

[Har78] Harris FJ (1978), "On the use of windows for harmonic analysis with discrete Fourier transform", Proc IEEE, 66:51-84.

[Kay81] Kay SM, Marple SL. Spectrum Analysis – A modern perspective. Proc IEEE, vol 69, pp. 1380-1419, 1981.

[Lju87] Ljung L. Identification Systems. Theory for the user, Prentice Hall, New Jersey, 1987.

[Mar77] Marple Jr., S.L. "Resolution of conventional Fourier, autoregressive and special ARMA methods of spectral analysis" IEEE International Conf. on ASSP pp 74–7, 1977.

[Mar82] Marple Jr., S.L. Frequency Resolution of Fourier and Maximum Entropy Spectral Estimates, Geophysics, vol. 47, No. 9, pp. 1303-1307, Sep. 1982.

[Mar87] Marple Jr, S.L. Digital Spectral Analysis with Applications. Prentice Hall, New Jersey, 1987.

[Par87] T.W. Parks and C.S. Burrus, Digital Filter Design, John Wiley and Sons, 1987, p226.

[Pro92] Proakis JG, Rader CM, Ling F, Nikias CL, "Advanced Digital Signal Processing", Macmillan Pub. Co., New York, 1992

[Ris78] Rissanen J. Modeling by the shortest data description. Automatica, vol 14, pp. 465-471, 1978.

[Sod88] Söderström T, Stoica P, "On some identification techniques for adaptive filtering", IEEE Trans Circuits Syst, CAS-35:457-461. 1988.

[Ste65] Steiglitz K, McBride LE, "A technique for the identification of linear systems", IEEE Trans Autom Control, AC-10:461-464. 1965

[Tas96] Task Force of The European Society of Cardiology and The North American Society of Pacing and Electrophysiology. Heart rate variability: standards of measurement, physiological interpretation, and clinical use. Eur Heart J 1996;17:354-

[The92] Ch. W. Therrien, Discrete Random Signals and Statistical Signal Processing. Englewood Cliffs, New Jersey: Prentice-Hall, pp. 550-575, 1992.