UNIVERSIDAD EAN

Classwork 1

Hecho por:

Leonardo Jiménez Ubaque

Ingeniería de Sistemas
Redes I

Docente
Alexander García Pérez

Bogotá D.C
27 de febrero de 2023

Índice

- **Introducción**

En el desarrollo de la unidad de estudio de Redes I, se han trabajado distintos conceptos de manera teórica y se ha logrado llevarlos más allá hacia la practica haciendo uso de software como Matlab para tratar los temas de Ondas Sinusoidales, Series de Fourier, Transformada de Fourier y la Modulación Analógica. De este modo, se ha logrado una mejor apropiación de la teoría lo que ha permitido realizar una serie de interfaces donde se plasma esta teoría en la práctica. A continuación, se repasará parte de la teoría y se explicará cómo se desarrolló las interfaces de cada tema.

- **Tema 1: Series de Fourier - Generador de señales**

Vamos a observar lo que son Series de Fourier, su definición y la forma de como se puede crear un generador de este tipo de señales en el software Matlab, esto mediante el uso de una interfaz de usuario que nos permitirá aplicar los conceptos de una manera más cómoda.

Como se menciona anteriormente, se desarrolló una interfaz de usuario para aplicar las Series de Fourier, esto mediante el uso de las ondas sinusoidales, aplicada a 3 tipos diferentes de Series de Fourier. Estos tipos de Series de Fourier son la (Square Wave) *onda cuadrada*, la (Triangular Wave) *onda triangular* y la (Sawtooth Wave) *onda diente sierra.* A continuación, vamos a hablar mas detalladamente sobre cada una de estas.

En primer lugar, encontramos la onda cuadrada, la cual se encuentra representada por la siguiente formula:

$$X(t) = \sum_{\substack{n=1 \\ n=odd}}^{\infty} \frac{4A}{n\pi} \sin\left(\frac{2n\pi t}{T}\right)$$

Donde se entiende que es la sumatoria de todos los numeres impares que van desde 1 hasta infinito de la formula, encontrado las variables:

- A = Amplitud
- T = 1/f = Periodo
- f = Frecuencia

Sin embargo, esta formula no es muy practica a la hora de emplearse para codificar en Matlab, para ello se busco la forma de reescribirla de una manera mas sencilla, esto mediante la aplicación de las matemáticas se logra reducir de la siguiente manera:

$$\frac{2 * n * \pi * t}{T} = \frac{2 * n * \pi * t}{\frac{1}{f}} = 2 * n * \pi * t * f$$

Asimismo, se puede reducir el termino que acompaña a la variable de la Amplitud mediante un proceso de normalización, donde el termino $\frac{4A}{\pi}$ pasa a ser una nueva variable N.

De este modo, se obtiene una formula más fácil de trabajar en el software Matlab.

$$X(t) = N * \sum_{\substack{n=1 \\ n=odd}}^{\infty} \frac{1}{n} \sin(2n\pi tf)$$

En segundo lugar, se tiene a la onda triangular, que se representa mediante la formula presentada a continuación:

$$X(t) = \sum_{\substack{n=1 \\ n=odd}}^{\infty} \frac{8A}{n^2\pi^2} \cos\left(\frac{2n\pi t}{T}\right)$$

Donde se entiende que es la sumatoria de todos los numeres impares que van desde 1 hasta infinito de la formula, encontrado las variables:

- A = Amplitud
- T = 1/f = Periodo
- f = Frecuencia

Para esta formula realizaremos el mismo proceso que hicimos con la fórmula de la onda cuadrada de tal modo que podemos reducir términos de la siguiente forma:

$$\frac{2 * n * \pi * t}{T} = \frac{2 * n * \pi * t}{\frac{1}{f}} = 2 * n * \pi * t * f$$

Y podemos volver a normalizar los valores que acompañan la variable amplitud, de tal manera que $\frac{8A}{\pi^2}$ pasa a ser una variable N. Obteniendo como resultado una formula mas sencilla la cual es:

$$X(t) = N * \sum_{\substack{n=1 \\ n=odd}}^{\infty} \frac{1}{n^2} \cos(2n\pi tf)$$

En tercer y ultimo lugar, se tiene a las ondas diente sierra, que se representan por la formula:

$$X(t) = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{2A}{n\pi} \sin\left(\frac{2n\pi t}{T}\right)$$

La cual podemos reducir de una manera similar a como hicimos con las dos anteriores, de este modo obteniendo la siguiente formula:

$$X(t) = N * \sum_{n=1}^{\infty} (-1)^{n+1} \frac{1}{n} \sin(2n\pi t f)$$

Ya con estas formulas, es mucho más fácil su implementación en el software de Matlab, por lo que para realizar las Series de Fourier se diseño el siguiente código.

```
clear all, close all,  clc, format compact
%Input-------------------------------------------------------------------
f = 3400;
A =5;
cp = 2;
num_cf = 4; %Cantidad Frecuencias
phi = 3*pi/2; % 0 (Sen)
```

Figura 1.1 Datos de Entrada

Lo primero que se realizo fue definir unos datos de entrada, los cuales son los que necesitamos en nuestras formulas, amplitud, frecuencia y otros nuevo que nos permitirá representar gráficamente nuevas cosas usando estas series.

```
%Process-----------------------------------------------------------------
T = 1/f;
t = linspace(0, cp*T, 1000);
N_square = (4*A)/(pi); %Normalizacion
N_sawtooth = (2*A)/(pi); %Normalizacion
N_Triangle = (8*A)/(pi^2); %Normalizacion

s_square = 0;
s_sawtooth = 0;
s_triangle = 0;
s_sinusoidal = 0;
```

Figura 1.2 Procesos

Ahora lo que se realizo fue definir el valor de las contantes que definimos al realizar la normalización y asimismo definir un vector t de tiempo y inicializar nuestras variables donde almacenaremos los diferentes tipos de ondas.

```
%Sinusoidal
for n=1:num_cf
    s_sinusoidal_f = A*sin(2*pi*f*t + phi);
    s_sinusoidal = s_sinusoidal + s_sinusoidal_f;
end


%Square
for n=1:2:(2*num_cf-1)
    s_square_f = (N_square/n)*sin(2*pi*n*f*t);
    s_square = s_square + s_square_f;
end

%Sawtooth
for n=1:num_cf
    s_sawtooth_f = ((-1)^(n+1))*(N_sawtooth/n)*sin(2*pi*n*f*t);
    s_sawtooth = s_sawtooth + s_sawtooth_f;
end

%Triangle
for n=1:2:(2*num_cf-1)
    s_triangle_f = (N_Triangle/n^2)*cos(2*pi*n*f*t);
    s_triangle = s_triangle + s_triangle_f;
end
```

Figura 1.3 Formulas de ondas en ciclos

En este punto ya tenemos definidas nuestras formulas en el código y listas para ejecutarse, por lo que para realizar la sumatoria de la formula se realiza un ciclo for que esta definido de maneras distintas para cada tipo de onda de acuerdo a lo que cada formula especifica, ya sea avanzar números pares o tomar todos los números.

```
%Output-------------------------------------------------------------------
%One graphic
figure(1)
subplot(2,2,1), plot(t,s_sinusoidal), title('Sinusoidal Signal'), grid on
subplot(2,2,2), plot(t,s_square), title('Square Signal'), grid on
subplot(2,2,3), plot(t,s_sawtooth), title('Sawtooth Signal'), grid on
subplot(2,2,4), plot(t,s_triangle), title('Triangle Signal'), grid on


%Sinusoidal graphic
figure(2)
plot(t, s_sinusoidal), title('Sinusoidal Signal')

%Square graphic
figure(3)
plot(t, s_square), title('Square Signal'), grid on

%Sawtooth graphic
figure(4)
plot(t, s_sawtooth), title('Sawtooth Signal'), grid on

%Triangle graphic
figure(5)
plot(t, s_triangle), title('Triangle Signal'), grid on
%NOTE, ADD SINOSUIDAL GRAHP
```

Figura 1.4 Imprimir las ondas

Ya mediante este código tenemos las diferentes Series de Fourier codificadas y listas para ser graficas. Sin embargo, este es un script, no es la codificación que esta en la interfaz, pero este mismo código será el que se implementara allá, la única diferencia es que se deberá de enlazar este código a ciertos elementos de la interfaz, lo que permitirá que con ingresar unos datos y dar clic a un boto se genere nuestras graficas sin necesidad de codificar, es por esto que nuestra interfaz de Series de Fourier quedo de la siguiente manera.

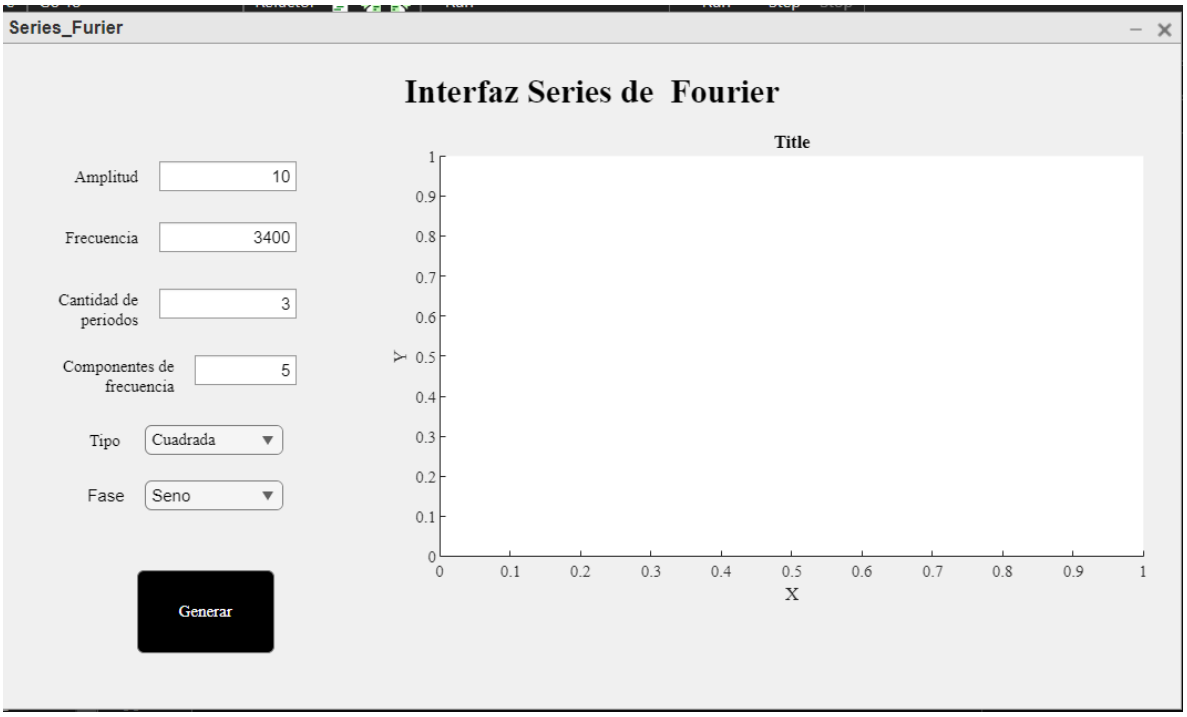

Figura 1.5 Interfaz Inicializada

- **Sinusoidal Seno**

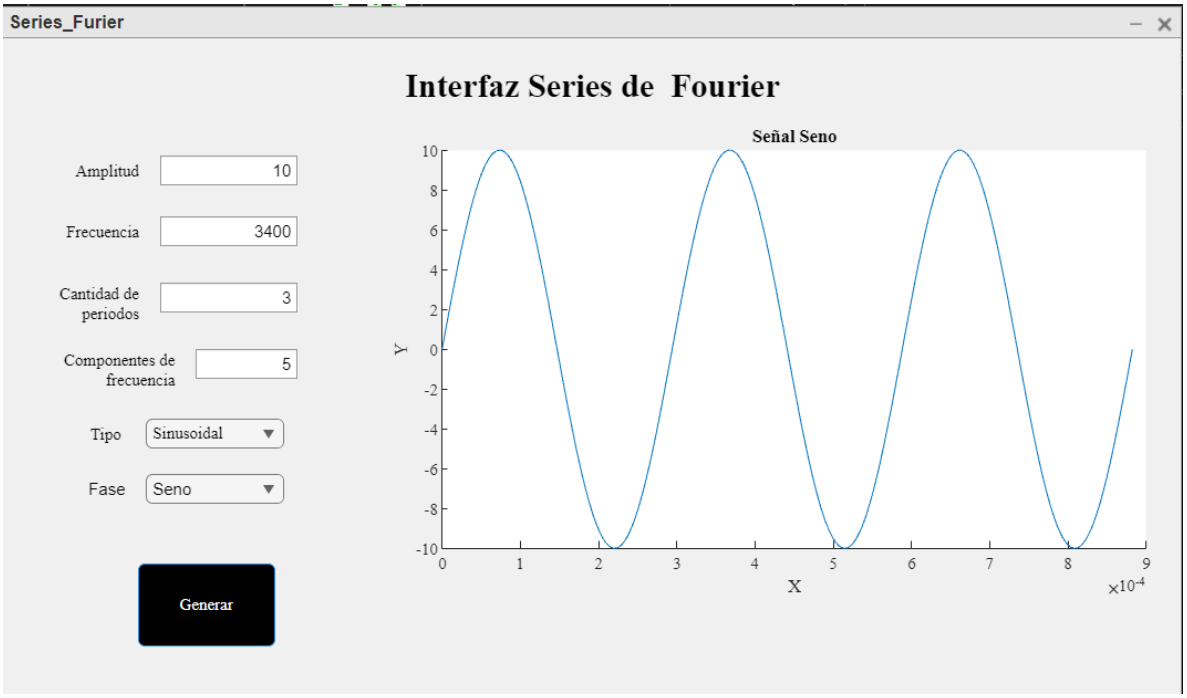

Figura 1.5 Interfaz gráfica sinusoidal Seno

- **Sinusoidal Coseno**

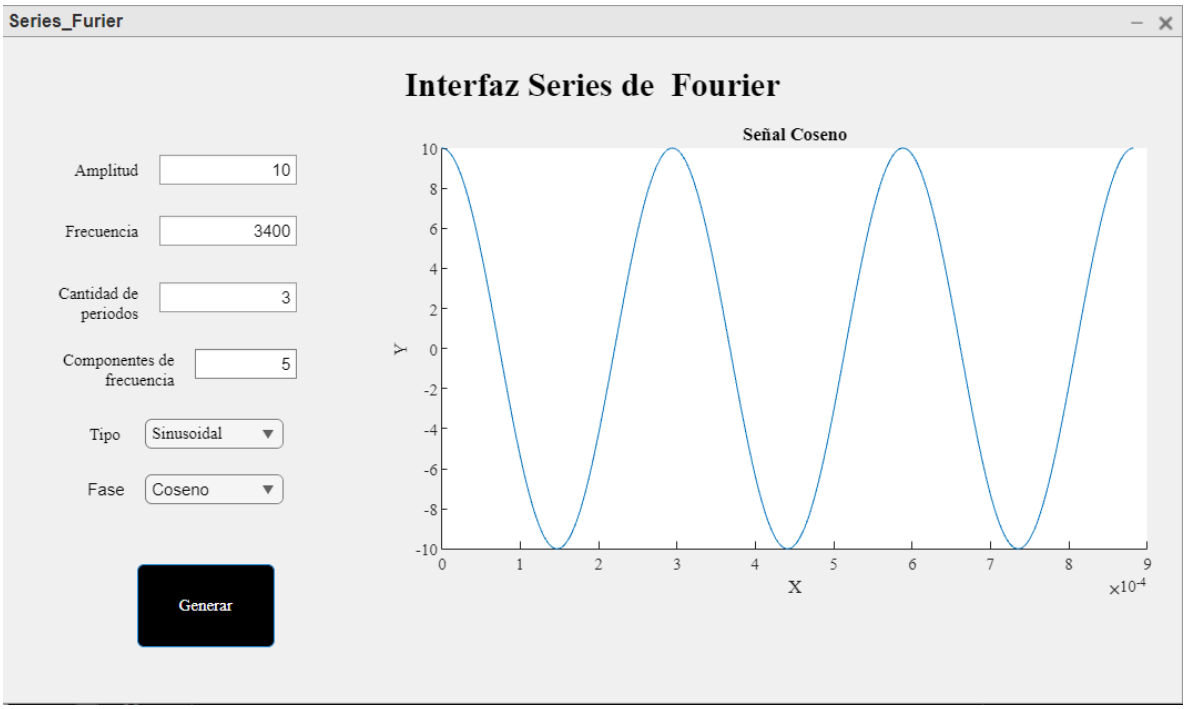

Figura 1.6 Interfaz gráfica sinusoidal Coseno

- **Sinusoidal - Sen**

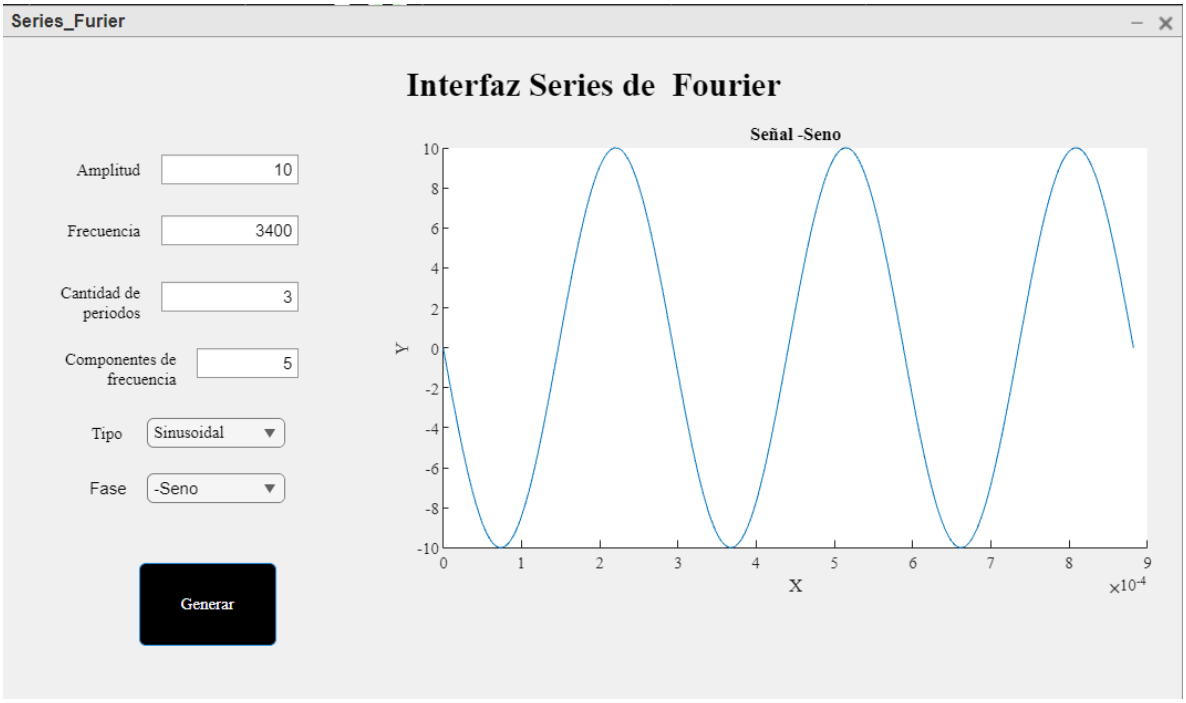

Figura 1.7 Interfaz gráfica sinusoidal – Sen

- **Sinusoidal - Coseno**

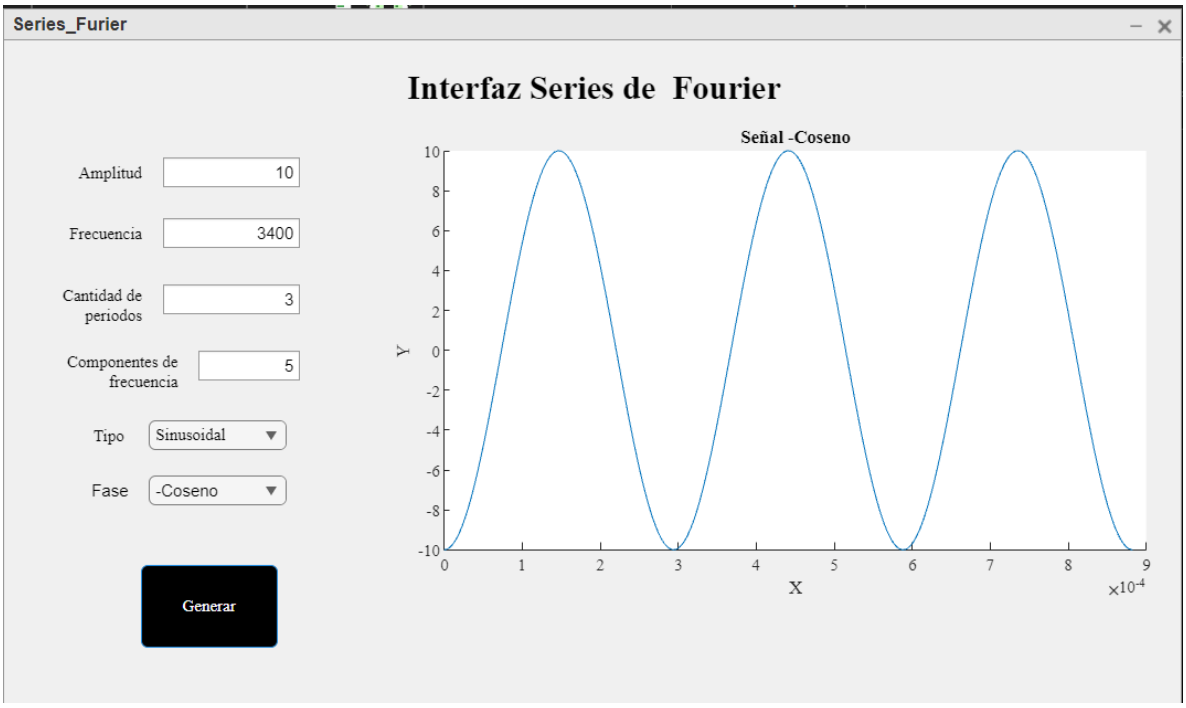

Figura 1.8 Interfaz gráfica sinusoidal – Coseno

- **Cuadrada**

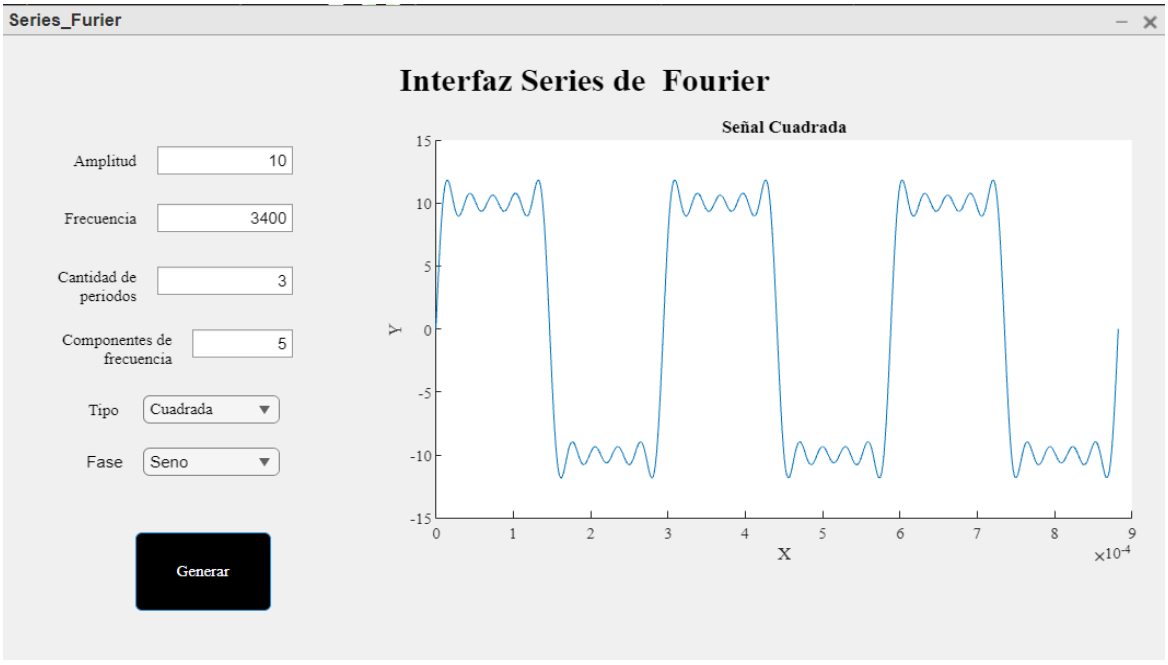

Figura 1.9 Interfaz gráfica Cuadrada

- **Triangular**

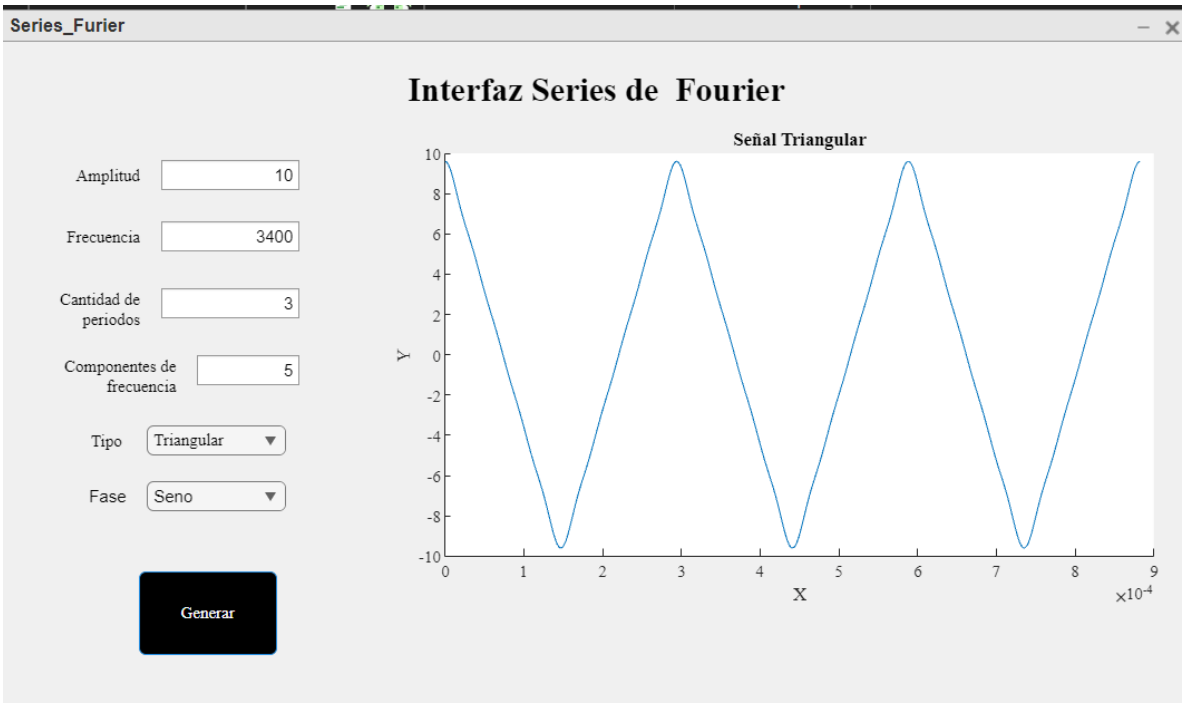

Figura 1.10 Interfaz gráfica Triangular

- **Diente Sierra**

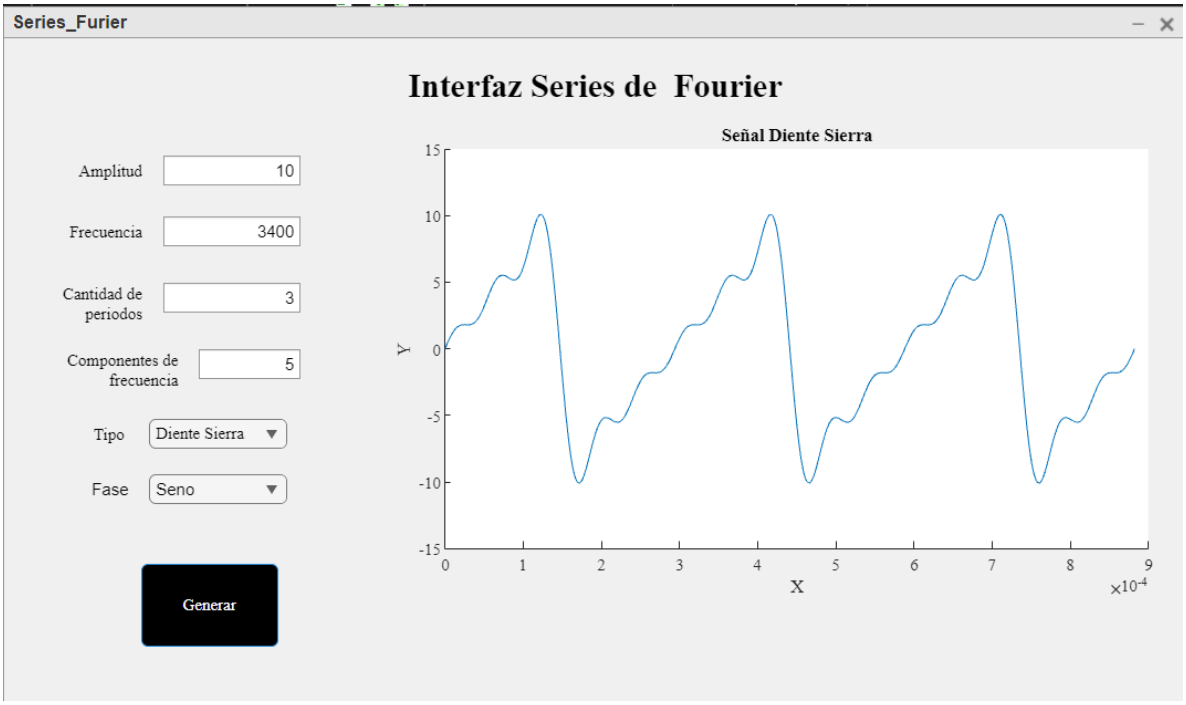

Figura 1.11 Interfaz gráfica Diente Sierra

- **Tema 2: Transformada de Fourier - Analizador de espectro**

Se desarrollo una interfaz, de la aplicación de la Transformada de Fourier para realizar un analizador de espectros, de tal manera que las señales que se encuentran en el dominio del tiempo podrán transformarse y pasarse al dominio de la frecuencia de esa misma señal, y viceversa. Esto mediante la aplicación de la siguiente formula.

$$X(f) = \int_{-\infty}^{\infty} x(t) * e^{-j2\pi f t} dt$$

Como se puede observar, vamos a usar la formula que esta en el dominio del tiempo (x(t)) para que cuando se integre en función del tiempo se pase al dominio de la frecuencia.

A la hora de pasar a la practica en Matlab, se descubrió que ya hay un comando diseñado para esta acción estos comando son fft() y shiftfft() los cuales nos permiten simular dicha integral sin definir límites si no definiendo exclusivamente la gráfica en el dominio del tiempo.

```
clear all, close all,  clc, format compact
%Input-------------------------------------------------------------------------
f = 3400;
A =5;
cp = 1;
num_cf = 4; %Cantidad Frecuencias

%Process------------------------------------------------------------------------
T = 1/f;
t = linspace(0, cp*T, 1000);
N_square = (4*A)/(pi); %Normalizacion



cf1 = (A/1)*sin(2*pi*1*f*t);
cf2 = (A/3)*sin(2*pi*3*f*t);
cf3 = (A/5)*sin(2*pi*5*f*t);
cf4 = (A/7)*sin(2*pi*7*f*t);

xt_cf1 = cf1;
xt_cf2 = cf1 + cf2;
xt_cf3 = cf1 + cf2 + cf3;
xt_cf4 = cf1 + cf2 + cf3 + cf4;

Xf_1cf = abs(fft(xt_cf1));
Xf_2cf = abs(fft(xt_cf2));
Xf_3cf = abs(fft(xt_cf3));
Xf_4cf = abs(fft(xt_cf4));

Xf_1cf_shift = abs(fftshift(fft(xt_cf1)));
Xf_2cf_shift = abs(fftshift(fft(xt_cf2)));
Xf_3cf_shift = abs(fftshift(fft(xt_cf3)));
Xf_4cf_shift = abs(fftshift(fft(xt_cf4)));

s_square = 0;
for n=1:2:(2*num_cf-1)
    s_square = s_square + (N_square/n)*sin(2*pi*n*f*t);
    s_square_tf = abs(fft(s_square));
end
```

Figura 2.1 Ingreso de datos y proceso de formulas

De acuerdo con la imagen anterior, se observa que ya definimos los datos de entrada y las formulas de las onda que se va a transformar, así como el comando que hará la transformación de Fourier, lo que en la interfaz nos dará como resultado los siguiente.

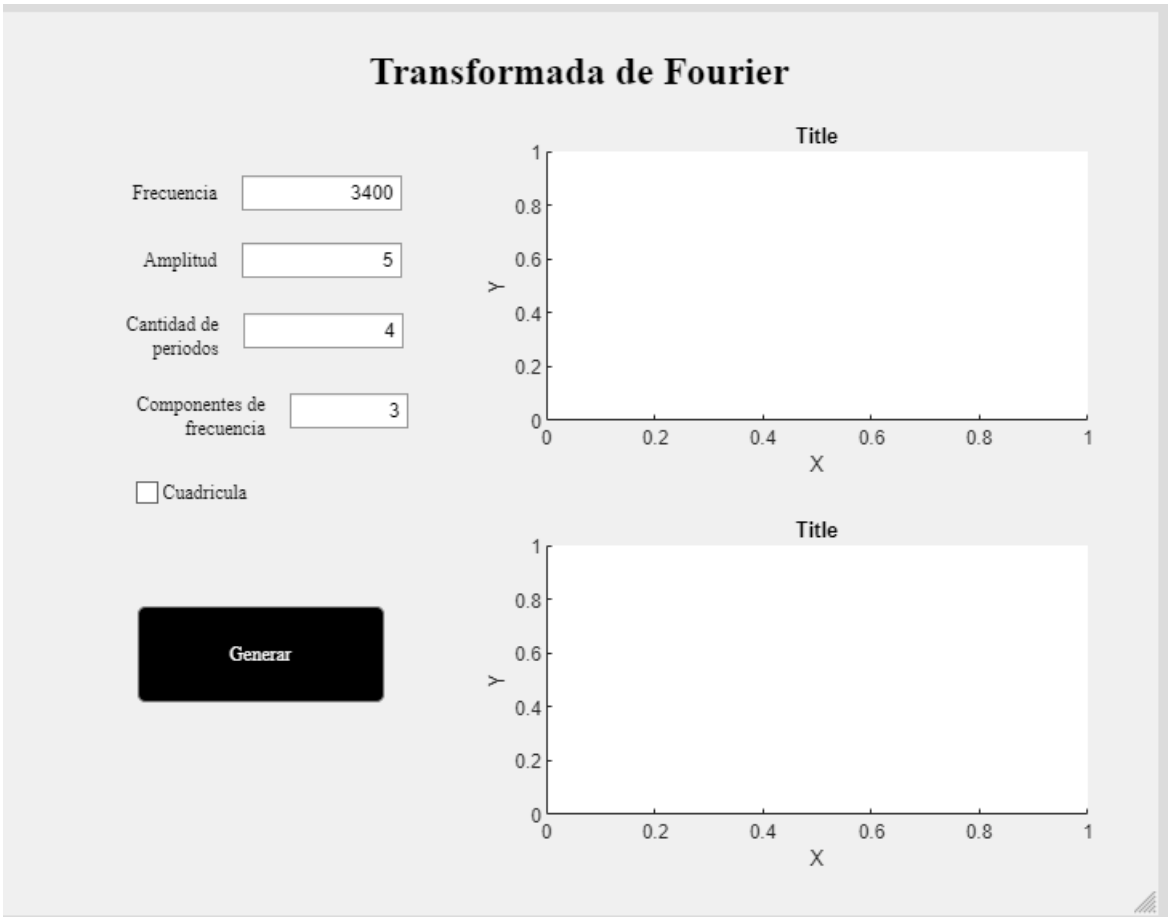

Figura 2.2 Interfaz inicializada

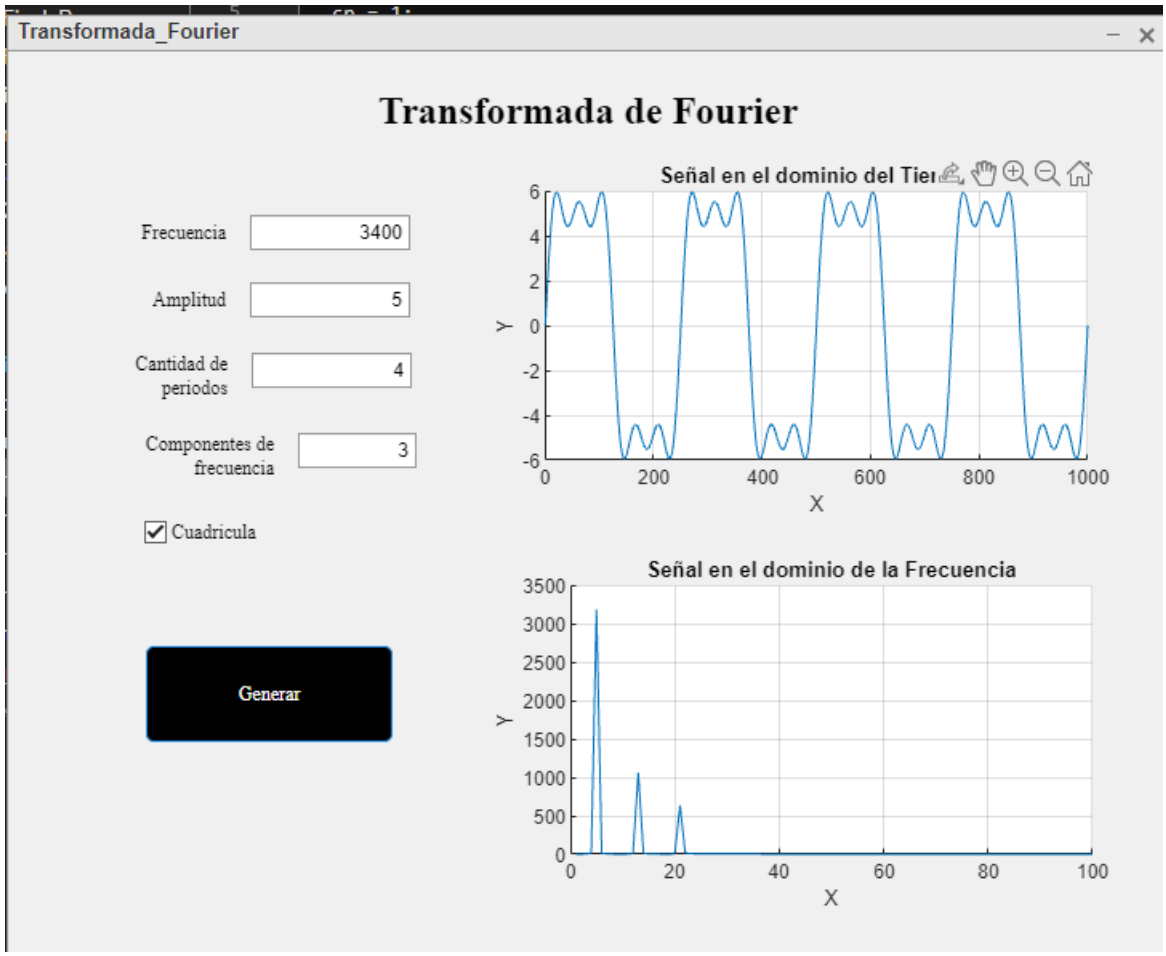

Figura 2.3 Interfaz Señal en dominio de tiempo y su transformada a dominio de frecuencia

- **Tema 3: Transmisores Analógicos - Modulación Analógica**

Este es el ultimo tema que se ha trabajado hasta el momento, se desarrollo una interfaz en Matlab, que nos permite simular la modulación analógica, lo que se conoce como convertir un dato de carácter analógico a una señal analógica, esto aplicando diversas fórmulas.

Pero debemos entender por que se debe modular los datos, esto se debe ya que permite que la información se transporte de manera eficiente y confiable a través de largas distancias, y transforma la información en una forma que se puede superponer en la señal de portadora y luego recuperarse en el receptor.

Formula de una modulación en amplitud:

$$Xam = [1 + m * x(t)] * carrier(t)$$

- Carrier(t) = Portador de datos.
- M = Índice de modulación
- X(t) = Datos analógicos

Formula de modulación en frecuencia podemos decir que:

$$Xfm = Ac * \sin(2\pi * fc * t + nfm * x(t))$$

- Ac = Amplitud del Carrier
- Fc = frecuencia del Carrier
- Nfm = Índice de modulación
- X(t) = Datos analógicos

Ya con estas formulas podemos diseñar nuestra código e interfaz para crear un modulador de señales análogas.

```
%Data In
Ax = 2;
fx = 3400;
cpx = 2;

%Carrier
Ac = 10;
fc = 1000e3;
cpc = 13;
phi_c = 0;

%Modulation Index
m = 0.6;                % 0<= m <=1
nfm = 2;                % nfm >= 2
npm = 2;                % npm >= 2

%Process-----------------------------------------------------------------
%xt
Tx = 1/fx;
tx = linspace(0, cpx*Tx, 1000);
xt = Ax*sin(2*pi*fx*tx);

%Carrier
Tc = 1/fc;
tc = linspace(0, cpc*Tc, 1000);
xc = Ac*sin(2*pi*fc*tc+phi_c);

%AM Modulation
xam = (1+m*xt).*xc;

%FM Modulation
xfm = Ac*sin(2*pi*fc*tc+nfm*xt);

%PM Modulation
Dxt = Ax*cos(2*pi*fx*tx);
xpm = Ac*sin(2*pi*fc*tc+npm*Dxt);
%Output------------------------------------------------------------------
figure(1)
subplot(5,1,1), plot(tc, xc), title('Analog Data - Carrier')
subplot(5,1,2), plot(tx, xt), title('Analog Data - Voice')
subplot(5,1,3), plot(tc, xam), title('Signal - AM')
subplot(5,1,4), plot(tc, xfm), title('Signal - FM')
subplot(5,1,5), plot(tc, xpm), title('Signal - PM')
```

Figura 3.1 Código modulación AM, FM y PM en Matlab

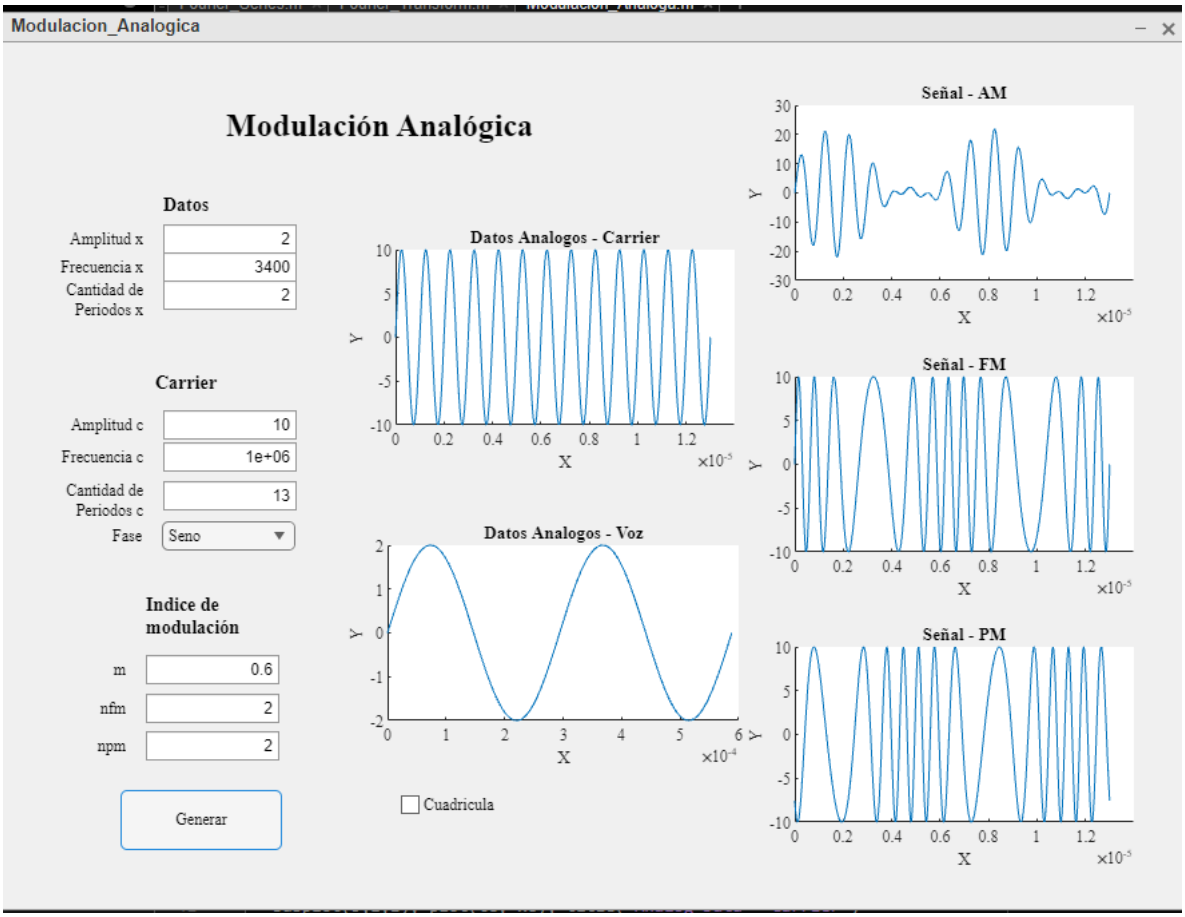

Gráfico 3.2 Interfaz Modulación Analógica

De este modo, se construyeron las interfaces sobre los temas que se analizaron y trabajaron a lo largo del corte, de tal modo que se logro aprender la teoría y llevar a la practica en un ambiente virtual como lo es Matlab.

Enlaces videos Interfaces:

1. https://youtu.be/dZ8B5CD_PlA

2. https://youtu.be/XOzr_7_Ti84

3. https://youtu.be/OklDvDX-Nm8