ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ESTUDIO Y COMPARACIÓN DE MÉTODOS DE ESTIMACIÓN ESPECTRAL APLICADOS A UNA SEÑAL ARTIFICIAL

ESTUDIO DE LOS FUNDAMENTOS MATEMÁTICOS E IMPLEMENTACIÓN DEL MÉTODO PARAMÉTRICO AR POR MEDIO DEL ALGORITMO DE BURG Y SU COMPARACIÓN CON LOS MÉTODOS NO PARAMÉTRICOS BASADOS EN FOURIER (PERIODOGRAMA SIMPLE Y MODIFICADO)

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

ALEX JAVIER RAMOS VÁSQUEZ

alex.ramos@epn.edu.ec

DIRECTOR: M.Sc. RICARDO XAVIER LLUGSI CAÑAR

ricardo.llugsi@epn.edu.ec

DMQ, agosto 2023

CERTIFICACIONES

Yo, ALEX JAVIER RAMOS VÁSQUEZ declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

_____

ALEX JAVIER RAMOS VÁSQUEZ

Certifico que el presente trabajo de integración curricular fue desarrollado por ALEX JAVIER RAMOS VÁSQUEZ, bajo mi supervisión.

_____

M.Sc. RICARDO XAVIER LLUGSI CAÑAR

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

ALEX JAVIER RAMOS VÁSQUEZ

RICARDO XAVIER LLUGSI CAÑAR

DEDICATORIA

No habrá satisfacción más grande, que ver cumplida una meta de vida tan importante, después de haber invertido mucho esfuerzo, tiempo, dedicación, y hasta algunas lágrimas.

Dedico el presente trabajo de titulación a mi amada familia, quienes han constituido mi principal motor y motivación a lo largo de estos años.

A mis padres, Daysi y Patricio, quienes me han dado la vida, me han guiado y me han enseñado valores, y me han brindado todo su apoyo para poder concluir mis estudios.

A mi hermano, Santiago, quien se está formando en la vida, a quien espero servir de ejemplo de esfuerzo y perseverancia.

A mis abuelitos paternos, quienes están en el cielo, pero sé que estarían muy orgullosos y contentos de verme cumplir esta meta.

A mis abuelitos maternos, a quienes todavía tengo la suerte de tenerlos con vida, de quienes he aprendido el valor de la humildad y el esfuerzo.

El camino no ha sido fácil, hemos pasado por muchas dificultades, pero a pesar de ello, después de toda esta lucha, y con la bendición de Dios, sé que vendrán tiempos mejores para nosotros.

Con mucho cariño para ustedes,

Alex.

AGRADECIMIENTO

A Dios por darme la fortaleza para seguir adelante cada día, especialmente en los momentos difíciles.

A mis padres Daysi y Patricio, agradecerles de manera especial por su apoyo incondicional por ser lo más sagrado que tengo en la vida. Por ser mi motivación en momentos de desaliento y debilidad, gracias a sus enseñanzas, me han brindado ese respaldo que necesitaba para finalizar esta importante etapa de mi vida.

A mis tíos, Lida y Edy, quienes me han brindado acogida en su hogar, sin duda han sido un gran apoyo para poder seguir adelante en mi carrera.

A mis tíos, primos y todos mis familiares que me han brindado su apoyo durante el curso de mis estudios.

A mis amigos de la universidad, Jonathan Villarreal, Edison Cabrera, Michelle Chiluisa, Alex Páez, Daysi Guano, Ibeth Sotalín, Pauli Altamirano, Cristian Lema, Elvis Toscano gracias por los buenos momentos que hemos compartido dentro y fuera del aula de clases, ustedes han sido no sólo compañeros de clases, sino muy buenos amigos. Muchas gracias por estar ahí, por todas las risas compartidas, por estar ahí en las buenas y en las malas. Siempre los tendré presentes y les desearé muchos éxitos. De igual manera, agradezco al resto de amistades que quizá se me haya pasado por alto nombrar.

A mi mejor amigo, Hesnaider, quien ha estado apoyándome, y dándome consejos y ánimos en todo momento, desde el primer hasta el último día en la universidad, gracias por haber creído en mí y recordarme que soy capaz de siempre dar lo mejor de mí.

Al Dr. Ricardo Llugsi, director de este proyecto, quien me ha brindado su apoyo para concluir satisfactoriamente este trabajo. Y de manera especial, al Dr. Robin Álvarez, por brindarme su tiempo, y sus valiosos conocimientos, los cuales fueron de gran utilidad para el desarrollo del presente proyecto.

Finalmente, agradezco a la Escuela Politécnica Nacional, por ser mi casa de estudios, en donde me formado tanto profesionalmente, como a nivel personal, y donde he adquirido una gran cantidad de conocimientos, los cuales, espero poner en práctica para contribuir a la sociedad.

Infinitas gracias.

ÍNDICE DE CONTENIDO

RESUMEN

El presente estudio se enfrenta al problema de la falta de publicaciones respecto a la comparación de los métodos de estimación espectral, de los cuales existe una gran cantidad de ellos, pero lamentablemente, no existe algún trabajo investigativo que los compare en base a una señal artificial, y en donde también se aborden los problemas de la resolución en frecuencia y la capacidad de detección de componentes de amplitudes muy pequeñas, problemas intrínsecos a los métodos tradicionales basados en Fourier, como es el caso del periodograma simple y el periodograma modificado.

Para enfrentar este problema, se planteó una comparación entre el método paramétrico, utilizando el Algoritmo de Burg, y el no paramétrico, mediante el periodograma simple y modificado, empezando primero con una presentación didáctica, de los fundamentos teóricos y matemáticos, para posteriormente llevar a cabo la implementación paso a paso del algoritmo de Burg en Matlab.

Finalmente, los resultados alcanzados mostraron que, si bien el método paramétrico de estimación espectral de Burg logró una alta resolución en frecuencia y, una extraordinaria capacidad de detección de componentes de amplitud muy pequeña, no encontró correctamente las amplitudes de la señal artificial. Sin embargo, al variar la duración de la señal, el método de Burg logró una ventaja, puesto que mantuvo una resolución de frecuencia constante, a diferencia del periodograma, cuyo espectro experimentó una disminución de su resolución en frecuencia y su capacidad de detección, a medida que se redujo el tiempo de duración de la señal artificial.

PALABRAS CLAVE: Estimación espectral, Métodos autorregresivos, Periodograma, Algoritmo de Burg, Matlab.

ABSTRACT

The present study faces the problem of the lack of publications about the comparison of spectral estimation methods. There are many spectral estimation methods, but unfortunately, there is no research that compares them based on an artificial signal. In addition, there are no investigations that face the problems of frequency resolution and the ability to detect components of very small amplitudes. These problems are common in traditional methods based on Fourier (simple and modified periodogram).

In order to deal with this problem, a comparison is proposed between the Burg Algorithm, a parametric method, and periodogram (simple and modified), a non-parametric method. To achieve this, the mathematical and theoretical content about the algorithm was presented, in a didactic way. And then, the algorithm was implemented in Matlab step by step.

Finally, the results obtained showed that Burg's parametric method achieved a high frequency resolution and an extraordinary ability to detect very small amplitude components. Although, it did not correctly find the amplitudes of the artificial signal. However, Burg's method achieved an advantage since it maintained a constant frequency resolution while the duration of the artificial signal was reduced. In contrast, periodogram experienced a decrease in its frequency resolution and its ability to detect components as the duration of the artificial signal decreased.

KEYWORDS: Spectral estimation, Autoregressive methods, Periodogram, Burg algorithm, Matlab.

1 INTRODUCCIÓN

En el vasto campo de la ciencia y la ingeniería, la necesidad de analizar y comprender señales y datos complejos se ha vuelto cada vez más crucial. Los fenómenos presentes en la naturaleza están representados por señales formadas por varias componentes, que tienen diferentes frecuencias y amplitudes. La estimación espectral, un pilar fundamental en el procesamiento de señales, desempeña un papel esencial, al proporcionar información detallada sobre la distribución de energía o potencia de las diferentes componentes de una señal [1]–[3].

Los avances en el campo del procesamiento de señales han sido notables, ya que, a lo largo del tiempo, ha surgido una amplia cantidad de métodos de estimación espectral, cada uno con sus propias particularidades [4], [5]. Sin embargo, existe una brecha significativa en la literatura respecto a la evaluación comparativa de estos métodos, puesto que, aunque existen algunos estudios que dicen compararlos [6]–[8], no existe un estudio que muestre claramente las ventajas y desventajas de dichos métodos, y que permita conocer cuál de ellos es el mejor. Prueba de ello, es que en estudios modernos se sigue empleando distintos métodos sin tener claras dichas ventajas o desventajas [9]–[11].

El presente trabajo se centra en la implementación y comparación de dos enfoques de estimación espectral: el método paramétrico[1] AR (*Autoregressive*, Autoregresivo) a través del algoritmo de Burg y, los métodos no paramétricos[2] basados en Fourier, específicamente el periodograma simple y el periodograma modificado.

Este estudio surge en respuesta a una problemática relevante en el ámbito de la investigación: la falta de análisis exhaustivos y comparativos de distintas técnicas de estimación espectral en aplicaciones científicas y de ingeniería, lo cual, por ser un tema amplio y complejo, puede representar un problema muy grave para los investigadores de otras áreas, puesto que, un uso incorrecto de estas técnicas o la incomprensión de sus limitaciones puede tener graves consecuencias a la hora de interpretar resultados.

La presente investigación tiene como objetivo abordar esta brecha al implementar el método paramétrico AR a través del algoritmo de Burg y, compararlo rigurosamente con

[1] **Métodos paramétricos:** Son métodos de estimación espectral en los que se asume que la señal satisface un modelo específico, con un número limitado de parámetros. Estos modelos pueden ser, por ejemplo, procesos AR (*Autoregressive*, Autoregresivo), procesos MA (*Moving Average*, Media Móvil), o procesos ARMA (*Autoregressive Moving Average*, Autoregresivo de Media Móvil) [12] .

[2] **Métodos no paramétricos:** Son métodos de estimación espectral en los que no se hacen suposiciones específicas sobre la forma de la señal, o cómo los datos fueron generados. Los métodos basados en la Transformada de Fourier, como el periodograma simple y el periodograma modificado, son ejemplos de métodos no paramétricos [13].

los métodos no paramétricos basados en Fourier, utilizando el periodograma simple y el periodograma modificado. Al lograrlo, se espera no solo contribuir al conocimiento fundamental en el campo de la estimación espectral, sino también proporcionar una herramienta valiosa para investigadores y profesionales de diversas disciplinas científicas y de ingeniería. Los resultados y conclusiones obtenidos aportarán una base teórica para la toma de decisiones informadas, al seleccionar el método de estimación espectral más adecuado según el contexto específico de aplicación.

En los siguientes capítulos, se detallará de forma didáctica el fundamento teórico y matemático del algoritmo de Burg, así como el proceso de implementación en Matlab. Posteriormente, se presentarán los resultados de la comparación entre los métodos, en base a una señal artificial y, finalmente, se discutirán las implicaciones de estos hallazgos.

1.1 OBJETIVO GENERAL

El objetivo principal es exponer de manera didáctica la teoría e implementación en Matlab del método paramétrico de Burg, para estimar el espectro de potencias y compararlo con el método no paramétrico basado en Fourier (el periodograma simple y modificado).

1.2 OBJETIVOS ESPECÍFICOS

1. Exponer de forma didáctica la teoría referente al Método de Burg para estimación espectral.
2. Implementar paso a paso el método de Burg en Matlab, de manera didáctica.
3. Realizar las pruebas correspondientes para verificar el correcto funcionamiento de la implementación anterior, en base a señales artificiales que contengan varias componentes que incluyan tanto el problema de resolución en frecuencia como el de detección de componentes muy pequeñas.

1.3 ALCANCE

El presente Trabajo de Integración Curricular tiene como finalidad exponer de manera didáctica el método paramétrico de Burg para estimación espectral, implementar paso a paso el algoritmo en Matlab y realizar una comparativa con los métodos no paramétricos (periodograma simple y modificado). Para ello, el trabajo se llevará a cabo siguiendo las etapas descritas a continuación:

A. Fase Teórica:

Revisar y estudiar el estado del arte de los modelos paramétricos de estimación espectral y, específicamente respecto del método de Burg, así como los fundamentos matemáticos, los parámetros necesarios y la implementación del mismo, de manera didáctica.

B. Fase de implementación:

Desarrollar una función en Matlab que implemente el método de Burg, de acuerdo con los parámetros estudiados en la fase teórica. Adicionalmente, desarrollar un script principal que permita comparar el método paramétrico de Burg con los métodos no paramétricos tradicionales (periodograma simple y modificado). Se hará uso de una señal artificial multicomponente declarada en el script, así como de la función propia desarrollada, y de las funciones propias de Matlab.

C. Fase de evaluación y análisis de resultados:

A partir del script y la función desarrollada en la fase de implementación, comparar los resultados obtenidos con el método de Burg respecto de los métodos basados en Fourier, por medio de la aplicación a una señal artificial, considerando la resolución en frecuencia y la capacidad de detección de componentes de amplitud muy pequeña. Para ello, se realizarán simulaciones considerando varios escenarios de resolución en frecuencia y capacidad de detección de componentes de amplitud muy pequeña.

1.4 MARCO TEÓRICO

1.4.1 REPASO HISTÓRICO DE LOS DE LOS MÉTODOS DE ESTIMACIÓN ESPECTRAL

La estimación espectral es una herramienta esencial en el análisis de señales y sistemas en diversas disciplinas científicas y de ingeniería, ya que permite descomponer una señal en sus componentes de frecuencia, revelando patrones y características ocultas en los datos [1]–[3] . A lo largo del tiempo, diferentes enfoques se han desarrollado para estimar la densidad espectral de potencia de una señal, cada uno con sus propias ventajas y limitaciones [4], [5].

En 1822, Jean-Baptiste Joseph Fourier desarrolló la Transformada de Fourier, un hito significativo que permitió analizar señales no periódicas y continuas en el dominio de la frecuencia [14] . Esta técnica se convirtió en la base para muchos métodos posteriores de

estimación espectral. Posteriormente, en 1836, Sturm y Liouville extendieron la teoría de Fourier para el caso de funciones ortogonales arbitrarias [15].

Para el siglo XX, los avances en la estimación espectral empiezan con aportes de alta relevancia, por parte de J. W. Tukey en 1949, los cuales se consideran la contrapartida estadística del trabajo de Fourier desarrollado 142 años antes [15]. Luego, para el año 1965 ocurrió un gran avance y un hito importante en el campo de la estimación espectral, con la publicación del algoritmo de la FFT (*Fast Fourier Transform*, Transformada Rápida de Fourier) por J. S. Cooley & J. W. Tukey [16], de tal manera que, con este nuevo algoritmo computacionalmente eficiente, el campo del análisis espectral fue tomando mucha más relevancia.

Posteriormente, se da comienzo a los métodos modernos de estimación espectral, con los trabajos publicados por Burg en 1967, introduciendo el método MEM (*Maximum Entropy Method*, Método de Máxima Entropía) y Parzen, en 1968, con el método de estimación espectral autorregresiva, en donde ambos métodos son equivalentes y se caracterizan por suponer previamente un modelo, basado en parámetros, que generan la señal destinada a obtener su espectro de potencia [17] .

1.4.2 CLASIFICACIÓN DE LOS MÉTODOS DE ESTIMACIÓN ESPECTRAL

A lo largo de las investigaciones en el campo del análisis espectral, se han desarrollado distintas estrategias para estimar el espectro de potencia en el dominio de la frecuencia. Estos enfoques se agrupan en dos categorías principales: métodos paramétricos y métodos no paramétricos.

Cada uno de estos enfoques presenta características únicas y aplicaciones específicas, lo que brinda a los investigadores una gama de herramientas flexibles para abordar diversos escenarios de análisis espectral. En este contexto, es esencial comprender las diferencias y ventajas de estos dos enfoques clave para lograr una interpretación precisa y significativa de los datos.

A continuación, se describirán estos dos enfoques de estimación espectral.

1.4.2.1. Métodos No Paramétricos

Los métodos no paramétricos de estimación espectral son aquellos que no asumen previamente un modelo de cómo fue generada la señal y, trabajan directamente con los datos o muestras de la señal a estudiar [13].

Estos métodos, cuyas técnicas están basadas en Fourier, se conocen también como "métodos clásicos" o "métodos tradicionales" de estimación espectral [12]. Dentro de éstos se encuentran: el periodograma simple, periodograma modificado, método de Bartlett (1948), método de Blackman-Tukey (1958) y el método de Welch (1967) [13].

Los métodos no paramétricos para estimar el espectro de potencia son relativamente simples, de comprensión clara y se pueden calcular con facilidad mediante el algoritmo de la FFT. No obstante, estos enfoques demandan la existencia de extensos conjuntos de datos para lograr la resolución de frecuencia requerida en diversas aplicaciones [13].

1.4.2.2. Métodos Paramétricos

Los métodos paramétricos de estimación espectral son aquellos que se caracterizan por hacer suposiciones previas específicas sobre la forma de la señal a estudiar, es decir, se asume que la señal satisface un determinado modelo, con un número limitado de parámetros. Esos métodos también se conocen como "métodos basados en modelos" y, entran en la categoría de las técnicas modernas de estimación espectral [12].

El enfoque paramétrico de estimación espectral involucra la selección de un modelo adecuado, la estimación de sus parámetros y posteriormente, la inserción de estos valores obtenidos en las ecuaciones teóricas de la PSD (*Power Spectral Density*, Densidad Espectral de Potencia) [18].

Dentro de este enfoque, se tienen los modelos AR (*Autoregressive*, Autoregresivo), MA (*Moving Average*, Media Móvil), y ARMA (*Autoregressive Moving Average*, Autoregresivo de Media Móvil). Estos modelos asumen que la señal a ser analizada puede ser representada mediante una ecuación en diferencias, lo que implica que la salida puede ser formulada en términos de la entrada actual, así como de entradas y salidas anteriores. De este modo, cuando se tiene una serie temporal[3] de datos, el modelo constituye una herramienta para entender y, aún más importante, para la predicción de los futuros valores de la serie [20] . En este contexto, se identifican los siguientes tipos de modelos:

- **Modelo Autoregresivo (AR):** Este modelo implica una estimación o predicción del valor presente de una serie temporal de datos, a partir de los valores pasados de la misma [21].

[3] **Serie temporal:** Una serie temporal es una colección de observaciones de elementos de datos bien definidos, adquiridos a través de mediciones repetidas a lo largo de un período de tiempo [19].

- **Modelo de Media Móvil (MA):** Es un modelo de series temporales que establece que el valor siguiente se calcula como el promedio de todos los valores anteriores [22].
- **Modelo Autoregresivo de Media Móvil (ARMA):** Es un modelo generalizado que resulta de la combinación del modelo AR y MA [23].

En la Figura 1, se muestra de manera resumida la clasificación de los métodos de estimación espectral, de tal manera que se proporciona una perspectiva más clara sobre estos enfoques.

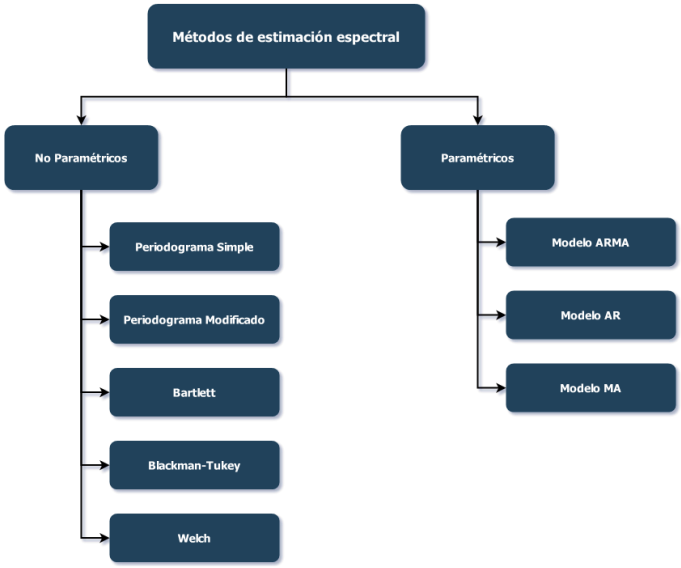


Figura 1. Clasificación de los métodos de estimación espectral.

1.4.3 ESTIMACIÓN ESPECTRAL PARAMÉTRICA

Los métodos tradicionales de estimación espectral, como el periodograma simple y el periodograma modificado, emplean la transformada de Fourier de datos enventanados para obtener el espectro de potencia. Estos enfoques asumen implícitamente que los valores de datos no observados o fuera de la ventana son nulos, lo cual es normalmente una suposición poco realista, que limita significativamente la resolución en frecuencia y la calidad del espectro estimado [18].

A menudo, se dispone de un mayor entendimiento del proceso del cual se toman las muestras de datos, o al menos se pueden hacer suposiciones más razonables que la de considerar ceros fuera de la ventana. Esto permite seleccionar un modelo más preciso para

el proceso que generó las muestras, o al menos, un modelo que se asemeje de manera efectiva al proceso real [18].

Un factor crucial detrás del interés en el presente estudio, en la estimación espectral basada en modelos paramétricos, radica en la aparente mejora en resolución y precisión en comparación con los estimadores espectrales tradicionales basados en Fourier, especialmente en situaciones con conjuntos de datos de corta duración. El grado de mejora en la resolución y fidelidad en el espectro depende de la capacidad de ajustar un modelo propuesto con un número reducido de parámetros [18]. Esta ventaja, se traduce en una menor influencia del efecto de la ventana en el estimador, lo que conlleva a eliminar la apariencia de sinc en el espectro estimado, como se aprecia en la Figura 2.

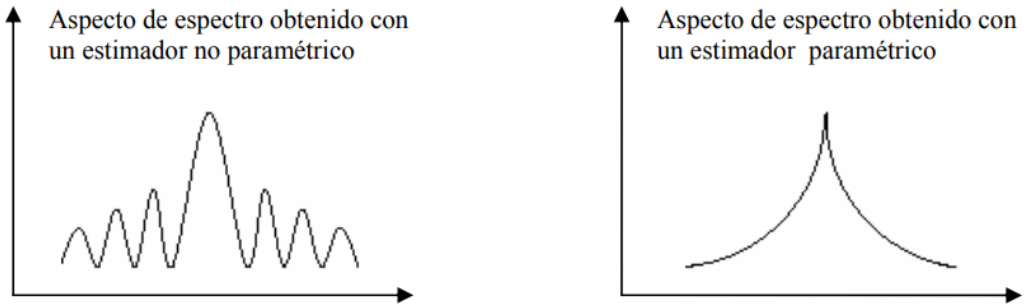


Figura 2. Comparación entre un espectro no paramétrico y paramétrico [24] .

En este contexto, la estimación espectral paramétrica es un proceso que sigue los siguientes pasos [24]:

1. **Elección de un modelo:** El primer paso consiste en la elección de un modelo adecuado para el proceso en cuestión. Esta decisión puede fundamentarse en el conocimiento previo acerca de cómo se ha generado el proceso o, en pruebas experimentales que respalden un buen funcionamiento del modelo específico elegido.
2. **Estimación de los parámetros:** Una vez que se ha seleccionado el modelo, se procede a la estimación de los parámetros correspondientes utilizando los datos disponibles, representados por la señal recibida $x[n]$. Esta señal se considera como un proceso estocástico[4], estacionario[5] y ergódico [6].

[4] **Proceso estocástico:** Es una colección de variables aleatorias indexadas en el tiempo, que representa la evolución de una cantidad a lo largo del tiempo de manera incierta. En otras palabras, es una secuencia de eventos o valores que evolucionan de manera aleatoria [25].
[5] **Proceso estacionario:** Es aquel en el que las propiedades estadísticas, como la media y la varianza, permanecen constantes en el tiempo [26].
[6] **Proceso ergódico:** Es aquel en el que las propiedades estadísticas calculadas a partir de una sola realización del proceso, son equivalentes a las propiedades estadísticas promediadas a lo largo del tiempo [17].

3. **Estimación del espectro de potencia:** Por último, se calcula el espectro de potencia del proceso estocástico, empleando la inserción de los parámetros del modelo estimado en la expresión teórica de la densidad espectral de potencia del modelo implicado.

1.4.4.1. Modelos temporales AR, MA y ARMA

En numerosos casos prácticos, los procesos aleatorios discretos en el tiempo pueden ser efectivamente representados mediante series temporales o mediante modelos de función de transferencia racional [18]. En este enfoque, se establece una relación entre una secuencia de entrada $u[n]$ y una secuencia de salida $x[n]$, que constituyen los datos del modelo, mediante la siguiente ecuación en diferencias lineal [18]:

$$x[n] = -\sum_{k=1}^{p} a_k \cdot x[n-k] + \sum_{k=0}^{q} b_k \cdot u[n-k] \quad (1)$$

El modelo descrito por la ecuación (1), se conoce como modelo ARMA de orden (p, q), de donde, los coeficientes a_k y b_k son coeficientes paramétricos, y se denominan coeficientes autorregresivos y coeficientes de medias móviles, respectivamente. Este modelo se denota como ARMA(p, q) el cual, representa el modelo lineal más general [13], [18].

A partir del modelo ARMA, se desprenden dos modelos más, que corresponden a los siguientes casos [18]:

i. Cuando todos los coeficientes a_k son nulos, excepto $a_0 = 1$, en el modelo ARMA, entonces, la ecuación (1) se transforma en:

$$x[n] = \sum_{k=0}^{q} b_k \cdot u[n-k] \quad (2)$$

y el proceso se denomina proceso MA de orden q, denotado como MA(q).

ii. Cuando todos los coeficientes b_k son nulos, excepto $b_0 = 1$, en el modelo ARMA, entonces, la ecuación (1) se transforma en:

$$x[n] = -\sum_{k=1}^{p} a_k \cdot x[n-k] + u[n] \tag{3}$$

y el proceso se denomina proceso AR de orden p, denotado como AR(p).

1.4.4.2. Función de Transferencia Racional y Transformada Z

En el contexto de la estimación espectral por métodos paramétricos, la transformada Z y las funciones de transferencia racional desempeñan un papel importante, ya que constituyen una herramienta para modelar y analizar señales [13].

La transformada Z, una herramienta poderosa en el análisis de señales discretas, permite representar señales en el dominio de la frecuencia compleja mediante polinomios en la variable Z. Esta representación, combinada con las funciones de transferencia racional, que son expresiones algebraicas que relacionan la entrada y la salida de sistemas lineales, proporciona un conocimiento teórico para la construcción de modelos ARMA y, por tanto, también para modelos MA y AR [13], [18].

La Transformada Z de una señal discreta $x[n]$ se define como [13]:

$$X[z] = \sum_{n=-\infty}^{\infty} x[n] z^{-n} \tag{4}$$

Donde z es una variable compleja.

Una familia importante de Transformadas Z, constituye aquellas para las que $X(z)$ es una función racional, es decir, una razón de dos polinomios. Así, para un modelo ARMA, definido en la ecuación (1), la función de transferencia $H(z)$, que caracteriza el sistema lineal, entre la entrada $u[n]$ y la salida $x[n]$, es la función racional [18]:

$$H[z] = \frac{B(z)}{A(z)} \tag{5}$$

De donde, $A(z)$ es la Transformada Z de la rama AR, definida como:

$$A[z] = \sum_{k=0}^{p} a[k] z^{-k} \tag{6}$$

y $B(z)$ es la Transformada Z de la rama MA, dada por:

$$B[z] = \sum_{k=0}^{q} b[k] z^{-k} \tag{7}$$

1.4.4.3. Correlación Cruzada y Autocorrelación

En el ámbito de la estimación espectral por medio de métodos paramétricos, es importante conocer el concepto de correlación, puesto que las funciones matemáticas, asociadas a este concepto, permiten establecer relaciones entre los parámetros $\{a_k\}$ y $\{b_k\}$ de un modelo ARMA y su función de transferencia racional $H(z)$.

La correlación se define como una medida de similitud entre dos señales. Se tienen dos tipos de correlaciones [27]:

- **Correlación Cruzada:** La correlación cruzada se refiere a la medida de coherencia o similitud entre dos señales, funciones o formas de onda distintas. Esta medida, se realiza comparando una señal con una versión retardada de otra señal.
- **Autocorrelación:** La autocorrelación se define como la medida de similitud o coherencia entre una señal y su versión retardada en el tiempo. Por tanto, la autocorrelación es la correlación de una señal consigo misma.

Suponiendo que se tienen dos secuencias, $x[n]$ y $y[n]$, correspondientes a dos señales reales, cada una de ellas con energía finita, la correlación cruzada de $x[n]$ y $y[n]$, denotada por $r_{xy}(m)$, se define por [13]:

$$r_{xy}[m] = \sum_{n=-\infty}^{\infty} x[n]y[n-m] = \sum_{n=-\infty}^{\infty} x[n+m]y[n] \quad m = 0, \pm 1, \pm 2, \ldots \tag{8}$$

En la ecuación (8), el valor de m indica el retardo temporal, y los subíndices xy indican las secuencias que se están correlacionando.

En el caso especial donde $y[n] = x[n]$, se tiene la autocorrelación de $x[n]$, la cual se define por [13]:

$$r_{xx}[m] = \sum_{n=-\infty}^{\infty} x[n]x[n-m] \quad m = 0, \pm 1, \pm 2, \ldots \tag{9}$$

Al tratar con secuencias de duración finita, como es el caso práctico, se acostumbra a expresar la autocorrelación y la correlación cruzada en términos de los límites finitos de la

suma [13]. En particular, si $x[n]$ y $y[n]$ son señales causales (aquellas definidas en un tiempo positivo) de longitud N, la correlación cruzada y la autocorrelación se pueden expresar como [20]:

$$r_{xy}[m] = \sum_{n=-(N-1)}^{N-1} x[n]y[n-m];\ \forall m \in [-(N-1),(N-1)] \tag{10}$$

$$r_{xx}[m] = \sum_{n=-(N-1)}^{N-1} x[n]x[n-m];\ \forall m \in [-(N-1),(N-1)] \tag{11}$$

1.4.4.4. Modelo Autorregresivo (AR)

El modelo paramétrico AR, objeto del presente estudio, destaca como el modelo predominante para la estimación de series temporales. Esto radica en la capacidad de lograr estimaciones precisas de los parámetros AR mediante la resolución de un conjunto de ecuaciones lineales. El estimador espectral AR también es reconocido con otros términos, como estimador espectral de máxima entropía y estimador espectral de predicción lineal [24]. Además, el modelo AR es adecuado para representar espectros con picos estrechos, es decir, tiene una alta resolución en frecuencia [13].

Bajo este modelo, una señal discreta $x[n]$, se puede formular como la salida resultante de un sistema vinculado a través de la siguiente relación:

$$\underbrace{x[n]}_{\text{Muestra actual}} = -\sum_{k=1}^{p} \underbrace{a_k}_{\text{Coeficientes paramétricos}} \underbrace{x[n-k]}_{\text{Muestras pasadas}} + \underbrace{w[n]}_{\text{Ruido Gaussiano}} \tag{12}$$

De donde:

- $x[n]$: Son las muestras actuales de la señal discreta.
- $x[n-k]$: Son las muestras pasadas de la señal discreta.
- $a_k,\ k = 1,2,\dots,p$: Son los coeficientes paramétricos (autorregresivos) del modelo AR(p).
- $w[n]$: Representa ruido blanco Gaussiano, con media igual a cero y varianza igual a σ^2

1.4.4.5. Ecuaciones de Yule-Walker

Los conceptos de correlación, desarrollados anteriormente, juegan un rol importante, puesto que, constituyen un nexo para establecer nuevas expresiones que permitan determinar los coeficientes a_k y b_k , de un modelo ARMA y, por consiguiente, también de un modelo AR y MA.

Para los desarrollos siguientes, es necesario definir la función de autocorrelación, desde una perspectiva estadística. Por tanto, en [5], se dice que en un proceso aleatorio estacionario, la función de autocorrelación se define como:

$$r_{xx}[k] = E\{x[n+k]x[n]\} = E\{x[n]x[n-k]\} \tag{13}$$

Donde $E\{\cdot\}$, representa el operador estadístico del valor esperado.

Bajo este escenario, un proceso ARMA es considerado como un proceso aleatorio estacionario, que incluye una secuencia de ruido Gaussiano $w[n]$, denotado por la siguiente expresión [13]:

$$x[n] = -\sum_{k=1}^{p} a_k \cdot x[n-k] + \sum_{k=0}^{q} b_k \cdot w[n-k] \tag{14}$$

Al multiplicar la ecuación (13) por $x[n-m]$, y aplicar el valor esperado, $E\{\cdot\}$, se obtiene:

$$E\{x[n]x[n-m]\} = -\sum_{k=1}^{p} a_k \cdot E\{x[n-k]x[n-m]\} + \sum_{k=0}^{q} b_k \cdot E\{w[n-k]x[n-m]\} \tag{15}$$

Por tanto, se obtiene la relación:

$$r_{xx}[m] = -\sum_{k=1}^{p} a_k \cdot r_{xx}[m-k] + \sum_{k=0}^{q} b_k \cdot r_{wx}[m-k] \tag{16}$$

Donde r_{wx} es la correlación cruzada entre w[n] y x[n].

El proceso modelado en la ecuación (14), puede ser visto como un filtro causal, con función de transferencia $H[z]$, por el cual entra la secuencia de ruido $w[n]$ y a la salida se obtiene $x[n]$ [13]. En la Figura 3, se observa de manera gráfica:

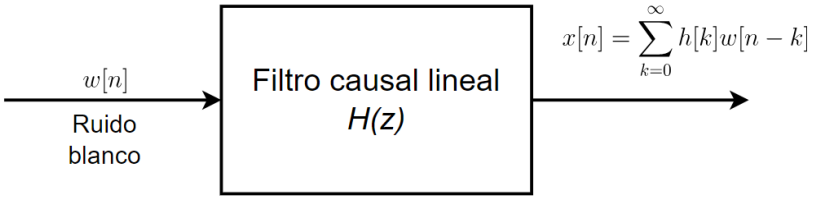


Figura 3. Modelo de filtro causal para la señal $x[n]$ [13].

La correlación cruzada r_{wx} se relaciona con la respuesta al impulso del filtro. Por lo que, desarrollando la expresión de r_{wx} , se tiene [13]:

$$\begin{aligned} r_{wx}[m] &= E\{x[n]w[n+m]\} \\ &= E\left\{\sum_{k=0}^{\infty} h[k]w[n-k]w[n+m]\right\} \\ &= \sigma_w^2 h[-m] \end{aligned} \tag{17}$$

En el último paso, el hecho de que se asuma que se tiene ruido blanco, permite determinar que [13]:

$$r_{xx}[m] = \begin{cases} 0, & m > 0 \\ \sigma_w^2 h[-m], & m \leq 0 \end{cases} \tag{18}$$

Al combinar las ecuaciones (16) y (18), se obtienen las siguientes expresiones [13]:

$$r_{xx}[m] = \begin{cases} -\sum_{k=1}^{p} a_k \cdot r_{xx}[m-k], & m > q \\ -\sum_{k=1}^{p} a_k \cdot r_{xx}[m-k] + \sigma_w^2 \sum_{k=0}^{q-m} h[k]b_{k+m}, & 0 \leq m \leq q \\ r_{xx}[-m], & m < 0 \end{cases} \tag{19}$$

Para un proceso ARMA, las relaciones descritas por la ecuación (19), representan una relación entre la autocorrelación $r_{xx}[m]$ y sus parámetros $\{a_k\}$,$\{b_k\}$. Luego, para un proceso AR, las ecuaciones se simplifican a la siguiente expresión [13]:

$$r_{xx}[m] = \begin{cases} -\sum_{k=1}^{p} a_k \cdot r_{xx}[m-k], & m > 0 \\ -\sum_{k=1}^{p} a_k \cdot r_{xx}[-k] + \sigma_w^2, & m = 0 \\ r_{xx}[-m], & m < 0 \end{cases} \tag{20}$$

El conjunto de ecuaciones (20), se denominan ecuaciones de Yule-Walker, las mismas que se pueden representar de forma matricial, de la siguiente manera [13]:

$$\underbrace{\begin{pmatrix} r_{xx}[0] & r_{xx}[1] & r_{xx}[2] & \cdots & r_{xx}[p-1] \\ r_{xx}[1] & r_{xx}[0] & r_{xx}[1] & \cdots & r_{xx}[p-2] \\ r_{xx}[2] & r_{xx}[1] & r_{xx}[0] & \cdots & r_{xx}[p-3] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{xx}[p-1] & r_{xx}[p-2] & r_{xx}[p-3] & \cdots & r_{xx}[0] \end{pmatrix}}_{\boldsymbol{R_{xx}}} \underbrace{\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_p \end{pmatrix}}_{\boldsymbol{a}} = -\underbrace{\begin{pmatrix} r_{xx}[1] \\ r_{xx}[2] \\ r_{xx}[3] \\ \vdots \\ r_{xx}[p] \end{pmatrix}}_{\boldsymbol{r_{xx}}} \tag{21}$$

Sistema de ecuaciones Yule-Walker

La ecuación (21) describe un sistema lineal de ecuaciones, llamado sistema de ecuaciones de Yule-Walker, cuyos elementos, se describen a continuación:

- $\boldsymbol{R_{xx}}$: Representa la matriz de autocorrelación de $x[n]$.
- $\boldsymbol{a}$: Representa el vector de coeficientes paramétricos del modelo AR(p)
- $\boldsymbol{r_{xx}}$: Representa el vector de autocorrelaciones de $x[n]$ desde 1 hasta p

Del conjunto de ecuaciones (20), se tiene que:

$$r_{xx}[0] = -\sum_{k=1}^{p} a_k \cdot r_{xx}[-k] + \sigma_w^2 \tag{22}$$

Por lo que el sistema de ecuaciones de Yule-Walker, también se puede representar como:

$$\begin{pmatrix} r_{xx}[0] & r_{xx}[1] & r_{xx}[2] & \cdots & r_{xx}[p] \\ r_{xx}[1] & r_{xx}[0] & r_{xx}[1] & \cdots & r_{xx}[p-1] \\ r_{xx}[2] & r_{xx}[1] & r_{xx}[0] & \cdots & r_{xx}[p-2] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{xx}[p] & r_{xx}[p-1] & r_{xx}[p-2] & \cdots & r_{xx}[0] \end{pmatrix} \begin{pmatrix} 1 \\ -a_1 \\ -a_2 \\ \vdots \\ -a_p \end{pmatrix} = \begin{pmatrix} \sigma_w^2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{23}$$

1.4.4.6. Algoritmo de Levinson-Durbin

Las ecuaciones de Yule-Walker desempeñan un rol fundamental en la estimación espectral en base a un modelo AR(p), puesto que, la resolución de este sistema de ecuaciones permite obtener los parámetros de este modelo, necesarios para la estimación de la PSD de la señal.

Al reescribir el sistema de ecuaciones (21), en forma compacta, se tiene:

$$\boldsymbol{R}_{xx}\boldsymbol{a} = -\boldsymbol{r}_{xx} \tag{24}$$

En la ecuación (24) se observa que el vector de coeficientes paramétricos $\boldsymbol{a}$, se puede obtener invirtiendo la matriz $\boldsymbol{R}_{xx}$. De donde:

$$\boldsymbol{a} = \boldsymbol{R}_{xx}^{-1}(-\boldsymbol{r}_{xx}) \tag{25}$$

Evidentemente, para órdenes altos de p, la complejidad de los cálculos para hallar la inversa de $\boldsymbol{R}_{xx}$ aumenta, lo cual, implicaría también un aumento en el tiempo de procesamiento, en una implementación por computadora.

No obstante, la matriz de autocorrelación $\boldsymbol{R}_{xx}$ tiene dos propiedades especiales: es hermitiana[7] y Toeplitz[8] [13]. Estas propiedades fueron aprovechadas por Levinson (1947), quien introdujo un algoritmo que permite obtener los parámetros del modelo AR de manera recursiva, el cual, fue mejorado por Durbin (1960), haciéndolo el doble de rápido. En consecuencia, la colaboración de ambos investigadores dio origen a un algoritmo recursivo altamente eficiente desde el punto de vista computacional, conocido como el algoritmo de Levinson-Durbin [17]. Las ecuaciones que conforman este algoritmo, se presentarán en el siguiente capítulo.

1.4.4.7. Algoritmo de Burg

En 1968 Burg [29] describió un método recursivo para estimar los parámetros del modelo AR, el cual se basa en la minimización de errores de predicción hacia adelante y hacia atrás, con la restricción de que los parámetros AR satisfacen la recursión de Levinson-Durbin.

Para comprender la esencia del algoritmo de Burg, es importante familiarizarse con los conceptos clave que lo sustentan. Los predictores lineales desempeñan un papel central, puesto que constituyen modelos matemáticos que permiten estimar un valor futuro de la señal en función de sus valores pasados [13]. En este sentido, en el algoritmo de Burg, se busca encontrar estos predictores de manera óptima para capturar la información espectral de la señal de entrada.

---

[7] **Matriz hermitiana:** Una matriz hermitiana o hermítica, es una matriz cuadrada con números complejos que se caracteriza por ser igual a su transpuesta conjugada. Matemáticamente, si $A = [a_{ij}]$ es hermitiana entonces $A = A^H$, donde A^H es la transpuesta conjugada de A [28].

[8] **Matriz Toeplitz:** Una matriz Toeplitz es aquella en la que todos los elementos de cualquier diagonal son constantes o iguales [18].

El proceso de estimación se lleva a cabo iterativamente, refinando continuamente los coeficientes del predictor lineal para reducir los errores de predicción hacia adelante y hacia atrás [13], [18]. El error de predicción hacia adelante se refiere a la diferencia entre el valor real y el valor estimado de la señal en un instante futuro, mientras que el error de predicción hacia atrás representa la diferencia entre el valor real y el valor estimado en un instante anterior. La combinación de estos dos enfoques de error proporciona una estimación más completa y precisa de la señal, lo que distingue al algoritmo de Burg y lo hace particularmente valioso en el análisis espectral [13].

Este algoritmo constituye la base del presente estudio, de tal manera que, los aspectos matemáticos y la implementación asociada al mismo, se analizarán en el siguiente capítulo.

1.4.4 SIGNAL PROCESSING TOOLBOX DE MATLAB

Signal Processing Toolbox de MATLAB, es un paquete de herramientas que proporciona una variedad de funciones y aplicaciones que permiten gestionar, analizar, preprocesar y extraer características de señales que han sido muestreadas de manera uniforme o no uniforme. El toolbox también dispone de utilidades para para diseñar y analizar filtros, remuestrear, suavizar, eliminar tendencias y estimar el espectro de potencia de las señales [30]. Este paquete de Matlab constituye una herramienta importante dentro del presente estudio, puesto que permitirá implementar el cálculo del espectro de una señal, mediante métodos tradicionales (usando la FFT), por medio del periodograma simple y el periodograma modificado, así como mediante métodos AR, por medio del algoritmo de Burg.

Las funciones más relevantes del Signal Processing Toolbox, que serán utilizadas en presente trabajo, se describen a continuación:

- **Función FFT:** Es una función que obtiene la DFT (*Discrete Fourier Transform*, Transformada Discreta de Fourier) de una señal de entrada, usando el algoritmo de la FFT [31].
- **Función RECTWIN:** Es una función que obtiene una ventana de tipo rectangular [32].
- **Función BLACKMANHARRIS:** Es una función que obtiene una ventana Blackman-Harris de cuatro términos (BH-4) [33].
- **Función PBURG:** Es una función que estima la PSD de una señal, utilizando el método de Burg [34].
- **Función ARBURG:** Es una función que retorna los parámetros correspondientes a un modelo AR(p) [35].

2 METODOLOGÍA

En este capítulo se aborda de manera detallada los fundamentos matemáticos del método de estimación espectral de Burg, los cuales servirán de base para su implementación en Matlab, y con esto lograr tener las herramientas para el posterior análisis, y de esta manera, cumplir con los objetivos planteados.

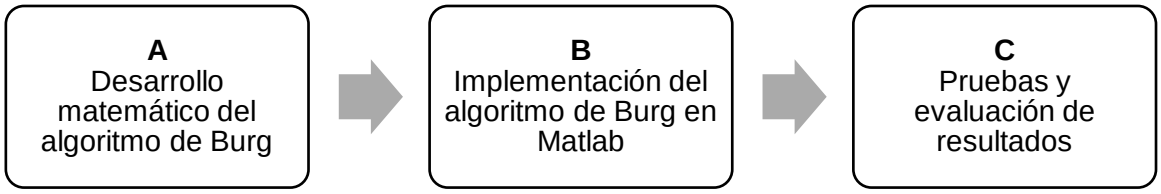


Figura 4. Diagrama de bloques de la metodología propuesta.

La metodología propuesta sigue tres etapas en forma secuencial, como se observa en la **Figura *4***. A continuación, se describen las etapas a llevarse a cabo:

A. En esta etapa, se revisará de forma minuciosa los aspectos matemáticos del método de Burg, mediante un desarrollo paso a paso, presentado de forma didáctica, para ayudar al lector a tener una mejor comprensión de lo que se va a implementar.

B. En esta etapa, se llevará a cabo la implementación paso a paso del método de Burg en Matlab. Para lo cual, en base a la teoría revisada en la etapa anterior, se desarrollará una función en formato *.m*, denominada *mi_pburg*, la cual será invocada en un script principal, para poder generar las gráficas comparativas de este método, junto con los métodos tradicionales de estimación espectral (periodograma simple y modificado). Dentro del script principal, se procederá a generar una señal artificial multicomponente, que aborde los problemas de resolución en frecuencia y la capacidad de detección de componentes de amplitud muy pequeña.

C. En esta última etapa, una vez se disponga del script y la función desarrollada en la fase anterior, se procederá a realizar las pruebas y simulaciones en Matlab, obteniendo las gráficas que permitan realizar el análisis de los resultados obtenidos, los cuales, serán presentados en el siguiente capítulo.

2.1 CONCEPTOS ASOCIADOS AL ALGORITMO DE BURG

Antes de adentrarse en el análisis matemático del algoritmo de Burg, es necesario abordar conceptos adicionales asociados al algoritmo, puesto que, esto permitirá una mejor comprensión del desarrollo matemático posterior, así como de la implementación a realizarse.

2.1.1. PREDICCIÓN LINEAL DE LOS PROCESOS AR

La predicción lineal es un tema importante en el procesamiento digital de señales, especialmente en el ámbito de los modelos autorregresivos (AR). En este contexto, se considera el problema de predecir linealmente el valor de un proceso aleatorio estacionario, ya sea hacia adelante o hacia atrás en el tiempo [13]. Esta formulación conduce a estructuras denominadas filtros de predicción, las cuales establecen vínculos con los modelos AR y, por consiguiente, con el algoritmo de Burg. Así, se empieza definiendo los conceptos de predicción lineal hacia adelante y predicción lineal hacia atrás.

- **Predicción Lineal Hacia Adelante:** La predicción lineal hacia adelante se refiere a la capacidad de predecir el valor futuro de la señal utilizando una combinación lineal de sus valores anteriores. En otras palabras, se utiliza una combinación ponderada de los valores pasados de la señal para predecir su siguiente valor. Esta predicción tiene como objetivo estimar cómo evolucionará la señal en el futuro, en función de su comportamiento pasado [13].

- **Predicción Lineal Hacia Atrás:** La predicción lineal hacia atrás implica estimar los valores pasados de una señal en función de sus valores futuros y presentes. Se utiliza una combinación lineal de los valores actuales y futuros de la señal para estimar sus valores pasados. Esto permite entender cómo evolucionaron los valores anteriores de la señal en función de la información disponible en el futuro [13].

Para el caso de la predicción lineal hacia adelante, se define el predictor lineal hacia adelante de un paso, de orden p, a aquel que forma la predicción del valor $x[n]$ mediante una combinación lineal ponderada, de sus valores anteriores $x[n-1], x[n-2], \dots, x[n-p]$. Esto se representa por la siguiente expresión [13]:

$$\hat{x}[n] = -\sum_{k=1}^{p} a_p[k] \cdot x[n-k] \tag{26}$$

De donde $\{-a_p[k]\}$ representan los pesos en la combinación lineal. Estos pesos se denominan coeficientes de predicción del predictor lineal de un paso hacia adelante, de orden p.

Luego, la diferencia entre el valor $x[n]$ y el valor predicho de $x[n]$, se denomina, error de predicción hacia adelante, denotado como $f_p[n]$, el cual, se define por:

$$\begin{aligned} f_p[n] &= x[n] - \hat{x}[n] \\ &= x[n] + \sum_{k=1}^{p} a_p[k] \cdot x[n-k] \end{aligned} \tag{27}$$

Por otro lado, para el caso de la predicción lineal hacia atrás, asumiendo que, a partir de un proceso aleatorio estacionario, se tiene una secuencia de datos $x[n], x[n-1], \dots, x[n-p+1]$, se desea predecir el valor de $x[n-p]$. En este caso, se define el predictor lineal hacia atrás de un paso de orden p, de la siguiente manera [13]:

$$\hat{x}[n-p] = -\sum_{k=1}^{p} b_p[k] \cdot x[n-k] \tag{28}$$

Donde $b_p[k]$ son los coeficientes de predicción del predictor lineal de un paso hacia atrás, de orden p.

La diferencia entre el valor de $x[n-p]$ y el estimado $\hat{x}[n-p]$ se denomina error de predicción hacia atrás, y se denota como $g_p[n]$:

$$\begin{aligned} g_p[n] &= x[n-p] + \sum_{k=0}^{p-1} b_p[k] \cdot x[n-k] \\ &= \sum_{k=0}^{p} b_p[k] \cdot x[n-k], \qquad b_p[p] = 1 \end{aligned} \tag{29}$$

Los coeficientes de ponderación en el predictor lineal hacia atrás $b_p[k]$, son los conjugados complejos de los coeficientes para el predictor lineal hacia adelante, pero ocurren en orden inverso. De esta manera, se tiene que [13]:

$$b_p[k] = a_p^*[p-k], \qquad k = 0,1, \dots, p \tag{30}$$

2.1.2. ECUACIONES RECURSIVAS DE LEVINSON-DURBIN

Bajo el contexto de los predictores lineales, si $x[n]$ es un proceso $AR(p)$, se dice que los coeficientes óptimos de predicción lineal de un paso, están dados por la solución de las ecuaciones de Yule-Walker [18].

En el algoritmo de Levinson-Durbin, se plantea el sistema de ecuaciones de Yule-Walker de la siguiente manera [36]:

$$\begin{pmatrix} r_{xx}[0] & r_{xx}[1] & r_{xx}[2] & \cdots & r_{xx}[p] \\ r_{xx}[1] & r_{xx}[0] & r_{xx}[1] & \cdots & r_{xx}[p-1] \\ r_{xx}[2] & r_{xx}[1] & r_{xx}[0] & \cdots & r_{xx}[p-2] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{xx}[p] & r_{xx}[p-1] & r_{xx}[p-2] & \cdots & r_{xx}[0] \end{pmatrix} \begin{pmatrix} 1 \\ -a_p[1] \\ -a_p[2] \\ \vdots \\ -a_p[p] \end{pmatrix} = \begin{pmatrix} \sigma_p \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{31}$$

Bajo el planteamiento de la ecuación (31), el algoritmo calcula de manera recursiva los conjuntos de parámetros $\{a_1[1], \sigma_1\}, \{a_2[1], a_2[2], \sigma_2\}, \dots, \{a_p[1], a_p[2], \dots, a_p[p], \sigma_p\}$. De manera que, el conjunto final en el orden p representa la solución deseada de las ecuaciones de Yule-Walker. De esta forma, si $x[n]$ es un proceso $AR(p)$, entonces $a_p[i] = a[i], \forall i = 1,2, \dots p$ y $\sigma_p = \sigma^2$ [18].

Utilizando notación matricial, el sistema de ecuaciones dado en la ecuación (31), se puede representar como [36]:

$$\boldsymbol{R}_{xx}[\boldsymbol{i}]\boldsymbol{a}[\boldsymbol{i}] = \boldsymbol{e}[\boldsymbol{i}] \tag{32}$$

Explicado en otras palabras, lo que hace el algoritmo de Levinson-Durbin es, obtener la solución *i*-ésima a partir del orden $[i-1]$. Es decir, la ecuación (32) se calcula en base a [36]:

$$\boldsymbol{R}_{xx}[\boldsymbol{i}-\boldsymbol{1}]\boldsymbol{a}[\boldsymbol{i}-\boldsymbol{1}] = \boldsymbol{e}[\boldsymbol{i}-\boldsymbol{1}] \tag{33}$$

A continuación, se sintetizan los pasos y ecuaciones que describen el algoritmo de Levinson-Durbin.

En primer lugar, se inicializan los siguientes parámetros [18]:

$$a_1[1] = -\frac{r_{xx}[1]}{r_{xx}[0]} \tag{34}$$

$$\sigma_1 = (1 - |a_1[1]|^2)\, r_{xx}[0] \tag{35}$$

Luego, empieza la recursión para $k = 2,3, \dots, p$,dada por las siguientes ecuaciones:

$$a_k[k] = \frac{r_{xx}[k] + \sum_{l=1}^{k-1} a_{k-1}[l] r_{xx}[k-l]}{\sigma_{k-1}} \tag{36}$$

$$a_k[i] = a_{k-1}[i] + a_k[k] a_{k-1}^*[k-i], \quad i = 1,2, \dots, k-1 \tag{37}$$

$$\sigma_k = (1 - |a_k[k]|^2)\sigma_{k-1} \tag{38}$$

Los coeficientes $a_k[k]$se denominan coeficientes de reflexión[9]. Estos coeficientes son de gran relevancia, puesto que el algoritmo de Burg se basa en el cálculo de estos coeficientes para obtener el conjunto $\{a_p[1], a_p[2], \dots, a_p[p], \sigma_p\}$, y con ello, obtener la PSD estimada.

2.2 DESARROLLO MATEMÁTICO DEL ALGORITMO DE BURG

Una vez abordados los conceptos y la teoría asociada a los métodos paramétricos de estimación espectral, específicamente del modelo paramétrico autorregresivo AR(p). Se procederá a detallar la teoría matemática referente al algoritmo, en la presente sección.

El método de Burg para estimar los parámetros AR es un algoritmo recursivo, en el orden del modelo (p) y, se basa en la minimización de los errores hacia adelante y hacia atrás en los predictores lineales, con la restricción de que los parámetros AR satisfacen la recursión de Levinson-Durbin. A continuación, se indica el procedimiento matemático del algoritmo de Burg [13], [18], [37]:

Para empezar el desarrollo del algoritmo, se procede a suponer un conjunto de muestras $x[n]$, a partir del modelo AR, como el que se formuló en la ecuación (12), con $n = 0,1, \dots, N-1$, y luego, se definen los predictores lineales de un paso, hacia adelante y hacia atrás, de orden p.

$$\hat{x}[n] = -\sum_{i=1}^{k} a_k[i] \cdot x[n-i] \quad \text{(Predictor lineal hacia adelante)} \tag{39}$$

$$\hat{x}[n-k] = -\sum_{i=1}^{k} a_k^*[i] \cdot x[n+i-k] \quad \text{(Predictor lineal hacia atrás)} \tag{40}$$

[9] **Coeficientes de reflexión:** Los coeficientes de reflexión son valores resultantes de la aplicación del algoritmo de Levinson-Durbin a una señal, los cuales describen la interacción de la señal con un modelo de filtro. Los coeficientes de reflexión reciben su nombre debido a la analogía con el fenómeno físico de la reflexión de ondas. Por ejemplo, en casos prácticos, los coeficientes de reflexión se emplean para modelar fenómenos como el tracto vocal humano y las capas de la tierra [13].

A continuación, se establecen los correspondientes errores de predicción hacia adelante y hacia atrás, $\hat{f}_k[n]$ y $\hat{g}_k[n]$, respectivamente.

Para el error de predicción hacia adelante, se tiene que:

$$\begin{aligned} \hat{f}_k[n] &= x[n] - \hat{x}[n] \\ &= x[n] + \sum_{i=1}^{k} a_k[i] \cdot x[n-i] \end{aligned} \tag{41}$$

Para el error de predicción hacia atrás, en cambio:

$$\begin{aligned} \hat{g}_k[n] &= x[n-k] - \hat{x}[n-k] \\ \hat{g}_k[n] &= x[n-k] + \sum_{i=1}^{k} a_k^*[i] \cdot x[n-k+i] \end{aligned} \tag{42}$$

El término, $a_k[i]$ representa los coeficientes de predicción, en donde:

$$0 \leq i \leq k-1 \text{ y } k = 1,2,\dots,p.$$

Seguidamente, se define el error de mínimos cuadrados ε_p, como la suma de las energías de los errores de predicción en adelanto y en retardo, de la siguiente manera:

$$\varepsilon_k = \sum_{n=k}^{N-1} \left[\left| \hat{f}_k[n] \right|^2 + |\hat{g}_k[n]|^2 \right] \tag{43}$$

Este error debe ser minimizado, seleccionando los coeficientes de predicción, y además está sujeto a la restricción de que cumpla la recursividad de Levinson, dada por:

$$\hat{a}_k[i] = \hat{a}_{k-1}[i] + \widehat{K}_k \hat{a}_{k-1}^*[k-i], \quad \begin{matrix} 1 \leq i \leq k-1 \\ 1 \leq k \leq p \end{matrix} \tag{44}$$

De donde $\widehat{K}_k = \hat{a}_k[k]$ es el k-ésimo coeficiente de reflexión del filtro de predicción y, $\hat{a}_1[1] = k_1$, por definición.

Sustituyendo la ecuación (44) en (41), se obtiene:

$$\hat{f}_k[n] = x[n] + \sum_{i=1}^{k} a_k[i] \cdot x[n-i]$$

$$\hat{f}_k[n] = x[n] + \sum_{i=1}^{k} \left(\hat{a}_{k-1}[i] + \hat{K}_k \hat{a}^*_{k-1}[k-i]\right) \cdot x[n-i] \tag{45}$$

$$\hat{f}_k[n] = x[n] + \sum_{i=1}^{k} \hat{a}_{k-1}[i] \cdot x[n-i] + \hat{K}_k \sum_{i=1}^{k} \hat{a}_{k-1}[k-i] \cdot x[n-i]$$

$$\therefore \hat{f}_k[n] = \hat{f}_{k-1}[n] + \hat{K}_k \cdot \hat{g}_{k-1}[n-1]$$

Seguidamente, al sustituir (44) en (42), se determina que:

$$\hat{g}_k[n] = x[n-k] + \sum_{i=1}^{k} a^*_k[i] \cdot x[n-k+i]$$

$$\hat{g}_k[n] = x[n-k] + \sum_{i=1}^{k} (\hat{a}_{k-1}[i] + \hat{K}_k \hat{a}^*_{k-1}[k-i]) \cdot x[n-k+i] \tag{46}$$

$$\hat{g}_k[n] = x[n-k] + \sum_{i=1}^{k} \hat{a}_{k-1}[i] \cdot x[n-k+i] + \hat{K}_k \sum_{i=1}^{k} \hat{a}_{k-1}[k-i] \cdot x[n-k+i]$$

$$\therefore \hat{g}_k[n] = \hat{g}_{k-1}[n-1] + \hat{K}^*_k \cdot \hat{f}_{k-1}[n]$$

De este modo, es notable que las ecuaciones (45) y (46) resultan expresiones recursivas para los errores de predicción hacia adelante y hacia atrás, respectivamente. Estas dos recursiones halladas, se reemplazan en la ecuación (44), de manera que:

$$\varepsilon_k = \sum_{n=k}^{N-1} \left[\left|\hat{f}_k[n]\right|^2 + |\hat{g}_k[n]|^2\right]$$

$$\varepsilon_k = \sum_{n=k}^{N-1} \left[\left|\hat{f}_{k-1}[n] + \hat{K}_k \cdot \hat{g}_{k-1}[n-1]\right|^2 + \left|\hat{g}_{k-1}[n-1] + \hat{K}^*_k \cdot \hat{f}_{k-1}[n]\right|^2\right] \tag{47}$$

A continuación, se realiza la minimización de ε_k , para ello, se deriva ε_k respecto a K_k, luego se iguala a cero:

$$\frac{\partial \varepsilon_k}{\partial \hat{K}_k} = 0 \tag{48}$$

Al calcular la derivada, se tiene que:

$$\frac{\partial}{\partial \hat{K}_k}\left(\sum_{n=k}^{N-1}\left[\left|\hat{f}_k[n]\right|^2+|\hat{g}_k[n]|^2\right]\right)=0$$

$$\sum_{n=k}^{N-1}\left[\frac{\partial}{\partial \hat{K}_k}\left|\hat{f}_k[n]\right|^2+\frac{\partial}{\partial \hat{K}_k}|\hat{g}_k[n]|^2\right]=0$$

$$\sum_{n=k}^{N-1}\left[\frac{\partial}{\partial K_k}\left|\hat{f}_{k-1}[n]+\hat{K}_k\cdot\hat{g}_{k-1}[n-1]\right|^2+\frac{\partial}{\partial K_k}\left|\hat{g}_{k-1}[n-1]+\hat{K}_k^*\cdot\hat{f}_{k-1}[n]\right|^2\right]=0$$

$$\sum_{n=k}^{N-1}\left[2\left(\hat{f}_{k-1}[n]+\hat{K}_k\cdot\hat{g}_{k-1}[n-1]\right)\hat{g}_{k-1}[n-1]+2\left(\hat{g}_{k-1}[n-1]+\hat{K}_k^*\cdot\hat{f}_{k-1}[n]\right)\hat{f}_{k-1}[n]\right]=0$$

$$\sum_{n=k}^{N-1}\left[2\left(\hat{f}_{k-1}[n]\hat{g}_{k-1}[n-1]+\hat{K}_k\cdot\hat{g}_{k-1}^2[n-1]\right)+2\left(\hat{g}_{k-1}[n-1]\hat{f}_{k-1}[n]+\hat{K}_k^*\cdot\hat{f}_{k-1}^2[n]\right)\right]=0$$

$$\sum_{n=k}^{N-1}\left[4\left(\hat{f}_{k-1}[n]\hat{g}_{k-1}[n-1]\right)+2K_k\left(\hat{f}_{k-1}^2[n]+\hat{g}_{k-1}^2[n-1]\right)\right]=0$$

$$4\sum_{n=k}^{N-1}\hat{f}_{k-1}[n]\hat{g}_{k-1}[n-1]+2K_k\sum_{n=k}^{N-1}\hat{f}_{k-1}^2[n]+\hat{g}_{k-1}^2[n-1]=0$$

$$2\sum_{n=k}^{N-1}\hat{f}_{k-1}[n]\hat{g}_{k-1}[n-1]+K_k\sum_{n=k}^{N-1}\hat{f}_{k-1}^2[n]+\hat{g}_{k-1}^2[n-1]=0$$

$$K_k\sum_{n=k}^{N-1}\hat{f}_{k-1}^2[n]+\hat{g}_{k-1}^2[n-1]=-2\sum_{n=k}^{N-1}\hat{f}_{k-1}[n]\hat{g}_{k-1}[n-1]$$

Luego, despejando $\hat{K}_k$, se llega a obtener la siguiente ecuación:

$$\hat{K}_k=\frac{-2\sum_{n=k}^{N-1}\hat{f}_{k-1}[n]\hat{g}_{k-1}^*[n-1]}{\sum_{n=k}^{N-1}|\hat{f}_{k-1}[n]|^2+|\hat{g}_{k-1}[n-1]|^2},\qquad 1\leq k\leq p \tag{49}$$

La expresión hallada, constituye la ecuación principal del algoritmo de Burg, en donde se estima el k-ésimo coeficiente de reflexión. Además, se cumple que $|K_k|\leq 1$, lo que proporciona estabilidad al modelo autorregresivo [18].

Adicionalmente, se calcula el error de mínimos cuadrados $\hat{\sigma}_k$, cuya relación de recursividad, establecida en el algoritmo de Levinson-Durbin, está dada por [18]:

$$\hat{\sigma}_k=(1-|K_k|^2)\hat{\sigma}_{k-1} \tag{50}$$

Los parámetros iniciales para el algoritmo de Burg, deben inicializarse de la siguiente manera:

$$\hat{r}_{xx}[0]=\frac{1}{N}\sum_{n=0}^{N-1}|x[n]|^2 \tag{51}$$

$$\hat{\sigma}_0 = \hat{r}_{xx}[0] \tag{52}$$

$$\hat{f}_0[n] = x[n] \quad n = 1,2,\dots,N-1 \tag{53}$$

$$\hat{g}_0[n] = x[n] \quad n = 0,1,\dots,N-2 \tag{54}$$

Por último, una vez que se terminen las iteraciones del algoritmo, se procede a calcular el espectro, mediante la ecuación que relaciona el modelo AR con su espectro de potencia, la cual está dada por [18]:

$$P_{xx}(f) = \frac{\hat{\sigma}_p}{\left|1 + \sum_{k=1}^{p} \hat{a}_p[k] e^{-j2\pi fk}\right|^2} \tag{55}$$

2.3 IMPLEMENTACIÓN DE ALGORITMO DE BURG EN MATLAB

En la presente sección, se detalla el proceso de implementación del algoritmo de Burg en Matlab, para ello, se creará una función denominada *mi_pburg*, la cual será análoga a la función propia de Matlab *pburg*, con los mismos parámetros de entrada.

2.3.1 ECUACIONES A IMPLEMENTAR

A continuación, se presenta un resumen de las ecuaciones del algoritmo de Burg, a manera de pseudocódigo [18]:

Inicialización

$$\hat{r}_{xx}[0] = \frac{1}{N}\sum_{n=0}^{N-1} |x[n]|^2 \quad \text{Ecuación (51)}$$

$$\hat{\sigma}_0 = \hat{r}_{xx}[0] \quad \text{Ecuación (52)}$$

$$\hat{f}_0[n] = x[n] \quad n = 1,2,\dots,N-1 \quad \text{Ecuación (53)}$$

$$\hat{g}_0[n] = x[n] \quad n = 0,1,\dots,N-2 \quad \text{Ecuación (54)}$$

FOR $\boldsymbol{k = 1, 2, \dots, p}$

$$\hat{K}_k = \frac{-2\sum_{n=k}^{N-1} \hat{f}_{k-1}[n]\hat{g}^*_{k-1}[n-1]}{\sum_{n=k}^{N-1} |\hat{f}_{k-1}[n]|^2 + |\hat{g}_{k-1}[n-1]|^2} \quad \text{Ecuación (49)}$$

$$\hat{\sigma}_k = (1 - |\hat{K}_k|^2)\hat{\sigma}_{k-1} \quad \text{Ecuación (50)}$$

$$\hat{a}_k[i] = \begin{cases} \hat{a}_{k-1}[i] + \widehat{K}_k \hat{a}^*_{k-1}[k-i] & \text{for } i = 1,2,\dots,k-1 \\ \widehat{K}_k & \text{for } i = k \end{cases}$$ Ecuación (44)

IF $k = 1, \hat{a}_1[1] = 1$

$$\hat{f}_k[n] = \hat{f}_{k-1}[n] + \widehat{K}_k \cdot \hat{g}_{k-1}[n-1] \quad n = k+1, k+2, \dots, N-1$$ Ecuación (45)

$$\hat{g}_k[n] = \hat{g}_{k-1}[n-1] + \widehat{K}^*_k \cdot \hat{f}_{k-1}[n] \quad n = k, k+1, \dots, N-2$$ Ecuación (46)

END FOR

$$P_{xx}(f) = \frac{\hat{\sigma}_p}{\left|1 + \sum_{k=1}^{p} \hat{a}_p[k] e^{-j2\pi fk}\right|^2}$$ Ecuación (55)

Fin de la función

2.3.2 PARÁMETROS DE ENTRADA Y SALIDA DE LA FUNCIÓN

La función *mi_pburg*, que se implementará, tendrá la siguiente sintaxis:

```
function [Pxx,F,a,k] = mi_pburg(x, p, nfft, Fs)
```

En la Tabla 1 se indican los parámetros de entrada de la función *mi_pburg*, los cuales, son los mismos a ser ingresados en la función de Matlab *pburg*.

Tabla 1. Parámetros de entrada de la función *mi_pburg*.

Parámetro de entrada	Tipo de dato	Descripción
x	Vector	Señal de entrada a ser analizada. Corresponde a la señal artificial multicomponente, de longitud N.
p	Escalar	Orden del modelo autorregresivo $AR(p)$. El parámetro debe ser un entero positivo.
nfft	Escalar	Número de puntos de la Transformada de Fourier (entero positivo potencia de 2).
Fs	Escalar	Frecuencia de muestreo de la señal x.

En la Tabla 2 se indican los parámetros de salida de la función *mi_pburg.*

Tabla 2. Parámetros de salida de la función *mi_pburg*.

Parámetro de salida	Tipo de dato	Descripción
Pxx	Vector	Vector de la PSD estimada de la señal de entrada x.
F	Vector	Vector en el dominio de la frecuencia, de la señal de entrada x.
a	Vector	Vector que contiene los coeficientes autorregresivos estimados $\{\hat{a}_p[k]\}$ del modelo AR.
k	Vector	Vector que contiene los coeficientes de reflexión $\{\widehat{K}_k\}$ estimados en el algoritmo de Burg.

2.3.3 CODIFICACIÓN DEL ALGORITMO

A continuación, se describe paso a paso la implementación de la función *mi_pburg*. Describiendo cada porción del código que compone esta función:

Paso 1: Inicialización de parámetros

Se tiene una señal de entrada $x[n]$, la cual está compuesta por un conjunto de muestras $\{x[1], x[2], \dots, x[N]\}$. Para lo cual, el valor de N, se calcula con la función *length*.

```
N  = length(x); % Longitud de la señal de entrada
```

Luego, se inicializan los vectores que almacenarán los errores de predicción hacia adelante y hacia atrás, dados por las ecuaciones (53) y (54)

$$\hat{f}_0[n] = x[n] \quad n = 1,2, \dots, N-1$$

$$\hat{g}_0[n] = x[n] \quad n = 0,1, \dots, N-2$$

En Matlab, se inicializan estos vectores, con las variables, *ef* y *eg,* para los errores de predicción hacia adelante y hacia atrás, respectivamente.

```
ef = x(2:N); % Inicializacion del vector de errores de prediccion hacia adelante
eg = x(1:N-1); % Inicializacion del vector de errores de prediccion hacia atrás
```

En las ecuaciones teóricas el valor de *n* inicia en 0, para el error de predicción hacia atrás, mientras que, en Matlab, el primer elemento de un vector tiene un índice igual a 1. Por lo

tanto, *ef* toma los elementos de x, desde el segundo hasta el último elemento (N). Mientras que *eg* toma los elementos de x desde el primero, hasta el penúltimo $(N-1)$. De esta manera, se cumple con el rango definido en las ecuaciones.

Después, se inicializa el vector de coeficientes de reflexión $\{K_k\}$ y el vector que almacena los errores de mínimos cuadrados $\{\hat{\sigma}_k\}$. En Matlab, estos vectores se declaran con las variables *K* y *sigma_vec*:

```
K = zeros(p,1); % Inicialización del vector de coeficientes de reflexión {Kk}
sigma_vec = zeros(p,1); % Inicialización del vector de errores de mínimos cuadrados
```

Como se puede observar, dichos vectores se inicializan como vectores columna de longitud p, cuyos elementos son todos cero. Esto, ya que se realizan p iteraciones más adelante.

A continuación, se calcula el error de mínimos cuadrados inicial, definido por las ecuaciones (51) y (52):

$$\hat{r}_{xx}[0] = \frac{1}{N}\sum_{n=0}^{N-1}|x[n]|^2$$

$$\hat{\sigma}_0 = \hat{r}_{xx}[0]$$

```
sigma0 = x*x'/N; % Primer elemento del vector de errores de mínimos cuadrados
sigma_vec(1) = sigma0; % Primer elemento del vector de errores de mínimos cuadrados
```

En el código, la operación *x*x'/N* calcula la sumatoria de los cuadrados de $x[n]$ y divide el valor obtenido para N. Luego, este valor se asigna como el primer elemento del vector *sigma_vec*.

Seguidamente, el último parámetro a inicializar es el primer coeficiente paramétrico,

$$\hat{a}_1 = 1$$

```
a = 1; % Primer elemento del vector de coeficientes {ak}
```

Paso 2: Lazo iterativo para el cálculo de los vectores $\{\hat{K}_k\}$,$\{\hat{a}_p[k]\}$ y $\{\hat{\sigma}_k\}$

El siguiente bloque de código, contiene un lazo for, que realiza las iteraciones desde 1 hasta el valor del orden p del modelo AR. En este lazo se aplica el algoritmo de Burg, y en la última iteración, se habrán calculado los coeficientes de reflexión $\{\hat{K}_k\}$, los coeficientes paramétricos $\{\hat{a}_p[k]\}$ y los errores de mínimos cuadrados $\{\hat{\sigma}_k\}$.

```
for k=1:p % el bucle se itera desde k=1 hasta p
    num = sum(ef.*eg); % numerador de la ecuacion ()
    den = sum(ef.^2 + eg.^2); % denominador de la ecuacion
    K(k) = -2*num/den; % Cálculo del coeficiente de reflexión
    sigma_vec(k+1) = (1-K(k)^2)*sigma_vec(k); % Cálculo recursivo del error de mínimos
cuadrados
    a = [a;0] + K(k) * [0;flipud(conj(a))]; % Actualización de los coeficientes {ak}
    ef_nuevo = ef   + K(k)*eg; % actualizacion de errores de prediccion hacia adelante
    eg_nuevo = K(k)*ef  + eg; % actualizacion de errores de prediccion hacia atrás
    ef = ef_nuevo(2:end); % actualización del vector ef
    eg = eg_nuevo(1:end-1); % actualización del vector eg
end
```

En este bucle, lo primero que se realiza es el cálculo de los coeficientes de reflexión, dado por la ecuación (49):

$$\hat{K}_k = \frac{-2\sum_{n=k}^{N-1} \hat{f}_{k-1}[n]\hat{g}^*_{k-1}[n-1]}{\sum_{n=k}^{N-1} |\hat{f}_{k-1}[n]|^2 + |\hat{g}_{k-1}[n-1]|^2}, \qquad 1 \le k \le p$$

```
num = sum(ef.*eg); % numerador de la ecuacion ()
den = sum(ef.^2 + eg.^2); % denominador de la ecuacion
K(k) = -2*num/den; % Cálculo del coeficiente de reflexión
```

Para hacer más legible el código del cálculo de $\hat{K}_k$ se realiza primero el cálculo del sumatorio del numerador, definido por la variable *num*, con la ayuda de la instrucción *sum(ef.*eg)*, en la cual, se multiplican elemento a elemento los vectores *ef* y *eg*, y luego se suman, con la ayuda de la función *sum*. Para el denominador, definido por la variable den, se procede de forma similar, con la diferencia de que se suman los cuadrados de los elementos de *ef* y *eg*, mediante la instrucción *sum(ef.^2+eg.^2)*. Por último, el coeficiente se calcula con la instrucción *-2*num/den*. Cabe recalcar que ef y eg se actualizan más adelante, para cumplir con la recursión.

Luego, se calcula el error de mínimos cuadrados, definido en la ecuación (50):

$$\hat{\sigma}_k = (1 - |\hat{K}_k|^2)\hat{\sigma}_{k-1}$$

```
sigma_vec(k+1) = (1-K(k)^2)*sigma_vec(k); % Cálculo recursivo del error de mínimos
cuadrados
```

A continuación, se calculan los coeficientes paramétricos, dados por la ecuación (44):

$$\hat{a}_k[i] = \begin{cases} \hat{a}_{k-1}[i] + \hat{K}_k \hat{a}^*_{k-1}[k-i] & \text{for } i = 1,2,\dots,k-1 \\ \hat{K}_k & \text{for } i = k \end{cases}$$

```
a = [a;0] + K(k) * [0;flipud(conj(a))]; % Actualización de los coeficientes {ap[k]}
```

Esta línea de código puede considerarse como la más importante, pues la obtención del vector $\{\hat{a}_p[k]\}$ permitirá calcular el espectro de la señal, como se verá más adelante.

Para entender cómo funciona esta línea, se toma como ejemplo $p = 3$, de este modo, las iteraciones se calculan de la siguiente manera:

$\boldsymbol{k = 1}$

$\hat{a}_1[1] = \hat{K}_1$

$\boldsymbol{k = 2}$

$\hat{a}_2[1] = \hat{a}_1[1] + \hat{K}_2\hat{a}_1[1] = \hat{K}_1 + \hat{K}_2\hat{K}_1$

$\hat{a}_2[2] = \hat{K}_2$

$\boldsymbol{k = 3}$

$\hat{a}_3[1] = \hat{a}_2[1] + \hat{K}_3\hat{a}_2[2] = \hat{K}_1 + \hat{K}_2\hat{K}_1 + \hat{K}_3\hat{K}_2$

$\hat{a}_3[2] = \hat{a}_2[2] + \hat{K}_3\hat{a}_2[1] = \hat{K}_2 + \hat{K}_3(\hat{K}_1 + \hat{K}_2\hat{K}_1) = \hat{K}_2 + \hat{K}_3\hat{K}_1 + \hat{K}_1\hat{K}_2\hat{K}_3$

$\hat{a}_3[3] = \hat{K}_3$

La función flipud invierte el orden de los elementos de un vector columna. Por ejemplo *flipud([1;2;3])* devuelve el vector *[3;2;1]* [38]. Teniendo esto en cuenta, se procede a realizar las iteraciones que se plantea en la línea de código:

$\boldsymbol{k = 1}$

```
a = [a;0] + K(k) * [0;flipud(conj(a))]
a = [1;0] + K(1) * [0;1]
a = [1;0] + [0;K(1)]
a = [1;K(1)]
```

$\boldsymbol{k = 2}$

```
a = [a;0] + K(k) * [0;flipud(conj(a))]
a = [[1;K(1)];0] + K(2) * [0; flipud([1;K(1)])]
a = [[1;K(1)];0] + K(2) * [0;[K(1);1]]
a = [1;K(1);0] + K(2) * [0;K(1);1]
a = [1;K(1);0] + K(2) * [0;K(1);1]
a = [1;K(1);0] + [0; K(2)K(1); K(2)]
a = [1;K(1)+ K(2)K(1); K(2)]
```

De esta manera, las operaciones matemáticas son equivalentes con la línea de código, así, el vector $\{\hat{a}_p[k]\}$ se va construyendo de forma iterativa.

Para concluir con las instrucciones del lazo *for*, se actualizan los vectores *ef* y *eg*, de acuerdo con las ecuaciones recursivas (45) y (46)

$$\hat{f}_k[n] = \hat{f}_{k-1}[n] + \hat{K}_k \cdot \hat{g}_{k-1}[n-1] \quad n = k+1, k+2, \dots, N-1$$
$$\hat{g}_k[n] = \hat{g}_{k-1}[n-1] + \hat{K}_k^* \cdot \hat{f}_{k-1}[n] \quad n = k, k+1, \dots, N-2$$

```
ef_nuevo = ef   + K(k)*eg; % actualizacion de errores de prediccion hacia adelante
eg_nuevo = K(k)*ef  + eg; % actualizacion de errores de prediccion hacia atrás
ef = ef_nuevo(2:end); % actualización del vector ef
eg = eg_nuevo(1:end-1); % actualización del vector eg
```

En las primeras dos líneas de este segmento de código, identificadas por colores rojo y azul, se aplican las ecuaciones (45) y (46), reconocidas por los mismos colores. Luego, se actualizan los vectores *ef* y *eg.* Tomando todos los elementos desde el segundo hasta el último elemento del vector, para el caso de *ef* (línea verde) y, de forma similar, tomando los elementos de *eg* desde el primero hasta el penúltimo (línea amarilla), para el caso de *eg*. De esta manera, los vectores *ef* y *eg*, cumplen la recursividad e ingresan de nuevo al bucle *for*, para una nueva iteración de *k*.

Paso 3: Cálculo de la PSD a partir de los parámetros estimados

Luego de completar todas las iteraciones del lazo *for*, desde $k = 1$ hasta p, se procede a calcular el espectro de potencia de la señal *x*, a partir de los coeficientes paramétricos $\{\hat{a}_p[k]\}$, mediante la ecuación (55):

$$P_{xx}(f) = \frac{\hat{\sigma}_p}{\left|1 + \sum_{k=1}^{p} \hat{a}_p[k] e^{-j2\pi f k}\right|^2}$$

```
Af = abs( fft( a, nfft )).^ 2; % Función de trasferencia del denominador del filtro
Pxx = sigma_vec(end) ./ Af; % Ecuación del espectro de potencia
```

Paso 4: Control de parámetros de salida

Una vez que se obtiene la PSD mediante el método de Burg, el último paso en la función *mi_pburg* consiste en obtener la salida del espectro unilateral de potencia. El siguiente bloque de código realiza este último paso:

```
if isreal(x)
   select = (1:floor(nfft/2)+1)'; % seleccion de las frecuencias positivas
   Pxx = Pxx(select); % Actualizacion del vector de la PSD estimada
   % Correcciones del espectro unilateral
   Pxx(1)=Pxx(1)/2;
   Pxx(end)=Pxx(end)/2;
else
   select = (1:nfft)';
   Pxx = Pxx(select);
end
freq = (select - 1)*Fs/nfft; % calculo del eje de frecuencias
```

Se explica a continuación, las líneas de código del bloque anterior:

1) *if isreal(x):* Se verifica si la señal de entrada x es una señal real o no.
2) *select = (1:floor(nfft/2)+1)'*: Se crea un vector *select* que contiene los índices de las frecuencias positivas del espectro. La instrucción *floor(nfft/2)+1* representa la cantidad de frecuencias positivas incluyendo la frecuencia cero, hasta la frecuencia de Nyquist. Se utiliza la función *floor* para redondear hacia abajo el valor de *nfft/2*.
3) *Pxx = Pxx(select)*: Aquí, se actualiza el vector de la PSD (*Pxx*) reemplazando sus valores con los valores correspondientes a las frecuencias positivas seleccionadas en el paso anterior. De esta manera, se obtiene el espectro unilateral.
4) *Pxx(1)=Pxx(1)/2* y *Pxx(end)=Pxx(end)/2*: Estas líneas realizan una corrección en las componentes de frecuencia cero y de Nyquist del espectro unilateral. Debido a la naturaleza de la transformada de Fourier, estas componentes son duplicadas en el espectro unilateral. Por lo tanto, se dividen por 2 para corregir este efecto y obtener una representación precisa del espectro unilateral de potencia.
5) Si la señal de entrada no es real (*else*), entonces simplemente se seleccionan todas las frecuencias hasta el valor de *nfft* en el espectro, ya que en este caso la señal es compleja y no existe la simetría que permite reducir a la mitad el espectro.

2.4 DESARROLLO DEL SCRIPT PRINCIPAL DE PRUEBAS

A continuación, se detalla en orden las secciones de código que conforman el script principal de pruebas, denominado *Principal_Burg.m*

2.3.1. GENERACIÓN DE LA SEÑAL ARTIFICIAL

Respecto de la señal artificial, se considerarán valores de frecuencia relativamente bajas para evitar una excesiva carga computacional. Para evitar realizar demasiadas pruebas, se considerará una sola señal artificial que incorpore tanto el problema de resolución en frecuencia, es decir, se considerarán valores de frecuencias cada vez más cercanas, como

el problema de detección de componentes muy pequeñas, es decir, se considerarán valores de amplitudes cada vez más pequeños.

En este caso, se considerará una señal senoidal de 16 componentes, las mismas que están en el rango de [100 – 241] [Hz], cuyas amplitudes van desde 1 hasta 0.0000001, disminuyendo en un factor de 10, cada 2 componentes. De esta manera, en la Tabla 3 se muestran los parámetros de amplitud y frecuencia de cada una de las 16 componentes de la señal artificial.

Tabla 3. Características de la señal multicomponente

Componente	Amplitud	Frecuencia [Hz]
1	1	100
2	1	108
3	0.1	120
4	0.1	127
5	0.01	140
6	0.01	146
7	0.001	160
8	0.001	165
9	0.0001	180
10	0.0001	184
11	0.00001	200
12	0.00001	203
13	0.000001	220
14	0.000001	222
15	0.0000001	240
16	0.0000001	241

A continuación, se muestra el bloque de código de la sección para la generación de la señal artificial multicomponente:

```
%% GENERACIÓN DE LA SEÑAL CON 16 COMPONENTES:
% En este bloque se genera la señal artificial para las pruebas
clear all; clc
rng default
duracion = 2; % duracion de la señal
Fmax = 250; % frecuencia maxima de la señal
Fs = 10*Fmax; % frecuencia de muestreo de la señal
Ts = 1/Fs; % periodo de muestreo de la señal
t=0:Ts:duracion; % vector de tiempos
```

```
% Declaración de la señal
x = 1*sin(2*pi*100*t)+1*sin(2*pi*108*t)+...
  0.1*sin(2*pi*120*t)+0.1*sin(2*pi*127*t)+...
  0.01*sin(2*pi*140*t)+0.01*sin(2*pi*146*t)+...
  0.001*sin(2*pi*160*t)+0.001*sin(2*pi*165*t)+...
  0.0001*sin(2*pi*180*t)+0.0001*sin(2*pi*184*t)+...
  0.00001*sin(2*pi*200*t)+0.00001*sin(2*pi*203*t)+...
  0.000001*sin(2*pi*220*t)+0.000001*sin(2*pi*222*t)+...
  0.0000001*sin(2*pi*240*t)+0.0000001*sin(2*pi*241*t);
% Grafica de la senial en el dominio del tiempo
f1 = figure;
%f1.Position = [420 200 700 560];
plot(t,x,'r','LineWidth',0.9)
title('Señal artificial multicomponente')
xlabel('Tiempo [s]')
ylabel('Amplitud')
grid minor
axis([0 duracion/5 -3.5 3.5])
```

Como puede observarse, la duración de la señal propuesta es de 2 segundos, y se considera una frecuencia máxima de 250 [Hz]. Luego, la frecuencia de muestreo *Fs* es 10 veces esta frecuencia, para cumplir con el Teorema de Nyquist. De esta manera, se crea el vector de tiempos de 0 [s] a 2 [s] en pasos del tiempo de muestreo *Ts,* donde *Ts = 1/Fs.*

Al ejecutar el bloque de código, se obtiene la gráfica de la señal artificial multicomponente, como se observa en la Figura 5.

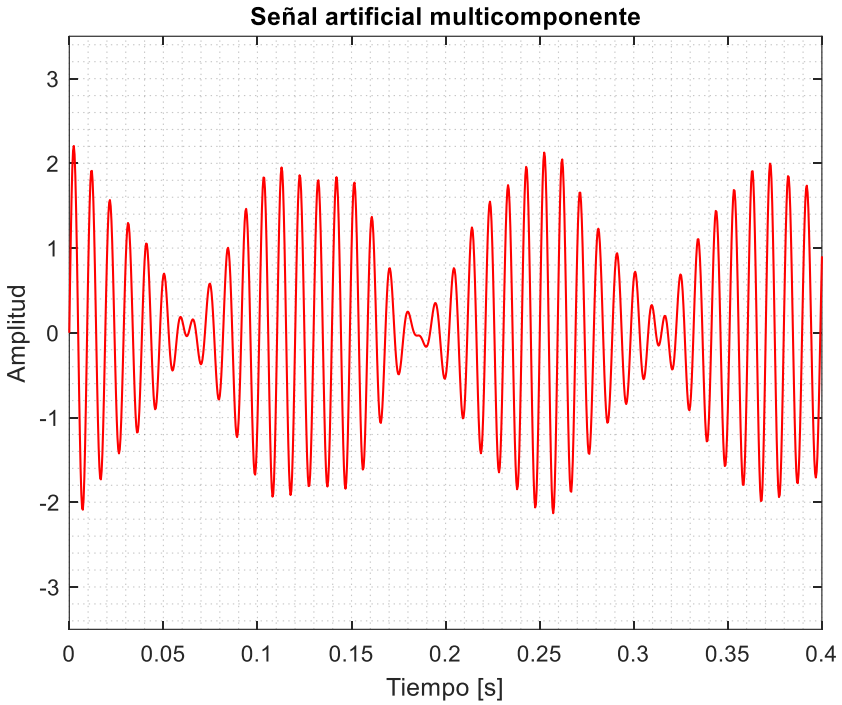


Figura 5. Señal artificial multicomponente para pruebas de los métodos de estimación espectral.

2.3.2. OBTENCIÓN DEL ESPECTRO CON MÉTODOS CLÁSICOS

Una vez generada la señal artificial en el dominio del tiempo, en esta sección se detalla la obtención de su espectro, por medio de los métodos tradicionales, periodograma simple y periodograma modificado, con la ayuda de las funciones del Signal Processing Toolbox de Matlab.

Para comenzar con el tratamiento de la señal en el dominio de la frecuencia, se tienen en cuenta los parámetros iniciales, que se muestran en la Tabla 4:

Tabla 4. Parámetros iniciales para de la FFT

Parámetro	Descripción
N	Número de muestras de la señal discretizada.
nFFT	Longitud de la transformada de Fourier. Una potencia de 2 tal que nFFT < N.
FACTOR	Factor para aumentar el número de muestras y tener un mejor resultado (múltiplo de 2).

Considerando la teoría de ventanas, se conoce que el periodograma simple puede obtenerse al hallar la FFT de la señal multiplicada por una ventana rectangular, mientras que, el periodograma modificado se obtiene al hallar la FFT de la señal multiplicada por una determinada ventana. En este caso, para el presente estudio se considera la ventana Blackman-Harris de 4 términos, por ser una ventana que proporciona al periodograma una capacidad de detección excepcional, ya que reduce el tamaño de los lóbulos secundarios hasta en -92 dB [39].

Luego, con la ayuda de las funciones *rectwin* y *blackmanharris* se procede a generar las señales enventanadas, de acuerdo a lo indicado en la Tabla 5.

Tabla 5. Cálculo de las señales enventanadas

Parámetro	Cálculo en Matlab	Descripción
vent_1	rectwin(N)	Ventana rectangular
vent_2	blackmanharris(N)	Ventana Blackman-Harris de 4 términos (BH-4)

x1_enventanada	x.*vent_1'	Señal artificial *x* multiplicada por la ventana rectangular.
x2_enventanada	x.*vent_2'	Señal artificial *x* multiplicada por la ventana Blackman-Harris de 4 términos (BH-4).

El siguiente segmento de código implementa lo mencionado en la Tabla 5:

```
% 1. GENERACION DE VENTANAS:
vent_1 = rectwin(N); % ventana rectangular
vent_2 = blackmanharris(N); % ventana blackmanharris
% 2. Enventanado de la senial temporal:
x1_enventanada = x.*vent_1'; % señal con ventana rectangular
x2_enventanada = x.*vent_2'; % señal con ventana blackmanharris
```

Una vez que se obtienen las señales enventanadas, se procede a hallar los periodogramas, con lo cual, al calcular el módulo de la FFT de la señal *x1_enventanada*, se obtiene el periodograma simple, y al calcular el módulo de la FFT de la señal *x2_enventanada*, se obtienen el periodograma modificado. Para ello, se utiliza la función *fft* de Matlab, de la siguiente manera:

```
% 3. Periodograma modificado:
Periodograma_Simple = 10*log10(abs(fft(x1_enventanada,nFFT)));
Periodograma_Modificado = 10*log10(abs(fft(x2_enventanada,nFFT)));
```

A continuación, se obtiene el vector de frecuencias con la ayuda de la función *linspace*, de la siguiente forma:

```
f_periodograma = linspace(0,Fs,nFFT); % vector de frecuencias
```

Una vez realizados estos pasos, se obtienen las señales en el dominio de la frecuencia, con lo cual se puede graficar para observar el espectro de la señal artificial. Todos los pasos descritos anteriormente, se reflejan en el siguiente bloque de código:

```
%% ESPECTRO DE LA SEÑAL CON LOS PERIODOGRAMAS
N = length(x); % numero de muestras de la señal discretizada
FACTOR = 64; % factor de la nFFT
nFFT = 2^ceil(log2(N))*FACTOR;
%nFFT = N;
% 1. GENERACION DE VENTANAS:
vent_1 = rectwin(N); % ventana rectangular
vent_2 = blackmanharris(N); % ventana blackmanharris
% 2. Enventanado de la senial temporal:
x1_enventanada = x.*vent_1'; % señal con ventana rectangular
x2_enventanada = x.*vent_2'; % señal con ventana blackmanharris
% 3. Periodograma modificado:
```

```
Periodograma_Simple = 10*log10(abs(fft(x1_enventanada,nFFT)));
Periodograma_Modificado = 10*log10(abs(fft(x2_enventanada,nFFT)));
%Se genera el eje de frecuencias:
f_periodograma = linspace(0,Fs,nFFT); % vector de frecuencias
f2 = figure;
%f2.Position = [420 200 700 560];
hold on
plot(f_periodograma,Periodograma_Simple);  %
plot(f_periodograma,Periodograma_Modificado,'LineWidth',1.0);
%axis([50 250 min(Periodograma_Simple) max(Periodograma_Simple)+50]);
xlim([80 250]);
xlabel('Frecuencia [Hz]');
ylabel('Amplitud [dB]')
legend('Periodograma Simple','Periodograma Modificado (Ventana BH-4)')
title('Periodograma de la señal de pruebas')
grid minor
hold off
```

Al ejecutar el código anterior, se obtienen los periodogramas simple y modificado, como se observa en la Figura 6.

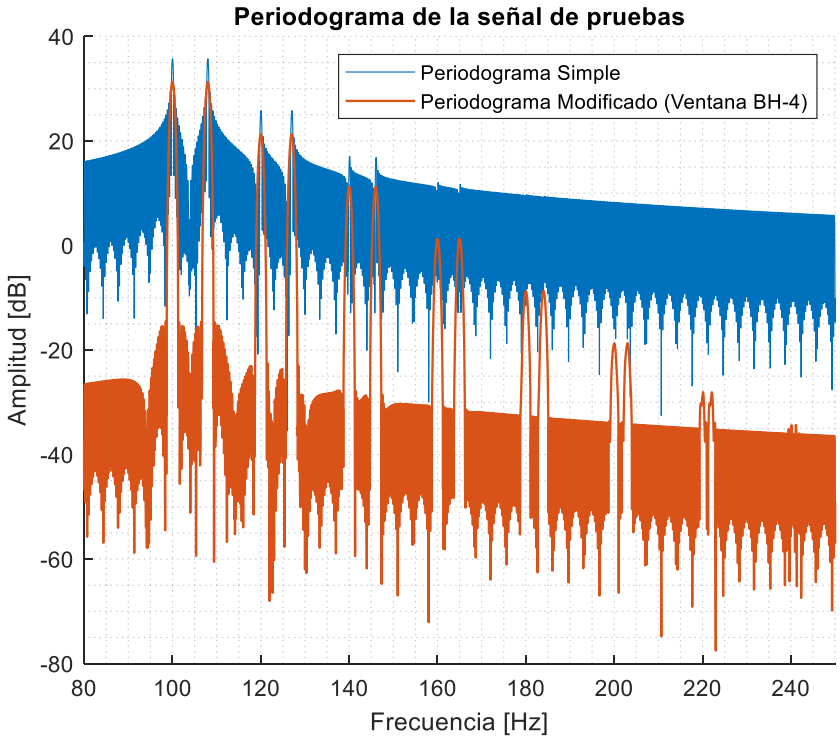


Figura 6. Periodograma simple y modificado de la señal de pruebas.

2.3.3. OBTENCIÓN DEL ESPECTRO CON EL MÉTODO DE BURG

En esta sección del script, se hallará el espectro de la señal artificial de pruebas mediante el método de Burg, usando la función *pburg*, de Matlab, así como de la función *mi_pburg*,

desarrollada anteriormente, con el propósito de comparar los resultados entre las dos funciones, y también, con el método del periodograma modificado.

La sintaxis de las funciones *pburg* y *mi_pburg*, son idénticas, la única diferencia radica en que *mi_pburg* tiene dos parámetros de salida adicionales, correspondientes al vector de coeficientes paramétricos $\{a_p[k]\}$, y al vector de coeficientes de reflexión $\{K_k\}$. Así, para estimar el espectro de la señal artificial multicomponente, se invoca a las funciones de acuerdo con la siguiente sintaxis:

```
% Espectro con la funcion pburg de Matlab
[pb1,fb1] = pburg(x,orden,nFFT,Fs);
% Espectro usando la funcion mi_pburg
[pb2,fb2,a_mipburg,k_mipburg] = mi_pburg(x,orden,nFFT,Fs);
```

Una vez que se obtienen los vectores del eje de frecuencias y el eje de densidad espectral de potencia (PSD), se procede a graficar el espectro con el método de Burg, y el método del periodograma modificado en una misma figura, como se muestra en el siguiente código:

```
%% ESPECTRO DE LA SEÑAL CON EL MÉTODO DE BURG
orden = 150; % orden del modelo AR(p)
% Espectro con la funcion pburg de Matlab
[pb1,fb1] = pburg(x,orden,nFFT,Fs);
f3 = figure;
%f3.Position = [420 200 700 560];
hold on
plot(f_periodograma,Periodograma_Simple,'m'); % grafico del periodograma
simple
plot(f_periodograma,Periodograma_Modificado,'b'); % grafico del periodograma
modificado
plot(fb1, 10*log10(pb1),'g','LineWidth',1.0) % grafico
% Espectro usando la funcion mi_pburg
[pb2,fb2,a_mipburg,k_mipburg] = mi_pburg(x,orden,nFFT,Fs);
%[pb3,fb3,cof2,ref_old] = mi_pburg(x,orden,nFFT,Fs);
plot(fb2,10*log10(pb2),'r','LineWidth',1.0)
xlabel('Frecuencia [Hz]');
ylabel('Amplitud [dB]')
legend('Periodograma Simple','Periodograma Modificado (BH-4)','Burg propia de
Matlab','mi\_pburg creada','Location','south')
title(['Periodograma vs Método de Burg (orden ',num2str(orden), ')'])
xlim([80 250]);
ylim([-250 50])
grid minor
```

Al ejecutar el código anterior, se obtiene una gráfica donde se muestra el espectro de la señal artificial de pruebas, tanto con el método de Burg, como con los métodos tradicionales

(Periodograma simple y modificado), en una misma gráfica, como se observa en la Figura 7.

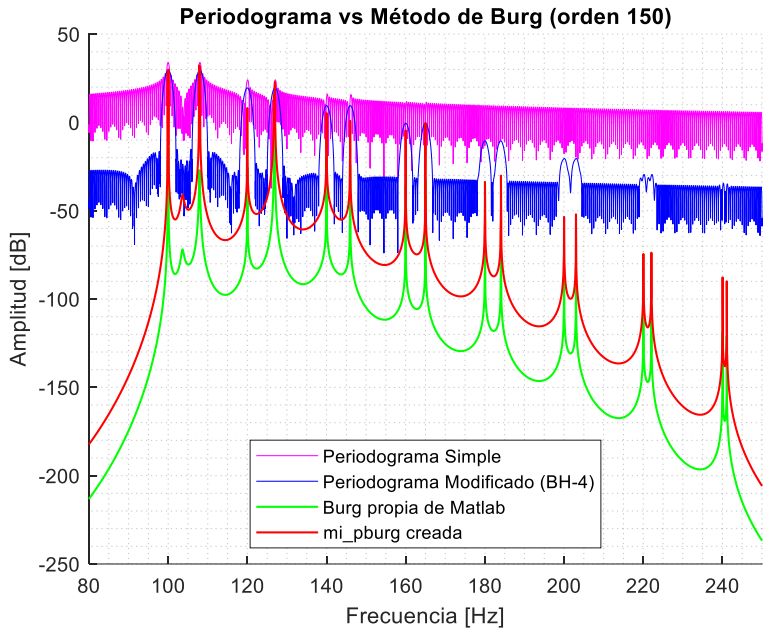


Figura 7. Espectro de la señal de pruebas obtenida con periodogramas y el método de Burg.

2.3.4. COMPROBACIÓN DE COEFICIENTES DE REFLEXIÓN

En esta sección del script, se hace uso de la función *arburg*, para obtener los coeficientes paramétricos y los coeficientes de reflexión.

La función se invoca de la siguiente manera:

```
[a_arburg,sigma_arburg,k_arburg] = arburg(x,orden);
```

Los parámetros de salida, devueltos por la función arburg, se explican a continuación [35]:

- `a_arburg:` Es el vector de coeficientes paramétricos $\{a_p[k]\}$
- `sigma_arburg:` Es el error de mínimos cuadrados σ_p
- `K_arburg:` Es el vector de coeficientes de reflexión $\{K_k\}$

Estos valores, se comparan numéricamente con los obtenidos por la función desarrollada *mi_pburg*, mediante la función *table.* Adicionalmente, en esta sección del script, se genera una gráfica de los coeficientes de reflexión obtenidos las funciones *mi_pburg* y *arburg*.

```
%% COMPROBACIÓN DE COEFICIENTES DE REFLEXION
% Obtencion de parametros con la funcion arburg
[a_arburg,sigma_arburg,k_arburg] = arburg(x,orden);
% comparacion entre los coeficientes devueltos por arburg y mi_pburg
T = table(k_arburg,k_mipburg)
% Grafica de los coeficientes de reflexion
figure
stem(k_mipburg,'filled')
hold on
stem(k_arburg,'filled')
title('Coeficientes de reflexión')
xlabel('Orden (p)')
legend('Función arburg (Matlab)','Función mi\_pburg (creada)')
grid minor
```

2.3.5. PRUEBAS PARA UN RANGO DE ORDEN P

Finalmente, para facilitar el análisis de los resultados, evitando realizar numerosas simulaciones, al cambiar el orden del modelo AR en cada ejecución del script, se propone una última sección, en donde se simulan un conjunto de gráficos comparativos entre el método de Burg y los periodogramas (como el de la Figura 7). Para ello, se generan múltiples gráficos por medio de un bucle *for,* que va iterando la variable *orden*. Esto se obtiene con el siguiente bloque de código:

```
%% PRUEBAS PARA UN RANGO DE ORDEN P
orden = 10:10:100;
for i=orden
    [pb1,fb1] = pburg(x,i,nFFT,Fs);
    [pb2,fb2] = mi_pburg(x,i,nFFT,Fs);
    figure
    plot(f_periodograma,Periodograma_Simple,'m',...
        'DisplayName','Periodograma Simple'); % grafico del periodograma
simple
    hold on
    plot(f_periodograma,Periodograma_Modificado,'b',...
        'DisplayName','Periodograma Modificado (BH-4)'); % grafico del
periodograma modificado
    plot(fb1, 10*log10(pb1),'g','LineWidth',1.0,...
        'DisplayName','Función pburg propia de Matlab') % grafico
    plot(fb2,10*log10(pb2),'r','LineWidth',1.0,...
        'DisplayName','Función mi\_pburg creada')
    xlabel('Frecuencia [Hz]');
    ylabel('Amplitud [dB]')
    legend('Location','south')
    title(['Periodograma vs Método de Burg (orden ',num2str(i), ')'])
    xlim([80 250]);
    ylim([-250 50])
    grid minor
    hold off
end
```

En el código anterior, se tiene el siguiente vector:

```
orden = 10:10:100;
```

Esto indica que se inicia con un orden de $p = 10$, hasta $p = 100$, en pasos de 10. Dicho esto, es posible modificar el vector de acuerdo con los órdenes de p, que se requieran observar, para realizar el análisis.

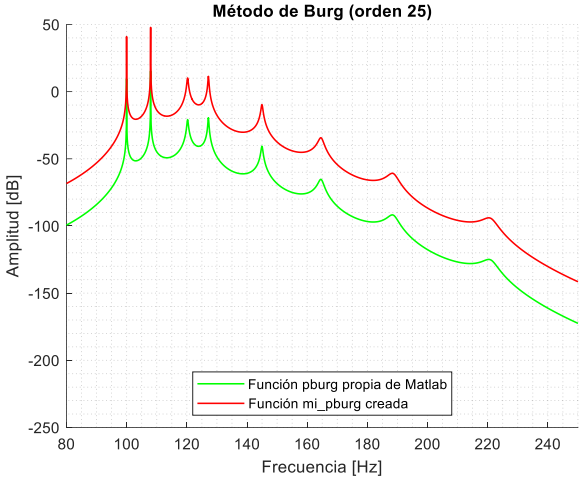
3 RESULTADOS

En el presente capítulo, se presentarán y discutirán los resultados obtenidos a partir de las simulaciones realizadas en Matlab. Los resultados se centran en la comparación de los métodos de estimación de espectral mediante el periodograma y el algoritmo de Burg por medio de la señal artificial multicomponente generada en el script principal de pruebas. Así mismo, también se aborda una comparación entre las funciones que proporciona Matlab y el algoritmo propio desarrollado en el capítulo anterior, en base a comparaciones gráficas y numéricas.

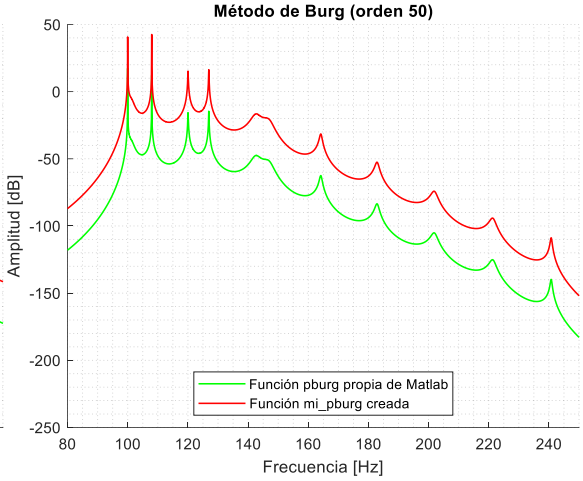
Finalmente, luego de las discusiones de los resultados, el presente Trabajo de Integración Curricular culmina con las conclusiones y recomendaciones, derivadas del desarrollo del trabajo.

3.1 COMPARACIÓN ENTRE LA FUNCIÓN CREADA Y LA DE MATLAB.

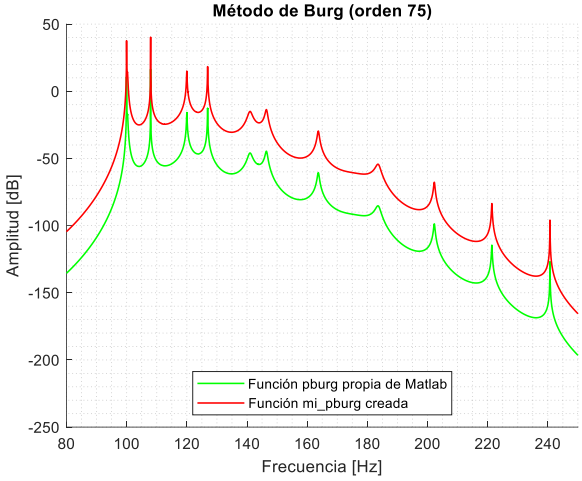
En primera instancia, se generaron los espectros de la señal artificial $x[n]$ mediante el algoritmo de Burg. En una misma gráfica, se presenta el espectro obtenido con el método de Burg, con la función de Matlab pburg, así como con la función propia *mi_pburg*. De esta manera, se generaron ocho espectros variando el orden del modelo AR desde $p = 25$ hasta $p = 200$, con p aumentando en pasos de 25.


Método de Burg (orden 25)
Amplitud [dB]
Frecuencia [Hz]
Función pburg propia de Matlab
Función mi_pburg creada

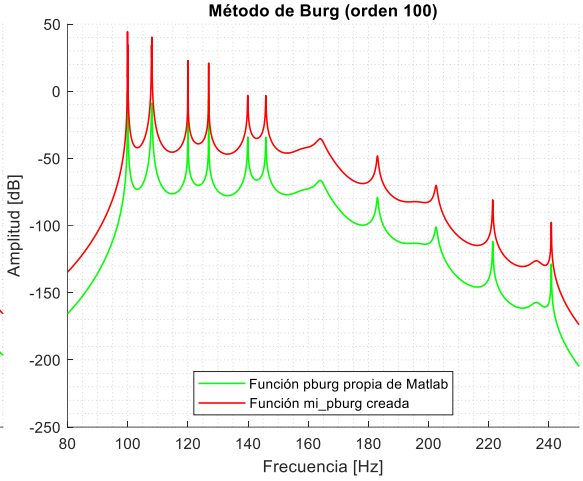
(a)


Método de Burg (orden 50)
Amplitud [dB]
Frecuencia [Hz]
Función pburg propia de Matlab
Función mi_pburg creada

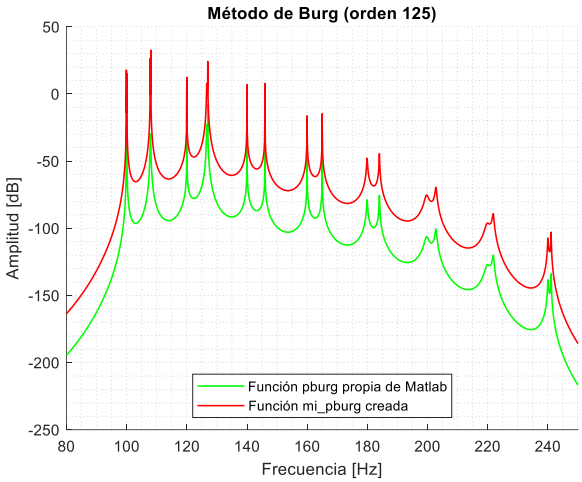
(b)


Método de Burg (orden 75)
Amplitud [dB]
Frecuencia [Hz]
Función pburg propia de Matlab
Función mi_pburg creada

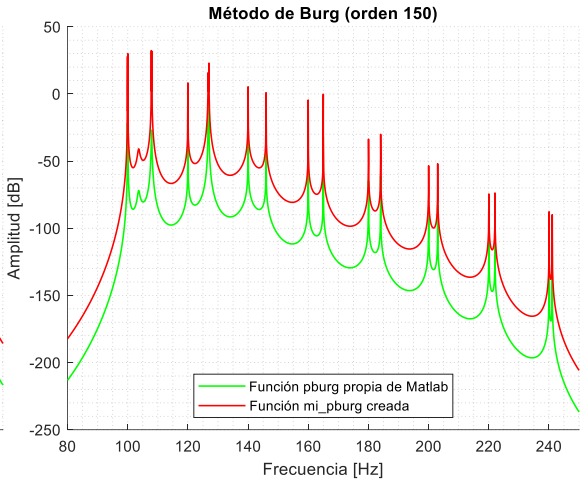
(c)


Método de Burg (orden 100)
Amplitud [dB]
Frecuencia [Hz]
Función pburg propia de Matlab
Función mi_pburg creada

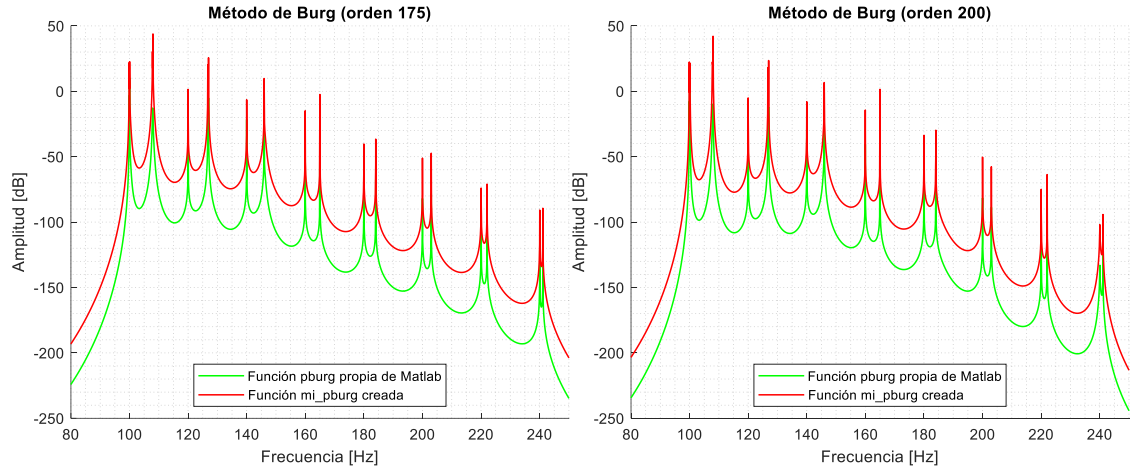
(d)


Método de Burg (orden 125)
Amplitud [dB]
Frecuencia [Hz]
Función pburg propia de Matlab
Función mi_pburg creada

(e)


Método de Burg (orden 150)
Amplitud [dB]
Frecuencia [Hz]
Función pburg propia de Matlab
Función mi_pburg creada

(f)



(g) (h)

Figura 8. Espectro obtenido con el método de Burg. (a) Orden 25, (b) Orden 50, (c) Orden 75, (d) Orden 100 , (e) Orden 125, (f) Orden 150, (g) Orden 175 y (h) Orden 200.

Como se observa en la Figura 8, los espectros obtenidos con ambas funciones se asemejan, salvo el hecho de que la función *mi_pburg*, muestra las amplitudes desplazadas más arriba en el eje vertical.

Más adelante se realizará el análisis de la variación de los picos en función del orden p del modelo AR.

3.2 COMPROBACIÓN DE COEFICIENTES DE REFLEXIÓN

Para comprobar de forma cuantitativa la similaridad entre los espectros obtenidos con funciones propias Matlab, y la función desarrollada *mi_pburg*, se realizaron pruebas con diferentes órdenes de p para obtener los valores de los coeficientes de reflexión $\{K_k\}$.

En la Figura 9, se obtuvieron los coeficientes de reflexión para $p = 5$. En la columna izquierda, se tienen los coeficientes devueltos por la función *arburg* de Matlab, mientras que en la columna derecha, se tienen los coeficientes devueltos como una salida de la función *mi_pburg*, desarrollada anteriormente. Como puede observarse, los valores de ambas columnas son iguales.

5×2 table

k_arburg	k_mipburg
-0.96585	-0.96585
0.99974	0.99974
-0.96288	-0.96288
0.99877	0.99877
-0.95186	-0.95186

Figura 9. Coeficientes de reflexión para orden 5 entre la función *arburg* y *mi_pburg*.

Luego, se realizó una prueba para $p = 10$. Como se observa en la Figura 10, los valores de ambas columnas también son iguales.

10×2 table

k_arburg	k_mipburg
-0.96585	-0.96585
0.99974	0.99974
-0.96288	-0.96288
0.99877	0.99877
-0.95186	-0.95186
0.99874	0.99874
-0.93529	-0.93529
0.99773	0.99773
-0.92437	-0.92437
0.99391	0.99391

Figura 10. Coeficientes de reflexión para orden 10 entre la función *arburg* y *mi_pburg*.

Seguidamente, se realizó una prueba numérica adicional con orden $p = 20$. En la Figura 11, se observa que casi todos los coeficientes son iguales. De esta forma, se observa que los valores tienden a ser iguales.

k_arburg	k_mipburg
-0.96585	-0.96585
0.99974	0.99974
-0.96288	-0.96288
0.99877	0.99877
-0.95186	-0.95186
0.99874	0.99874
-0.93529	-0.93529
0.99773	0.99773
-0.92437	-0.92437
0.99391	0.99391
-0.92237	-0.92237
0.9778	0.9778
-0.012278	-0.012287
-0.67251	-0.67251
0.13595	0.13595
0.80392	0.80393
0.55353	0.55351
0.19334	0.19334
-0.6134	-0.61341
0.17582	0.17584

Figura 11. Coeficientes de reflexión para orden 20 entre la función *arburg* y *mi_pburg*.

Por último, para corroborar la similitud de los coeficientes de reflexión, se generó una gráfica de los coeficientes hasta un orden $p = 100$. Como se observa en la Figura 12, los coeficientes de reflexión obtenidos con las funciones arburg y mi_pburg son idénticos, por lo que ambas gráficas se solapan. Además, con este gráfico se comprueba de forma experimental que $|K_k| \leq 1$.

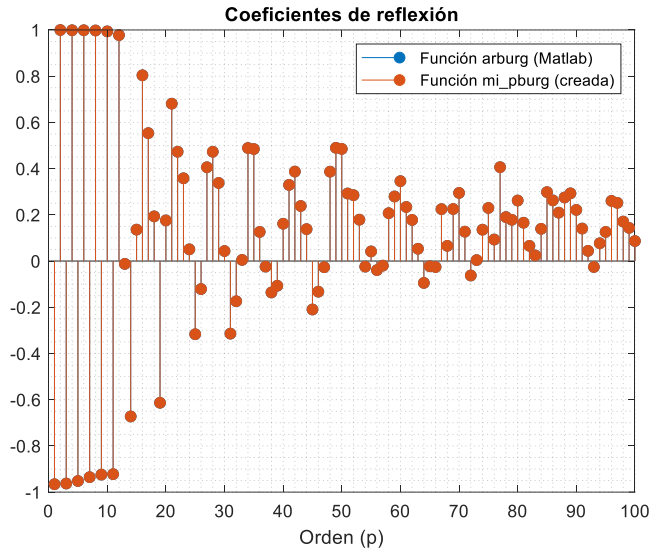


Figura 12. Coeficientes de reflexión en función del orden.

3.3 EVALUACIÓN AL VARIAR EL ORDEN DEL MÉTODO DE BURG

Para empezar con el análisis, se muestran los resultados obtenidos al estimar el espectro de la señal artificial mediante el periodograma simple y el periodograma modificado, con ventana Blackman-Harris de 4 términos.

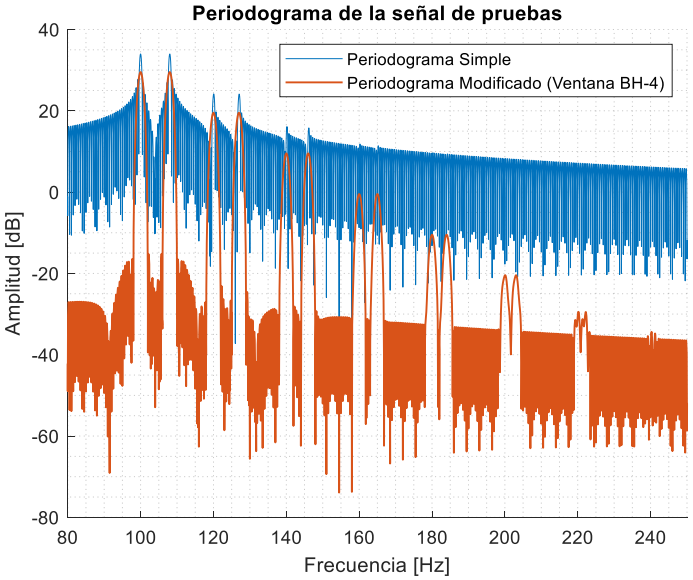
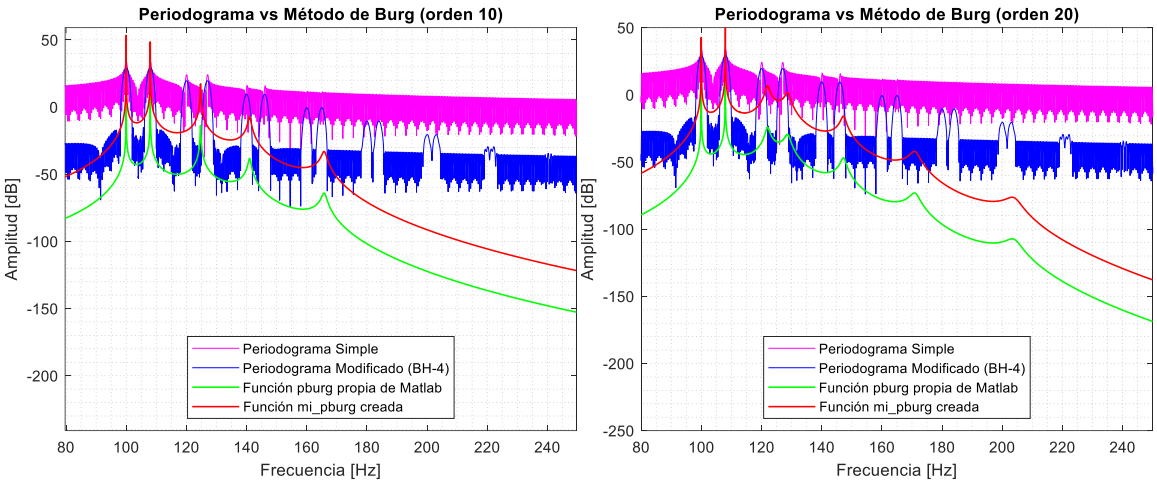


Figura 13. Espectro de la señal artificial en base al periodograma.

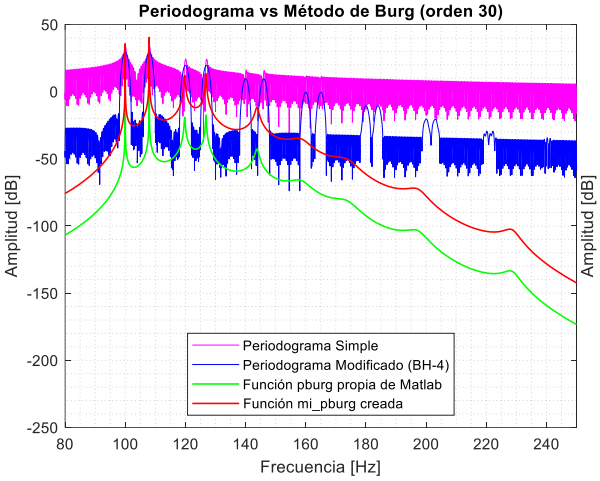
La señal artificial generada tiene 16 componentes de frecuencia, descritas en la Tabla 3. Una vez que se estimó su espectro, en la Figura 13, se observa que, el periodograma simple es capaz de detectar legiblemente, las 4 primeras componentes de frecuencia que se encuentran en 100, 108, 120 y 127 [Hz]. Las 4 componentes siguientes, que se encuentran en 140, 146, 160 y 165 [Hz] se distinguen con dificultad y, las 8 componentes restantes, definitivamente ya no se pueden observar.

En el caso del periodograma modificado, la detección de componentes mejora significativamente, por lo que, se pueden observar con claridad las 8 primeras componentes ya mencionadas y, se observan las siguientes componentes, que se encuentran en 180, 184, 200 y 203 [Hz]. Las componentes que están a 220 y 222 [Hz] ya no se distinguen fácilmente y, las últimas componentes, que se encuentran en 240 y 241 [Hz] ya no son detectables.

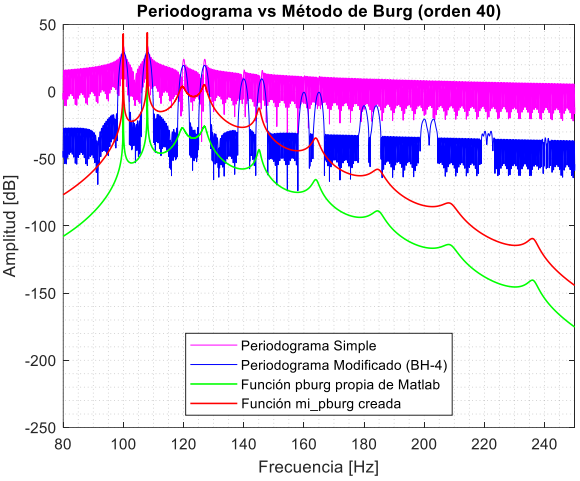
A continuación, en la Figura 14 se muestran los resultados de la comparación entre el periodograma simple, periodograma modificado con ventana Blackman-Harris de 4 términos y, el método de Burg (función propia y función de Matlab), con órdenes de p entre 10 y 100. En una misma gráfica se presentan los 4 espectros, que se pueden distinguir por su leyenda.



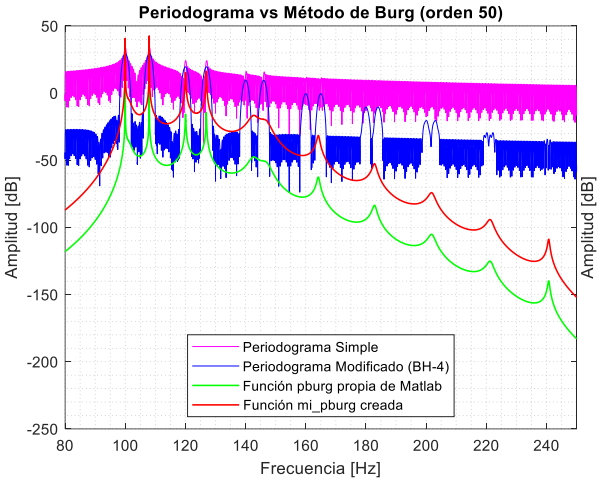
(a) (b)


Periodograma vs Método de Burg (orden 30)
Amplitud [dB]
Frecuencia [Hz]
Periodograma Simple
Periodograma Modificado (BH-4)
Función pburg propia de Matlab
Función mi_pburg creada

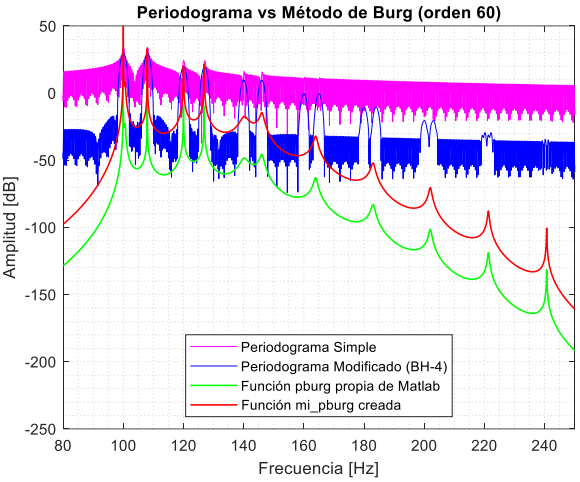
(c)


Periodograma vs Método de Burg (orden 40)
Amplitud [dB]
Frecuencia [Hz]
Periodograma Simple
Periodograma Modificado (BH-4)
Función pburg propia de Matlab
Función mi_pburg creada

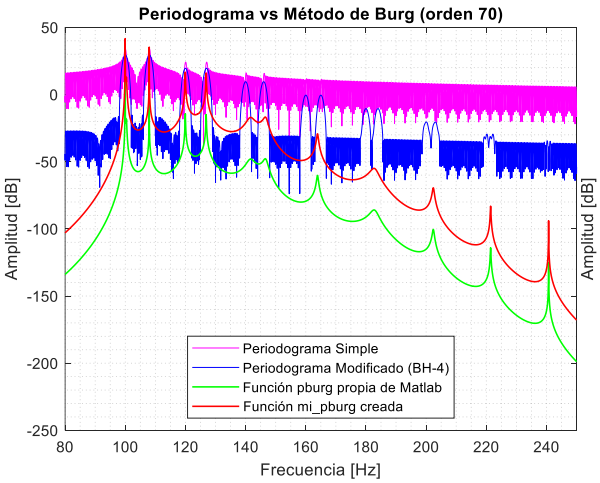
(d)


Periodograma vs Método de Burg (orden 50)
Amplitud [dB]
Frecuencia [Hz]
Periodograma Simple
Periodograma Modificado (BH-4)
Función pburg propia de Matlab
Función mi_pburg creada

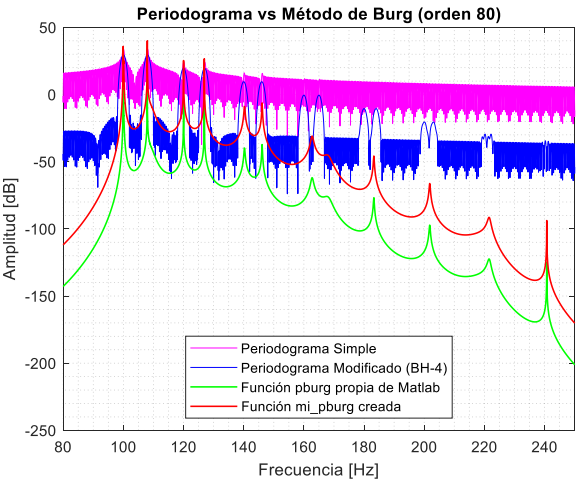
(e)


Periodograma vs Método de Burg (orden 60)
Amplitud [dB]
Frecuencia [Hz]
Periodograma Simple
Periodograma Modificado (BH-4)
Función pburg propia de Matlab
Función mi_pburg creada

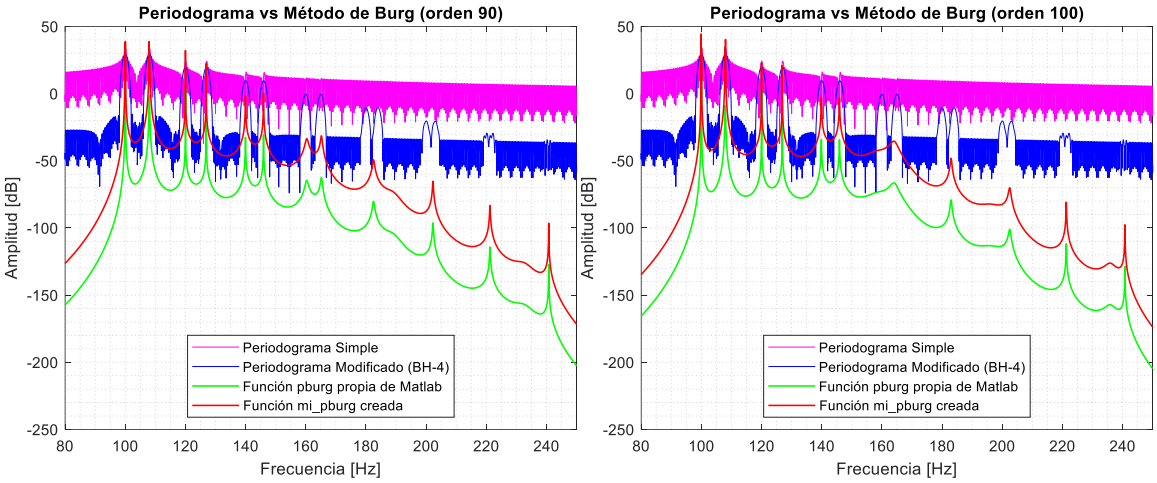
(f)


Periodograma vs Método de Burg (orden 70)
Amplitud [dB]
Frecuencia [Hz]
Periodograma Simple
Periodograma Modificado (BH-4)
Función pburg propia de Matlab
Función mi_pburg creada

(g)


Periodograma vs Método de Burg (orden 80)
Amplitud [dB]
Frecuencia [Hz]
Periodograma Simple
Periodograma Modificado (BH-4)
Función pburg propia de Matlab
Función mi_pburg creada

(h)

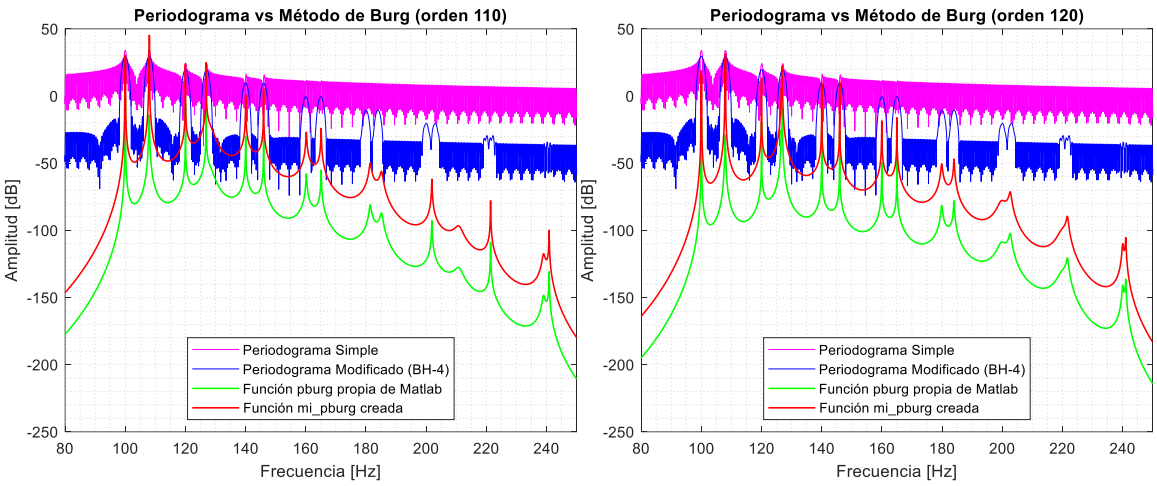


(i) (j)

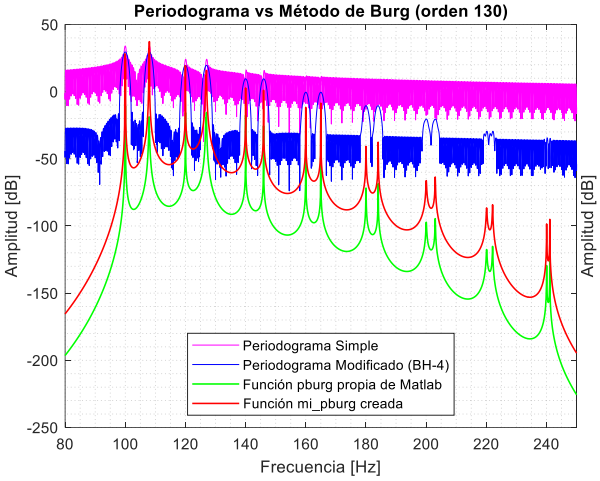
Figura 14. Comparación entre el periodograma y el método de Burg. (a) Orden 10, (b) Orden 20, (c) Orden 30, (d) Orden 40, (e) Orden 50, (f) Orden 60, (g) Orden 70, (h) Orden 80, (i) Orden 90 y (j) Orden 100.

Como se puede observar, al empezar con un orden 10, el método de Burg encuentra 5 picos de frecuencias, de los cuales, únicamente los dos primeros (100 y 108 [Hz]), fueron detectados correctamente. Luego, a medida que el orden aumenta, se puede observar que, se añaden más picos de frecuencia y, al llegar al orden 100, se tienen 11 componentes de frecuencia, en los que los 6 primeros, se ubican en la frecuencia correcta, mientras que los restantes, están ubicados incorrectamente. Además, no se llega a detectar todas las 16 componentes.

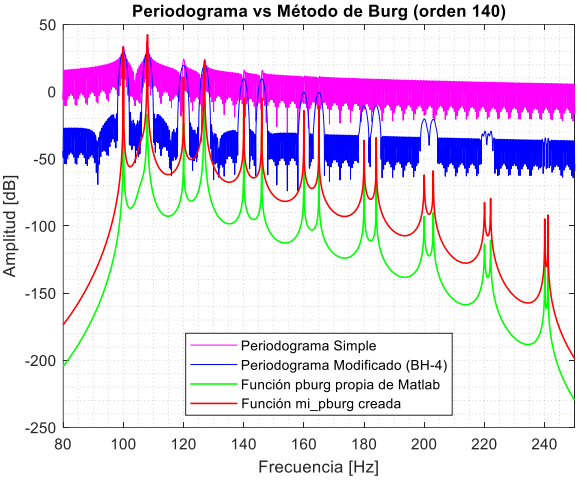
A continuación, se muestran los resultados de las simulaciones desde un orden de p de 110, hasta el orden 200.



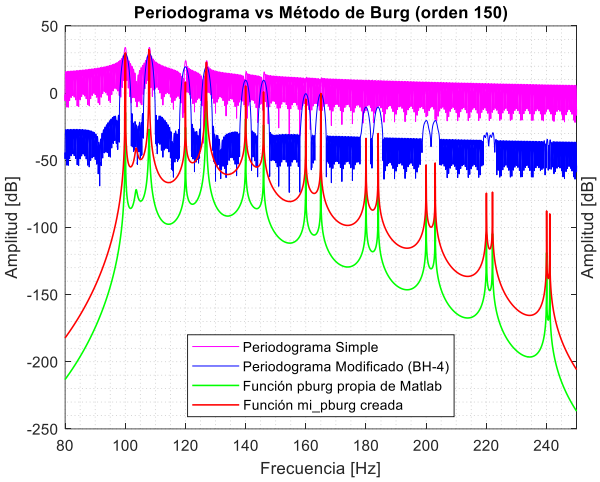
(a) (b)


Periodograma vs Método de Burg (orden 130)
Amplitud [dB]
Frecuencia [Hz]
Periodograma Simple
Periodograma Modificado (BH-4)
Función pburg propia de Matlab
Función mi_pburg creada

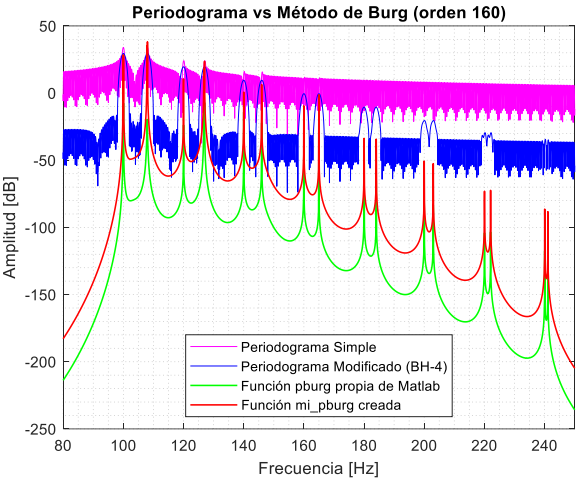
(c)


Periodograma vs Método de Burg (orden 140)
Amplitud [dB]
Frecuencia [Hz]
Periodograma Simple
Periodograma Modificado (BH-4)
Función pburg propia de Matlab
Función mi_pburg creada

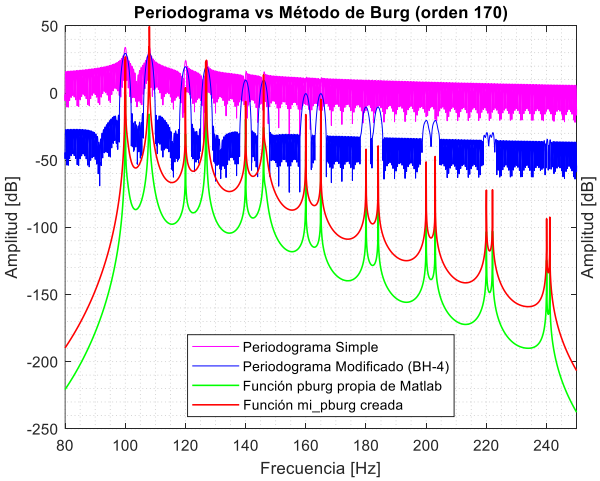
(d)


Periodograma vs Método de Burg (orden 150)
Amplitud [dB]
Frecuencia [Hz]
Periodograma Simple
Periodograma Modificado (BH-4)
Función pburg propia de Matlab
Función mi_pburg creada

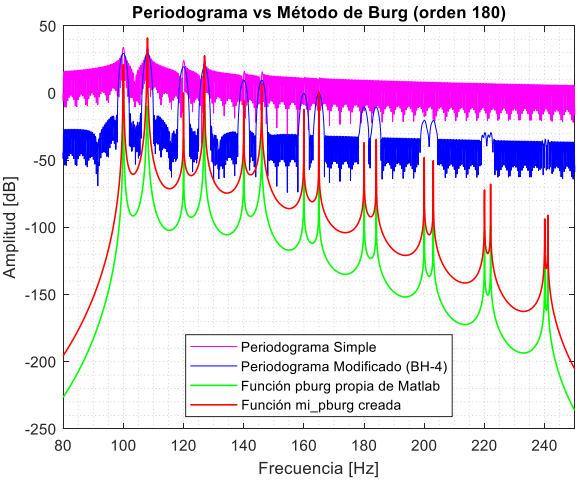
(e)


Periodograma vs Método de Burg (orden 160)
Amplitud [dB]
Frecuencia [Hz]
Periodograma Simple
Periodograma Modificado (BH-4)
Función pburg propia de Matlab
Función mi_pburg creada

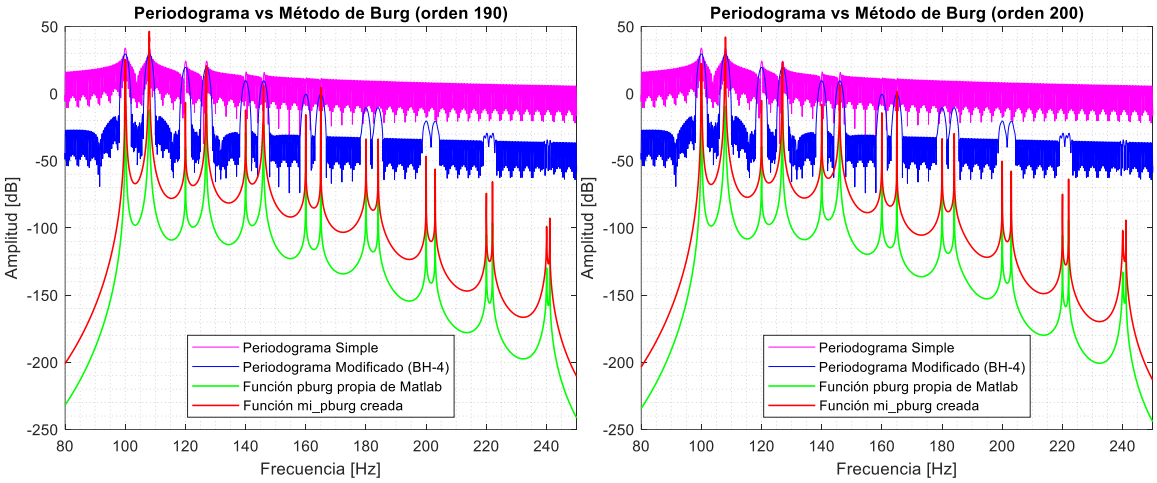
(f)


Periodograma vs Método de Burg (orden 170)
Amplitud [dB]
Frecuencia [Hz]
Periodograma Simple
Periodograma Modificado (BH-4)
Función pburg propia de Matlab
Función mi_pburg creada

(g)


Periodograma vs Método de Burg (orden 180)
Amplitud [dB]
Frecuencia [Hz]
Periodograma Simple
Periodograma Modificado (BH-4)
Función pburg propia de Matlab
Función mi_pburg creada

(h)



(i) (j)

Figura 15. Comparación entre periodograma y método de Burg. (a) Orden 110, (b) Orden 120, (c) Orden 130, (d) Orden 140, (e) Orden 150, (f) Orden 160, (g) Orden 170, (h) Orden 180, (i) Orden 190 y (j) Orden 200.

Como se muestra en la Figura 15, se observa que a partir del orden 130 el método de Burg ya es capaz de mostrar todas las componentes de frecuencia y, a partir de ese orden, el espectro tiende a mantenerse estable y, se muestran todas las componentes de frecuencia. No obstante, las amplitudes generadas en el espectro de Burg, no son correctas, puesto que los pares de componentes no se visualizan al mismo nivel de potencia.

En la Figura 16, se muestra una vista más amplia del gráfico de la Figura 15(c), con las marcas en las últimas 8 componentes de frecuencia de la señal analizada. Como se puede observar, el método de Burg proporciona una alta resolución en frecuencia y a la vez, tiene una alta capacidad de detección de componentes muy pequeñas, pues detecta sin problemas el par de componentes localizado a 220 y 222 [Hz], y también el par localizado a 240 y 241 [Hz], cuyas componentes son las más cercanas, con una diferencia de 1 [Hz] entre ellas.

Posteriormente, se realizaron pruebas adicionales del método de Burg desde el orden 300 hasta el orden 1000, en pasos de 100, para determinar el comportamiento del espectro en órdenes altos de p.

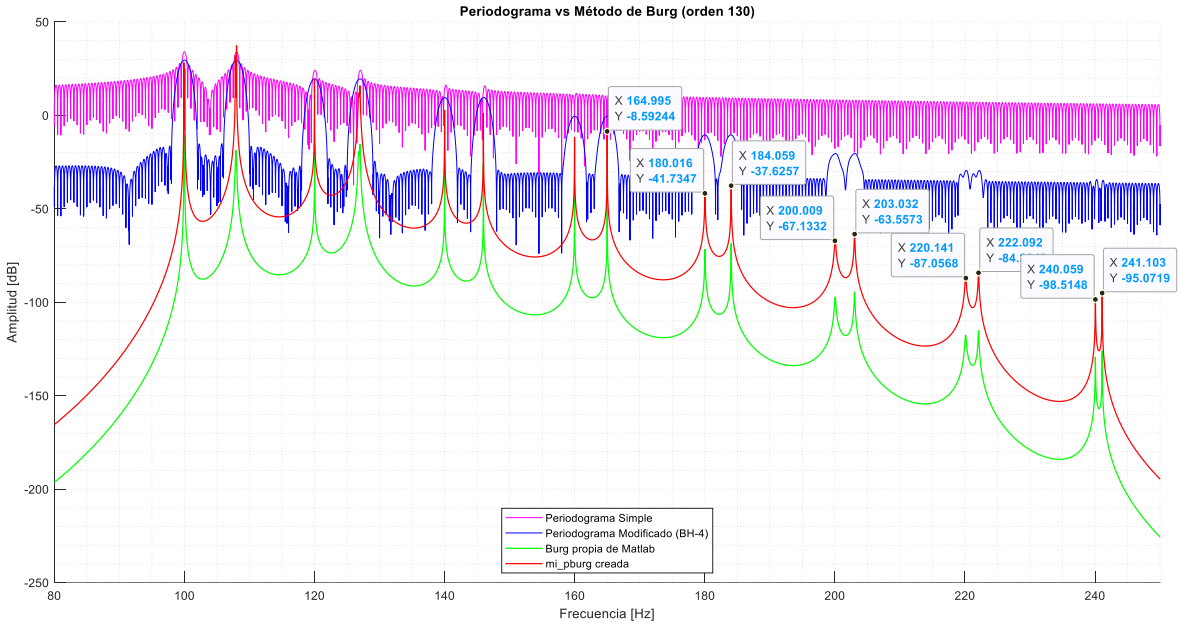
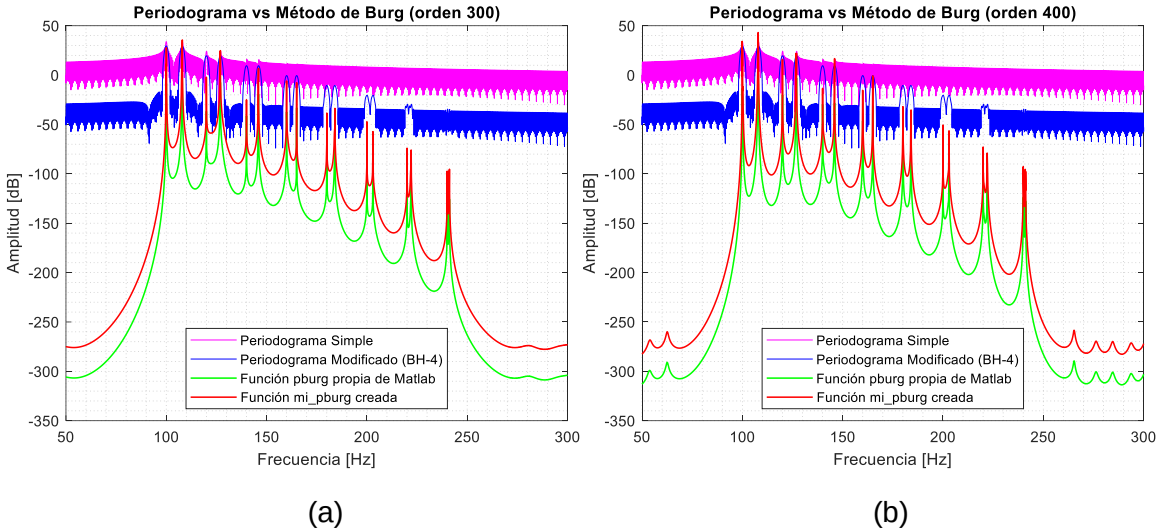
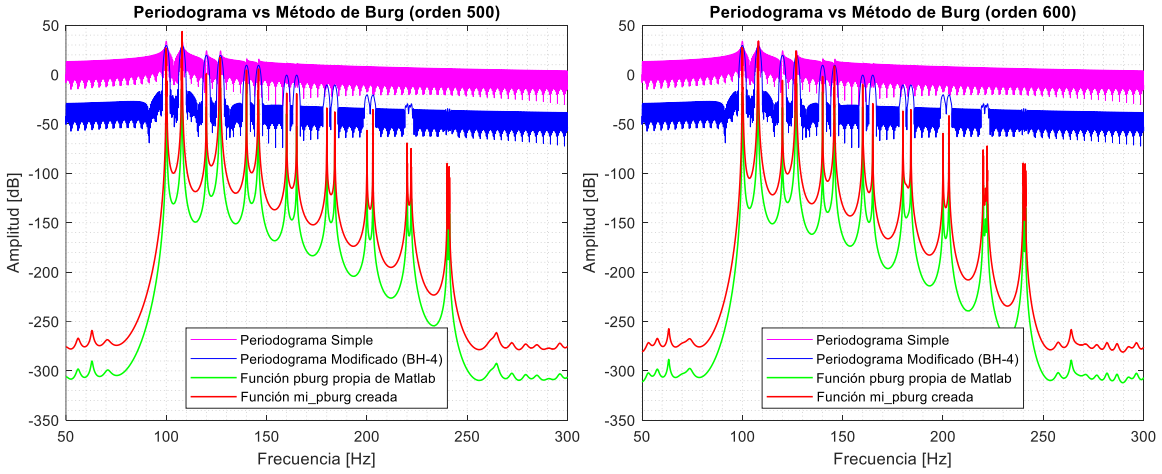


Figura 16. Comparación entre periodograma y método de Burg de orden 130.



(a) (b)



(c) (d)

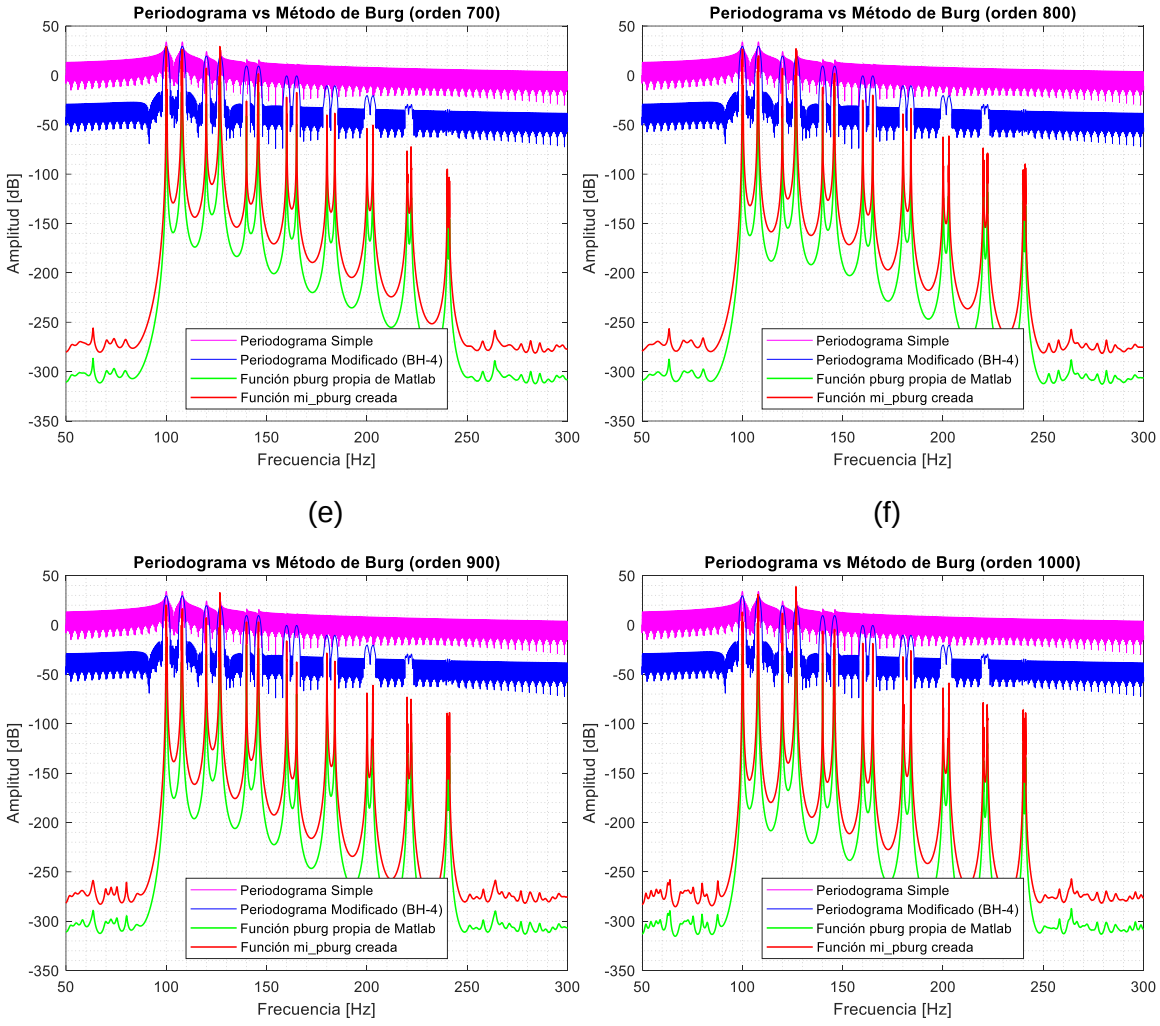
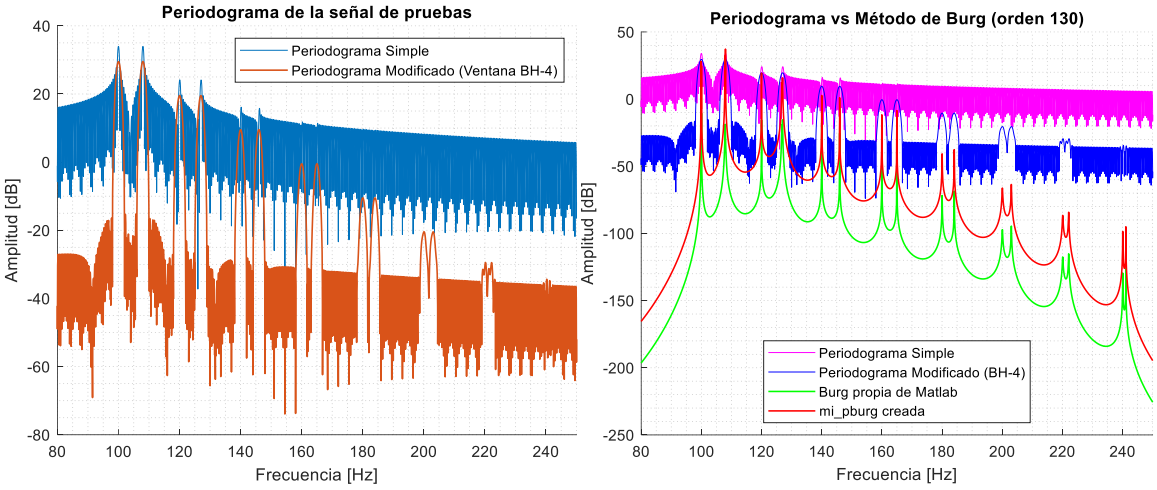


Figura 17. Comparación entre periodograma y método de Burg. (a) Orden 300, (b) Orden 400, (c) Orden 500, (d) Orden 600, (e) Orden 700, (f) Orden 800, (g) Orden 900 y (h) Orden 1000.

Al visualizar los resultados de las simulaciones, en la Figura 17, se observa que, a medida que se aumenta el orden de p, empiezan a aparecer componentes espurias o indeseadas, que se ubican antes de los 100 [Hz] y después de los 250 [Hz], las cuales, tienen amplitudes mucho menores a las componentes reales de la señal artificial.

3.4 EVALUACIÓN AL VARIAR LA DURACIÓN DE LA SEÑAL

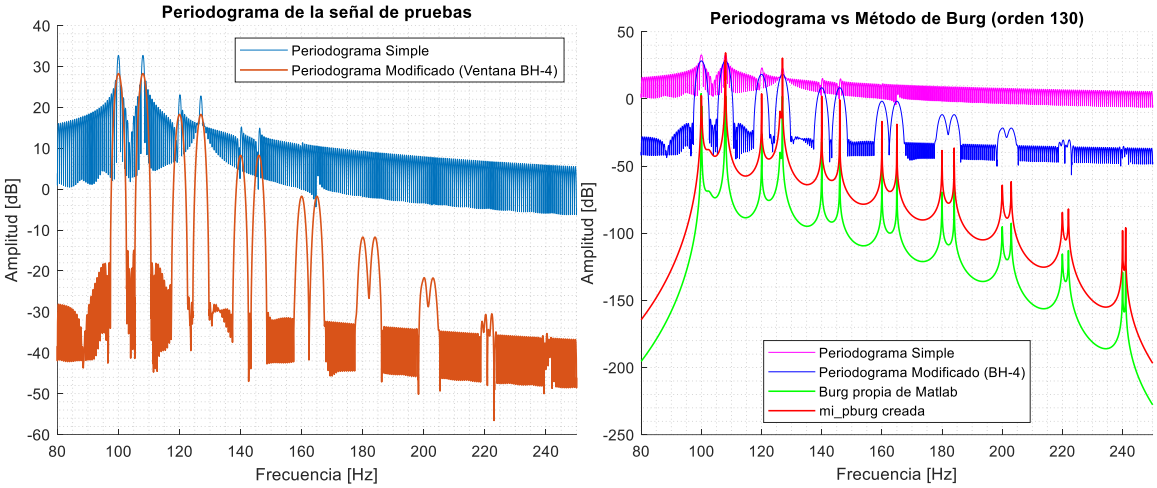
En estas pruebas, se consideró variar la duración de la señal, manteniendo fijo el orden del método de Burg. En este caso, se trabajó con un orden de $p = 130$, para el cual, anteriormente se determinó que este es el orden mínimo, en el que se pueden detectar todas las componentes de frecuencia de la señal artificial de pruebas. De esta manera, se realizaron simulaciones para una duración de 2, 1.5, 1 y 0.5 segundos.



(a) (b)

Figura 18. Comparación entre el periodograma y el método de Burg para una duración de la señal de 2 segundos. (a) Periodograma y (b) Método de Burg de orden 130.

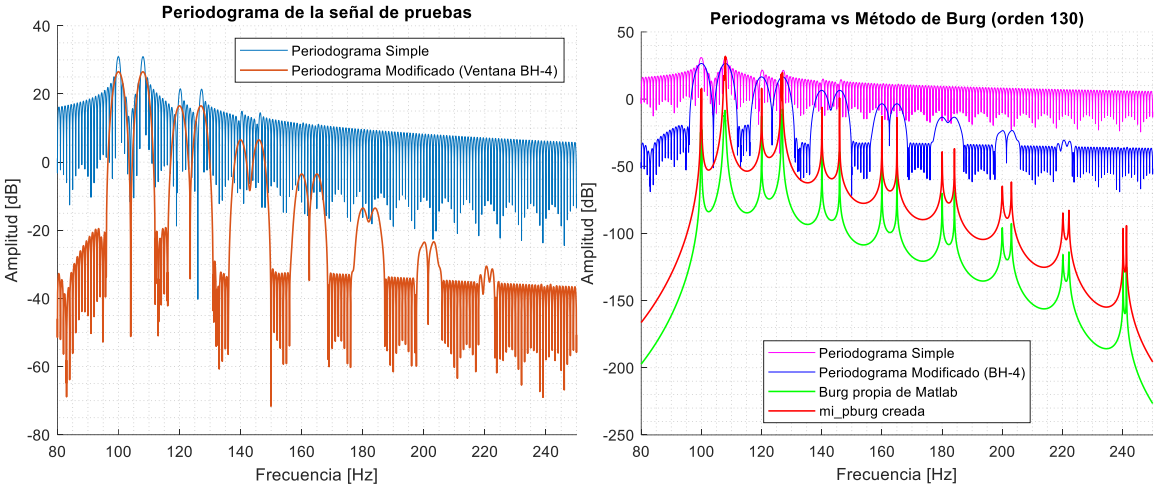
Los gráficos mostrados en la Figura 18, corresponden a la duración original de la señal, cuyos resultados ya se mostraron anteriormente.



(a) (b)

Figura 19. Comparación entre el periodograma y el método de Burg para una duración de la señal de 1.5 segundos. (a) Periodograma y (b) Método de Burg de orden 130.

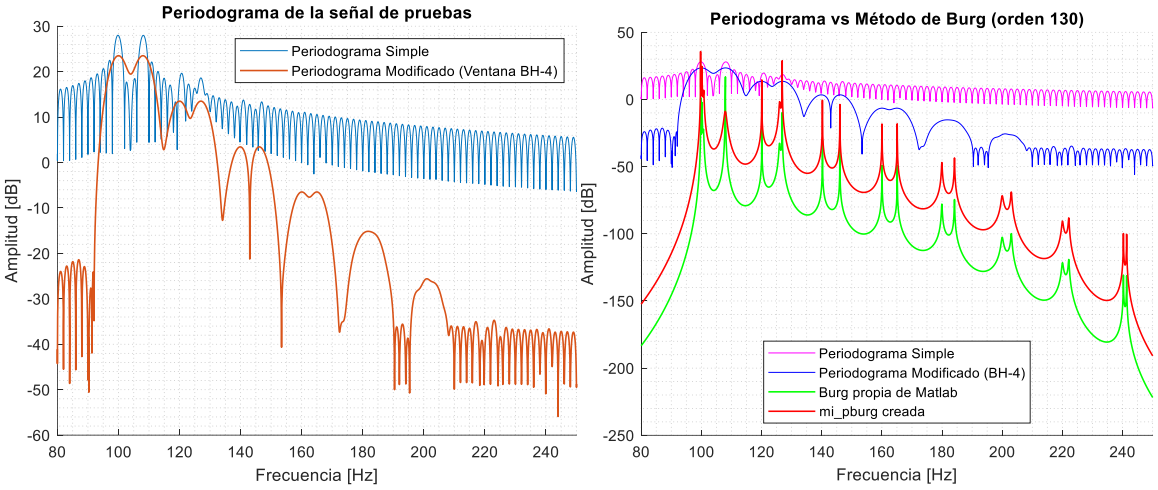
En la Figura 19(a), se observa que el periodograma empieza a perder un poco de resolución en frecuencia, ya que los lóbulos principales ubicados a partir de la octava componente, empiezan a juntarse, mientras que en la Figura 19(b), se observa que, los picos detectan las 16 componentes de frecuencia.



(a) (b)

Figura 20. Comparación entre el periodograma y el método de Burg para una duración de la señal de 1 segundo. (a) Periodograma y (b) Método de Burg de orden 130.

Luego, en la Figura 20(a), las componentes del periodograma se juntaron más, por lo que la calidad del espectro baja. Mientras que, en la Figura 20(b), se puede visualizar que el método de Burg no ha resultado afectado.



(a) (b)

Figura 21. Comparación entre el periodograma y el método de Burg para una duración de la señal de 0.5 segundos. (a) Periodograma y (b) Método de Burg de orden 130.

Por último, en la Figura 21(a) se distingue un periodograma muy degradado, en el que ya no se pueden distinguir todas las componentes de frecuencia, además de que algunas componentes muy cercanas se han unido en un solo lóbulo. Por tanto, para esta duración de la señal, el periodograma ya perdió resolución en frecuencia, y tampoco se pueden detectar las componentes más pequeñas. Por otro lado, en la Figura 21(b), el espectro del método de Burg tiene una ligera afectación en las amplitudes de las primeras 4 componentes, no obstante, aún fue capaz de detectar todas las 16 componentes de frecuencia.

4 CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

La teoría referente al algoritmo de Burg es amplia y engloba una variedad de temas que deben ser abordados. A pesar de esta complejidad, a través de un enfoque cuidadosamente estructurado en los conceptos clave y una organización adecuada, se logró desarrollar los pasos matemáticos de manera didáctica y esclarecedora. Esta metodología permitió que se pueda adquirir una base comprensible sobre la implementación.

Se procedió a la implementación didáctica del algoritmo de Burg utilizando el entorno de programación Matlab. Este proceso involucró una explicación detallada, paso a paso, de las líneas de código que conforman la función implementada *mi_pburg*. Las líneas de código se acompañaron con las correspondientes ecuaciones relacionadas, garantizando así una implementación transparente. De tal manera que se consiguió una presentación clara y concisa de la implementación del algoritmo de Burg, facilitando su comprensión y asimilación por parte del lector.

La implementación del método de Burg en la función *mi_pburg* fue sometida a pruebas para validar su correcto funcionamiento. En este proceso de verificación, se lograron obtener coeficientes de reflexión que demostraron ser idénticos a aquellos obtenidos mediante la función *arburg*. Adicionalmente, se llevó a cabo una evaluación gráfica comparativa de los espectros generados por las funciones *mi_pburg* y *pburg*. Se constató que estos espectros presentaban una forma similar, destacando que la única variación significativa radicaba en un desplazamiento vertical en el espectro generado por *mi_pburg*. No obstante, este desplazamiento no ejerció ningún impacto en la capacidad de detectar componentes de amplitud muy pequeña, ni en la resolución en frecuencia.

De acuerdo con los resultados obtenidos en las simulaciones, el método de Burg proporciona una alta resolución en frecuencia y, también es capaz de detectar componentes de amplitud muy pequeña. Además, no presenta el problema de los lóbulos secundarios de las sincs, característico del periodograma, pues el espectro de Burg es suave y continuo. Asimismo, las componentes son fácilmente identificables por sus picos prominentes. Sin embargo, la desventaja del método de Burg está presente en las amplitudes de los picos, pues éstos no se muestran al mismo nivel de potencia en cada par de componentes, como fueron definidas en la señal artificial.

Al manipular la duración de la señal, se pudo constatar de manera evidente la superioridad del método de Burg en comparación con el periodograma simple y el periodograma modificado. A medida que la duración de la señal se reducía, se observó cómo los periodogramas experimentaban una disminución en su resolución en frecuencia y su capacidad de detección. Contrariamente, los espectros generados mediante el método de Burg mantenían una resolución constante en frecuencia. Como resultado, las 16 componentes de frecuencia se identificaron sin dificultades. No obstante, persistió el problema de la amplitud de los picos.

4.2 RECOMENDACIONES

Dentro del contexto del modelo paramétrico autorregresivo desarrollado en este estudio, también debe tomarse en cuenta el problema de la selección del orden del modelo. Como se evidenció en el trabajo, se llevaron a cabo simulaciones en un rango de p, con el propósito de identificar el orden más apropiado. En este sentido, se recomienda examinar la literatura existente sobre los enfoques paramétricos de estimación espectral, puesto que, existen criterios fundamentados para escoger un orden adecuado del modelo $AR(p)$.

Dado que el algoritmo de Burg fue concebido como un algoritmo computacionalmente eficiente, se recomienda valuar la carga computacional al emplear la función *pburg* en contraste con la función *mi_pburg*, puesto que al realizar comparativas durante la simulación con diferentes órdenes de p, puede arrojar una perspectiva adicional sobre la correcta implementación del algoritmo.

Por último, se recomienda ampliar las pruebas del método de Burg mediante la exploración de diversas señales. Estas señales podrían abarcar tanto aquellas de naturaleza determinística como reales, provenientes de fenómenos inherentes a la naturaleza. Esta recomendación se desprende del entendimiento teórico de que los modelos paramétricos autorregresivos se originaron en la modelización de procesos aleatorios discretos estacionarios. Mediante la aplicación del método de Burg a una gama más amplia de señales, se podría afianzar aún más su versatilidad y validez en una variedad de contextos. Esto permitiría comprender su capacidad para capturar características esenciales de señales provenientes de diferentes dominios, enriqueciendo así su aplicabilidad y su valor como herramienta analítica.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] L. H. Koopmans, *The spectral analysis of time series*, 2nd ed. en Probability and mathematical statistics, no. v. 22. San Diego: Academic Press, 1995.

[2] R. N. Bracewell, *The Fourier transform and its applications*, 3rd ed. en McGraw-Hill series in electrical and computer engineering. Boston: McGraw Hill, 2000.

[3] P. Bloomfield, *Fourier analysis of time series: an introduction*, 2nd ed. en Wiley series in probability and statistics. Applied probability and statistics section. New York: Wiley, 2000.

[4] S. L. Marple, «A tutorial overview of modern spectral estimation», en *International Conference on Acoustics, Speech, and Signal Processing*, Glasgow, UK: IEEE, 1989, pp. 2152-2157. doi: 10.1109/ICASSP.1989.266889. Disponible en: http://ieeexplore.ieee.org/document/266889/. [Accedido: 18 de julio de 2023]

[5] S. M. Kay y S. L. Marple, «Spectrum analysis—A modern perspective», *Proc. IEEE*, vol. 69, n.º 11, pp. 1380-1419, 1981, doi: 10.1109/PROC.1981.12184

[6] W. H. Foy, «Comparison of methods for spectral estimation with interrupted data», *IEEE Trans. Signal Process.*, vol. 41, n.º 3, pp. 1449-1453, mar. 1993, doi: 10.1109/78.205754

[7] A. Mordojovich y R. Roberts, «A comparison of spectral estimators for real data», en *ICASSP '81. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Atlanta, Georgia, USA: Institute of Electrical and Electronics Engineers, 1981, pp. 492-495. doi: 10.1109/ICASSP.1981.1171200. Disponible en: http://ieeexplore.ieee.org/document/1171200/. [Accedido: 18 de julio de 2023]

[8] G. B. Rossi, F. Crenna, V. Piscopo, y A. Scamardella, «Comparison of Spectrum Estimation Methods for the Accurate Evaluation of Sea State Parameters», *Sensors*, vol. 20, n.º 5, p. 1416, mar. 2020, doi: 10.3390/s20051416

[9] I. Seroussi y N. Sochen, «Spectral Analysis of a Non-Equilibrium Stochastic Dynamics on a General Network», *Sci Rep*, vol. 8, n.º 1, p. 14333, sep. 2018, doi: 10.1038/s41598-018-32650-5

[10] G. F. Lothian, «Spectral Analysis», *Nature*, vol. 234, n.º 5327, pp. 278-278, dic. 1971, doi: 10.1038/234278a0

[11] G. Froyland, D. Giannakis, B. R. Lintner, M. Pike, y J. Slawinska, «Spectral analysis of climate dynamics with operator-theoretic approaches», *Nat Commun*, vol. 12, n.º 1, p. 6570, nov. 2021, doi: 10.1038/s41467-021-26357-x

[12] P. Stoica y R. L. Moses, *Spectral analysis of signals*. Upper Saddle River, N.J: Pearson/Prentice Hall, 2005.

[13] J. G. Proakis y D. G. Manolakis, *Digital signal processing*, 4th ed. Upper Saddle River, N.J: Pearson Prentice Hall, 2007.

[14] J. B. J. Fourier, *The Analytical Theory of Heat*, 1.ª ed. Cambridge University Press, 2009. doi: 10.1017/CBO9780511693205. Disponible en: https://www.cambridge.org/core/product/identifier/9780511693205/type/book. [Accedido: 21 de julio de 2023]

[15] E. A. Robinson, «A historical perspective of spectrum estimation», *Proc. IEEE*, vol. 70, n.º 9, pp. 885-907, sep. 1982, doi: 10.1109/PROC.1982.12423

[16] J. W. Cooley y J. W. Tukey, «An algorithm for the machine calculation of complex Fourier series», *Mathematics of computation*, vol. 19, n.º 90, pp. 297-301, 1965.

[17] A. Prieto-Guerrero y G. Espinosa-Paredes, «Linear signal processing methods and decay ratio estimation», en *Linear and Non-Linear Stability Analysis in Boiling Water Reactors*, Elsevier, 2019, pp. 269-314. doi: 10.1016/B978-0-08-102445-4.00006-0. Disponible en: https://linkinghub.elsevier.com/retrieve/pii/B9780081024454000060

[18] S. M. Kay, *Modern spectral estimation: theory and application*. en Prentice-Hall signal processing series. Englewood Cliffs, N.J: Prentice Hall, 1988.

[19] «Time Series Analysis: The Basics». Disponible en: https://www.abs.gov.au/websitedbs/d3310114.nsf/home/time+series+analysis:+the+basics

[20] R. Álvarez, «Apuntes de Cátedra de Procesamiento Digital de Señales».

[21] TrainDataHub, «Clear Explanations of AR, MA, ARMA, and ARIMA in Times Series Analysis», *Medium*, 3 de enero de 2022. Disponible en: https://medium.com/@ooemma83/clear-explanations-of-ar-ma-arma-and-arima-in-times-series-analysis-9a72ff569dee

[22] J. E. Kim Ari Arango, Joshua, *Chapter 2 Modelling Time Series | Time Series for Beginners*. Disponible en: https://bookdown.org/JakeEsprabens/431-Time-Series/modelling-time-series.html. [Accedido: 24 de agosto de 2023]

[23] Mathuranathan, «Understand AR, MA and ARMA models», *GaussianWaves*, 22 de mayo de 2014. Disponible en: https://www.gaussianwaves.com/2014/05/lti-system-models-for-random-signals-ar-ma-and-arma-models/

[24] M. Ríos, «Análisis espectral multivariable aplicado a señales cerebrales reales (EEG)», Universidad de Sevilla, Sevilla, 2005.

[25] «Stochastic Process - Definition, Classification, Types and Facts», *VEDANTU*. Disponible en: https://www.vedantu.com/maths/stochastic-process, https://www.vedantu.com/maths/stochastic-process

[26] N. Zorba, «Stationarity Vs Ergodicity In Time Series Data In Econometrics», 4 de febrero de 2023. Disponible en: https://learneconometricsfast.com/stationarity-vs-ergodicity-in-time-series-data-in-econometrics

[27] «What is Correlation in Signals and Systems». Disponible en: https://www.tutorialspoint.com/what-is-correlation-in-signals-and-systems

[28] «Hermitian Matrix - Definition, Properties, Examples, and FAQs», *GeeksforGeeks*, 24 de marzo de 2023. Disponible en: https://www.geeksforgeeks.org/hermitian-matrix/

[29] J. P. Burg, *Maximum entropy spectral analysis.* Stanford University, 1975.

[30] «Signal Processing Toolbox Documentation - MathWorks América Latina». Disponible en: https://la.mathworks.com/help/signal/

[31] «Fast Fourier transform - MATLAB fft - MathWorks». Disponible en: https://la.mathworks.com/help/matlab/ref/fft.html?lang=en

[32] «Ventana rectangular - MATLAB rectwin - MathWorks América Latina». Disponible en: https://la.mathworks.com/help/signal/ref/rectwin.html

[33] «Ventana de Blackman-Harris mínima de cuatro términos - MATLAB blackmanharris - MathWorks América Latina». Disponible en: https://la.mathworks.com/help/signal/ref/blackmanharris.html

[34] «Autoregressive power spectral density estimate — Burg's method - MATLAB pburg - MathWorks América Latina». Disponible en: https://la.mathworks.com/help/signal/ref/pburg.html

[35] «Autoregressive all-pole model parameters — Burg's method - MATLAB arburg - MathWorks América Latina». Disponible en: https://la.mathworks.com/help/signal/ref/arburg.html. [Accedido: 24 de agosto de 2023]

[36] A. V. Oppenheim y R. W. Schafer, *Discrete-time signal processing*, 3rd ed. Upper Saddle River: Pearson, 2010.

[37] M. Bayas, «Modelos Autorregresivos de Análisis Espectral», Escuela Politécnica Nacional, Quito, 1984.

[38] «Invertir un arreglo de arriba abajo - MATLAB flipud - MathWorks América Latina». Disponible en: https://la.mathworks.com/help/matlab/ref/flipud.html

[39] F. J. Harris, «On the use of windows for harmonic analysis with the discrete Fourier transform», *Proc. IEEE*, vol. 66, n.º 1, pp. 51-83, 1978, doi: 10.1109/PROC.1978.10837

6 ANEXOS

ANEXO I. Código de la función *mi_pburg.m (MATLAB)*

ANEXO II. Código del script *Principal_Burg.m* (MATLAB)

ANEXO I. CÓDIGO DE LA FUNCIÓN MI_PBURG.M (MATLAB)

```
function [Pxx,freq,a,K]= mi_pburg(x,p,nfft,Fs)
% FUNCIÓN MI_PBURG
% ALEX JAVIER RAMOS VÁSQUEZ
% Esta función estima el espectro de una señal, por medio del algoritmo de
% Burg
%%%%%%%%%%%% Parámetros de entrada %%%%%%%%%%%%%%%%%%%
% x: Señal de entrada
% p: orden del modelo AR
% nfft: tamaño de la FFT
% Fs: frecuencia de muestreo de la señal x
%%%%%%%%%%%% Parámetros de salida %%%%%%%%%%%%%%%%%%%%
% Pxx: Densidad espectral de potencia estimada de x
% F: Vector del eje de frecuencias
% a: Vector de coeficientes paramétricos {ap[k]}
% k: Vector de coeficientes de reflexion Kk
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Paso 1: Inicialización de los parámetros
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N  = length(x); % Longitud de la señal de entrada
ef = x(2:N); % Inicializacion del vector de errores de prediccion hacia
adelante
eg = x(1:N-1); % Inicializacion del vector de errores de prediccion hacia
atrás
K = zeros(p,1); % Inicialización del vector de coeficientes de reflexión {Kk}
sigma_vec = zeros(p,1); % Inicialización del vector de errores de mínimos
cuadrados
sigma0 = x*x'/N; % Primer elemento del vector de errores de mínimos cuadrados
sigma_vec(1) = sigma0; % Primer elemento del vector de errores de mínimos
cuadrados
a = 1; % Primer elemento del vector de coeficientes {ap[k]}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Paso 2: Lazo iterativo para el cálculo de K,{ap[k]}, y sigma
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k=1:p % el bucle se itera desde k=1 hasta p
    num = sum(ef.*eg); % numerador de la ecuacion de Kk
    den = sum(ef.^2 + eg.^2); % denominador de la ecuacion Kk
    K(k) = -2*num/den; % Cálculo del coeficiente de reflexión
    sigma_vec(k+1) = (1-K(k)^2)*sigma_vec(k); % Cálculo recursivo del error
de mínimos cuadrados
    a = [a;0] + K(k) * [0;flipud(conj(a))]; % Actualización de los
coeficientes {ap[k]}
    ef_nuevo = ef   + K(k)*eg; % actualizacion de errores de prediccion hacia
adelante
    eg_nuevo = K(k)*ef  + eg; % actualizacion de errores de prediccion hacia
atrás
    ef = ef_nuevo(2:end); % actualización del vector ef
    eg = eg_nuevo(1:end-1); % actualización del vector eg
end
% Fin del algoritmo Iterativo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Paso 3: Estimación de la PSD a partir de los parámetros
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Af = abs( fft( a, nfft )).^ 2; % Función de trasferencia del denominador del
filtro
Pxx = sigma_vec(end) ./ Af; % Ecuación del espectro de potencia
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Paso 4: Control de parámetros de salida
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Si la señal de entrada es real, se selecciona la 1era mitad del espectro
if isreal(x)
   select = (1:floor(nfft/2)+1)'; % seleccion de las frecuencias positivas
   Pxx = Pxx(select); % Actualizacion del vector de la PSD estimada
   % Correcciones del espectro unilateral
   Pxx(1)=Pxx(1)/2;
   Pxx(end)=Pxx(end)/2;
else
   select = (1:nfft)'; % seleccion de frecuencias hasta nfft
   Pxx = Pxx(select); % Actualizacion del vector de PSD estimada
end
freq = (select - 1)*Fs/nfft; % calculo del eje de frecuencias
% Variables de salida
% P = Pxx; % PSD estimada
% freq = ff; % Vector de frecuencias
% Kk = K; % Coeficientes de reflexion {Kk}
% ak = a; % Coeficientes parametricos {ap[k]}
end
```

ANEXO II. CÓDIGO DEL SCRIPT PRINCIPAL_BURG.M (MATLAB)

```
% ESCUELA POLITÉCNICA NACIONAL
% TRABAJO DE INTEGRACIÓN CURRICULAR
% COMPARACIÓN DEL MÉTODO DE BURG CON EL PERIODOGRAMA SIMPLE Y MODIFICADO
% ALEX JAVIER RAMOS VÁSQUEZ
% --- SCRIPT PRINCIPAL ---
%% GENERACIÓN DE LA SEÑAL CON 16 COMPONENTES:
% En este bloque se genera la señal artificial para las pruebas
clear all; clc
rng default
duracion = 2; % duracion de la señal
Fmax = 250; % frecuencia maxima de la señal
Fs = 10*Fmax; % frecuencia de muestreo de la señal
Ts = 1/Fs; % periodo de muestreo de la señal
t=0:Ts:duracion; % vector de tiempos
% Declaración de la señal
x = 1*sin(2*pi*100*t)+1*sin(2*pi*108*t)+...
  0.1*sin(2*pi*120*t)+0.1*sin(2*pi*127*t)+...
  0.01*sin(2*pi*140*t)+0.01*sin(2*pi*146*t)+...
  0.001*sin(2*pi*160*t)+0.001*sin(2*pi*165*t)+...
  0.0001*sin(2*pi*180*t)+0.0001*sin(2*pi*184*t)+...
  0.00001*sin(2*pi*200*t)+0.00001*sin(2*pi*203*t)+...
  0.000001*sin(2*pi*220*t)+0.000001*sin(2*pi*222*t)+...
  0.0000001*sin(2*pi*240*t)+0.0000001*sin(2*pi*241*t);
% Grafica de la senial en el dominio del tiempo
f1 = figure;
%f1.Position = [420 200 700 560];
plot(t,x,'r','LineWidth',0.9)
title('Señal artificial multicomponente')
xlabel('Tiempo [s]')
ylabel('Amplitud')
grid minor
axis([0 duracion/5 -3.5 3.5])

%% ESPECTRO DE LA SEÑAL CON LOS PERIODOGRAMAS
N = length(x); % numero de muestras de la señal discretizada
FACTOR = 64; % factor de la nFFT
nFFT = 2^ceil(log2(N))*FACTOR;
%nFFT = N;
% 1. GENERACION DE VENTANAS:
vent_1 = rectwin(N); % ventana rectangular
vent_2 = blackmanharris(N); % ventana blackmanharris
% 2. Enventanado de la senial temporal:
x1_enventanada = x.*vent_1'; % señal con ventana rectangular
x2_enventanada = x.*vent_2'; % señal con ventana blackmanharris
% 3. Periodograma modificado:
Periodograma_Simple = 10*log10(abs(fft(x1_enventanada,nFFT)));
Periodograma_Modificado = 10*log10(abs(fft(x2_enventanada,nFFT)));
%Se genera el eje de frecuencias:
f_periodograma = linspace(0,Fs,nFFT); % vector de frecuencias
f2 = figure;
%f2.Position = [420 200 700 560];
hold on
plot(f_periodograma,Periodograma_Simple);  %
plot(f_periodograma,Periodograma_Modificado,'LineWidth',1.0);
%axis([50 250 min(Periodograma_Simple) max(Periodograma_Simple)+50]);
xlim([80 250]);
```

```
xlabel('Frecuencia [Hz]');
ylabel('Amplitud [dB]')
legend('Periodograma Simple','Periodograma Modificado (Ventana BH-4)')
title('Periodograma de la señal de pruebas')
grid minor
hold off
%% ESPECTRO DE LA SEÑAL CON EL MÉTODO DE BURG
orden = 25; % orden del modelo AR(p)
% Espectro con la funcion pburg de Matlab
[pb1,fb1] = pburg(x,orden,nFFT,Fs);
f3 = figure;
%f3.Position = [420 200 700 560];
hold on
plot(f_periodograma,Periodograma_Simple,'m'); % grafico del periodograma
simple
plot(f_periodograma,Periodograma_Modificado,'b'); % grafico del periodograma
modificado
plot(fb1, 10*log10(pb1),'g','LineWidth',1.0) % grafico
% Espectro usando la funcion mi_pburg
[pb2,fb2,a_mipburg,k_mipburg] = mi_pburg(x,orden,nFFT,Fs);
%[pb3,fb3,cof2,ref_old] = mi_pburg(x,orden,nFFT,Fs);
plot(fb2,10*log10(pb2),'r','LineWidth',1.0)
xlabel('Frecuencia [Hz]');
ylabel('Amplitud [dB]')
legend('Periodograma Simple','Periodograma Modificado (BH-4)','Burg propia de
Matlab','mi\_pburg creada','Location','south')
title(['Periodograma vs Método de Burg (orden ',num2str(orden), ')'])
xlim([80 250]);
ylim([-250 50])
grid minor
%% COMPROBACIÓN DE COEFICIENTES DE REFLEXION
% Obtencion de parametros con la funcion arburg
[a_arburg,sigma_arburg,k_arburg] = arburg(x,orden);
% comparacion entre los coeficientes devueltos por arburg y mi_pburg
T = table(k_arburg,k_mipburg)
% Grafica de los coeficientes de reflexion
figure
stem(k_mipburg,'filled')
hold on
stem(k_arburg,'filled')
title('Coeficientes de reflexión')
xlabel('Orden (p)')
legend('Función arburg (Matlab)','Función mi\_pburg (creada)')
grid minor
%% PRUEBAS PARA UN RANGO DE ORDEN P
orden = 10:10:100;
for i=orden
    [pb1,fb1] = pburg(x,i,nFFT,Fs);
    [pb2,fb2] = mi_pburg(x,i,nFFT,Fs);
    figure
    plot(f_periodograma,Periodograma_Simple,'m',...
        'DisplayName','Periodograma Simple'); % grafico del periodograma
simple
    hold on
    plot(f_periodograma,Periodograma_Modificado,'b',...
        'DisplayName','Periodograma Modificado (BH-4)'); % grafico del
periodograma modificado
    plot(fb1, 10*log10(pb1),'g','LineWidth',1.0,...
```

```
        'DisplayName','Función pburg propia de Matlab') % grafico
    plot(fb2,10*log10(pb2),'r','LineWidth',1.0,...
        'DisplayName','Función mi\_pburg creada')
    xlabel('Frecuencia [Hz]');
    ylabel('Amplitud [dB]')
    legend('Location','south')
    title(['Periodograma vs Método de Burg (orden ',num2str(i), ')'])
    xlim([80 250]);
    ylim([-250 50])
    grid minor
    hold off
end
```