

IMPLEMENTACION PROPIA DE LA FUNCION PBURG PARA LA OBTENCION DEL ESPECTRO DE POTENCIAS

cREACION E IMPLEMENTACION DE UNA FUNCION EN MATLAB QUE REALICE EL METODO DE BURG PARA LA OBTENCION DE COEFICIENTES DE REGRESION Y ESTIMACION DE ESPECTRO DE POTENCIAS

PhD. rOBIN aLVAREZ

Fierro Martin – Erick Moreira

# Planteamiento del Problema

Implementar nuestra propia función **mi\_pburg** para determinar los coeficientes de la regresión y con estos determinar el espectro de potencias. Debería obtener el mismo resultado que con la función de Matlab **pburg**.

# Resolucion Teorica

Metodos Parametricos

Los métodos paramétricos permiten una estimación espectral de alta resolución, y se han mostrado como una alternativa interesante a la TF en el análisis de bioseñales, como, por ejemplo, en el caso de series temporales RR [1].

El modelado espectral, en este contexto será un proceso de 3 pasos:

1. El primer paso para seguir es elegir un modelo apropiado para el proceso. Esta elección puede estar basada en el conocimiento a priori de cómo ha sido generado el proceso, o en resultados experimentales que indiquen que un modelo en particular funciona bien. Los modelos que vamos a estudiar son AR, MA y ARMA.
2. Una vez elegido el modelo, se han de estimar los parámetros del modelo para los datos disponibles que es la señal recibida , que se supone una realización de un proceso estocástico estacionario y ergódico .
3. Finalmente, se estima el espectro de potencia del proceso estocástico sustituyendo los parámetros del modelo estimado en la PSD teórica del modelo implicado.

COEFICIENTES DE MODElo

La estimación de los coeficientes (parámetros) del modelo (autorregresivo) se realiza a partir de las muestras de la señal. Como regla general, si tenemos N muestras de la señal, suele utilizarse la mitad para estimar los coeficientes y el resto para validar el modelo.

Se han desarrollado diversos algoritmos para obtener los coeficientes del modelo. Los más utilizados son: a) Forward-backward; b) Least-squares; c) Yule-Walker; **d) Burg**; e) Geometric lattice. Los dos últimos utilizan la estructura AR lattice (representación como filtro de celosía de la función de transferencia del modelo AR) para obtener los coeficientes.

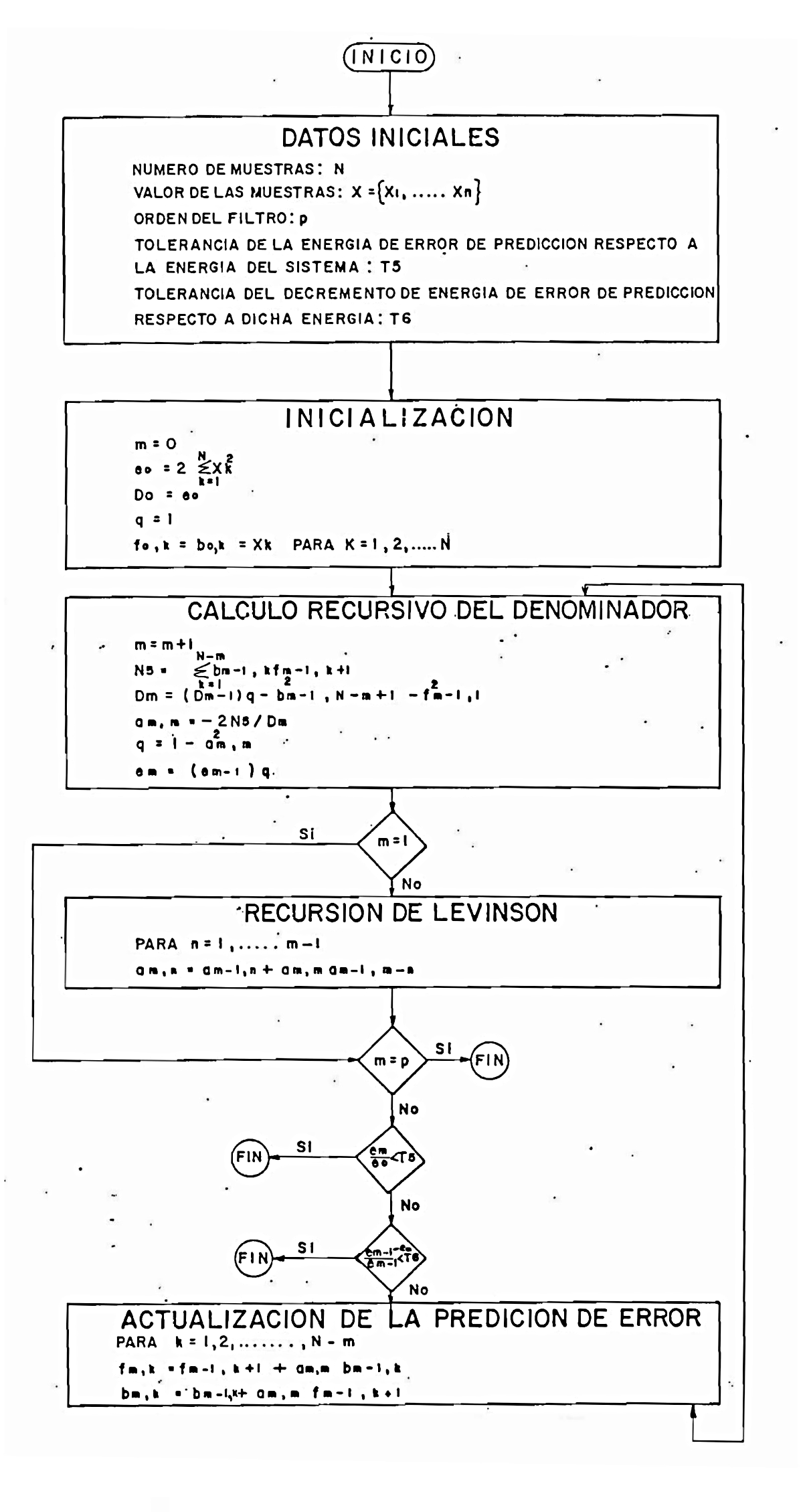
Metodo de burg

Algoritmo

Fue planteado en 1968. Es un algoritmo eficiente que ofrece importantes resultados al analizar sinusoides que contienen ruido aditivo.

Burg planteo que los parámetros AR deben ser tales que minimicen la suma de energías de error de predicción en adelanto y retardo.

A continuación, se muestra un diagrama de flujo y el seudocódigo del algoritmo de Burg en el que se basará la implementación de la función de Matlab, cabe destacar que el algoritmo es amplio para abarcarlo en su totalidad , motivo por el cual se muestra solo el seudocódigo del método rápido de Burg, para más información consultar las referencias bibliográficas [3]:

FIG. 1 Diagrama de Flujo de el algoritmo de Burg [2].

Algoritmo Matemático Pseudocódigo (Método Convencional de Burg)

Para una señal de entrada , se calcula los coeficientes de reflexión, los coeficientes de error de predicción y el componente de error Epsilon de la siguiente manera:

1. Inicialización:

(1)

(2)

(3)

(4)

1. Se remueve el primer elemento de y el ultimo de :

(5)

(6)

1. Cálculo del coeficiente de reflexión:

(7)

1. Cálculo de los coeficientes de error de predicción:

. (8)

Con siendo una matriz de intercambio:

1. **Si**

Terminado

**Fin si**

1. Se actualizan los errores de predicción:

(9)

(10)

(11)

(12)

(13)

1. Repetir desde 1. [3]

# Resolucion PRÁCTICA paso a paso

Implementación del algoritmo de Burg en Matlab [4]:

function varargout= mi\_pburg( x, orden, nfft, ventana, Fs)

%PBURGW Estimación Espectral de Potencia usando Burg enventanando la señal

% Pxx = PBURGW(x, Orden, nfft, Ventana) es la estimación de la D.E.P.(Densidad

% Espectral de Potencia) de la señal x, usando el método de Burg con

% enventanado. Orden es el orden del modelo Auto Regresivo de predicción lineal.

% NFFT es la longitud de la FFT que determina las frecuencias a que se

% estima la D.E.P. Ventana es el nombre de la función Matlab que realiza

% una determinada ventana, se introduce en comillas simples como texto Ej.: 'blackmanharris'.

% Pxx tendrá longitud (NFFT/2+1) para NFFT par, (NFFT+1)/2 para NFFT impar,

% y NFFT si x es una señal compleja. NFFT por defecto vale 256.

% Si no se especifica la ventana, se utiliza por defecto la ventana

% rectangular(rectwin).

%

% [Pxx, F] = PBURGW(x, Orden, nfft, Ventana, Fs) devuelve, además, el vector de

% frecuencias en que se evalúa en la D.E.P., para una frecuencia de muestreo Fs.

% Por defecto usa una Fs de 2 [Hz].

%

% PBURGW sin argumentos de salida grafica la Densidad Espectral de Potencias.

%1 - Revisión de los parámetros de entrada

narginchk(2,5);

if isempty(orden)

error('Necesita indicar el orden del modelo.');% Se tiene que especificar el orden del modelo

end

if nargin < 5, Fs = []; end

if nargin < 4, ventana = 'rectwin'; end % Ventana rectangular

if nargin < 3, nfft = []; end

if isempty(nfft), nfft = 256; end

if isempty(Fs), Fs = 2; end

El método de Burg opera convencionalmente en la serie temporal datos, en iteraciones. La iteración i encuentra un coeficiente de reflexión, , minimizando los vectores de error de predicción hacia adelante y hacia atrás y .

Para una señal de entrada calcula los coeficientes de reflexión de la siguiente manera, con y :

(11)

Se remueve el primer elemento de y el último elemento de

(12)

(13)

%2 - Inicialización de los parámetros

N = length(x);

fi\_t = x(2:end); % fragmento de vector de errores de predicción hacia adelante, se omite el primer elemento i = 1

gi\_t = x(1:end-1); % fragmento de vector de errores de predicción hacia atrás, se omite el último elemento

a = 1; % Inicialización del vector de estimadores de los coeficientes de predicción (a\_k) del filtro Auto Regresivo

Epsilon = x\*x'/N; % (Epsilon\_0)(Error de mín. cuadrados)

K = zeros(orden,1); % Estima de los coeficientes de reflexión

Una serie de tiempo proveniente de un proceso estocástico puede ser analizada a través de un modelo paramétrico autorregresivo. la muestra de una serie de tiempo en un momento dado, puede ser expresada como una sumatoria lineal ponderada de valores previos de la serie temporal más una componente de error (Epsilon) [6]:

(14)

En el código p es el orden del modelo AR.

%3 - Cálculo de estimadores de K, E.

%-----Algoritmo Iterativo

for i=1:orden

Se realiza un producto de una ventana con los vectores de predicción hacia adelante y hacia atrás y , para el cálculo del coeficiente de reflexión , para obtener valores en tiempo finito con la longitud de la señal ,

(15)

% Cálculo de la ventana:

% Se puede usar cualquier ventana que este implementada como una función Matlab

% Pero se recomienda utilizar la por defecto:

vent = ['v=' ventana '(length(fi\_t));'];

eval(vent);

v = v(:)'; % Conversión a vector fila(tanto si era columna como fila)

K(i)=-((v.\*fi\_t)\*gi\_t')/(((v.\*fi\_t) \* fi\_t' + (v.\*gi\_t) \* gi\_t')/2);

Aunque generalmente no se usa en el método Burg, los coeficientes de error de predicción se pueden encontrar a partir de los coeficientes de reflexión utilizando la recursividad de Levinson.

, (16)

Donde es una matriz de intercambio, que da la vuelta a un vector o matriz de izquierda a derecha si es horizontal, y de arriba abajo en vertical y viceversa. El tamaño de es igual a , cabe destacar que en el código no se la utiliza debido a la función flipud de Matlab que realiza el trabajo de intercambio, pero solo en vertical.

Consecuentemente se actualizan los vectores de error de predicción, y la componente de error:

(17)

(18)

(19)

(20)

(21)

a = [a;0] + K(i) \* [0;flipud(conj(a))]; % Actualización de los coeficientes de predicción {ak}

fi\_t\_nuevo = fi\_t + K(i) \* gi\_t;

gi\_t\_nuevo = K(i)' \* fi\_t + gi\_t;

fi\_t = fi\_t\_nuevo(2:end);

gi\_t = gi\_t\_nuevo(1:end - 1);

Epsilon(i+1) = (1-K(i)\*K(i)') \* Epsilon(i); % Error de mín. cuadrados

end

% Fin del algoritmo Iterativo.

Después de acabar el algoritmo de Burg se utiliza la transformada rápida de Fourier para la estimación de la densidad espectral de potencias DEP.

%4 - Estimación de la DEP usando la FFT(periodograma)

Af = abs( fft( a, nfft ) ) .^ 2; % Función de trasferencia del denominador del filtro

Pxx = Epsilon(end) ./ Af; % Función de trasferencia teniendo en cuenta la entrada(ruido)

El control de la salida retornara mínimo un resultado y máximo dos siendo el primero el espectro de potencias de la DEP, y el segundo el dominio de frecuencias. En el caso de que no se especifique una salida la función graficará la Densidad Espectral de Potencia en decibeles.

%5 - Control de la salida

% Si la entrada es real, se selecciona sólo la primera mitad

if isreal(x)

select = (1:floor(nfft/2)+1)'; %Siempre es unilateral(Solo frecuencias positivas)

Pxx = Pxx(select);

%Para que sea unilateral hay que hacer una corrección

Pxx(1)=Pxx(1)/2;

Pxx(end)=Pxx(end)/2;

else

select = (1:nfft)';

Pxx = Pxx(select);

end

ff = (select - 1)\*Fs/nfft;

%Argumentos de salida:

if nargout == 0

newplot;

plot(ff,10\*log10(Pxx)); grid on;

xlabel('Frecuencia [Hz]');

ylabel('Módulo de la Densidad Espectral de Potencia [dB]');

title('Estimación Espectral de Burg');

end

if nargout >= 1

varargout{1} = Pxx;

end

if nargout >= 2

varargout{2} = ff;

end [5]

**Vamos a probar nuestra función en los diferentes escenarios en los que se presentaría una señal, en este caso vamos a utilizar una señal seno de suma de frecuencias.**

xn = 1\*sin(2\*pi\*100\*t) + 0.5\*sin(2\*pi\*105\*t) + ...

0.1\*sin(2\*pi\*110\*t) + 0.01\*sin(2\*pi\*115\*t) + ...

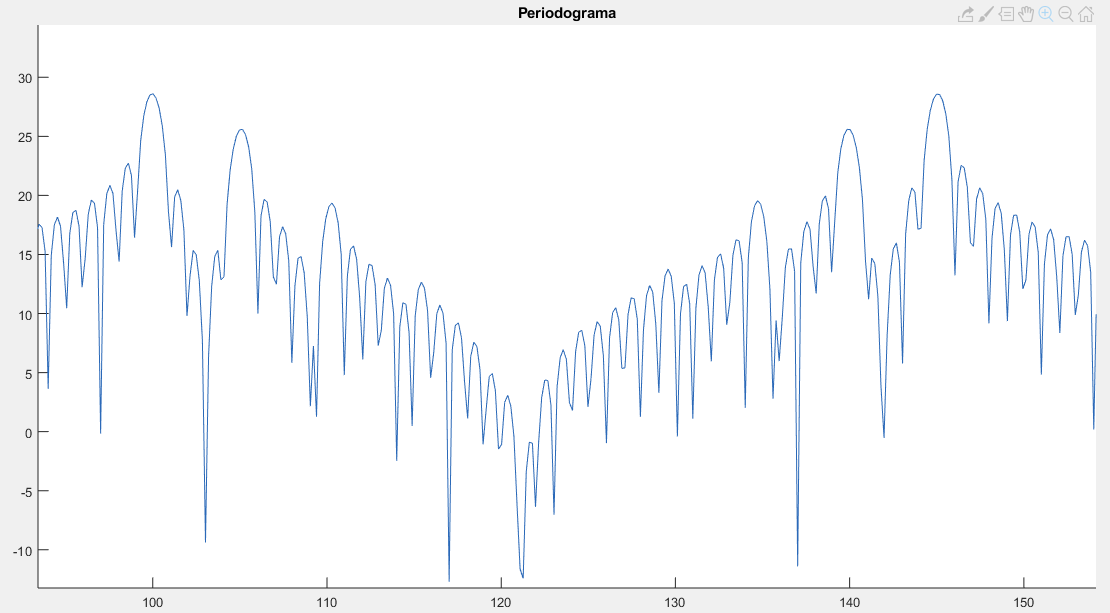
0.001\*sin(2\*pi\*120\*t) + 0.001\*sin(2\*pi\*125\*t) + ...

0.01\*sin(2\*pi\*130\*t) + 0.1\*sin(2\*pi\*135\*t) + ...

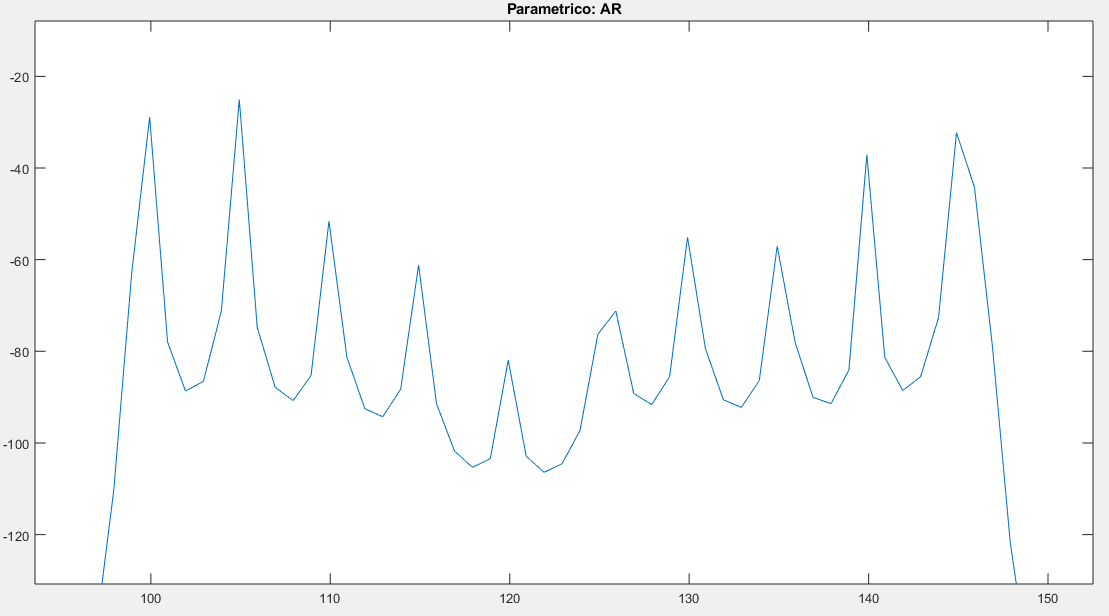
0.5\*sin(2\*pi\*140\*t) + 1\*sin(2\*pi\*145\*t);

Y de este caso base tenemos las siguientes graficas:

Utilizando un enventanado normal:

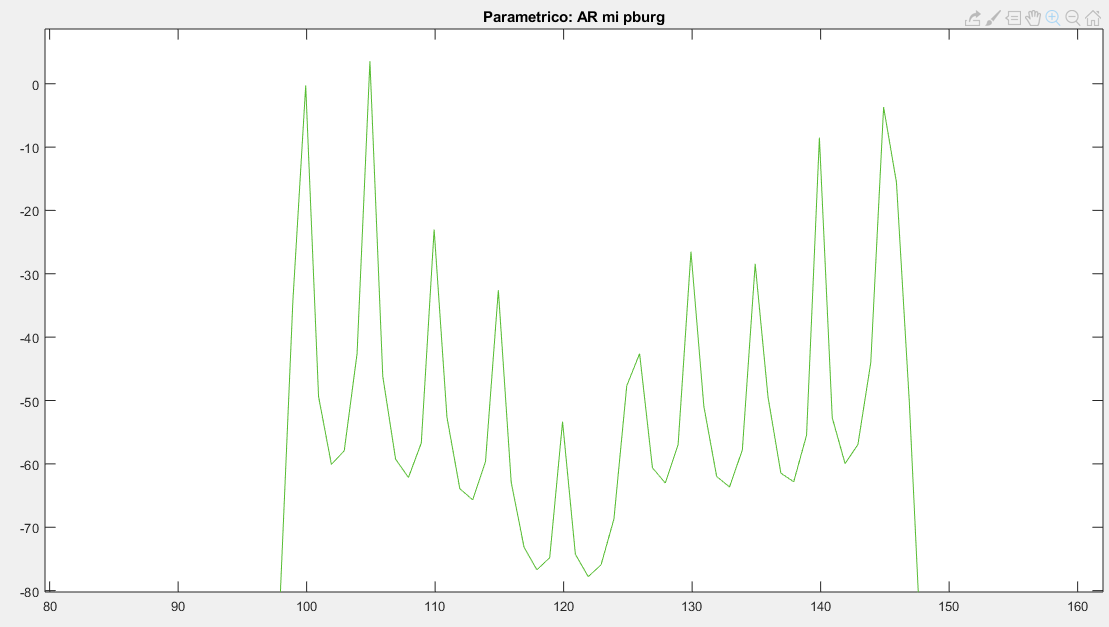


Utilizando la función propia de Matlab “pburg”:



Añadir código del ejemplo

Con nuestra implementación de la “pburg”:



Ahora para poner a prueba el método, hacemos más pequeña la variación de frecuencias

xn = 1\*sin(2\*pi\*100\*t) + 0.5\*sin(2\*pi\*100.1\*t) + ...

0.1\*sin(2\*pi\*100.2\*t) + 0.01\*sin(2\*pi\*100.3\*t) + ...

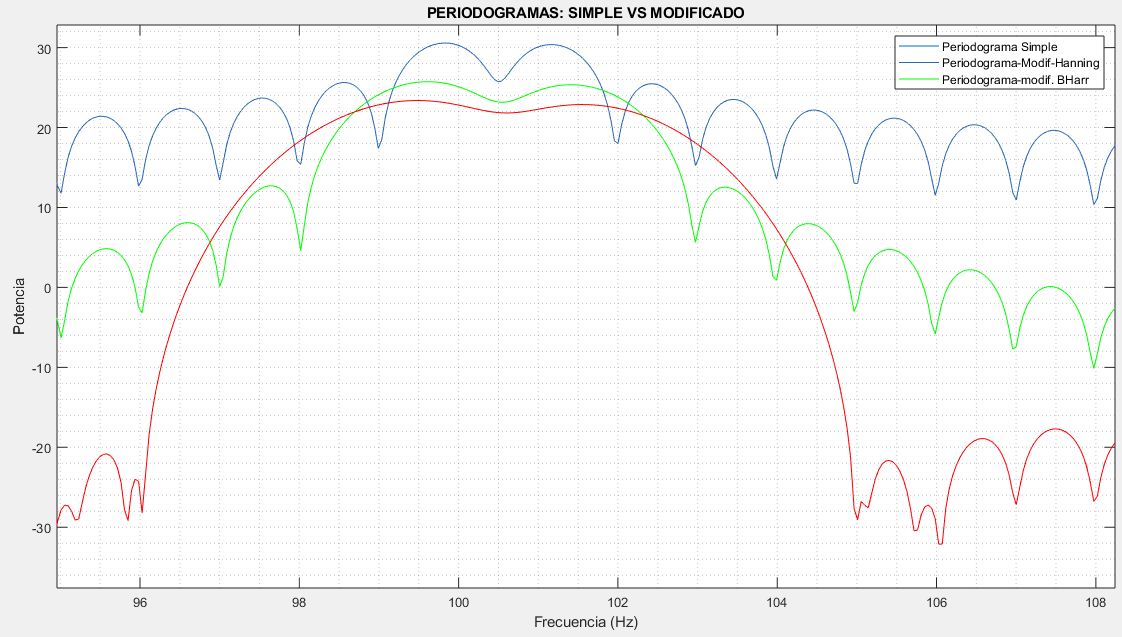
0.001\*sin(2\*pi\*100.4\*t) + 0.001\*sin(2\*pi\*100.5\*t) + ...

0.01\*sin(2\*pi\*100.6\*t) + 0.1\*sin(2\*pi\*100.7\*t) + ...

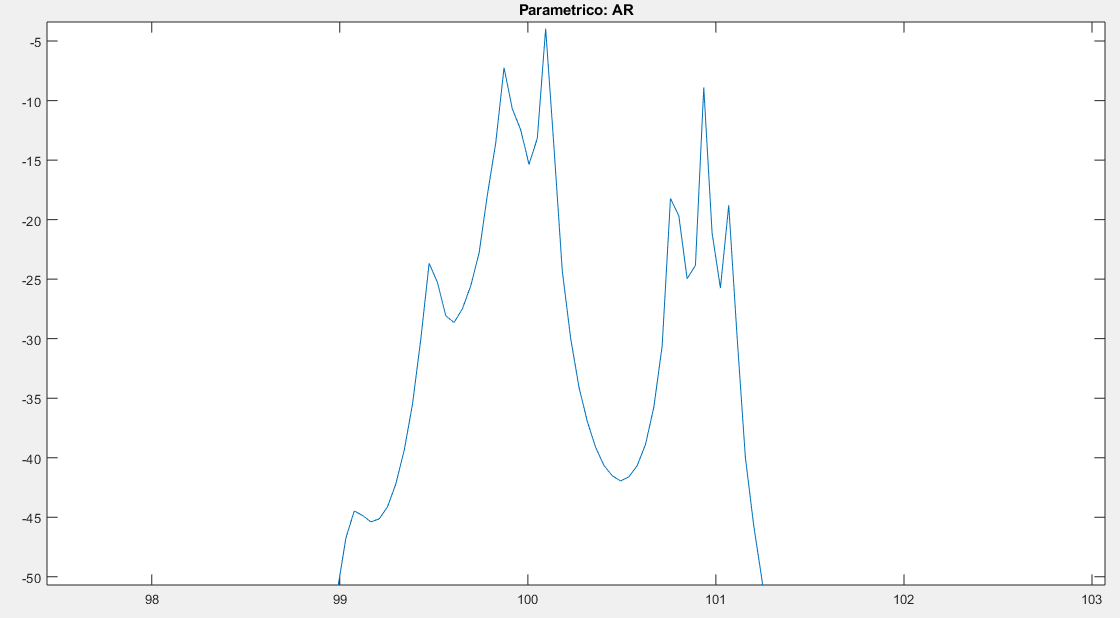
0.5\*sin(2\*pi\*100.8\*t) + 1\*sin(2\*pi\*101\*t);

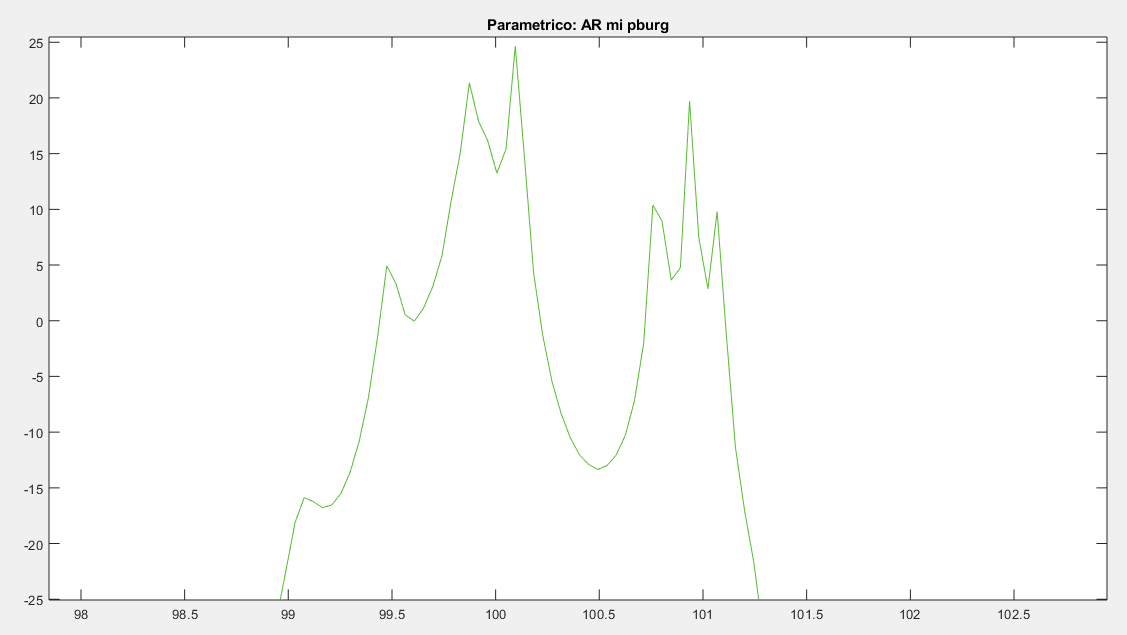
Añadir código del ejemplo

Periodogramas con enventanado



Con “pburg”:





Ahora a las frecuencias de la señal original se las va a reducir su amplitud y aplicar su respectivo ruido.

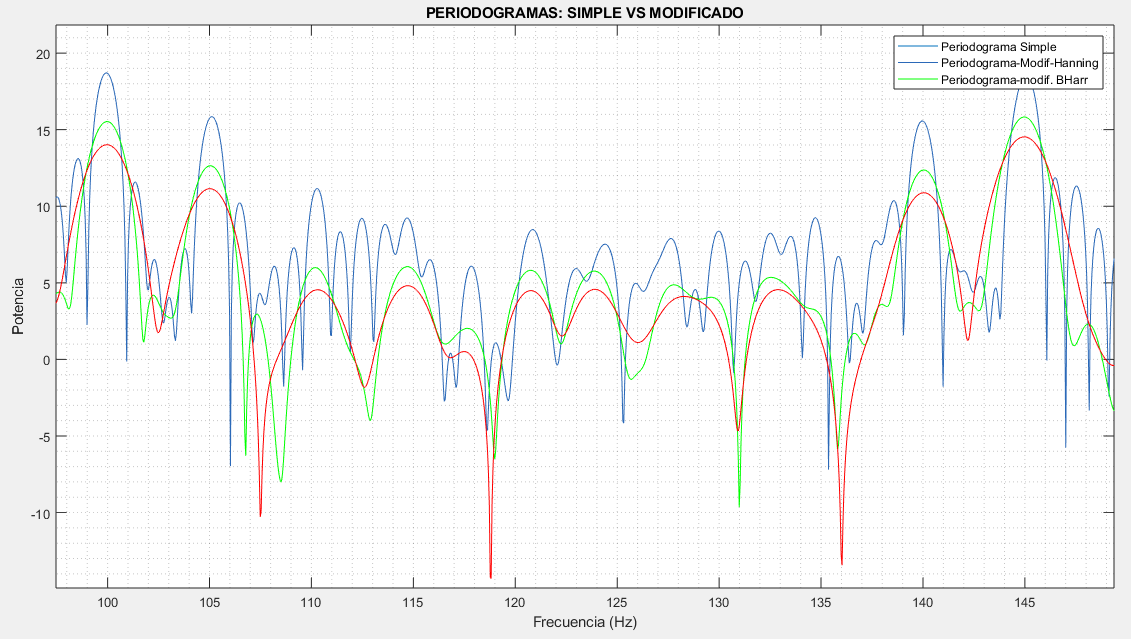
xn = 1\*sin(2\*pi\*100\*t) + 0.5\*sin(2\*pi\*100.1\*t) + ...

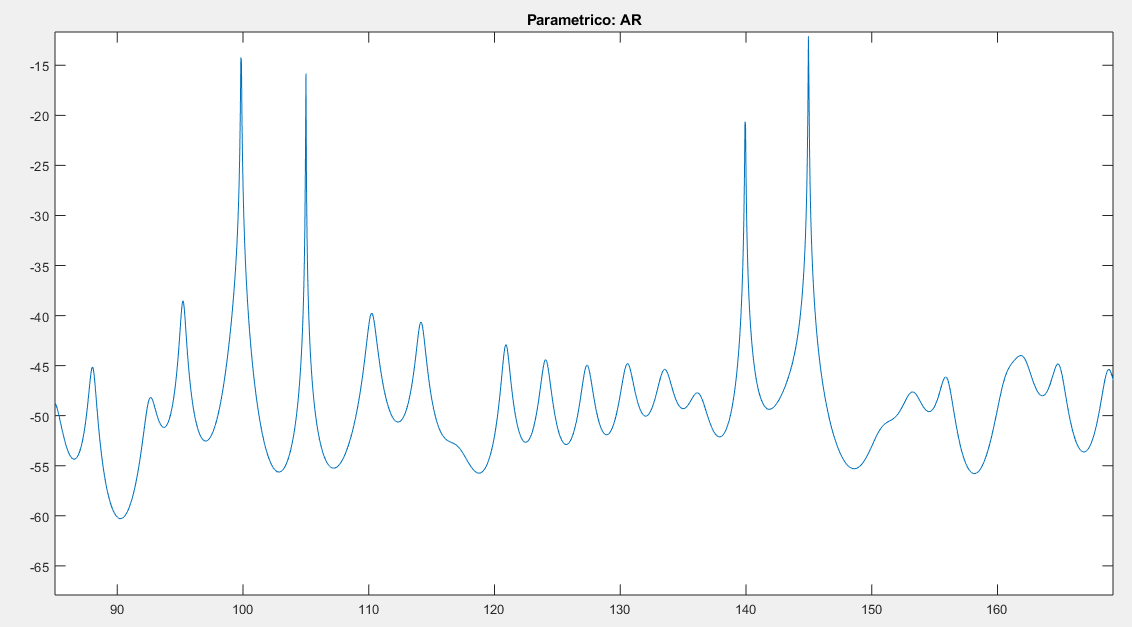
0.1\*sin(2\*pi\*100.2\*t) + 0.01\*sin(2\*pi\*100.3\*t) + ...

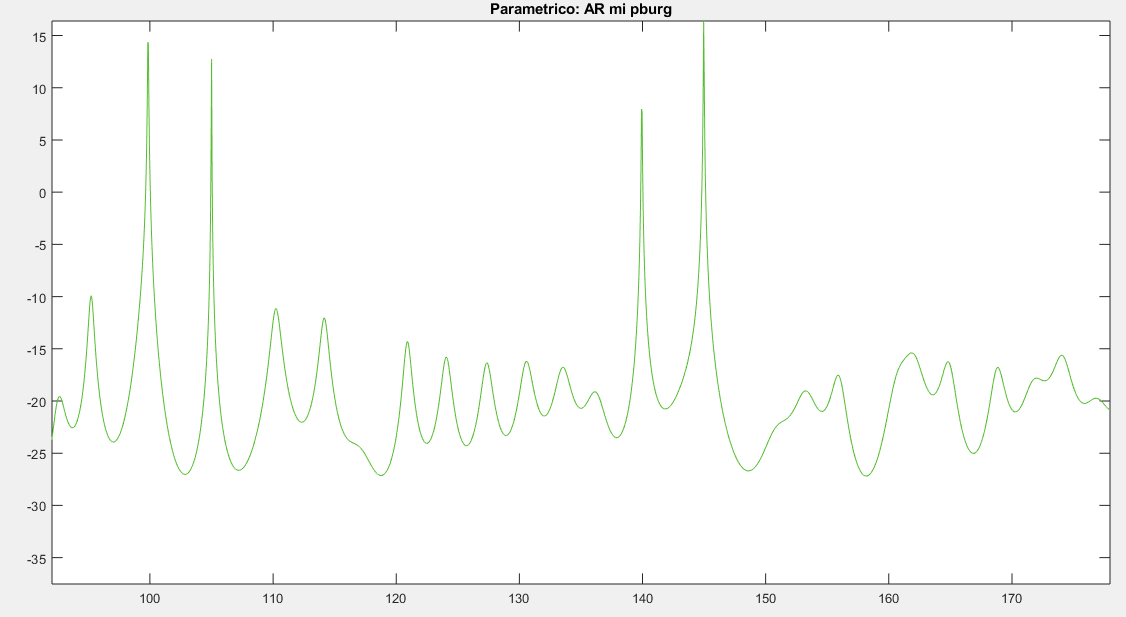
0.001\*sin(2\*pi\*100.4\*t) + 0.001\*sin(2\*pi\*100.5\*t) + ...

0.01\*sin(2\*pi\*100.6\*t) + 0.1\*sin(2\*pi\*100.7\*t) + ...

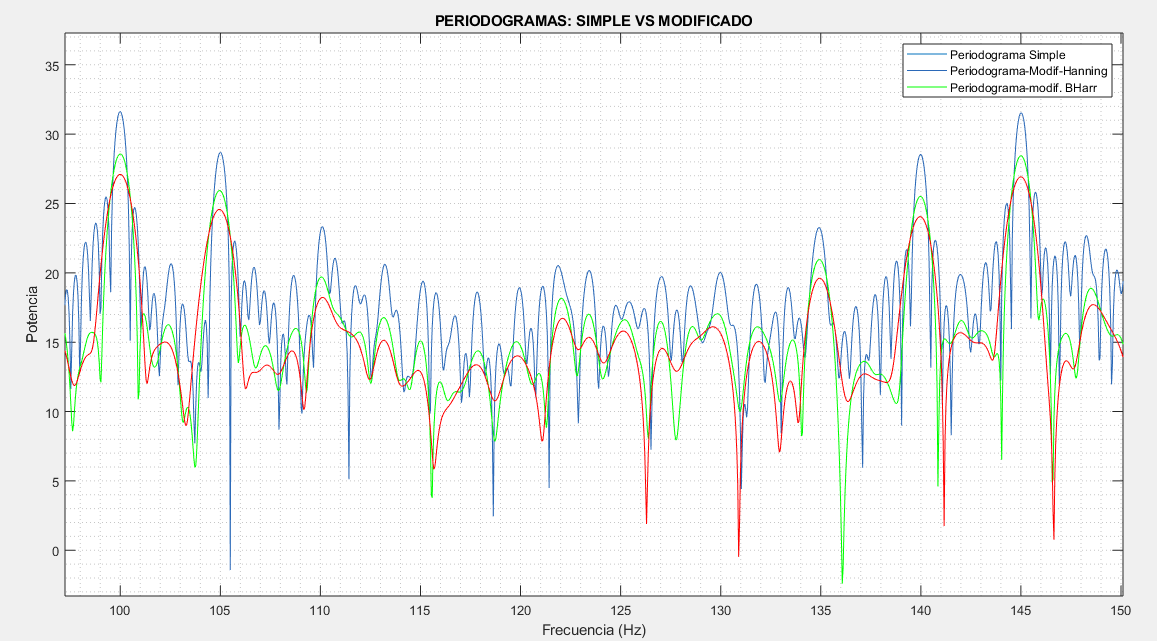
0.5\*sin(2\*pi\*100.8\*t) + 1\*sin(2\*pi\*101\*t);

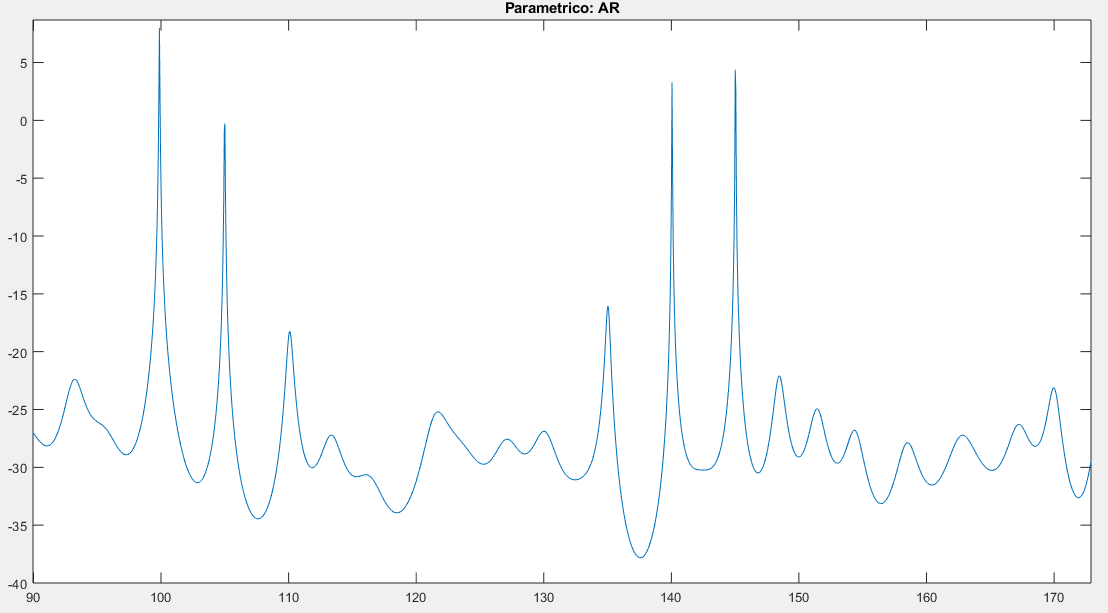


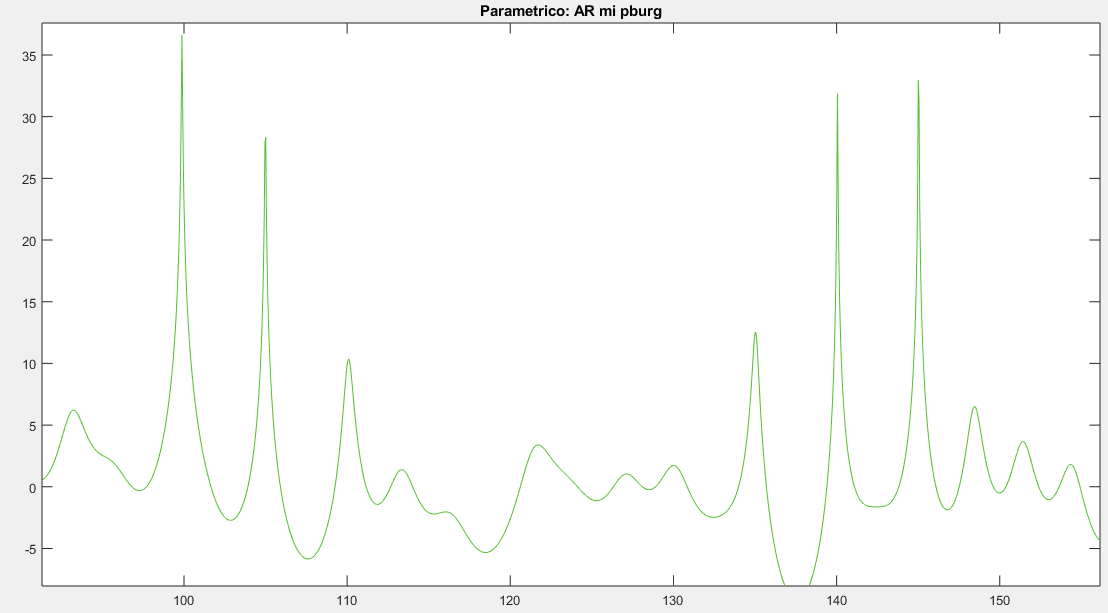




Ahora aumentamos la amplitud del ruido.







# Resultados y conclusiones

Como podemos ver en cada uno de los casos en el que probamos nuestra propia implementación de la función pburg , se obtiene el mismo resultado que la propia de Matlab , con lo que se comprueba su buen funcionamiento.

En el primer caso, variando las frecuencias se puede notar como ambas implementaciones de la pburg funcionan correctamente ya que logran obtener 6 de 10 armónicos que con métodos no paramétricos anteriormente aprendidos no se logran ver.

En el segundo caso, variando las amplitudes de la señal podemos ver resultados no favorables en su totalidad, ya que, el método no logra adquirir todas las componentes mientras que, con los anteriores, no paramétricos, si se logran obtener, por lo que se debería investigar de mejor manera como se podría obtener mejores resultados con este método.

En el tercer caso, nos muestra la capacidad de esta función ya que logra distinguir de una manera muy eficiente la señal del ruido y evidencia que este método se pude utilizar para trasmisión y recepción de señales.

De todo esto, se concluye que depende del tratamiento que se le va a dar a una señal para optar por un método paramétrico o no paramétrico en el mismo.

Se recomienda que en “mi\_pburg” se utilice la ventana por defecto, ya que, el uso de otras ventanas muestra un resultado distinto de la pburg original.

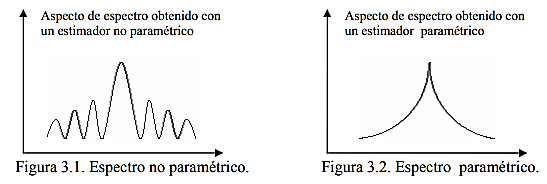
 En la obtención de la densidad espectral de potencias los métodos paramétricos representan una mejor herramienta con características como mayor resolución y mejora sobre los estimadores espectrales DFT tradicionales.

FIG. 2 Espectro no Paramétrico vs. Espectro Paramétrico.

# Referencias

|  |  |
| --- | --- |
| [1] | J. Guerrero Martínez, «Tema 5: Estimación Espectral,» 2010. [En línea]. Available: http://ocw.uv.es/ingenieria-y-arquitectura/1-5/ib\_material/IB\_T5\_OCW.pdf. [Último acceso: 21 Julio 2019]. |
| [2] | M. Bayas Paredes, «MODELOS AUTORREGRESIVOS DE ANÁLISIS ESPECTRAL,» Diciembre 1984. [En línea]. Available: https://bibdigital.epn.edu.ec/bitstream/15000/5721/1/T607.pdf. [Último acceso: 21 Julio 2019]. |
| [3] | K. Vos, «A Fast Implementation of Burg’s Method,» Agosto 2013. [En línea]. Available: https://www.opus-codec.org/docs/vos\_fastburg.pdf. [Último acceso: 24 Julio 2019]. |
| [4] | J. G. Proakis y D. G. Manolakis, Tratamiento Digital de Señales, principios, algoritmos y aplicaciones, Mexico: Prentice-Hall, 1997. |
| [5] | J. de la Torre Peláez, «Windowed Burg algorithms,» 6 Marzo 2002. [En línea]. Available: https://www.mathworks.com/matlabcentral/fileexchange/1565-windowed-burg-algorithms. [Último acceso: 24 Julio 2019]. |
| [6] | M. J. Ríos Aguaded, «Capítulo 3 ESTIMACIÓN ESPECTRAL PARAMÉTRICA UNIVARIABLE,» 3 Enero 2017. [En línea]. Available: http://bibing.us.es/proyectos/abreproy/11067/fichero/DOCUMENTACI%C3%93N+%252F06+-+Cap%C3%ADtulo+3+-+Estimaci%C3%B3n+espectral+param%C3%A9trica+univari.pdf+. [Último acceso: 30 Julio 2019]. |