



# DESARROLLO DE UNA INTERFAZ GRÁFICA EN APP DESIGNER DEL MATLAB PARA REFORZAR LA ENSEÑANZA DEL ANÁLISIS TEMPORAL-FRECUENCIAL DE SEÑALES DIGITALES

## *DEVELOPMENT OF GRAPHIC INTERFACE IN MATLAB APP DESIGNER TO REINFORCE TEACHING OF TEMPORAL-FREQUENCY ANALYSIS OF DIGITALS SIGNALS*

Pedro Huamaní Navarrete  
Universidad Ricardo Palma, Lima, Perú

RECIBIDO: 20 de septiembre de 2023.  
ACEPTADO: 30 de noviembre de 2023

### RESUMEN

Este artículo describe el procedimiento de implementación de una interfaz gráfica para usuarios en el que se utiliza el entorno de desarrollo App Designer y el Toolbox Signal Processing del software Matlab para reforzar la enseñanza del análisis en el dominio del tiempo y la frecuencia de señales periódicas y típicas empleadas en la asignatura Procesamiento Digital de Señales. Esta pertenece al programa de Ingeniería Electrónica de la Universidad Ricardo Palma. Dicha interfaz fue desarrollada a partir de varios componentes como DropDown, Button, Knob, Axes, Slider, entre otros más. Estos permitieron la elección del tipo de señal desde una lista, elección de la frecuencia de muestreo y del porcentaje de sobreposición, el tamaño de la ventana para segmentar la señal temporal, y la representación gráfica temporal, frecuencial y temporal-frecuencial. Así, el código fuente de la implementación de la interfaz gráfica es compartida con los estudiantes, lo que les permitió interactuar directamente con la interfaz gráfica y manipularla. Finalmente, se presentan cuatro resultados de las simulaciones, así como distintas frecuencias de muestreo y porcentaje de sobreposición.

**Palabras claves:** Espectrograma, Transformada Discreta de Fourier, entorno App Designer, software Matlab, Librería de Procesamiento de Señales.

### Cómo citar

P. F. Huamaní Navarrete, «Desarrollo de una Interfaz gráfica en App Designer del Matlab para reforzar la enseñanza del análisis temporal-frecuencial de señales digitales», *Perfiles\_Ingenieria*, vol. 19, n.º 20, pp. 147–166, dic. 2023.

### ABSTRACT

This article describes the implementation procedure of a graphical user interface in which the App Designer development environment and the Signal Processing Toolbox of the Matlab software are used to reinforce the teaching of analysis in the time and frequency domain of periodic signals and typical techniques used in the Digital Signal Processing subject. This belongs to the Electronic Engineering program at the Ricardo Palma University. Likewise, this interface was developed using various components such as DropDown, Button, Knob, Axes, Slider, among others. These allowed the choice of the type of signal from a list, choice of the sampling frequency and the percentage of overlap, the size of the window to segment the temporal signal, and the temporal, frequency and time-frequency graphical representation. Thus, the source code of the graphical interface implementation is shared with the students, which allowed them to interact directly with the graphical interface and manipulate it. Finally, four simulation results are presented, as well as different sampling frequencies and percentage of overlap.

**Keywords:** Spectrogram, Discrete Fourier Transform, App Designer environment, Matlab software, Signal Processing Toolbox.

© Los autores. Este artículo Open Access esta publicado bajo la Licencia Creative Commons Atribución 4.0 Internacional. (CC-BY 4.0)



## 1. Introducción

Con el auge de la tecnología, muchos de los programas universitarios, a nivel de pregrado y posgrado, han incluido en sus mallas curriculares asignaturas relacionadas al tratamiento o procesamiento temporal y frecuencial de señales digitales a nivel de software y/o hardware, principalmente señales de voz, audio, imágenes y video. Por eso, la Universidad Ricardo Palma, en sus actuales Planes Curriculares de los Programas de Ingeniería Electrónica y Mecatrónica, en pregrado, continúan incluyendo asignaturas de este tipo con horas de dictado para la teoría y otras para el laboratorio [1, 2].

En ese sentido, el programa de Ingeniería Electrónica de la Facultad de Ingeniería de la Universidad Ricardo Palma (URP), ubicado en Lima, Perú, imparte la asignatura obligatoria de Procesamiento Digital de Señales (cuatro horas semanales – 2 Teoría – 2 Laboratorio) en el octavo semestre académico. Por lo tanto, el contenido de este artículo se orienta principalmente a la segunda unidad temática de esta asignatura [3]. Esta unidad temática lleva por nombre Transformada Discreta de Fourier y su contenido trata de la teoría y aplicación de la Transformada Discreta de Fourier (DFT) y la Transformada de Fourier corta en el tiempo (STFT) sobre señales periódicas y reales.

Asimismo, la URP cuenta con la licencia del software Matlab y es utilizada para reforzar la enseñanza-aprendizaje de los estudiantes de pregrado para diferentes asignaturas de los programas universitarios de Ingeniería Electrónica y Mecatrónica. Con la intención de facilitar dicho aprendizaje, en uno de los temas importantes del área de procesamiento digital de señales, se ha realizado la implementación de una interfaz gráfica que utiliza el entorno de desarrollo App Designer del Matlab. De esta manera, se pone a disposición del estudiante el uso de esta interfaz gráfica para visualizar mejor aún el comportamiento de una señal temporal desde el punto de vista frecuencial, y con la opción de añadir y/o modificar el código de programación según el requerimiento necesario.

El uso de la interfaz gráfica, en diferentes áreas, se viene realizando desde algunos años atrás, porque ofrece flexibilidad y dinamismo sobre todo para utilizarlas en las clases no presenciales. Por ejemplo, algunas de esas experiencias se encuentran en [4], donde se propuso el desarrollo de una interfaz gráfica para el área de diseño de un generador síncrono saliente. Asimismo, en [5], se utilizó el App Designer para monitorear las principales

variables de un vehículo eléctrico que fueron adquiridas a través del NI USB 6001. Adicionalmente, en [6] el entorno App Designer fue empleado como una novedosa herramienta de simulación con alta fidelidad para optimizar el diseño de convertidores resonantes cc-cc. Por otro lado, se encuentran las investigaciones que hicieron uso del análisis tiempo-frecuencia para determinadas situaciones. Ese es el caso de [7], en el que se realizó una discriminación de las señales musicales con voz y sin voz utilizando características de co-ocurrencia basadas en imágenes de espectrograma.

## 2. Materiales y Métodos

La Transformada Discreta de Fourier (DFT) es una de las principales herramientas utilizadas en el análisis de las señales discretas en el dominio de la frecuencia; sin embargo, se requiere previamente conocer los conceptos de la Transformada Discreta de Fourier. Por ello, antes de mencionar la metodología empleada, se hace una breve revisión de la DFT y su aplicación sobre señales temporales a través de una representación gráfica: tiempo versus frecuencia. Luego, se continúa con la metodología desarrollada e inicia el diseño de la interfaz gráfica de usuario utilizando el App Designer del Matlab. Luego seguirá el procedimiento de concatenación de las señales periódicas y típicas en la asignatura Procesamiento Digital de Señales, y se finalizará con el desarrollo de la programación con apoyo del Toolbox Signal Processing del Matlab y la interfaz gráfica App Designer.

### 2.1. Transformada de Fourier y Transformada Discreta de Fourier Corta en el Tiempo

De acuerdo con [8], las señales y sistemas en el tiempo discreto,  $x[n]$ , pueden ser caracterizadas en el dominio de la frecuencia a través de su Transformada de Fourier, la cual es periódica con periodo  $2\pi$ ; sin embargo, en dicha caracterización se observa una dependencia a la variable continua “ $\omega$ ”, conocida como frecuencia; por ello, no se hace posible el procesamiento de señales en el tiempo discreto en computadoras digitales. A continuación, en (1), se muestra la expresión matemática de la representación de la Transformada de Fourier.

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n} \quad (1)$$

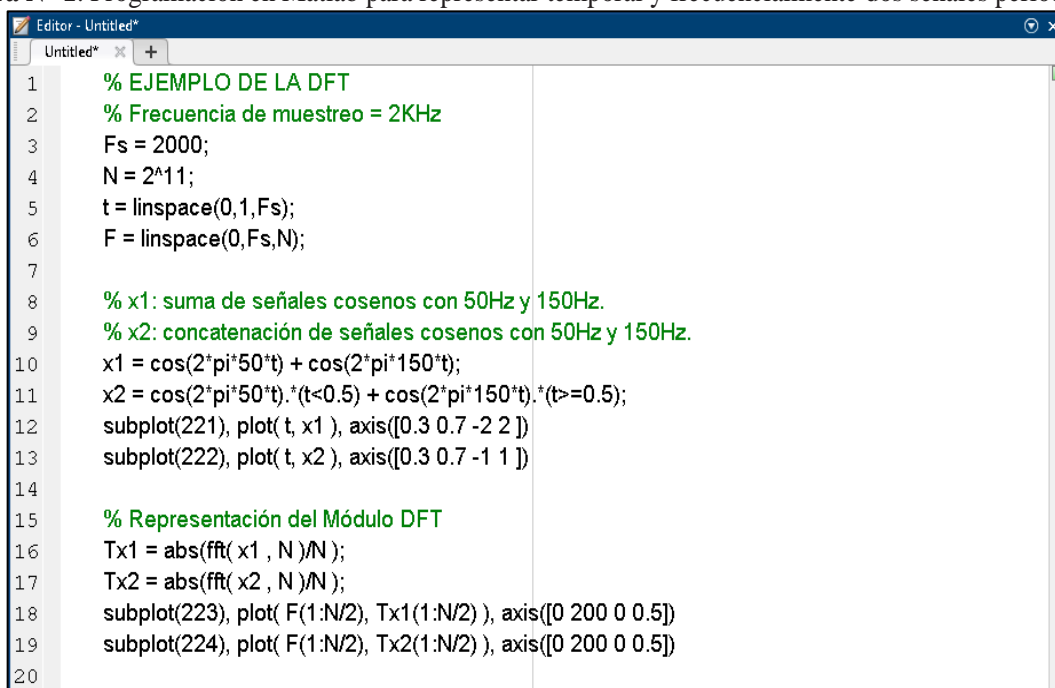
En vista de lo comentado anteriormente, surge la Transformada Discreta de Fourier

(DFT) que es prácticamente el muestreo uniforme de la variable continua de frecuencia; es decir, el mapeo de una señal que depende de una variable discreta de tiempo “n” en una transformada que depende de una variable discreta de frecuencia “k”. Líneas abajo se muestra la representación matemática de la DFT donde la variable “N” representa al número de muestras uniformemente espaciadas entre 0 y  $2\pi$  [8].

$$X\left(e^{j\frac{2\pi}{N}k}\right) = X(k) = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn} \quad \text{para: } 0 \leq k \leq N-1 \quad (2)$$

Sin embargo, el uso de la DFT no otorga información referente a la presencia de los componentes de frecuencia por intervalos de tiempo; es decir, al aplicar la DFT a una señal variable en frecuencia solamente será posible conocer todos los componentes de frecuencia que originaron dicha señal, pero se ignorarán los instantes en los cuales fueron apareciendo tales componentes. Como ejemplo, en la Figura N° 1, se comparte un código de programación en Matlab que genera dos señales periódicas siendo una de ellas la suma de dos cosenos con frecuencias de 50 Hz y 150 Hz, mientras que la otra señal corresponde a la concatenación de las mismas ondas cosenos; es decir, una a continuación de la otra; además, dicho código contiene el procedimiento para la obtención del módulo de la DFT de tales señales periódicas a una tasa de muestreo de 2000 muestras/segundo.

Figura N° 1. Programación en Matlab para representar temporal y frecuencialmente dos señales periódicas



```

Editor - Untitled*
Untitled* x +
1 % EJEMPLO DE LA DFT
2 % Frecuencia de muestreo = 2KHz
3 Fs = 2000;
4 N = 2^11;
5 t = linspace(0,1,Fs);
6 F = linspace(0,Fs,N);
7
8 % x1: suma de señales cosenos con 50Hz y 150Hz.
9 % x2: concatenación de señales cosenos con 50Hz y 150Hz.
10 x1 = cos(2*pi*50*t) + cos(2*pi*150*t);
11 x2 = cos(2*pi*50*t).*(t<0.5) + cos(2*pi*150*t).*(t>=0.5);
12 subplot(221), plot( t, x1 ), axis([0.3 0.7 -2 2 ])
13 subplot(222), plot( t, x2 ), axis([0.3 0.7 -1 1 ])
14
15 % Representación del Módulo DFT
16 Tx1 = abs(fft( x1 , N )/N );
17 Tx2 = abs(fft( x2 , N )/N );
18 subplot(223), plot( F(1:N/2), Tx1(1:N/2) ), axis([0 200 0 0.5])
19 subplot(224), plot( F(1:N/2), Tx2(1:N/2) ), axis([0 200 0 0.5])
20

```

Fuente: Captura de pantalla de una simulación en Matlab.

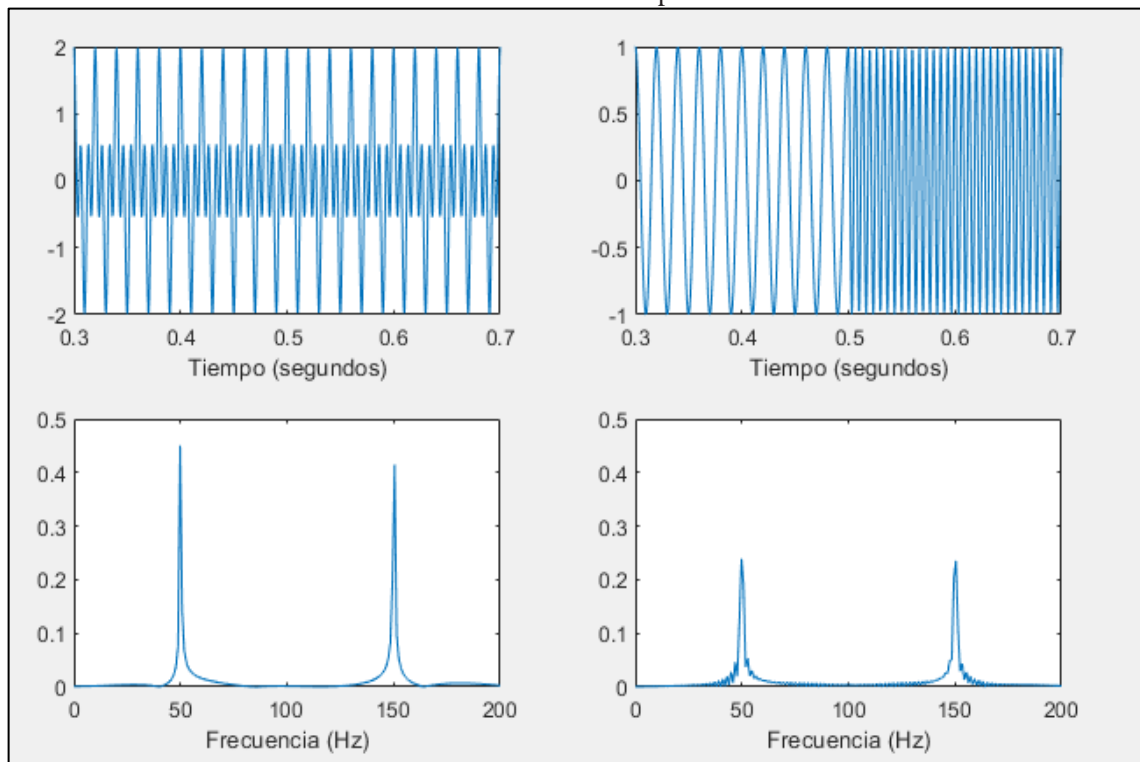
A continuación, en la Figura N° 2, se muestra la representación gráfica en el dominio del tiempo y la frecuencia de tales señales por separado y como resultado de emplear el código de programación compartido en la Figura N° 1. Asimismo, en la Figura N° 2, es posible apreciar que la primera señal (ubicada en la parte superior-izquierda de la figura) está compuesta por la suma de dos ondas cosenos con frecuencias de 50 Hz y 150 Hz, mientras que la segunda señal (ubicada en la parte superior-derecha de la figura) se implementó por la concatenación de las mismas ondas cosenos; no obstante, los módulos de sus respectivas Transformadas Discretas de Fourier (ubicadas en la parte inferior izquierda y derecha) muestran la misma cantidad y distribución de frecuencias, pero con amplitudes diferentes, por lo cual se desconoce si las frecuencias de los cosenos cambian o no en cada instante de tiempo.

Sin embargo, en aplicaciones prácticas de modelos de señales sinusoidales las propiedades o características de amplitud, frecuencia y fase, cambian con el transcurrir del tiempo; tal es el caso de señales de radar, sonar, voz, comunicación de datos, entre otras más. Esto indica que el uso de una simple DFT no es suficiente para describir o analizar tales señales, y como resultado de ello se requiere de los conceptos de la Transformada de Fourier Corta en el Tiempo (STFT) o también conocida como Transformada de Fourier Dependiente del Tiempo [9].

De esta forma, la STFT se representa a través de la expresión matemática en (3) en la que  $w[m]$  es una secuencia discreta finita denominada ventana, que recorre toda la señal desde el principio hasta el final con una determinada cantidad de muestras sobrepuestas. Por otro lado, se tiene a la señal  $x[n]$ , que es una función de una simple variable discreta convertida en una función de dos dimensiones de la misma variable temporal “n”; luego la variable frecuencia “ $\lambda$ ”, la cual es continua, y la función exponencial que es propia de la Transformada de Fourier.

$$X[n, \lambda] = \sum_{m=-\infty}^{\infty} x[n + m] w[m] e^{-j \lambda m} \quad (3)$$

**Figura N° 2.** Arriba: Representación de señales temporales. Abajo: Representación de los módulos de la DFT de las señales temporales



Fuente: Captura de pantalla de una simulación en Matlab

Además, la STFT es periódica en  $\lambda$  con periodo  $2\pi$ , y por lo tanto se requiere solamente considerar los valores de  $\lambda$  para  $0 \leq \lambda \leq 2\pi$ , u otro intervalo de longitud  $2\pi$  [9]. Entonces, con el fin de visualizar la aplicación de la STFT sobre las dos señales descritas anteriormente, a continuación, en la Figura N° 3, se comparte el código de programación utilizado en Matlab para graficar tal representación temporal versus frecuencia. En esta, “N” representa al número de muestras de la DFT aplicado con una VENTANA del tipo Hamming, y con una sobreposición aproximada del 30% tal como se ha asignado en la variable NOVERLAP; asimismo, en la Figura N° 4, se muestra la representación gráfica de la STFT para el caso de la suma y concatenación de las dos señales sinusoidales propuestas anteriormente, en la que el gráfico de la izquierda corresponde a la suma de las dos señales y el gráfico de la derecha a la concatenación de estas.

## *2.2. Diseño de la Interfaz Gráfica de Usuario en App Designer*

El App Designer es un entorno de desarrollo interactivo que permite diseñar una aplicación y programar su comportamiento; además, proporciona una versión totalmente integrada del editor de MATLAB® y un gran conjunto de componentes interactivos de la Interfaz de Usuario [10], tales como botones, casillas de verificación, elementos de medición, indicadores luminosos, entre otros más [11, 12].

De esta manera, para el diseño de la interfaz gráfica, se optó por utilizar un panel de configuración conformado por tres perillas que a su vez presentan cuatro opciones para elegir el tamaño de la ventana, el porcentaje de sobre posición y la frecuencia de muestreo, respectivamente. El primer parámetro determina el tamaño de la resolución que se desea obtener, sea a nivel de tiempo o de frecuencia. Por lo general, al elegir el mayor tamaño se tendrá una mayor resolución de frecuencia, pero menor en cuanto a tiempo. De la misma manera, si se elige un parámetro de porcentaje de sobre posición pequeño, el resultado gráfico se observa con menor resolución cuando se realiza la transición de frecuencia. Asimismo, se cuenta con un arreglo de doce componentes DropDown. Cada uno tiene asignado un tipo de señal sinusoidal, cuadrada o triangular con una frecuencia específica por columna; de esta manera, es posible construir una señal con múltiples frecuencias por intervalos diferentes de tiempo, porque existe la posibilidad de elegir hasta tres tipos de señales por cada columna del arreglo logrando así la concatenación de señales arbitrariamente. Se debe tener en cuenta que la elección de las señales por cada columna será sumada, y posteriormente concatenadas con las otras señales de las demás columnas. En la parte inferior del arreglo de componentes DropDown, se cuenta con un componente del tipo Axes en el que se representa el espectrograma de la señal anteriormente concatenada.

Figura N° 3. Código de programación para representar la STFT de dos señales discretas

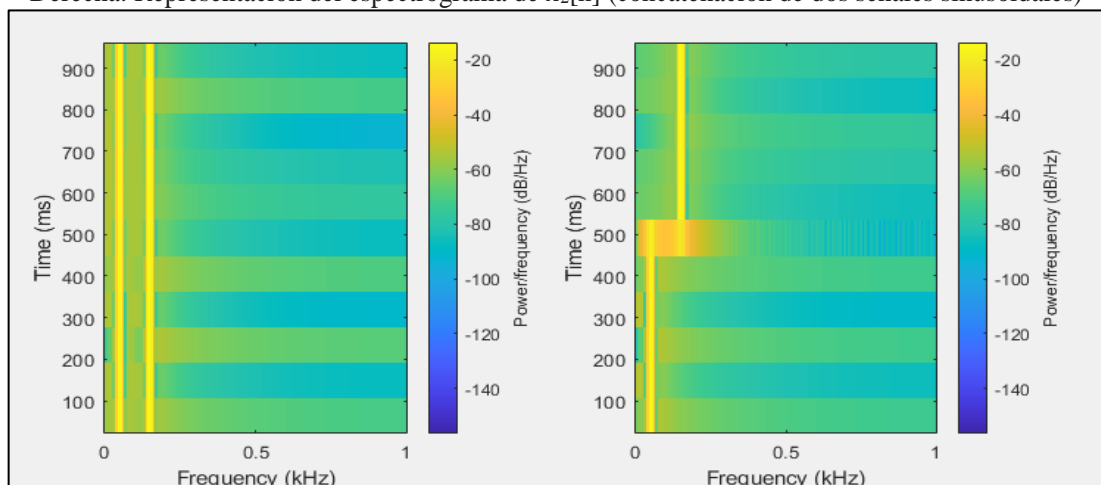
```

Editor - G:\URP\Docs 2022\Articulos2022\SCIENTIA 2022\Codigo_articulo_Scientia_2022.m*
Codigo_articulo_Scientia_2022.m* x +
22 % Frecuencia de muestreo = 2KHz
23 Fs = 2000;
24 N = 2^8;
25 t = linspace(0,1,Fs);
26 % x1: suma de señales cosenos con 50Hz y 150Hz.
27 % x2: concatenación de señales cosenos con 50Hz y 150Hz.
28 x1 = cos(2*pi*50*t) + cos(2*pi*150*t);
29 x2 = cos(2*pi*50*t).*(t<0.5) + cos(2*pi*150*t).*(t>=0.5);
30 %Parámetros para la STFT
31 VENTANA = hamming(N);
32 NOVERLAP = round( N / 3 );
33 subplot(121), spectrogram( x1 , VENTANA , NOVERLAP , N , Fs)
34 subplot(122), spectrogram( x2 , VENTANA , NOVERLAP , N , Fs)

```

Fuente: Captura de pantalla de una simulación en Matlab

Figura N° 4. Izquierda: Representación del espectrograma de  $x_1[n]$  (suma de dos señales sinusoidales). Derecha: Representación del espectrograma de  $x_2[n]$  (concatenación de dos señales sinusoidales)

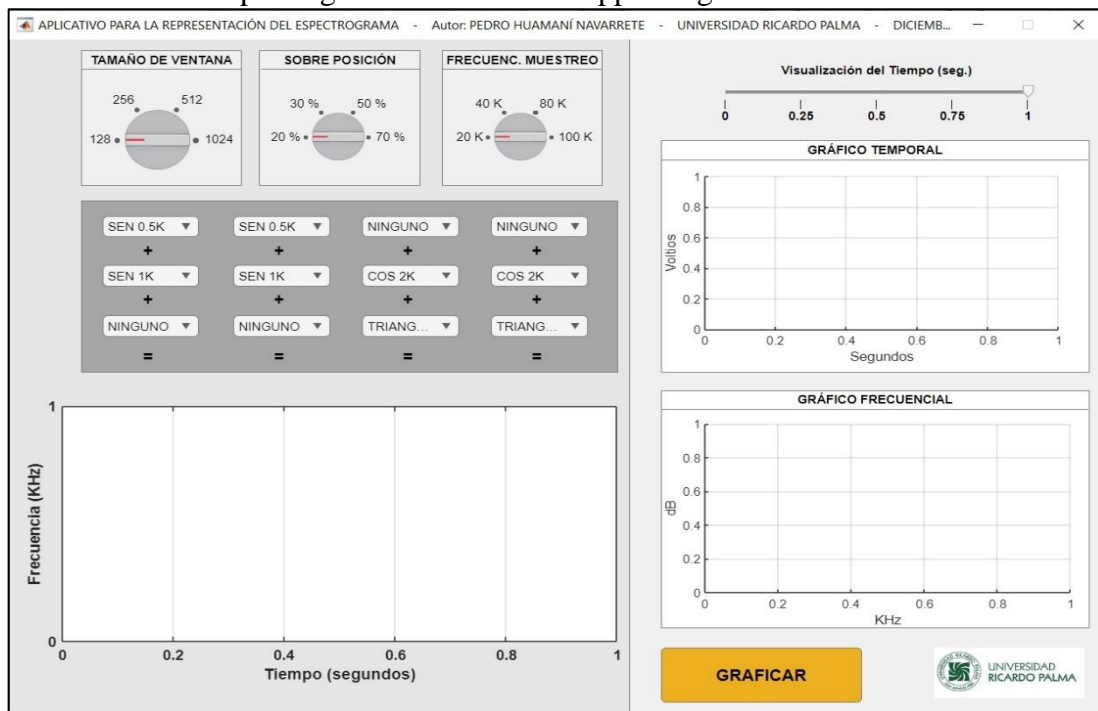


Fuente: Captura de pantalla de una simulación en Matlab

Igualmente, en el lado derecho de la interfaz gráfica se incluyeron dos componentes del tipo Axes. Estos permiten visualizar la representación temporal y frecuencial de la señal concatenada, que, a su vez, es restringida por el componente Slider que condiciona la visualización de 0 hasta 1 segundo de duración. La representación gráfica de frecuencia es obtenida con el módulo de la DFT de toda la señal generada por el arreglo de componentes DropDown. Seguidamente, en la Figura N° 5, se muestra una captura de pantalla del diseño de la interfaz gráfica que indica las opciones por defecto que presenta cada componente utilizado.



Figura N° 5. Captura de pantalla del diseño de la Interfaz Gráfica para la Representación del Espectrograma utilizando el App Designer del Matlab.



Fuente: Captura de pantalla de una simulación en Matlab

Adicionalmente, tal como se aprecia en la figura anterior, la interfaz gráfica incluye un componente del tipo Button etiquetado como GRAFICAR, el cual está encargado de ejecutar la concatenación de las señales seleccionadas del arreglo de componentes en una sola variable del tipo vector, para posteriormente representarla en forma gráfica a través de un espectrograma, así como también en un gráfico temporal y frecuencial.

### 2.3. Concatenación de Señales Periódicas

Conociendo que existe una cantidad infinita de señales a considerar, se determinó emplear una combinación de cuatro señales sinusoidales con frecuencias distintas (0.5 KHz, 1 KHz, 2 KHz y 4KHz), una señal cuadrada de 2 KHz, dos señales triangulares de 1 KHz y 2KHz, un ruido blanco y una componente DC. De esta manera, con el arreglo de los 12 componentes DropDown conformados por tres filas y cuatro columnas, fue posible concatenar diferentes tipos de señales hasta por un intervalo de 1 segundo; y por cada columna del arreglo fue posible sumar hasta tres tipos de señales distintas con una correspondencia de 0.25 segundos. Es decir, si en la primera columna del arreglo de

componentes se eligieron las opciones SENO\_0.5K, SEN\_1K y RUIDO, el resultado de ello fue la suma de dos señales sinusoidales con frecuencias diferentes más un ruido blanco; de la misma manera, si en la segunda columna del arreglo de componentes se eligieron las opciones COS\_1K, COS\_2K y NINGUNO, el resultado fue la suma de solamente dos señales sinusoidales con frecuencias diferentes.

Posteriormente, se estableció una función con código de programación y denominada como “Obtiene Señal” con trece variables de entrada y una de salida. De las trece, una de ellas corresponde a la frecuencia de muestreo elegida desde la tercera perilla, mientras que las otras doce variables representan a la opción seleccionada en cada uno de los componentes DropDown. A continuación, en la Figura N° 6, se muestra el código de programación en Matlab utilizado para concatenar las señales de la primera y segunda parte de la tercera columna del arreglo de componentes. Similarmente, se desarrolló el código fuente para las otras dos columnas. Adicionalmente, tal como se puede apreciar en dicha figura, se añadió la opción de NINGUNO en cada componente del arreglo para evitar la participación de alguna señal en determinado tiempo de la concatenación.

Por otro lado, la única variable de salida de la función “Obtiene Señal” contuvo dos vectores que correspondieron al tiempo y a la señal resultante de la concatenación.

**Figura N° 6.** Captura de pantalla del código de programación de la función “Obtiene Señal” en el App Designer del Matlab para concatenar las señales temporales y típicas en Procesamiento de Señales

```

110 function S = ObtieneSeñal(app,FMUESTREO,N01,N02,N03,N04,N05,N06,N07,N08,N09,N10,N11,N12);
111 %Input: selección de tipo señal
112 %Output: señal compuesta y vector de tiempo
113
114 t = linspace(0,0.25,0.25*FMUESTREO);
115 x1 = 0; x2 = 0; x3 = 0 , x4 = 0;
116
117 %===== 1RA COLUMNA =====
118 if strcmp(N01,'SEN 0.5K')
119     x1 = sin(2*pi*500*t);
120 elseif strcmp(N01,'COS 1K')
121     x1 = cos(2*pi*1000*t);
122 elseif strcmp(N01,'SEN 4K')
123     x1 = sin(2*pi*4000*t);
124 elseif strcmp(N01,'DC')
125     x1 = 0.2*ones(size(t));
126 elseif strcmp(N01,'NINGUNO')
127     x1 = 0.2*zeros(size(t));
128 end
129 %-----
130 if strcmp(N02,'SEN 1K')
131     x1 = sin(2*pi*1000*t) + x1;
132 elseif strcmp(N02,'COS 2K')
133     x1 = cos(2*pi*2000*t) + x1;
134 elseif strcmp(N02,'SEN 4K')
135     x1 = sin(2*pi*4000*t) + x1;
136 elseif strcmp(N02,'DC')
137     x1 = 0.2*ones(size(t)) + x1;
138 elseif strcmp(N02,'NINGUNO')
139     x1 = 0.2*zeros(size(t)) + x1;
140 end
141 %-----
142 if strcmp(N03,'TRIANG 1K')
143     x1 = sawtooth(2*pi*1000*(0.5) + x1;
144 elseif strcmp(N03,'TRIANG 2K')
145     x1 = sawtooth(2*pi*2000*(0.5) + x1;
146 elseif strcmp(N03,'CUAD 2K')
147     x1 = square(2*pi*2000*(0.5) + x1;
148 elseif strcmp(N03,'RUIDO')
149     x1 = 0.1*randn(size(t)) + x1;
150 elseif strcmp(N03,'NINGUNO')
151     x1 = 0.2*zeros(size(t)) + x1;
152 end
153 %===== 2DA COLUMNA =====
154 if strcmp(N04,'SEN 0.5K')
155     x2 = sin(2*pi*500*t);
156 elseif strcmp(N04,'COS 1K')
157     x2 = cos(2*pi*1000*t);
158 elseif strcmp(N04,'SEN 4K')
159     x2 = sin(2*pi*4000*t);
160 elseif strcmp(N04,'DC')
161     x2 = 0.2*ones(size(t));
162 elseif strcmp(N04,'NINGUNO')
163     x2 = 0.2*zeros(size(t));
164 end
165 %-----
166 if strcmp(N05,'SEN 1K')
167     x2 = sin(2*pi*1000*t) + x2;
168 elseif strcmp(N05,'COS 2K')
169     x2 = cos(2*pi*2000*t) + x2;
170 elseif strcmp(N05,'SEN 4K')
171     x2 = sin(2*pi*4000*t) + x2;
172 elseif strcmp(N05,'DC')
173     x2 = 0.2*ones(size(t)) + x2;
174 elseif strcmp(N05,'NINGUNO')
175     x2 = 0.2*zeros(size(t)) + x2;
176 end
177 %-----
178 if strcmp(N06,'TRIANG 1K')
179     x2 = sawtooth(2*pi*1000*(0.5) + x2;
180 elseif strcmp(N06,'TRIANG 2K')
181     x2 = sawtooth(2*pi*2000*(0.5) + x2;
182 elseif strcmp(N06,'CUAD 2K')
183     x2 = square(2*pi*2000*(0.5) + x2;
184 elseif strcmp(N06,'RUIDO')
185     x2 = 0.1*randn(size(t)) + x2;
186 elseif strcmp(N06,'NINGUNO')
187     x2 = 0.2*zeros(size(t)) + x2;
188 end
189 %===== 3RA COLUMNA =====
190 if strcmp(N07,'SEN 0.5K')
191     x3 = sin(2*pi*500*t);
192 elseif strcmp(N07,'COS 1K')
193     x3 = cos(2*pi*1000*t);
194 elseif strcmp(N07,'SEN 4K')
195     x3 = sin(2*pi*4000*t);

```

Fuente: Captura de pantalla de una simulación en Matlab

#### 2.4. Desarrollo de la Programación

El código de programación se realizó a través del editor del entorno App Designer del Matlab, el cual se visualiza cuando se elige la opción “Code View” de la ventana central de Diseño. Asimismo, es importante aclarar que este entorno genera de manera automática un código de programación relacionado a los componentes que se van añadiendo en la interfaz gráfica diseñada.

Por lo tanto, los valores asignados a través de las perillas y los componentes DropDown y Slide fueron asignados a variables específicas dentro del código de programación, para luego utilizarlos en el momento de la representación gráfica del espectrograma con apoyo del comando SPECTROGRAM del Toolbox Signal Processing. A continuación, la Figura N° 7 muestra parte del código de programación utilizado para asignar a los componentes las alternativas de frecuencias de muestreo, sobreposición y tamaño de ventana.

Figura N° 7. Captura de pantalla del código de programación

```
% Callback function: GRAFICARButton, UIFigure
function GRAFICARButtonPushed(app, event)

    A = app.FMUESTREOKnob.Value;
    if strcmp(A,'20 K'), Fs = 20000;
    elseif strcmp(A,'40 K'), Fs = 40000;
    elseif strcmp(A,'80 K'), Fs = 80000;
    elseif strcmp(A,'100 K'), Fs = 100000;
    end

    A = app.SOBREPOSIKKnob.Value;
    if strcmp(A,'20 %'), SOBREPONE = 0.2;
    elseif strcmp(A,'30 %'), SOBREPONE = 0.3;
    elseif strcmp(A,'50 %'), SOBREPONE = 0.5;
    elseif strcmp(A,'70 %'), SOBREPONE = 0.7;
    end

    VENTANA = str2num( app.TAMAVENTANAKnob.Value );

    N01 = app.DropDown_1.Value; N02 = app.DropDown_2.Value; N03 = app.DropDown_3.Value;
    N04 = app.DropDown_6.Value; N05 = app.DropDown_5.Value; N06 = app.DropDown_4.Value;
    N07 = app.DropDown_9.Value; N08 = app.DropDown_8.Value; N09 = app.DropDown_7.Value;
    N10 = app.DropDown_12.Value; N11 = app.DropDown_11.Value; N12 = app.DropDown_10.Value;

    S = ObtieneSenal(app,Fs,N01,N02,N03,N04,N05,N06,N07,N08,N09,N10,N11,N12);
    t = S(1,:); x = S(2,:);
```

Fuente: Captura de pantalla de una simulación en Matlab

### 3. Resultados

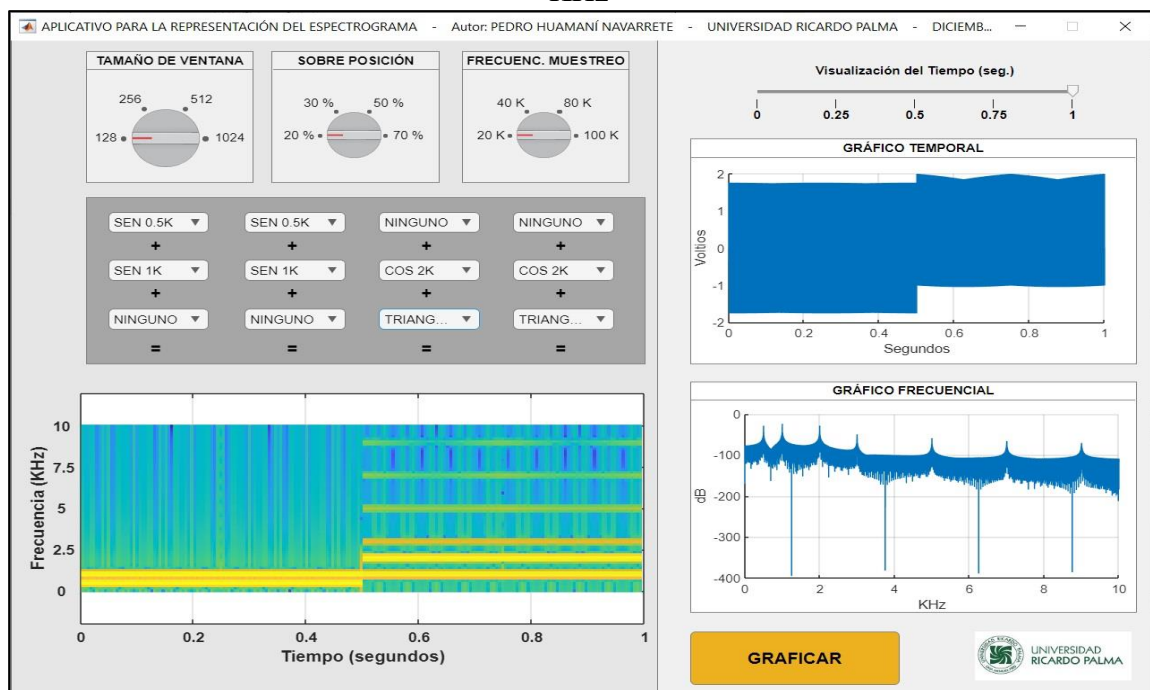
Respecto a los resultados alcanzados, se comparten cuatro casos que representan situaciones diferentes en cuanto a la suma y concatenación de señales sinusoidales, triangulares, cuadradas, con nivel de continua, y ruido blanco de media cero.

a) Primer resultado. Se utilizó un tamaño de ventana igual a 128, una sobreposición al 20% y una Frecuencia de Muestreo de 20 KHz. La concatenación fue realizada de la siguiente manera:

- En la primera y segunda columna de componentes DropDown se seleccionó la suma de dos señales sinusoidales de 0.5 KHz y 1 KHz.
- En la tercera y cuarta columna de componentes DropDown se seleccionó la suma de un coseno de 2 KHz y una triangular de 1 KHz.

Por eso, en el gráfico del espectrograma, se aprecian dos trazos paralelos en el intervalo de 0 a 0.5 segundos los cuales corresponden a las frecuencias de 1 KHz y 0.5 KHz; asimismo, para el otro intervalo de 0.5 a 1 segundo se aprecian varios trazos paralelos cada vez con menor intensidad los cuales corresponden a las armónicas de la onda triangular. De la misma manera, en los gráficos temporal y frecuencial del lado derecho de la interfaz gráfica, se observa la señal concatenada en el tiempo de 0 a 1 segundo y su respectivo módulo de la DFT de manera global donde se observan todas las componentes de frecuencia que aparecen en el espectrograma (ver la Figura N° 8).

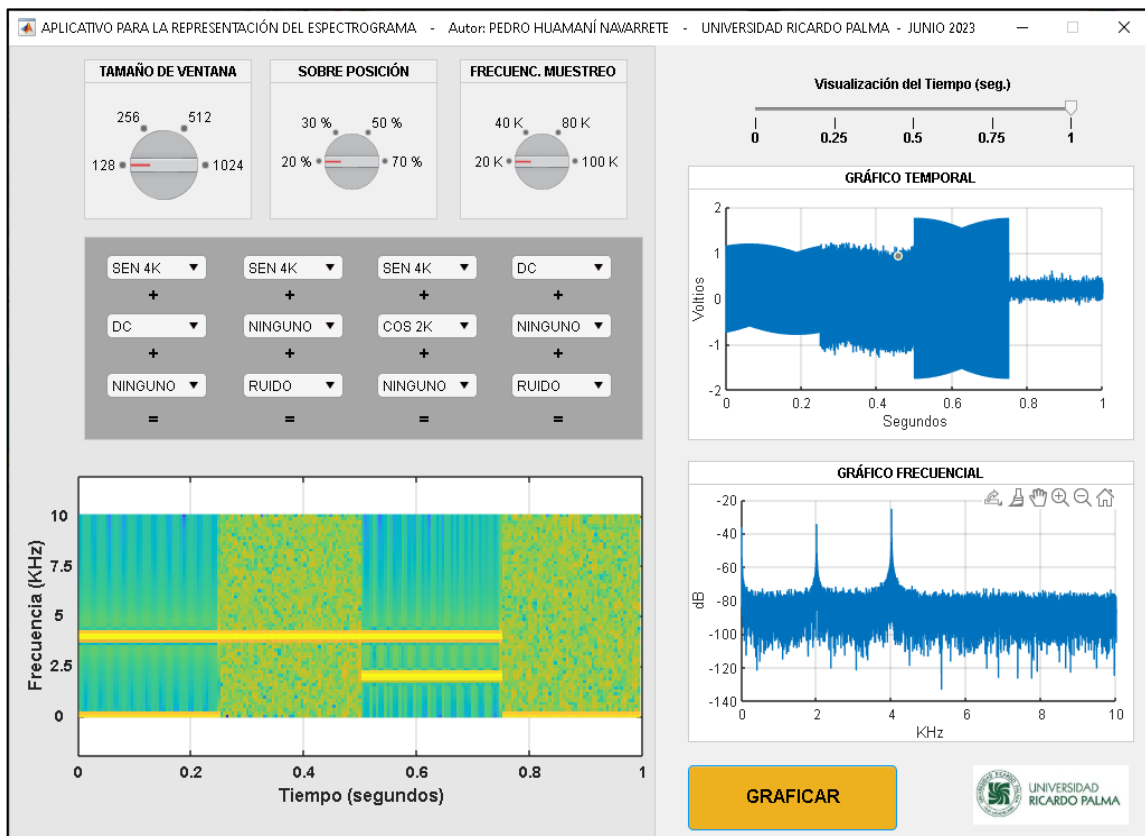
Figura N° 8. Interfaz gráfica para representar el espectrograma de múltiples señales sinusoidales y triangulares con un nivel de continua, a una Frecuencia de Muestreo de 20 KHz



Fuente: Captura de pantalla de una simulación en Matlab

En consecuencia, en el gráfico del espectrograma se aprecian dos trazos paralelos correspondientes a la frecuencia de 4 KHz y 0 Hz para el intervalo de 0 a 0.25 segundos; asimismo, de 0.25 a 0.5 segundos se aprecia el ruido en todo el espectro y la continuidad del trazo para la frecuencia de 4 KHz. Luego, de 0.5 a 0.75 segundos se observa nuevamente el trazo correspondiente a la frecuencia de 4 KHz y una nueva para la frecuencia de 2 KHz. Para el intervalo de 0.75 a 1 segundo se aprecia el ruido blanco en todo el espectro de frecuencia más un trazo que representa la frecuencia en 0 Hz. De la misma manera, en los gráficos temporal y frecuencial del lado derecho de la interfaz gráfica, se observa la señal concatenada en el tiempo de 0 a 1 segundo y su respectivo módulo de la DFT de manera global donde se observan todas las componentes de frecuencia que aparecen en el espectrograma, lo que incluye el ruido blanco (ver la Figura N° 9).

Figura N° 9. Interfaz gráfica para representar el espectrograma de múltiples señales sinusoidales con un nivel de continua y presencia de ruido a una Frecuencia de Muestreo de 20 KHz



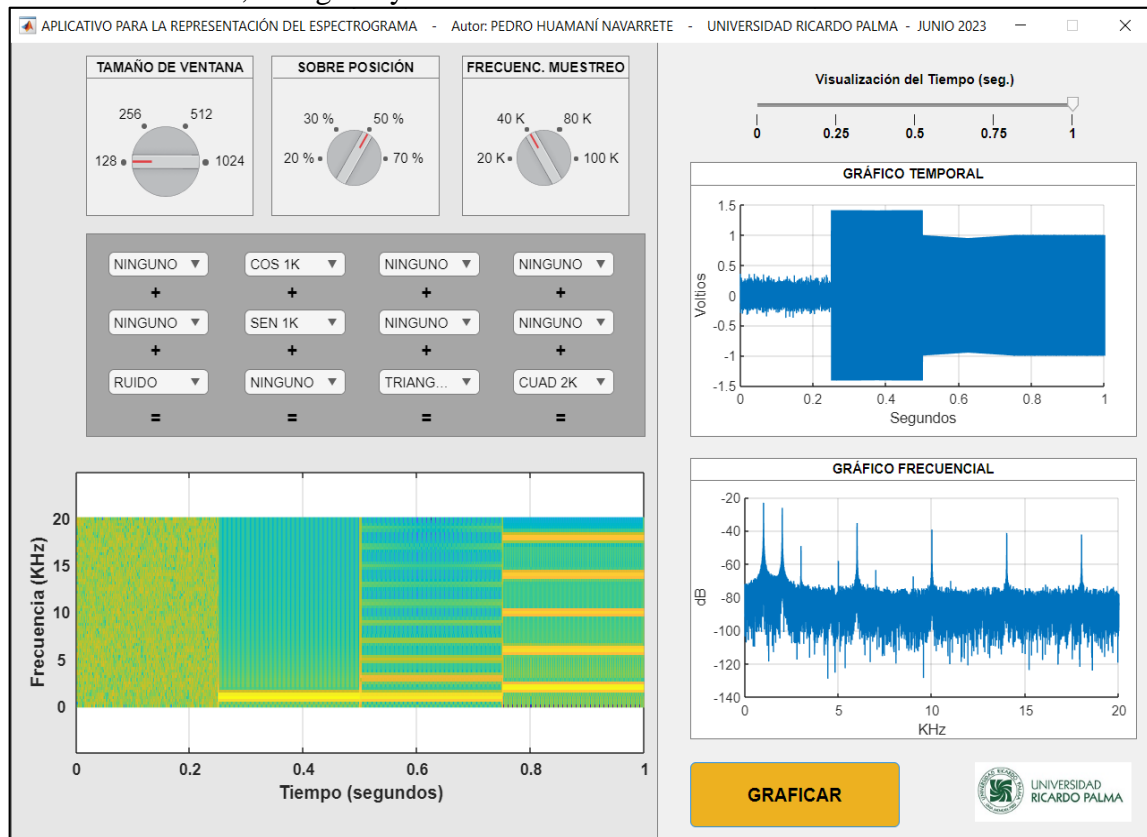
Fuente: Captura de pantalla de una simulación en Matlab

b) Tercer resultado. Se utilizó un tamaño de ventana igual a 128, una sobreposición al 50% y una Frecuencia de Muestreo de 40 KHz. La concatenación fue realizada de la siguiente manera:

- En la primera columna de componentes DropDown se seleccionó la señal de ruido blanco de media cero.
- En la segunda columna de componentes DropDown se seleccionó la suma de dos señales coseno y seno, ambas de 1 KHz.
- En la tercera columna de componentes DropDown se seleccionó una señal triangular con frecuencia fundamental de 1 KHz.
- En la tercera columna de componentes DropDown se seleccionó una señal cuadrada con frecuencia fundamental de 2 KHz.

Por eso, en el gráfico del espectrograma, se aprecia el ruido blanco disperso en todo el espectro de frecuencia desde 0 hasta 20 KHz para el intervalo de 0 a 0.25 segundos; asimismo, de 0.25 a 0.5 segundos se aprecia un trazo correspondiente a la suma de una señal seno y coseno con la misma frecuencia de 1 KHz por lo que se observa una sola línea. Luego, de 0.5 a 0.75 segundos se observan las armónicas de la señal triangular de 1 KHz con trazos paralelos que incrementan cada 2 KHz hasta los 19 KHz. De igual forma, de 0.75 a 1 segundo se observan las armónicas de la señal cuadrada de 2 KHz con trazos paralelos que incrementan cada 2 KHz hasta los 18 KHz. Asimismo, en los gráficos temporal y frecuencial del lado derecho de la interfaz gráfica, se observa la señal concatenada en el tiempo de 0 a 1 segundo y su respectivo módulo de la DFT de manera global donde se observan todas las componentes de frecuencia que aparecen en el espectrograma, incluyendo el ruido blanco (ver la Figura N° 10).

Figura N° 10. Interfaz gráfica para representar el espectrograma de una señal sinusoidal, cuadrada, triangular y con ruido a una Frecuencia de Muestreo de 40 KHz



Fuente: Captura de pantalla de una simulación en Matlab

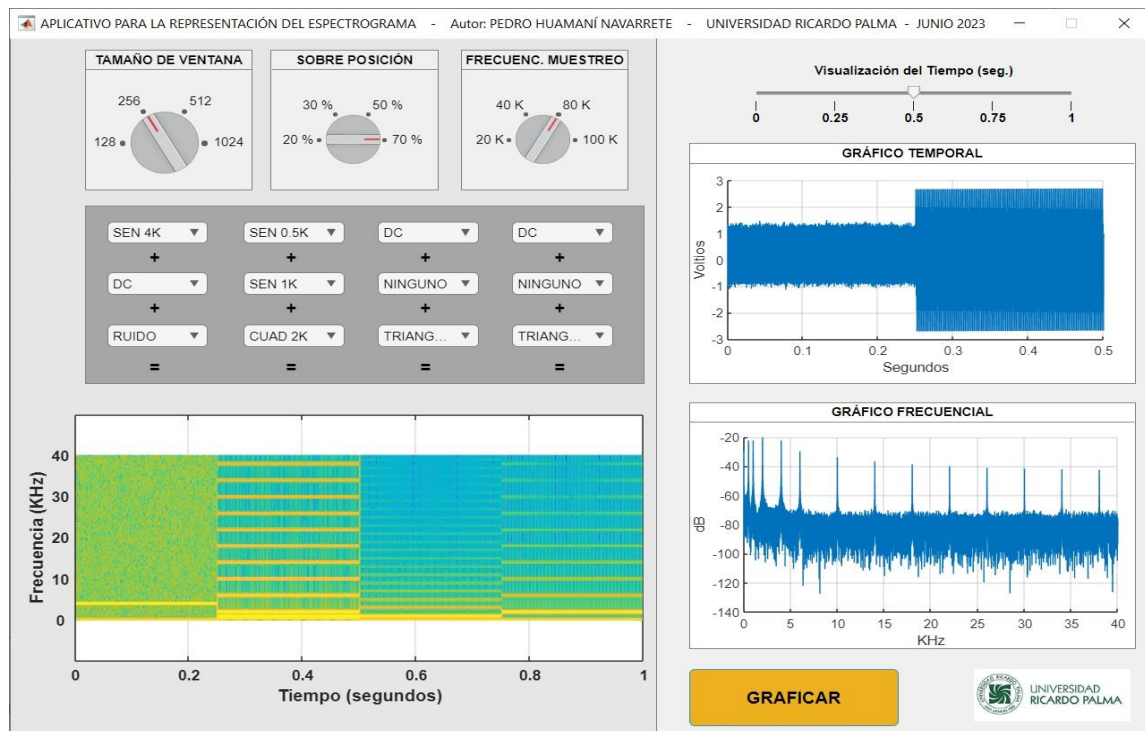
c) Cuarto resultado. Se utilizó un tamaño de ventana igual a 256, una sobreposición al 70% y una Frecuencia de Muestreo de 80 KHz. La concatenación fue realizada de la siguiente manera:

- En la primera columna de componentes DropDown se seleccionó la suma de una señal de ruido blanco una componente DC y una señal sinusoidal de 4 KHz.
- En la segunda columna de componentes DropDown se seleccionó la suma de tres señales: una onda seno de 0.5 KHz, una onda seno de 1 KHz y una onda cuadrada de 2 KHz.
- En la tercera columna de componentes DropDown se seleccionó la suma de una señal triangular con frecuencia fundamental de 1 KHz y una componente DC.
- En la cuarta columna de componentes DropDown se seleccionó la suma de una señal triangular con frecuencia fundamental de 2 KHz y una componente DC.



Por ello, en el gráfico del espectrograma, se aprecia el ruido blanco disperso en todo el espectro de frecuencia desde 0 hasta 40 KHz más una señal sinusoidal de 4 KHz y un nivel de continua, para el intervalo de 0 a 0.25 segundos; asimismo, de 0.25 a 0.5 segundos se aprecian varios trazos que corresponden a la suma de dos señales sinusoidales de 0.5 KHz y 1 KHz más una señal cuadrada con frecuencias armónicas que inician en 2 KHz y finalizan en 38 KHz. Luego, de 0.5 a 0.75 segundos se observan las armónicas de la señal triangular de 1 KHz representados por trazos paralelos. Y, de igual forma, de 0.75 a 1 segundo, se observan las armónicas de otra señal triangular de 2 KHz representados también por trazos paralelos. De la misma manera, en los gráficos temporal y frecuencial del lado derecho de la interfaz gráfica, se observa la señal concatenada en el tiempo de 0 a 0.5 segundos y su respectivo módulo de la DFT de manera global donde se observan todas las componentes de frecuencia que aparecen en el espectrograma, que incluyen el ruido blanco (ver la Figura N° 11).

Figura N° 11. Interfaz gráfica para representar el espectrograma de señales sinusoidales, cuadradas y triangulares con ruido y nivel de continúa muestreadas a una Frecuencia de Muestreo de 80 KHz



Fuente: Captura de pantalla de una simulación en Matlab



#### 4. Conclusiones

- Se desarrolló una interfaz gráfica empleando el entorno de desarrollo App Designer del Matlab de fácil manipulación y comprensión para los estudiantes; asimismo, se les facilitó el código fuente de la programación para que ellos mismos interactúen directamente incrementando o cambiando el tipo y número de señales periódicas, y para actualizar los parámetros de la digitalización y del análisis frecuencial.
- Se estableció un arreglo de 12 componentes DropDown en el entorno de desarrollo App Designer del Matlab con la finalidad de distribuir la elección de distintos tipos de señales periódicas con frecuencias diferentes, tales como sinusoidales, triangulares y cuadradas, así como también con la opción de elegir una señal de ruido blanco de media cero o una componente de continua. Posteriormente, con un código de programación implementado en una función denominada “ObtieneSenal”, se realizó la concatenación de todas las señales elegidas a través de los componentes DropDown; esto permitió realizar la representación de toda la señal en una sola variable para posteriormente graficar en el dominio del tiempo y de la frecuencia.
- La interfaz gráfica en App Designer estuvo conformada por tres componentes Axes, que permitió realizar la representación gráfica temporal de la señal concatenada, así como también la representación frecuencial utilizando el módulo de la Transformada Discreta de Fourier de toda la señal concatenada, y una representación gráfica temporal-frecuencial a través de un espectrograma donde se permitió visualizar los cambios de las componentes de frecuencia con el transcurrir del tiempo.
- La representación del espectrograma, cuando se eligió una señal de ruido blanco con media cero, permitió corroborar el comportamiento de dicha señal en el dominio de la frecuencia, que mostró un espectro amplio desde 0 Hz hasta la mitad de la frecuencia de muestreo elegida desde el componente Knob.

## 5. Referencias bibliográficas

- [1] Universidad Ricardo Palma. (10 julio 2022). Plan Curricular – Información Académica – Ingeniería Electrónica – Escuelas – Facultad de Ingeniería – Pregrado [En línea]. Disponible: <https://www.urp.edu.pe/pregrado/facultad-de-ingenieria/escuelas/ingenieria-electronica/informacion-academica/>
- [2] Universidad Ricardo Palma. (10 julio 2022). Sumillas y Sílabos – Información Académica – Ingeniería Mecatrónica – Escuelas – Facultad de Ingeniería – Pregrado [En línea]. Disponible: <https://www.urp.edu.pe/pregrado/facultad-de-ingenieria/escuelas/ingenieria-mecatronica/informacion-academica/>
- [3] Universidad Ricardo Palma. (10 julio 2019). Sílabos – Información Académica – Ingeniería Electrónica – Escuelas – Facultad de Ingeniería – Pregrado [En línea]. Disponible: <https://www.urp.edu.pe/pdf/id/2507/n/plan-curricular-2015-ii-adecuado-a-la-ley-30220>
- [4] Petropol, G., Petropol, I. (October 2018). Designing a Graphical User Interface with Matlab to select the ideal covering coefficient of the polar pitch for a salient poly synchronous generator calculation. International Conference on Applied and Theoretical Electricity (ICATE). Conferencia llevada a cabo en Craiova, Rumania.
- [5] Gallego, J., Ciriero, J. and Romero, E. (May 2017). Electric vehicle monitoring system by using Matlab/App Designer. International Young Engineers Forum (YEF-ECE). Conferencia llevada a cabo en Costa da Caparica, Portugal.
- [6] Kumar A., Awasthi, A., Salari, O., Laha, A., Mathew, A. and Jain, P. (March 2020). A Time-Domain Based APP Designer For Resonant Converters With GUI Features. 2020 IEEE Applied Power Electronics Conference and Exposition (APEC). Conferencia llevada a cabo en New Orleans, LA, USA.
- [7] Ghosal, A., Chakraborty, R., Bibhas, D. y Kumar, S. (September 2012). Song/Instrumental Classification using Spectrogram Based Contextual Features. Proceedings of the CUBE International Information Technology Conference. Conferencia llevada a cabo en India.
- [8] Ramirez Diniz, P., Barros da Silva, E. e Lima Netto S. (2004). Processamento Digital de Sinais Projeto e Análise de Sistemas. Editorial Bookman.

- [9] Oppenheim, A. and Schafer, R. (1989). Discrete-time signal processing. Editorial Prentice Hall International.
- [10] Mathworks (2023, marzo). Centro de Ayuda. Desarrollar apps mediante App Designer [En línea]. Disponible: <https://la.mathworks.com/help/matlab/app-designer.html>
- [11] Matlab (2023, marzo). Cree apps web y de escritorio en Matlab [En línea]. Disponible: <https://la.mathworks.com/products/matlab/app-designer.html>
- Matlab (2023, marzo). Signal Processing Toolbox. Realice procesamiento y análisis de señales [En línea]. Disponible: <https://la.mathworks.com/products/signal.html>

**Pedro Huamani Navarrete**

Universidad Ricardo Palma, Lima, Perú.

Ingeniero Electrónico titulado por la Universidad Ricardo Palma (1999), con el grado de Maestría en Ingeniería Eléctrica en el área de Procesamiento de Señales y Control de Procesos por la Pontificia Universidad Católica de Río de Janeiro, Brasil (1997), y el grado de Doctor en Ingeniería de Sistemas por la Universidad Alas Peruanas (2013). Cuenta con sólida experiencia en el desarrollo de proyectos de investigación en las áreas de procesamiento digital de señales e imágenes, e inteligencia artificial, así como automatización y control de procesos. Amplia experiencia académica a nivel de pregrado y posgrado en universidades públicas y privadas, desempeño profesional en el área de investigación y desarrollo en empresas del sector público y privado, asesor de tesis de pregrado y posgrado, publicaciones en revistas indizadas a SCOPUS, revisor de artículos en congresos internacionales, y participación como ponente en diversos eventos académicos. Miembro activo del IEEE y del Colegio de Ingenieros del Perú.

**Autor correspondiente:** [phuamani@urp.edu.pe](mailto:phuamani@urp.edu.pe)

**Orcid:** <https://orcid.org/0000-0002-3753-9777>

**Financiamiento**

Esta investigación no ha recibido financiación externa.

**Conflicto de intereses**

El autor declara que no existe conflicto de intereses en la elaboración del presente artículo

**Responsabilidad ética y moral**

El autor confirma que las fuentes utilizadas en esta investigación fueron debidamente verificadas. Se cita, de manera textual o parafraseada, ideas provenientes de otras investigaciones, reconociendo la autoría correspondiente.

**Correspondencia:** [phuamani@urp.edu.pe](mailto:phuamani@urp.edu.pe)