

ESCUELA POLITÉCNICA NACIONAL

INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

PROCESAMIENTO DIGITAL DE SEÑALES

ALUMNO: Marialejandra Contreras y Melanny Dávila

SEMESTRE: 2020-B

PARALELO: GR1

FECHA: 02/12/2020

PROFESOR:

DR. ROBIN ÁLVAREZ

TEMA: Señales Analógicas Básicas

# **CAPÍTULO 1.2**

**GENERACIÓN DE ONDAS DETERMINISTAS CON MATLAB Y EL PUERTO DE AUDIO**

En el capítulo anterior se empleó un **instrumento virtual** para generar señales **reales** **elementales**, en el este capítulo se implementarán **generadores de señal** **personalizados** mediante algoritmos propios desarrollados en Matlab. Iniciamos primeramente con señales deterministas elementales y luego se verán casos de señales más complicadas, imposibles de ser generadas mediante instrumentos reales o virtuales.

**1.2.1 GENERACIÓN DE UNA SEÑAL SINUSOIDAL**

Al tratar de generar una **onda continua**, sin importar el tipo, el computador, que es un elemento discreto, es incapaz de trabajar con un número infinito de valores. Una posible solución es seleccionar solamente ciertas muestras que permitan tener una forma aproximada de dicha onda (Figura 1.2.1). Estas muestras son las que serán enviadas hacia el puerto de salida del puerto de audio. El tiempo entre dos muestras consecutivas se conoce como *tiempo entre muestras (Ts)* y su inverso será el *número de muestras por unidad de tiempo (Fs).* De esta forma, mientras más muestras se consideren, es decir, mientras más pequeño sea el tiempo entre muestras Ts, se obtendrá una señal más aproximada a la original. Este proceso se realizará en toda la duración de la señal que, en el caso del ejemplo es de 3 segundos.



**Figura 1.2.1** Proceso de selección de ciertas muestras, cada tiempo Ts, de una onda continua.

En cualquier caso de que la señal a generar sea una onda periódica, la expresión que relaciona el tiempo entre muestras (Ts) con el período de la onda (T) es la siguiente:

Entonces, la representación de la señal en tiempo discreto estaría dada por:

Donde sería la frecuencia de la onda y sería la fase inicial.

Veamos el siguiente ejemplo práctico que está divido en etapas para mejor comprensión.

**EJEMPLO: (a) generar en Matlab una onda sinusoidal con amplitud máxima de 1 (sin unidades), frecuencia 1000 (Hertz o ciclos / segundo), duración de pocos ciclos y fase inicial de 180 grados.**

**Resolución**: primeramente se establecen los valores de todos los parámetros solicitados, excepto la amplitud que aquí lo ponemos como 1 y que luego, mediante el control de volumen, será configurado al valor requerido por el usuario, por ejemplo, 0.5 V pico-pico. Si la Fo es de 1000 Hz, entonces, pocos ciclos podrían ser 2 (duración = 1/ (2T)) para poder visualizarlo perfectamente, luego ya el usuario podría poner la duración en segundos que desee:

|  |
| --- |
| clc; clear all; close all;  %% Configuración de los parámetros de la onda:  %Amplitud máxima  Amax=1;  %Frecuencia  Fo=1000;  %Período  T=1/Fo;  %Fase incial  Phy=180;  %Duración de la onda  duracion=5T; |

De acuerdo a la teoría anterior, si tomamos 100 muestras por cada período, el **tiempo entre muestras** será T / 100. En base a esto, conformamos un **vector de tiempos discretos** que comienza en 0 segundos con saltos de Ts hasta alcanzar el valor igual a la **duración**. Esto quiere decir que cada Ts segundos se estará tomando una muestra de la señal.

|  |
| --- |
| %Período de muestreo:  Ts=T/100;  %Vector de tiempos discretos:  tn=0:Ts:duracion; |

Una vez definido el **vector de tiempos discretos**, para cada una de las muestras debemos encontrar su amplitud correspondiente, que en este caso está dado por la función **seno**. Esta tarea podríamos hacerla mediante un **lazo FOR** como se lo hace tradicionalmente en otros lenguajes de programación; sin embargo, Matlab tiene una manera más simple de realizarlo y que se denomina **FORMA VECTORIAL,** esta consiste en aplicar una función a todo un vector, en este caso, se aplica la función seno al vector de tiempos discretos **tn** y como resultado se obtiene un vector **yn** con los valores correspondientes que lo denominaremos **vector de amplitudes**:

|  |
| --- |
| %% Construcción de la señal -- vector de amplitudes:  yn=Amax\*sin(2\*pi\*Fo\*tn+Phy); |

Una vez que tenemos ambos vectores, se procede a graficar la señal mediante el comando ***plot****.*

|  |
| --- |
| %% Gráfica de la señal  plot(tn,yn,':ob','MarkerSize',4);  %Etiquetas y título  xlabel('Tiempo');  ylabel('Amplitud');  title('Onda sinusoidal de 1 KHz'); |

Luego de correr este programa, Matlab genera el gráfico mostrado en la Figura 1.2.2 donde se puede **confirmar los parámetros dados** inicialmente: fase= 180 grados, T = 1/F = 1 ms, es decir, F = 1000 Hz y duración igual a 2 períodos de la señal.

**(b) Convertir esta señal simulada en una SEÑAL REAL de modo que se escuche a través de los altavoces del computador.**

En este punto se requiere de una instrucción que transforme los valores numéricos del **vector de amplitudes** **yn** en niveles de voltaje, los mismos que son enviados hacia la salida del puerto de audio. Como ya sabemos, **los niveles máximos permitidos en el puerto de audio son de +/- 1 voltios,** si estos niveles son sobrepasados, la señal será cortada y debido a esto aparecerán armónico.

Una de las instrucciones que permite hacer esto es la instrucción ***sound*** cuya sintaxis es la siguiente:

***sound*** (yn, Fs), donde Fs es el inverso de Ts: Fs =1/Ts

**Nota**. Se debe tomar en cuenta que, dentro de la ayuda de instrucción **Sound,** la frecuencias de muestreo (Fs) tiene la restricción que debe estar en el rango **(1000 Hz- 384000 Hz)**. Para evitar que se generen errores al momento de ejecutar el programa, se debe verificar que dicha Fs esté en este rango, si no es así, se podría modificar el valor de Ts=T / #muestras, según el caso.

**Al ejecutar el programa,** ya se escucha un sonido; sin embargo, la **verificación final** se la realizará mediante un osciloscopio real o virtual (audio tester).

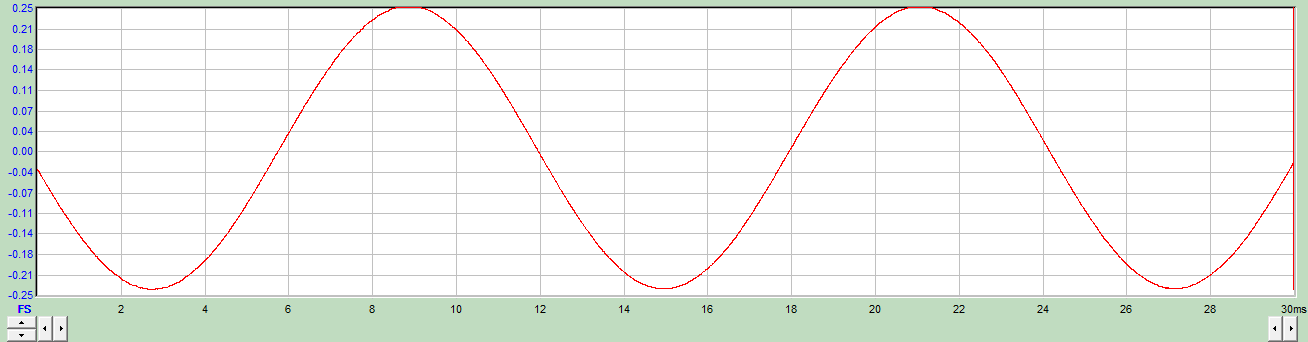
Imagen que contiene mesa, mapa

Descripción generada automáticamente

**Figura 1.2.2** Onda sinusoidal de amplitud máxima = 1 (sin unidades), frecuencia 1 KHz y fase inicial de 180 grados, generada con Matlab.

**(c) Verificar, mediante el instrumento virtual audio tester, que la señal real generada tenga los parámetros solicitados. Variar el control del volumen del dispositivo hasta obtener el valor de 0.5 Vpico-pico (0.25 V positivos y 0.25 V negativos).**

Empleando el audiotester, luego de colocar el **cable de puente entre la salida y la entrada** del puerto de audio, en la Figura 1.2.3 se puede observar la amplitud pico-pico de la señal, que ha sido obtenida variando el control del volumen del dispositivo hasta obtener el valor solicitado de 0.5 Vpico-pico (0.25 V positivos y 0.25 V negativos). El período de la señal en el dominio del tiempo es de 1 ms, lo cual correspondería a una Frecuencia de 1 KHz, y esto es confirmada en el dominio de frecuencia donde el pico de la componente fundamental está ubicado exactamente en 1 KHz (Figura 1.2.4).

****

**Figura 1.2.3** Señal sinusoidal de amplitud pico-pico 0.5V, generada a 1KHz

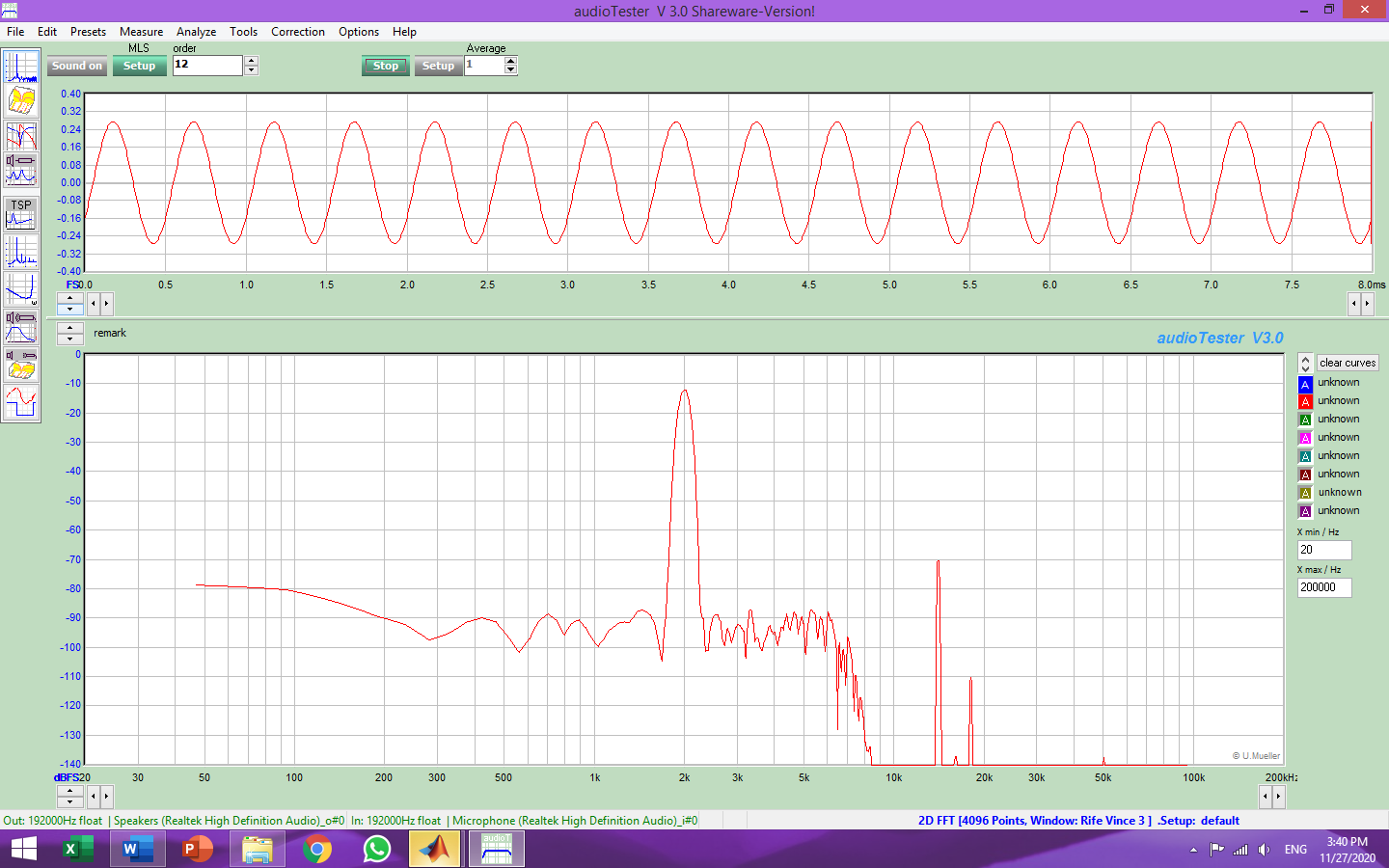


**Figura 1.2.4** Espectro de la señal sinusoidal de 1KHz generada en Matlab y analizada desde audio tester.

**Observación:** Por defecto la instrucción sound está enviando el vector de amplitudes **a los dos canales de salida del puesto de audio** por lo que, en las gráficas anteriores, están superpuestos en color azul y rojo ambas señales. Más adelante se explicará cómo generar dos señales distintas, la una que salga por el canal 1 y la otra por el canal 2.

DEBER: REHACER PARA 2 KHz





Resultado obtenido de la onda sinusoidal a 2[KHz]

**1.2.2 GENERACIÓN DE UNA SEÑAL RECTANGULAR**

Toda la teoría anterior sigue siendo válida para generar cualquier otra señal determinista y lo único que cambia es la función de Matlab que genera el **vector de amplitudes**:

Para onda **rectangular** con tiempo en alto ***ta*** que es conocido como **ancho de pulso** osi está en porcentaje se lo conoce como **ciclo útil**.

**EJEMPLO: generar una onda rectangular bipolar de 1 KHz con un ciclo útil del 80% y analizarla en Audio Tester**

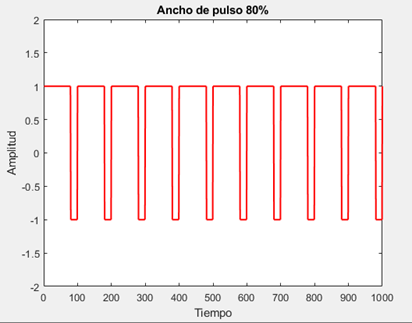
En el procedimiento anterior, la única línea de código que cambia es siguiente:

%tiempo de subida 80% y bajada 20%

y1=square(2\*pi\*Fo\*tn,**80**);

Esta función genera por defecto una onda **bipolar**, es decir, entre +/- 1.

Luego de cambiar el título correspondiente, la señal que se obtiene en Matlab se muestra en la Figura 1.2.5.

******

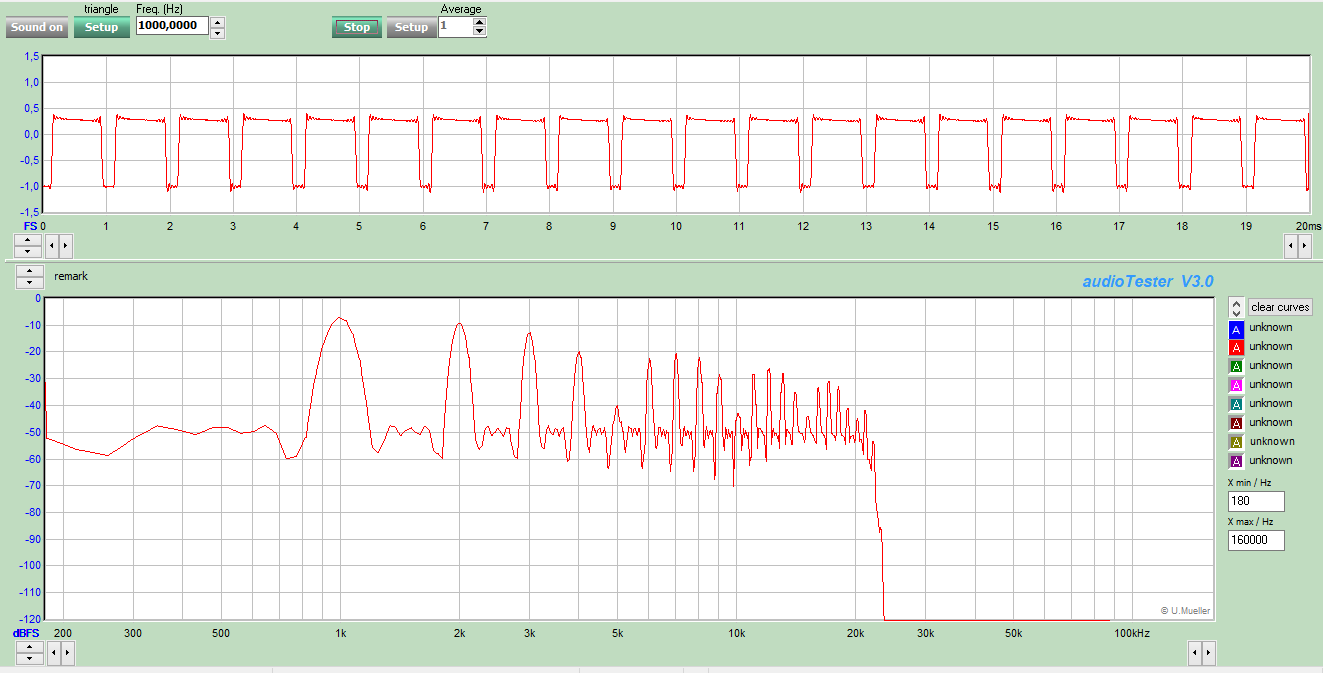
**Amplitud**

**Tiempo en alto = 80%**

**Ciclo de 100 ms**

**Figura 1.2.5** Resultado de la generación en Matlab de una onda rectangular bipolar con 80% de tiempo en alto y Fo = 1 KHz.

A la salida del puerto de audio, por medio del audio tester, en el dominio de frecuencia, se puede ver que se tiene la componente fundamental ubicada en 1 KHz y armónicos impares ubicados en 3, 5, 7, 9 KHz, etc (Figura 1.2.6). Estos armónicos, si bien en teoría van hasta el infinito, debido a las limitaciones en frecuencia del puerto de audio, se cortan aproximadamente en 20 KHz.



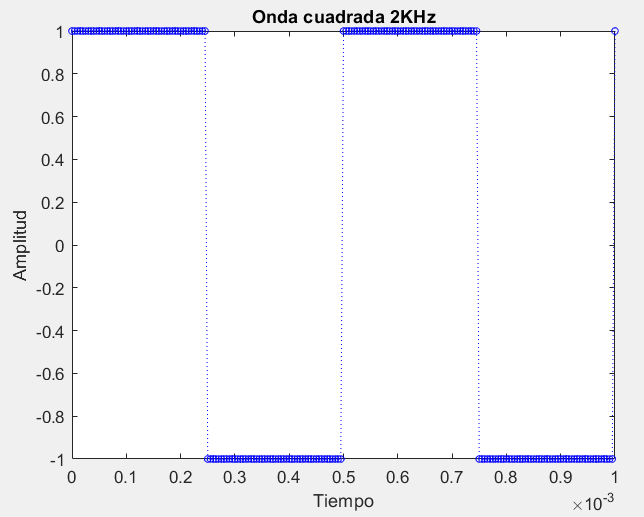
**Figura 1.2.6** Resultado obtenido mediante el audio tester al analizar la onda rectangular **bipolar** con 80% de ciclo útil.

Si se deseara generar una onda cuadrada, el tiempo en alto coincidiría con el tiempo en bajo y la instrucción sería:

% onda cuadrada: tiempo de subida 50% y bajada 50%

y1=square(2\*pi\*Fo\*tn,**50**);

DEBER: REHACER PARA 2 KHz





Resultado obtenido de la onda cuadrada a 2[KHz]

**1.2.3 GENERACIÓN DE UNA SEÑAL TRIANGULAR**

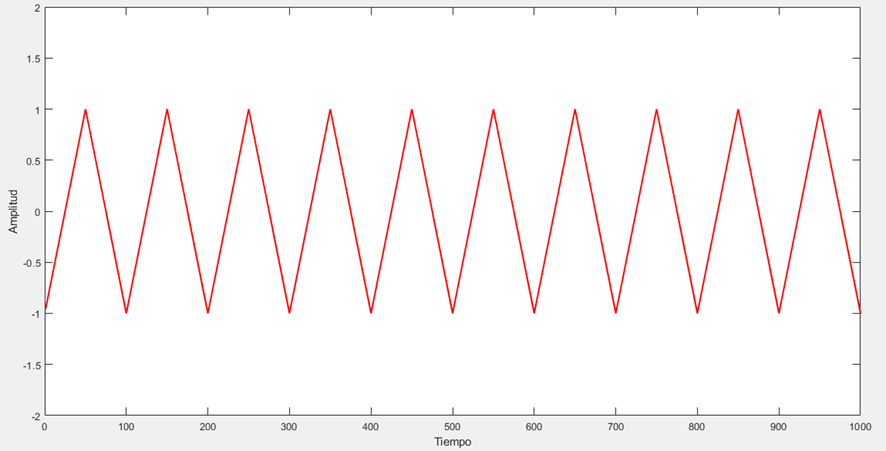
En este caso, para onda **triangular** con tiempo de subida en porcentaje ***ta,*** la instrucción es la siguiente:

**Ejemplo: generar una onda triangular bipolar de 1 kHz con tiempo de subida 50% y analizarla en Audio Tester.**

Respecto del código inicial, la única instrucción que cambia es la siguiente:

y= **sawtooth** (2\*pi\*Fo\*tn, **50 / 100**);

La onda obtenida en Matlab se muestra en la Figura 1.2.7 y los resultados en audio tester muestran que solamente se tiene armónicos impares (Figura 1.2.8)

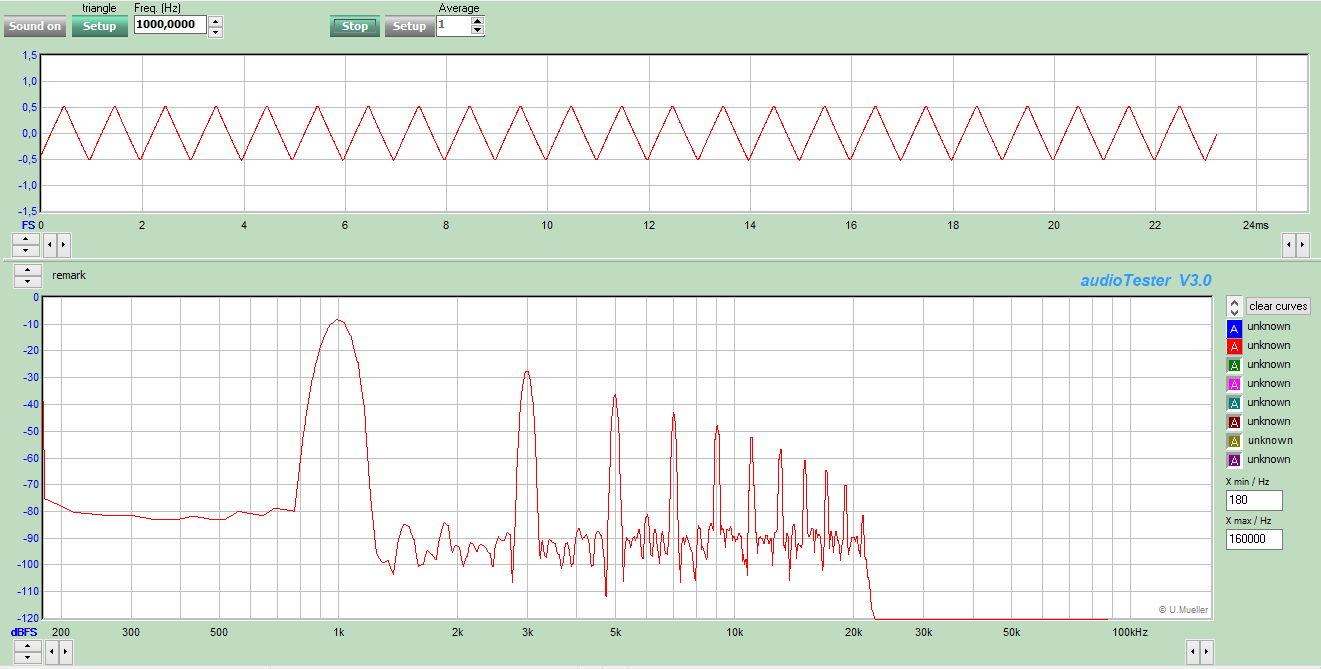


**Amplitud**

**Ciclo de 100 ms**

**50% de subida**

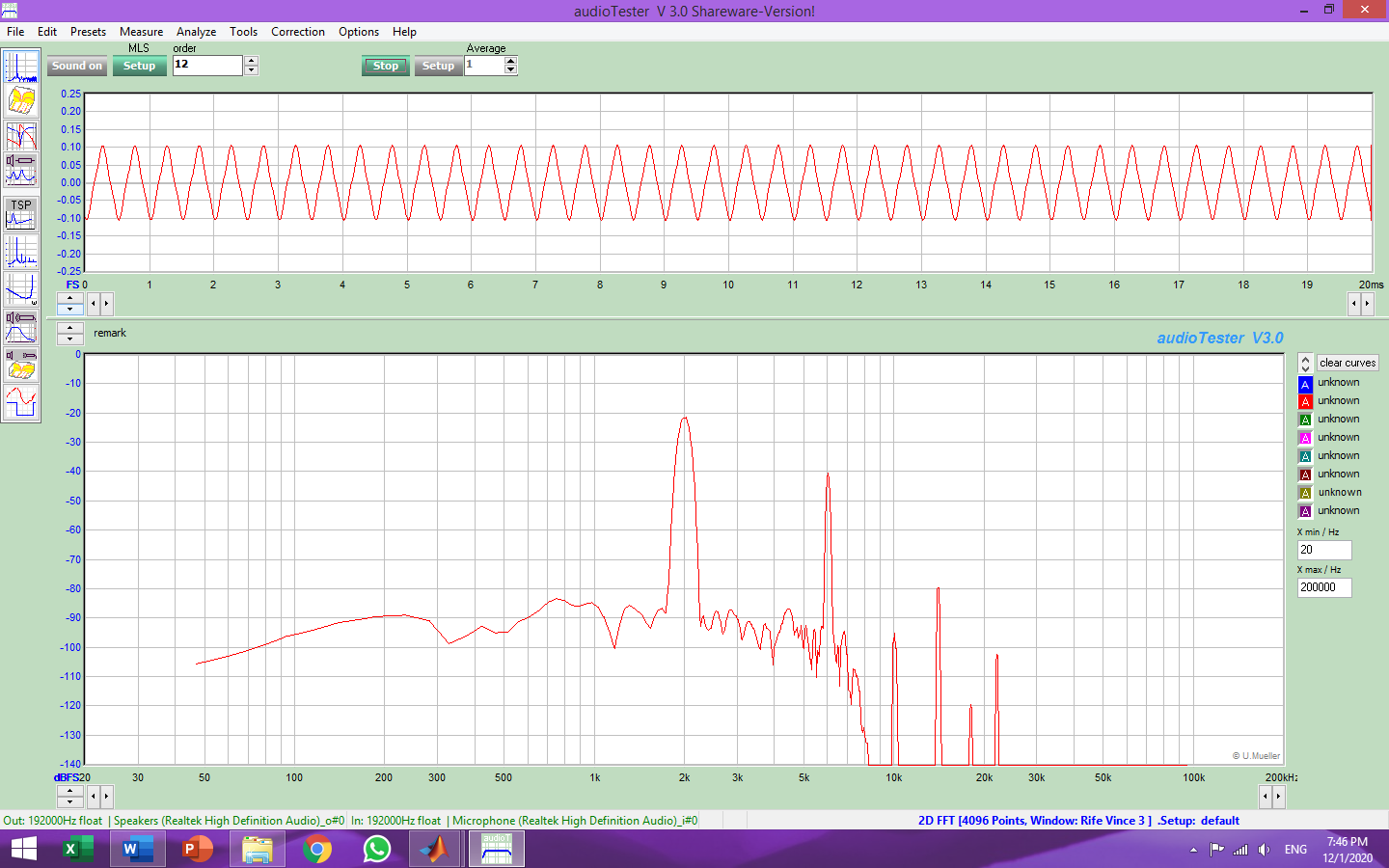
**Figura 1.2.7.** Señal triangular bipolar con 50% de subida, generada en Matlab.



**Figura 1.2.8.** Respuesta en tiempo y frecuencia de la señal triangular con 50% de tiempo de subida

DEBER: REHACER PARA 2 KHz, ESCALA LINEAL

****



Resultado obtenido de la onda triangular a 2[KHz]

**1.2.4 GENERACIÓN DE UNA SEÑAL DIENTE DE SIERRA**

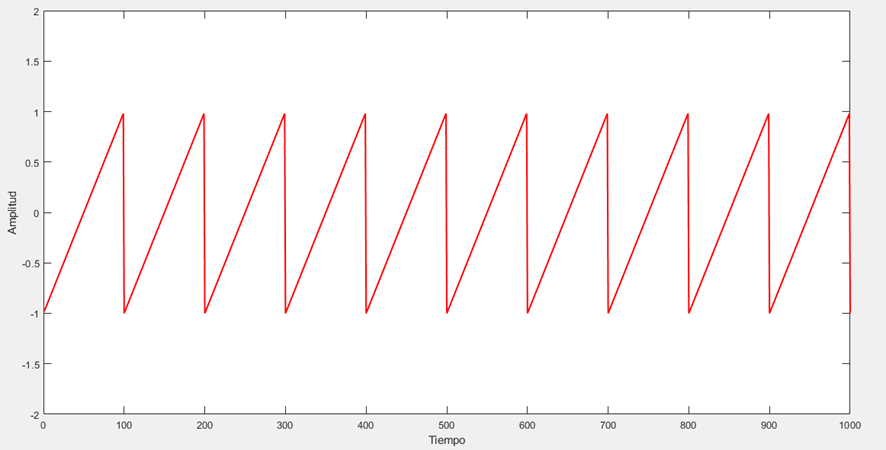
En este caso, para onda **diente de sierra*,*** la instrucción es la misma que para la señal triangular solo que el tiempo de subida ocupa el 100%:

**Ejemplo: generar una onda diente de sierra bipolar de 1 kHz y analizarla en Audio Tester.**

Respecto del código inicial, la única instrucción que cambia es la siguiente:

y= **sawtooth** (2\*pi\*Fo\*tn, 1);

La respuesta obtenida en Matlab se la puede ver en la Figura 1.2.9.

******

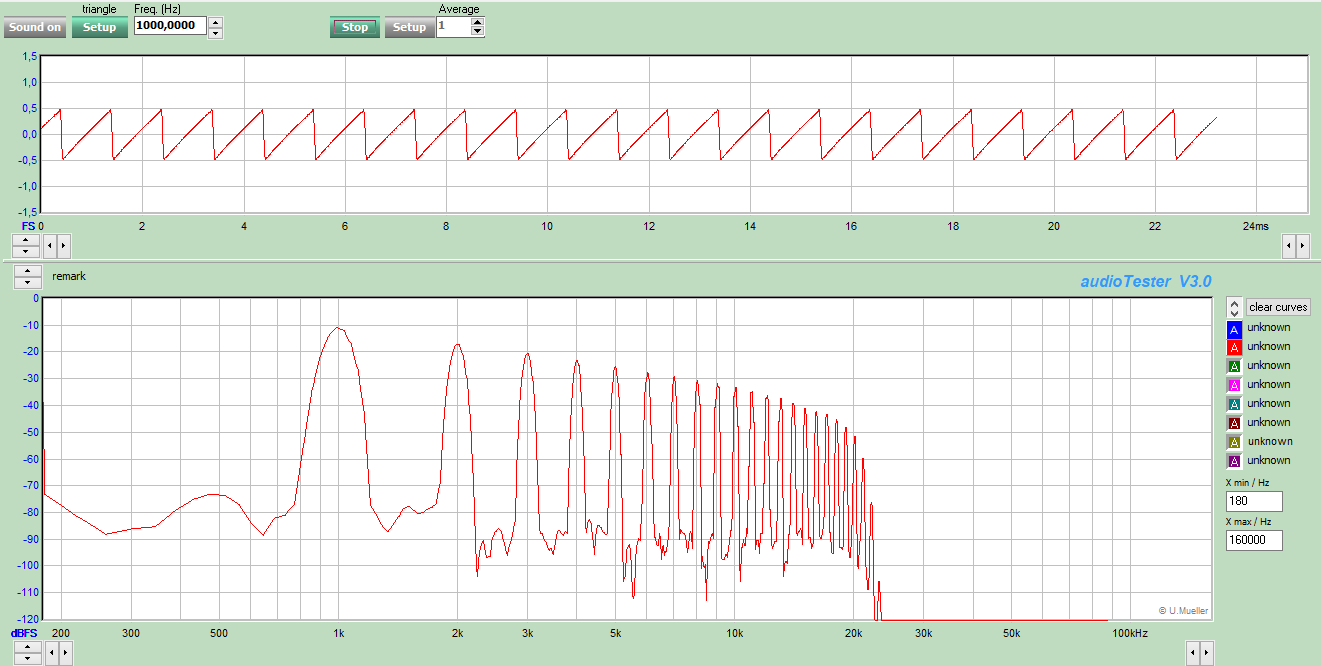
**Amplitud**

**Ciclo de 100 ms**

**100% de subida**

**Figura 1.2.9**. Señal diente de sierra generada en Matlab

A la salida del puerto de audio tenemos el resultado mostrado en la Figura 1.2.10.



**Figura 1.2.10.** Respuesta a la salida del puerto de audio correspondiente a la señal diente de sierra obtenida mediante audio tester.

Como se puede observar, se confirma el resultado esperado para esta onda diente de sierra: se tiene todos los armónicos tanto pares como impares, únicamente limitados por la respuesta en frecuencia de la tarjeta de sonido.

**1.2.5 GENERACIÓN DE OTROS TIPOS DE SEÑALES DETERMINÍSTICAS MÁS ELABORADAS**

Como se comentó al inicio del capítulo**,** el objetivo se implementar **generadores de señal** **personalizados** mediante algoritmos propios desarrollados en Matlab, los que serían difíciles o imposibles de obtener con generadores comerciales virtuales o reales.

**A. SINUSOIDE RECTIFICADA COMPLETA:**

Para obtener este tipo de onda, partiendo de la señal sinusoidal ya explicada anteriormente, solamente habría queinvertir los semiciclos negativos, es decir, obtener su **valor absoluto (**comando ***abs***). Para evitar realizar una amplificación en una zona, se determina una **duración** de pocos ciclos y luego de confirmar sus parámetros, esta puede ser cambiada según el tiempo que desee el usuario.

**Ejemplo: generar una sinusoide rectificada completa de frecuencia 100 Hz.**

En esta ocasión ponemos el programa completo y solamente se muestra la señal generada en Matlab, lo obtenido a la salida del puerto de audio queda para ser realizado por el lector.

Clc, clear all, close all

duracion=0.02;

f=100;

T=1/f;

% tiempo entre muestras (consideramos 100 muestras/período)

Ts=T/100;

tn=0:Ts:uración; %inicio:salto:fin

% Vector de amplitudes:

y = sin(2\*pi\*f\*tn);

% La onda rectificada completa es obtenida al invertir los semiciclos negativos por medio del comando abs:

y\_abs=**abs**(y);

% Verificación visual:

figure % Abre una nueva ventana de gráfico.

Stem(tn,y\_abs) % Grafica el Eje del Tiempo vs. Eje de Amplitudes.

Grid on % Agrega las divisiones menores en el fondo del grafico.

Xlabel(‘Tiempo(seg)’) % Titulo del Eje X

ylabel(‘Amplitud’) % Titulo del Eje Y

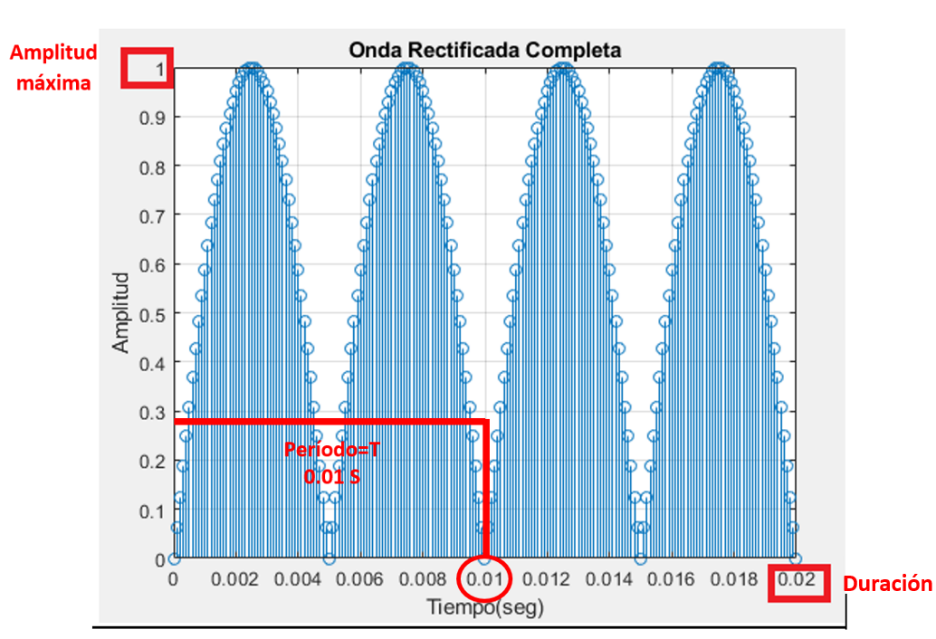
title(‘Onda Rectificada Completa’) % Titulo del Grafíco

% Verificación auditiva:

Fs = 1/Ts;

sound (y\_abs, Fs)

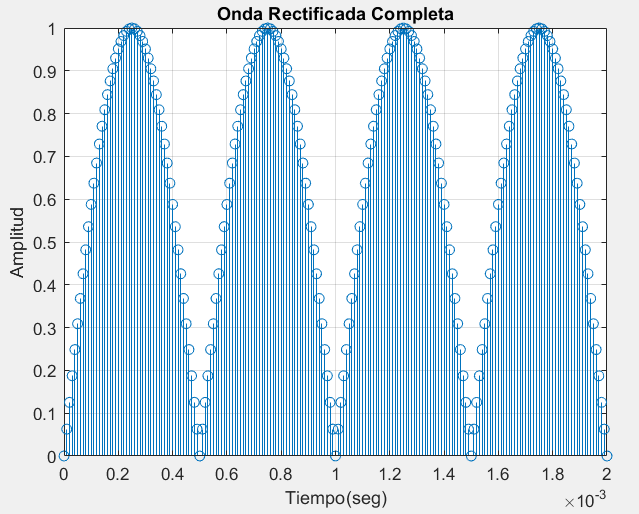
El resultado lo vemos en la Figura 1.2.11.

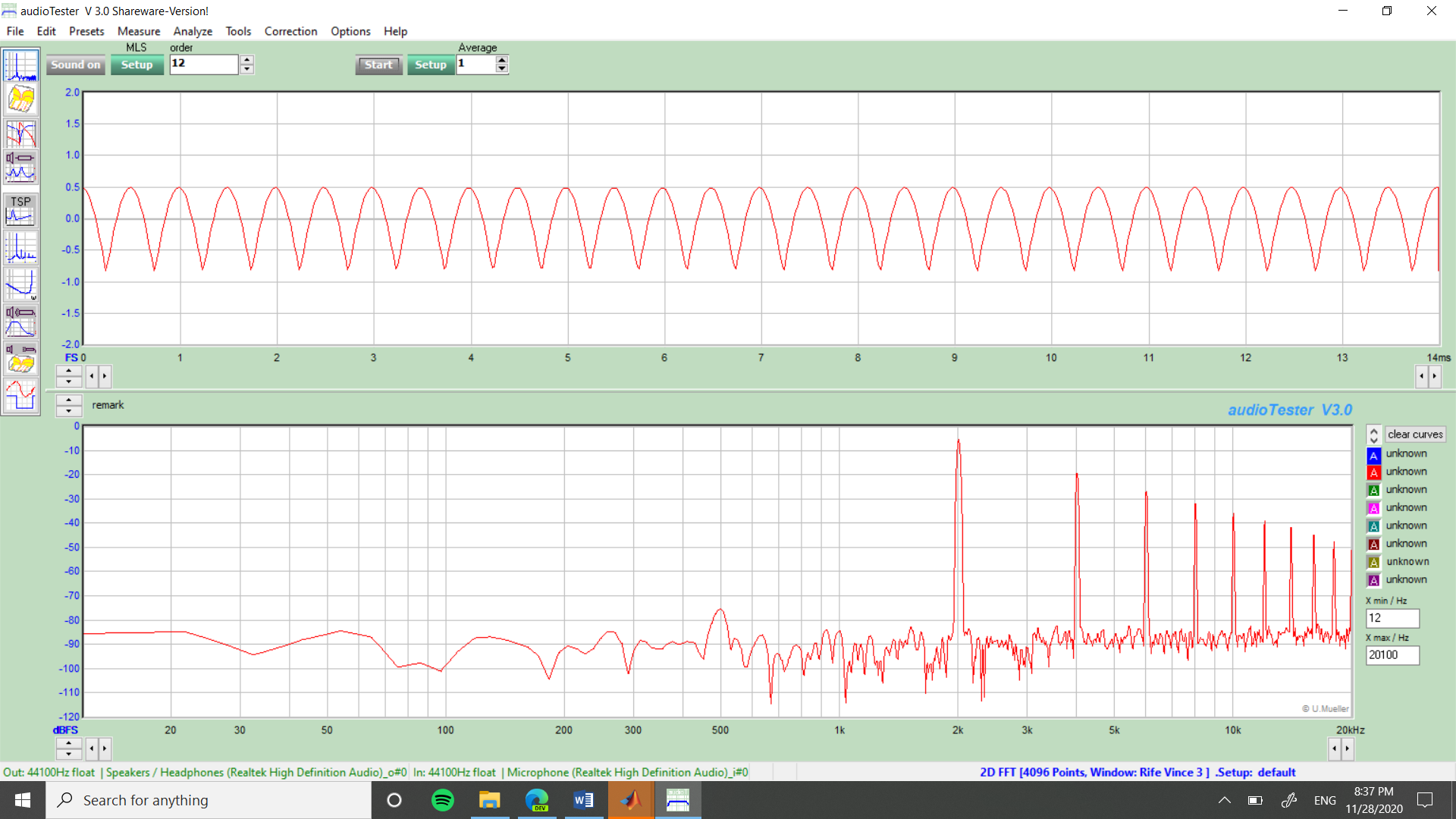


**Figura 1.2.11** Onda rectificada completa a una frecuencia de 100 Hz (Período=0.01 s), generada en Matlab.

DEBER: Resultado obtenido en el audio tester colocando el período que demuestra la frecuencia solicitada pero con una amplitud pico real de 0.5 Voltios,,, 1KHz

….





**Figura 1.2.12 Resultado obtenido de rectificada completa a 1kHz**

**B. ONDA RECTIFICADA DE ONDA COMPLETA TRUNCADA HASTA 0.7**

Para truncar la onda rectificada completa, necesitamos encontrar todos los valores mayores a 0.7 y una vez encontradas estas posiciones, mediante un lazo **for**, las cambiamos por la constante 0.7:

clc, clear all, close all

duracion=0.02;

f=100;

T=1/f;

% tiempo entre muestras (consideramos 100 muestras/período):

Ts=T/100;

tn=0:Ts:duracion; %inicio:salto:fin

% Vector de amplitudes:

y = sin(2\*pi\*f\*tn);

y\_abs=abs(y); % Obtiene la onda rectificada deseada

y1=y\_abs; % Vector auxiliar

% Se encuentra las posiciones de los valores mayores a 0.7 y mediante un lazo for, las cambiamos por la constante 0.7

j=1;

for i=tn

if y1(j)>0.7;

y1(j)=0.7;

end

j=j+1;

end

% Verificacion visual:

stem(tn,y1) % Grafica el Eje del Tiempo vs. Eje de Amplitudes.

xlabel('Tiempo(seg)') % Titulo del Eje X

ylabel('Amplitud') % Titulo del Eje Y

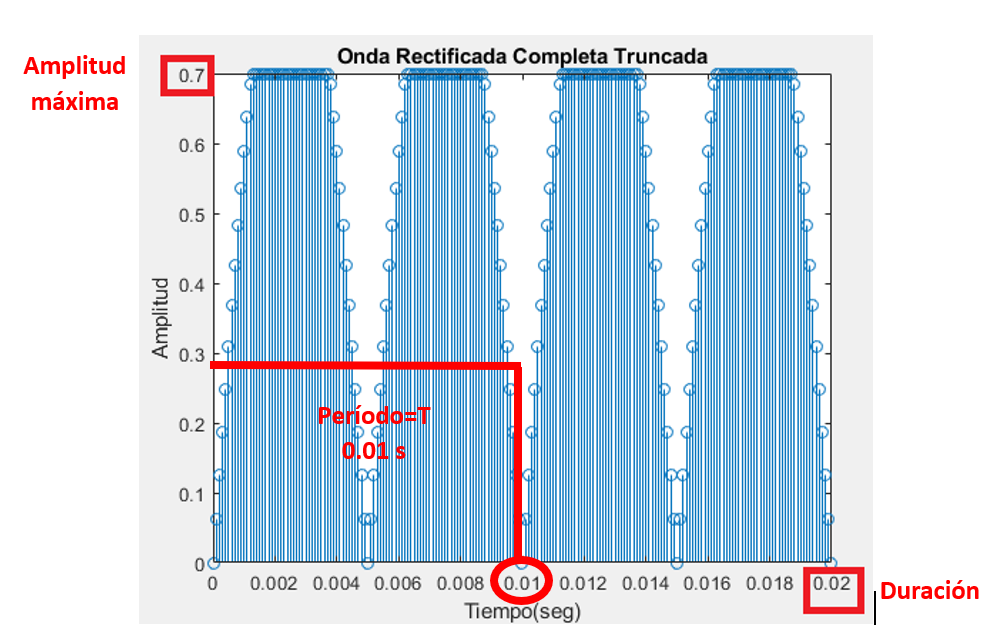
title('Onda Rectificada Completa Truncada') % Titulo del Gráfico

% Verificación auditiva:

Fs = 1/Ts;

sound (y\_abs, Fs)

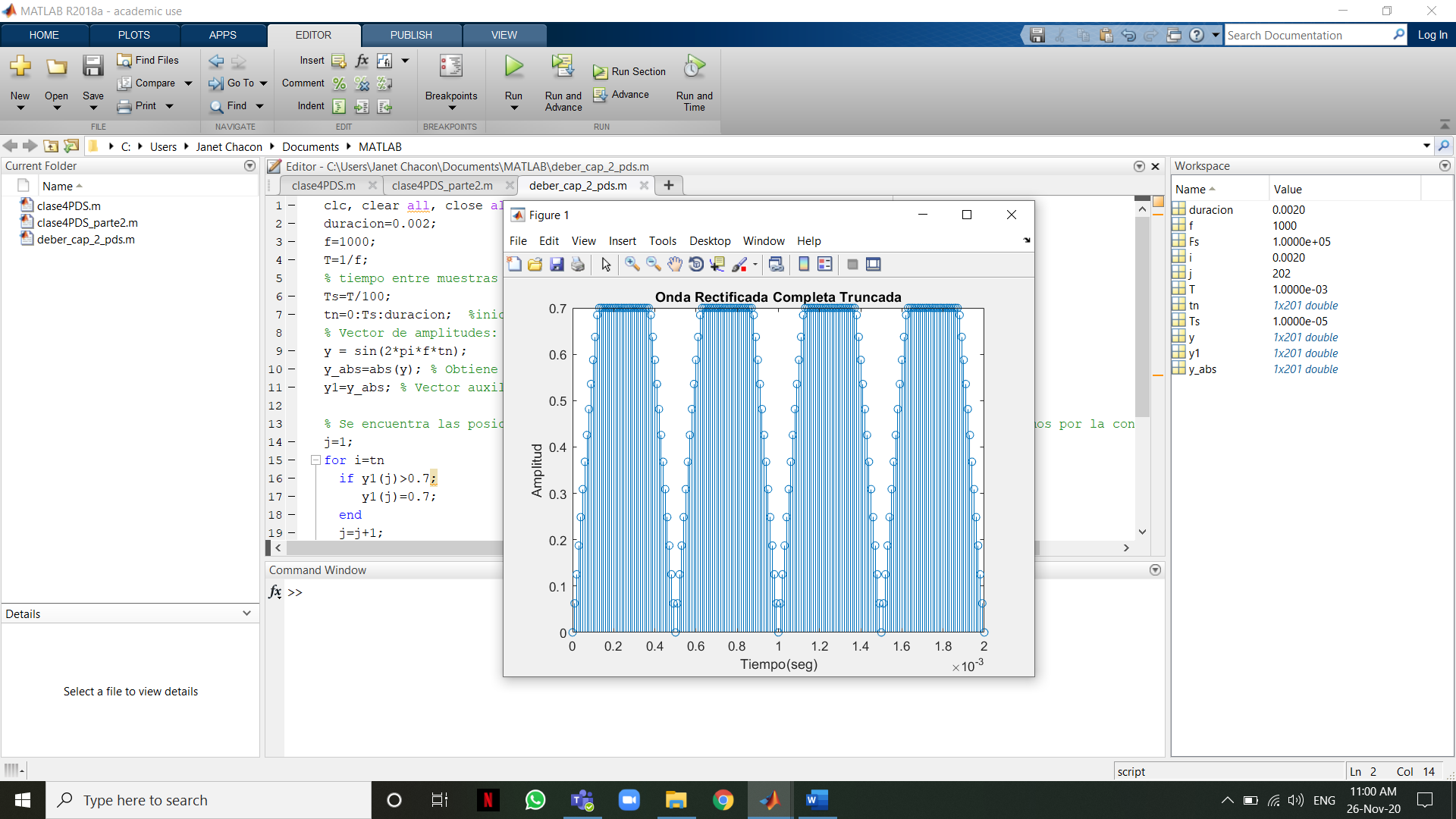
El resultado lo vemos en la Figura 1.2.13.



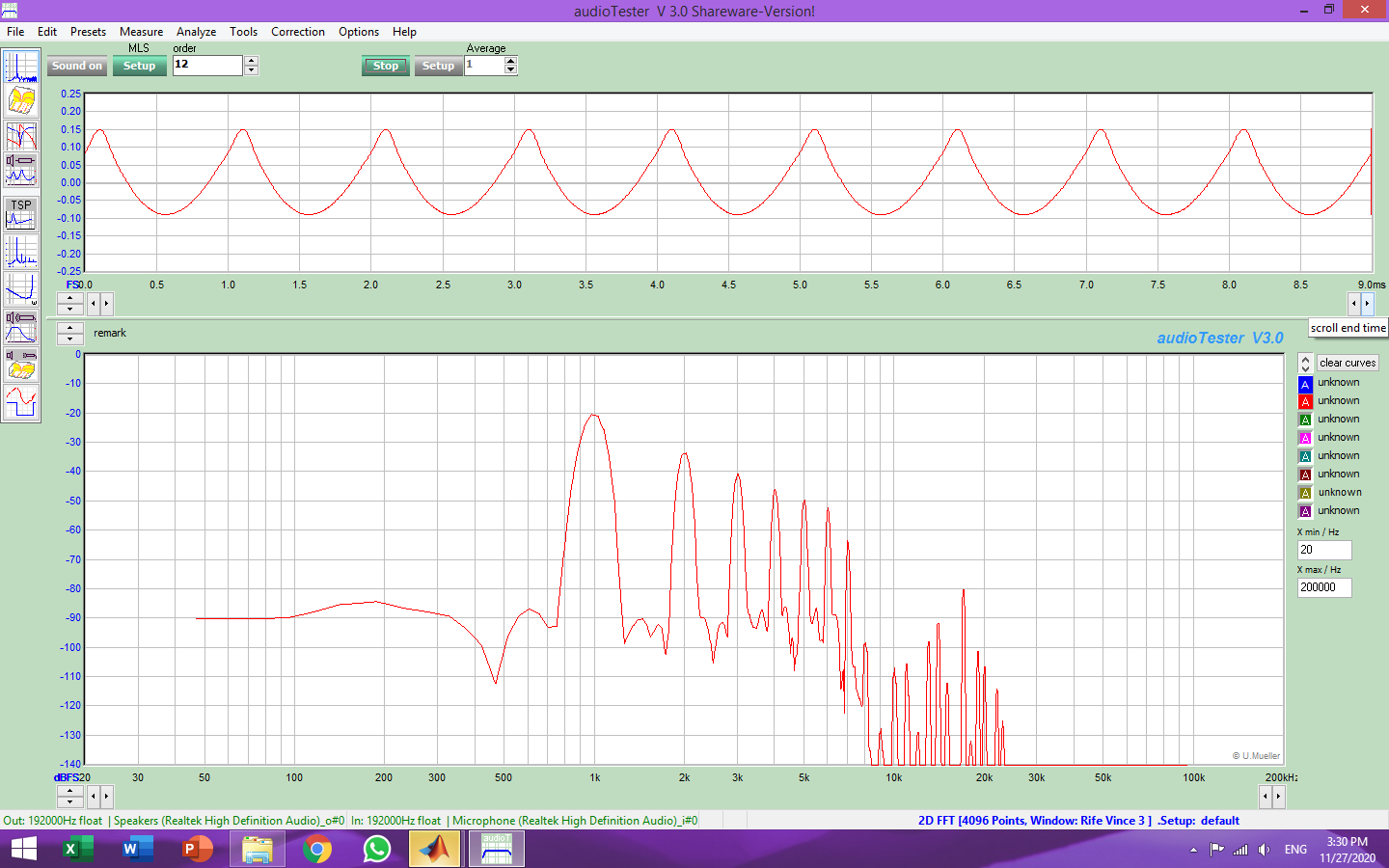
**Figura 1.2.13** Onda rectificada completa truncada de F = 100 Hz (Período=0.01) ..1KHz

DEBER: Resultado obtenido en el audio tester colocando el período que demuestra la frecuencia solicitada y la amplitud solicitada de 0.7 voltios pico real

….



**Figura 1.2.14** Onda rectificada completa truncada de F = 1000 Hz (Período=0.001)



**Figura** Onda rectificada completa truncada de F = 1000 Hz (Período=0.001), obtenida en el AudioTester

Análisis armónicos: Podemos observar que el armónico principal se encuentra en 1kHz y que esta señal cuenta con todos los armónicos.

**C. RECTIFICADOR DE MEDIA ONDA**

Un rectificador de media onda puede ser obtenido remplazando la parte negativa de una señal sinusoidal con ceros:

clc, clear all, close all

duracion=0.02;

f=100;

T=1/f;

% tiempo entre muestras (100 muestras/período)

Ts=T/100;

tn=0:Ts:duracion; %inicio:salto:fin

y1 = sin(2\*pi\*f\*tn);

% Un rectificador de media onda puede ser obtenido remplazando

% la parte negativa de una señal sinusoidal con ceros:

y2 = y1;

j=1;

for i=tn

if y1(j)<0;

y2(j)=0;

end

j=j+1;

end

stem(tn,y2);

xlabel('Tiempo(seg)') % Titulo del Eje X

ylabel('Amplitud') % Titulo del Eje Y

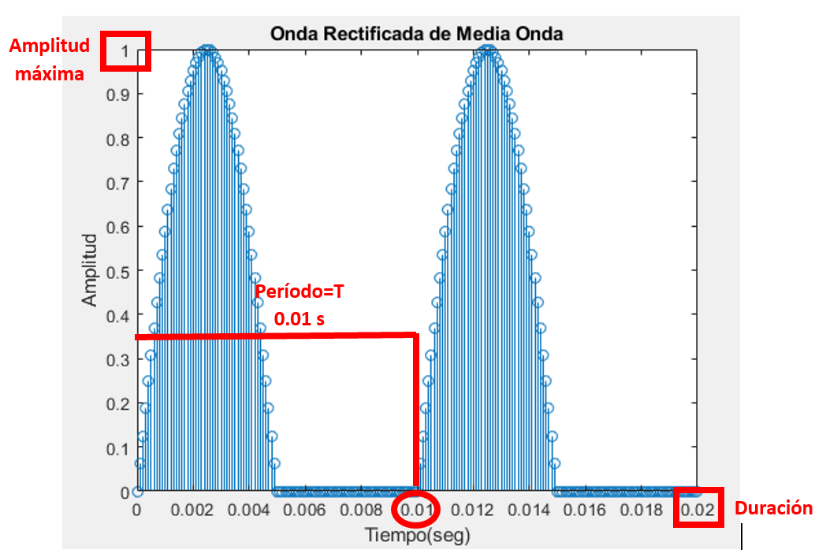
title('Onda Rectificada de Media Onda') % Titulo del Gráfico

% Verificación auditiva:

Fs = 1/Ts;

sound (y2, Fs)

El resultado lo vemos en la Figura 1.2.15.

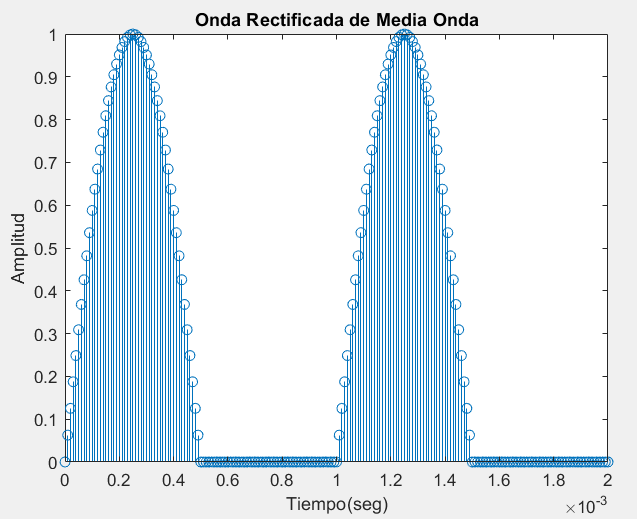


**Figura 1.2.15** Onda rectificada e media onda generada en Matlab a una frecuencia de 100 Hz (Período=0.01 s).

DEBER: Resultado obtenido en el audio tester colocando el período que demuestra la frecuencia solicitada pero con una amplitud real de 0.5 voltios pico. 1KHz

….

ANALISIS DE ARMÓNICOS





**Figura 1.2.16 Resultado obtenido del rectificador de media onda a 1kHz**

**Análisis de armónicos: En este caso solo tiene los armónicos pares es decir en 2kHz, 4Khz, etc además del armónico principal en 1kHz.**

**D. GENERAR UNA ONDA HÍBRIDA (SINUSOIDAL – TRIANGULAR) COMO SE INDICA EN LA FIGURA 1.2.17, DE FRECUENCIA 100 HZ.**



**Figura 1.2.17** Onda híbrida sinusoidal – triangular a ser generada en Matlab

Antes de generar nuestra onda híbrida, analicemos cómo se generan individualmente cada una de las señales que vamos a usar. Como vemos en la Figura 1.2.17, la primera parte (onda sinusoidal) no debe ser alterada mientras que la segunda parte (la onda triangular), como solo es monopolar (solo positiva), se debería multiplicar por 0.5 y añadirla un nivel de DC de 0.5. Veámoslo en el siguiente programa:

**RESOLUCIÓN:**

clc, close all, clear all

duracion=0.02;

f=100;

T=1/f;

Ts=T/100; % tiempo entre muestras

tn=0:Ts:duracion; %inicio:salto:fin

y1=sin(2\*pi\*f\*tn);

y2=0.5\*sawtooth(2\*pi\*f\*tn,0.5)+0.5;

subplot(3,1,1)

plot(tn,y1, 'b')

grid minor

subplot(3,1,2)

plot(tn,y2, 'r')

grid minor

subplot(3,1,3)

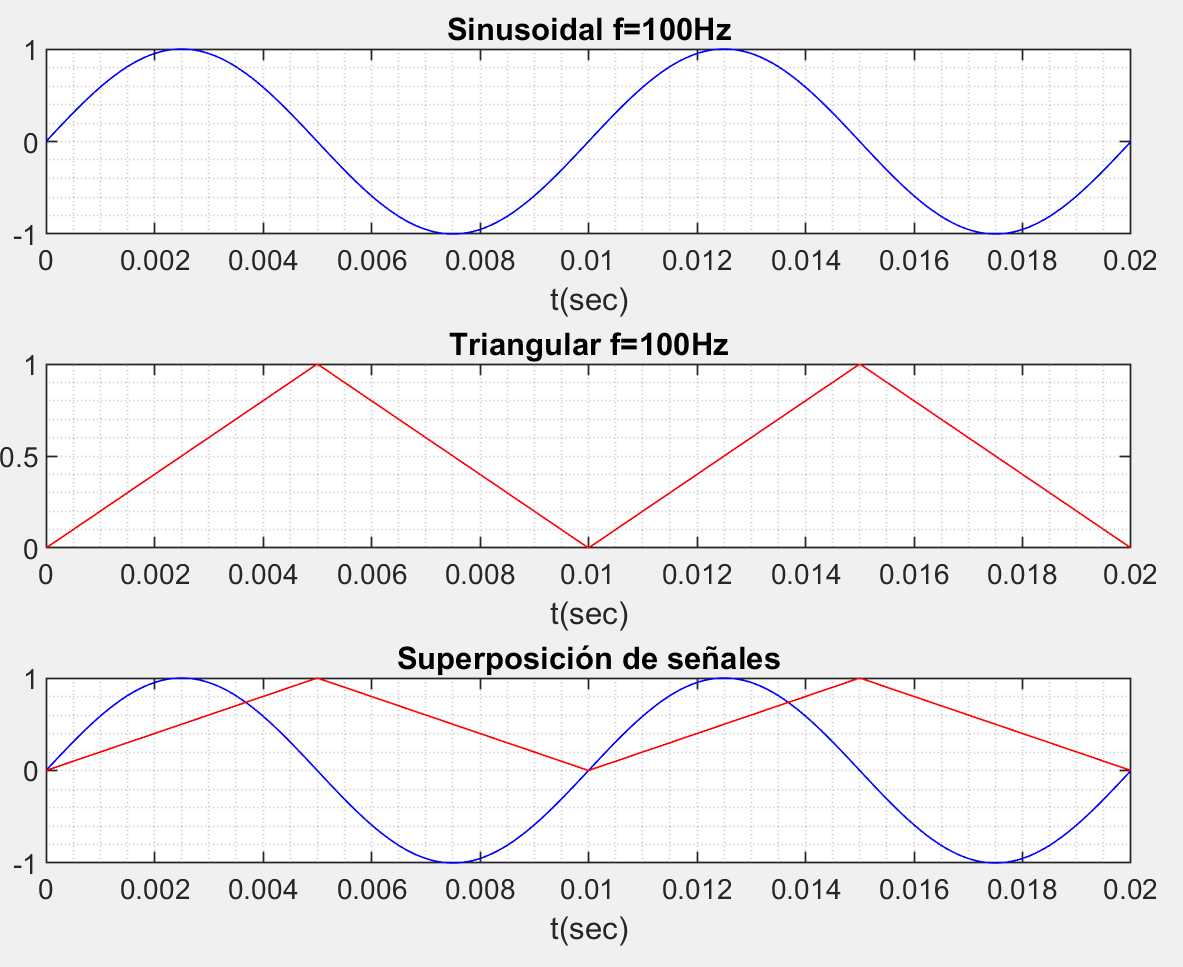
plot(tn,y1,'b')

hold on

plot(tn,y2, 'r')

grid minor

El resultado lo vemos en la Figura 1.2.18 donde se puede observar que cuandosobreponemos ambas gráficas,la onda triangular y sinusoidal tienen la misma frecuencia de 100 Hz; sin embargo, si analizamos lo que aparecería en cada mitad del periodo podemos ver que de 0 a pi/2 la parte positiva de la señal sinusoidal saldría muy bien, pero desde pi/2 a pi sale únicamente la parte de la pendiente de bajada de la onda triangular. Por lo tanto, se debe corregir el encuadre de la triangular para que se ubique según lo deseado.



**Amplitud máxima**

**Período=T**

**0.01 s**

**Figura 1.2.18.** Ondas sinusoidal y triangular empleadas para la formaciónla onda híbrida.

Esto podemos hacerlo aumentando la frecuencia de la onda triangular al doble (Figura **1.2.19**).

% Para encuadrar la triangular de modo que se ubique según lo deseado: hay que

% aumentar al doble de frecuencia:

y2\_2f=0.5\*sawtooth(2\*pi\*2\*f\*tn,0.5)+0.5;

figure

subplot(3,1,1)

plot(tn,y1, 'b')

grid minor

subplot(3,1,2)

plot(tn,y2\_2f, 'r')

grid minor

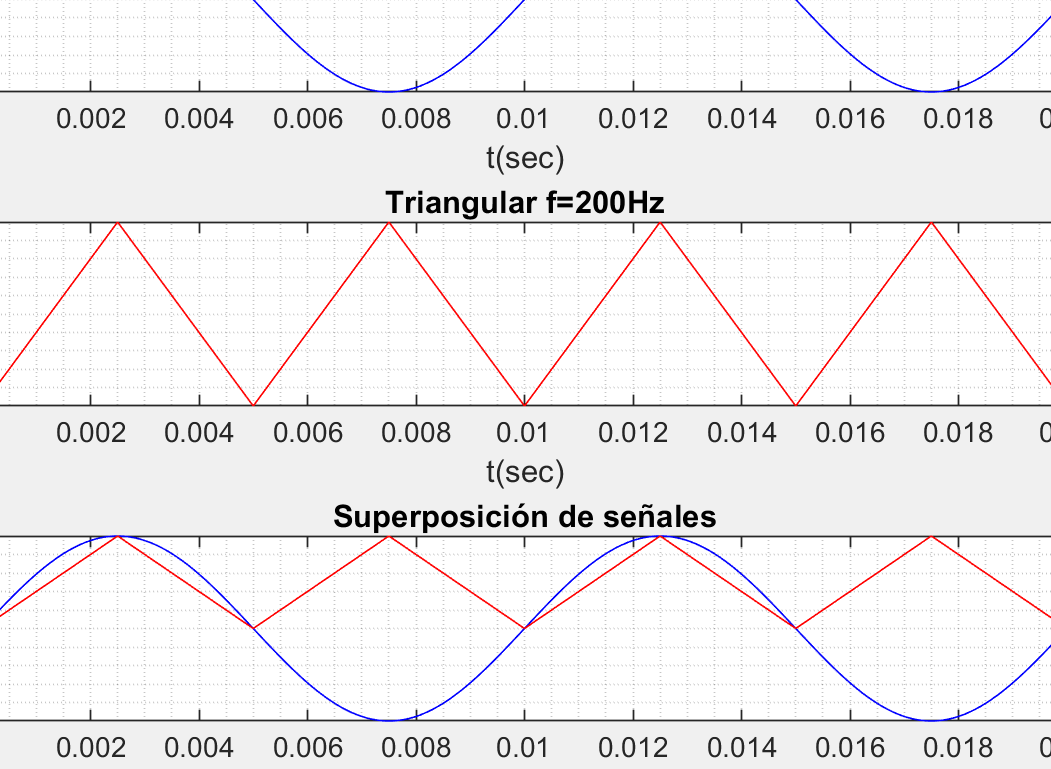
subplot(3,1,3)

plot(tn,y1, 'b')

hold on

plot(tn,y2\_2f, 'r')

grid minor



**Figura 1.2.19** Onda triangular aumentada al doble de frecuencia para la formaciónuna onda híbrida de 100 Hz

Como se puede observar, se logró que en la segunda mitad del período la onda triangular se posicione correctamente. Ahora, solo se debe hacer que la **parte negativa de la sinusoidal** **sea sustituida por la triangular.**

% Ahora, si la sinusoidal es negativa, hay que poner la triangular:

y\_final = y1;

for n=1:length(tn)

if y1(n)<=0

y\_final(n)=y2\_2f(n);

end

end

figure

subplot(3,1,1)

plot(tn,y1, 'b')

title('Sinusoidal')

xlabel('t(sec)')

grid minor

subplot(3,1,2)

plot(tn,y2\_2f, 'r')

title('triangular con duplicada su frecuencia')

xlabel('t(sec)')

grid minor

subplot(3,1,3)

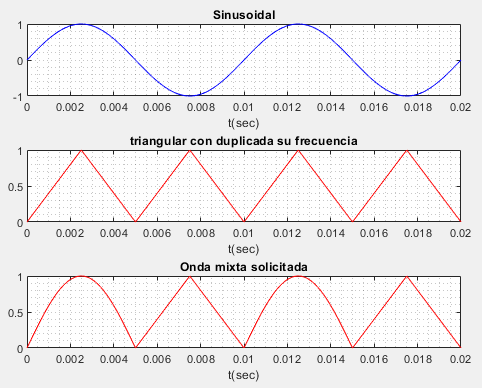
plot(tn,y\_final, 'r')

title('Onda mixta solicitada')

xlabel('t(sec)')

grid minor

El resultado lo vemos en la Figura 1.2.20 donde se muestra que el resultado ya es el deseado, la señal mixta o híbrida es solo **positiva** y su frecuencia es de 0.01 segundos correspondiente a la **Frecuencia** esperada de **100 Hz**. En este punto, el usuario puede poner la **duración** que desee.



**Amplitud máxima**

**Período =T = 0.01 s**

**Período=2T**

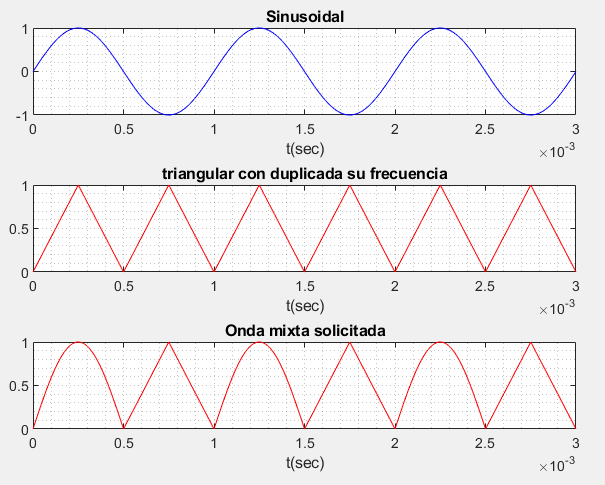
**Período = T = 0.01 s**

**Figura 1.2.20** Onda mixta (sinusoidal–triangular) de frecuencia de 100 Hz (Período=0.01 s), generada en Matlab.

DEBER: Resultado obtenido en el audio tester colocando el período que demuestra la frecuencia solicitada pero con una amplitud real de 0.5 voltios pico. 1KHz

….

ANALISIS DE ARMÓNICOS





**Figura 1.2.21Resultado obtenido de la onda mixta solicitada a 1kHz**

**En este caso se tienen todos los armónicos es decir pares e impares debido a la combinación de estas dos señales. Los armónicos impares son de menor magnitud que los armónicos pares.**

**E. GENERACIÓN DE UNA ONDA HÍBRIDA (TRIANGULAR CON PORCENTAJE DE SUBIDA 0.8 - SINUSOIDAL) SIMILAR A LA DE LA FIGURA 1.2.17 PERO DE FRECUENCIA 1 KHz Y ANALIZAR EL COMPORTAMIENTO DE SUS ARMÓNICOS MEDIANTE AUDIO TESTER**

Realizando el mismo procedimiento empleado en la onda híbrida anterior, tenemos el siguiente código:

clc, close all, clear all

duracion=0.02;

f=100;

T=1/f;

Ts=T/100; % tiempo entre muestras

tn=0:Ts:duracion; %inicio:salto:fin

y1=0.5\*sawtooth(2\*pi\*f\*tn,0.8)+0.5;

y2=sin(2\*pi\*f\*tn+pi); %senoidal desplazada 180 grados

subplot(3,1,1)

plot(tn,y1, 'b')

title('title triangular 0.8 pendiende de subida f=100Hz')

xlabel('t(sec)')

grid minor

subplot(3,1,2)

plot(tn,y2, 'r')

title('Sinusoidal f=100Hz')

xlabel('t(sec)')

grid minor

subplot(3,1,3)

plot(tn,y1, 'b')

hold on

plot(tn,y2, 'r')

title('Superposición de señales')

xlabel('t(sec)')

grid minor

% Para encuadrar la triangular de modo que se ubique según lo deseado: hay que

% aumentar al doble de frecuencia:

y1\_f=0.5\*sawtooth(2\*pi\*2\*f\*tn,0.8)+0.5;

figure

subplot(3,1,1)

plot(tn,y1\_f, 'b')

title(' triangular 0.8 pendiende de subida duplicada su frecuencia')

xlabel('t(sec)')

grid minor

subplot(3,1,2)

plot(tn,y2, 'r')

title('Sinusoidal f=100Hz')

xlabel('t(sec)')

grid minor

subplot(3,1,3)

plot(tn,y1\_f, 'b')

hold on

plot(tn,y2,'r')

title('Superposición de señales')

xlabel('t(sec)')

grid minor

% Ahora, si la sinusoidal es negativa, hay que poner la triangular:

y\_final = y2;

for n=1:length(tn)

if y2(n)<=0

y\_final(n)=y1\_f(n);

end

end

figure

subplot(3,1,1)

plot(tn,y1\_f, 'b')

title('triangular 0.8 pendiende de subida duplicada su frecuencia')

xlabel('t(sec)')

grid minor

subplot(3,1,2)

plot(tn,y2, 'r')

title('Sinusoidal')

xlabel('t(sec)')

grid minor

subplot(3,1,3)

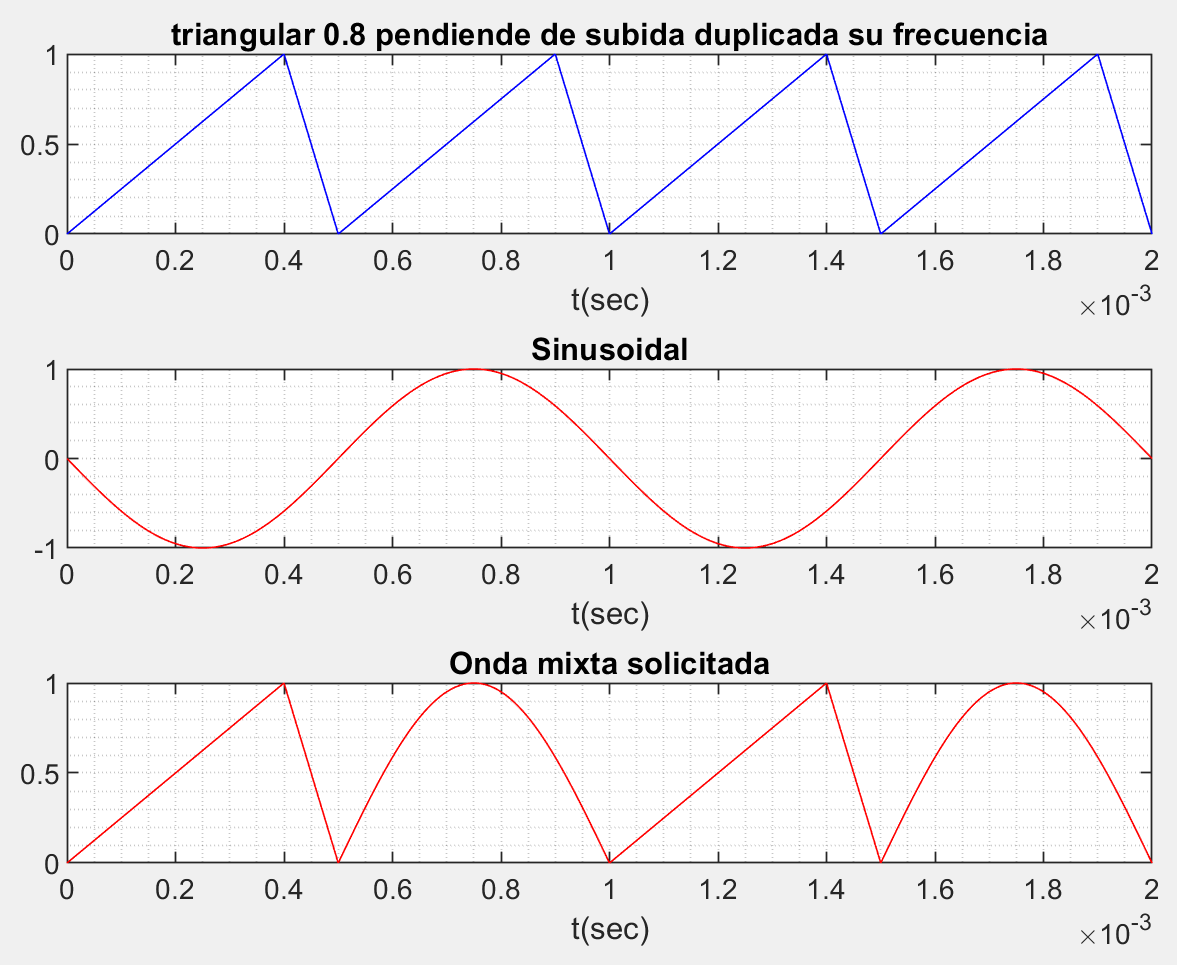
plot(tn,y\_final, 'r')

title('Onda mixta solicitada')

xlabel('t(sec)')

grid minor

El resultado lo vemos en la Figura 1.2.22 donde se muestra que el resultado ya es el deseado, la señal mixta o híbrida es solo **positiva** y su frecuencia es de 0.001 segundos correspondiente a la **Frecuencia** esperada de **1000 Hz**. En este punto, el usuario puede poner la **duración** que desee.



**Amplitud máxima**

**Período onda mixta=T**

**0.001 s**

**Período=T = 0.001 s**

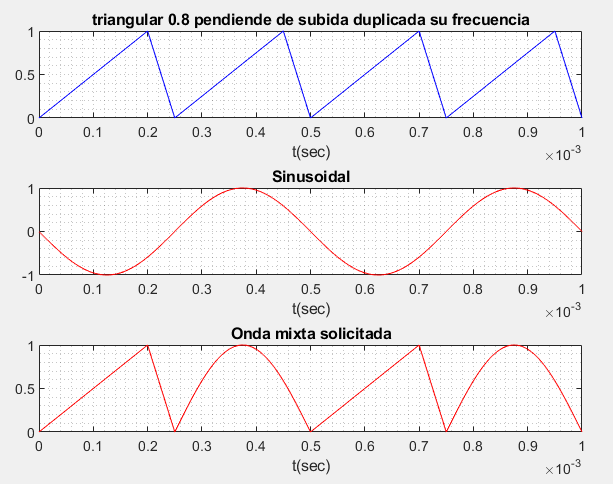
**Período=2T**

**Phy=180°**

**Figura 1.2.22.** Onda mixta (triangular-sinusoidal) de frecuencia de 1000 Hz (Período=0.001 s), generada en Matlab.

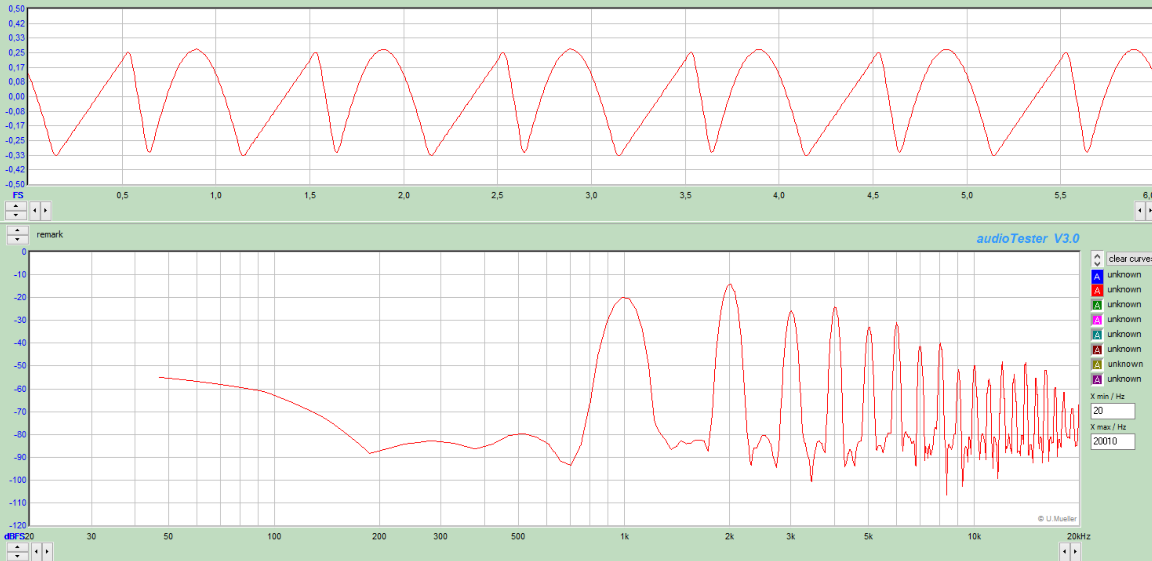
DEBER: Confirme el resultado obtenido en el audio tester (en la Figura 1.2.23) colocando el período que demuestra la frecuencia solicitada pero con una amplitud real de 0.5 voltios pico. 2KHz

ANALISIS DE ARMÓNICOS





**Comprobación en audiotester a 2kHz**



**Figura 1.2.23.** Onda mixta (triangular-sinusoidal) analizada mediante audio tester.

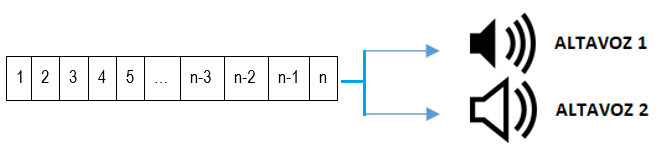
|  |
| --- |
| CONFIRME LO SIGUIENTE: Según la Figura 1.2.23, la señal híbrida (triangular-sinusoidal) construida tiene su frecuencia fundamental en 1Khz tal como se esperaba y además posee todos los armónicos (pares e impares). Un detalle muy importante es que los armónicos impares tienen picos menores a los pares y que su primer armónico en 2 KHz es mayor a la componente fundamental. Este hecho debería ser confirmado por el lector mediante un analizador espectral real. |

Análisis: Esto se debe a que las señales suman su frecuencia fundamental de 1kHz provocando que se genere ese fenómeno donde el armónico en 2kHz sea mayor que el pico de la señal.

**1.2.6 USO DE LOS DOS CANALES DE SALIDA DEL PUERTO DE AUDIO**

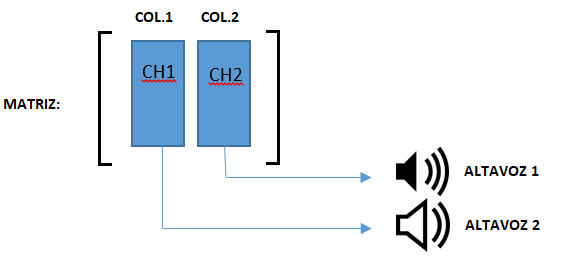
Ya que los humanos tenemos dos oídos para que nuestro cerebro pueda identificar el origen espacial de un sonido, para hacerlo más realista y obtener una sensación tridimensional, el puerto de audio tiene dos salidas que van a los dos altavoces. Entonces, se debe aprender a usar los dos canales de modo que cada uno genere una señal independiente.

Al usar las funciones ***sound***(yn,Fs) o ***audioplayer***, si **yn** es un vector, como en los ejemplos anteriores, este irá automáticamente hacia ambos canales, es decir, hacia los dos altavoces (Figura 1.2.24).



**Figura 1.2.24.** Si **yn** es un vector, la función sound envía la señal hacia los dos altavoces de forma automática.

Si se desea enviar por el canal izquierdo una seña de frecuencia F1 y por el canal derecho otra señal de frecuencia F2, primeramente se debería construir ambos vectores de amplitudes de forma independiente y la única consideración a tomar en cuenta es que se tiene un **único vector de tiempos** que se lo conformaría considerando **la frecuencia mayor o más exigente:** si este vector de tiempos funciona para la frecuencia más exigente, con mayor razón lo hará para la inferior. Con estos dos vectores de amplitudes debe conformar una matriz donde el canal izquierdo sería la columna izquierda y el canal derecho la columna derecha (Figura 1.2.25), de esta manera cada columna o señal iría a su respectivo altavoz.



**Figura 1.2.25.** Si **yn** es una matriz de dos columnas, cada columna va hacia un altavoz diferente.

Veamos el siguiente ejemplo:

**Ejemplo: por el canal 1 genere una señal sinusoidal (de 500 Hz) y por el canal 2 genera otra señal sinusoidal de frecuencia 1000 Hz, la duración puede ser de la que desee el usuario. Realice la verificación de las 2 ondas generadas empleando el instrumento virtual audio tester**.

Siguiendo los pasos anteriores y considerando una duración pequeña para evitar realizar una amplificación para visualización, la generación de las **señales independientes** quedaría de la siguiente manera:

%% Programa para manejar los 2 canales de salida del puerto de audio de manera independiente

clc,close all,clear all

Amax=1;%Amplitud de las señales

F1=500; %Frecuencia 1

F2=1000; %Frecuencia 2

d=0.003; %Duracion en segundos

% Tiempo entre muetras conisderando la Frecuencia más exigente (F2):

T = 1/F2;

Ts= T/ 100; % Considerando 100 muestras por cada período.

tn1=[0:Ts:d]; %Vector de tiempo discreto

%Generación de señales sinusoides de diferente frecuencia

%Se multiplica 0.75, para que la señal no sobrepase la amplitud de +/-1 voltio.

yn1=Amax\*sin(2\*pi\*F1\*tn1);

yn2=Amax\*sin(2\*pi\*F2\*tn1);

%Ploteo de la señal a salir por el puerto de audio

plot(tn1,yn1,tn1,yn2);

legend('Canal izquierdo: sinusoide de 450Hz ','Canal derecho: sinusoide de 1750Hz')

xlabel('Tiempo (s)');

ylabel('Amplitud');

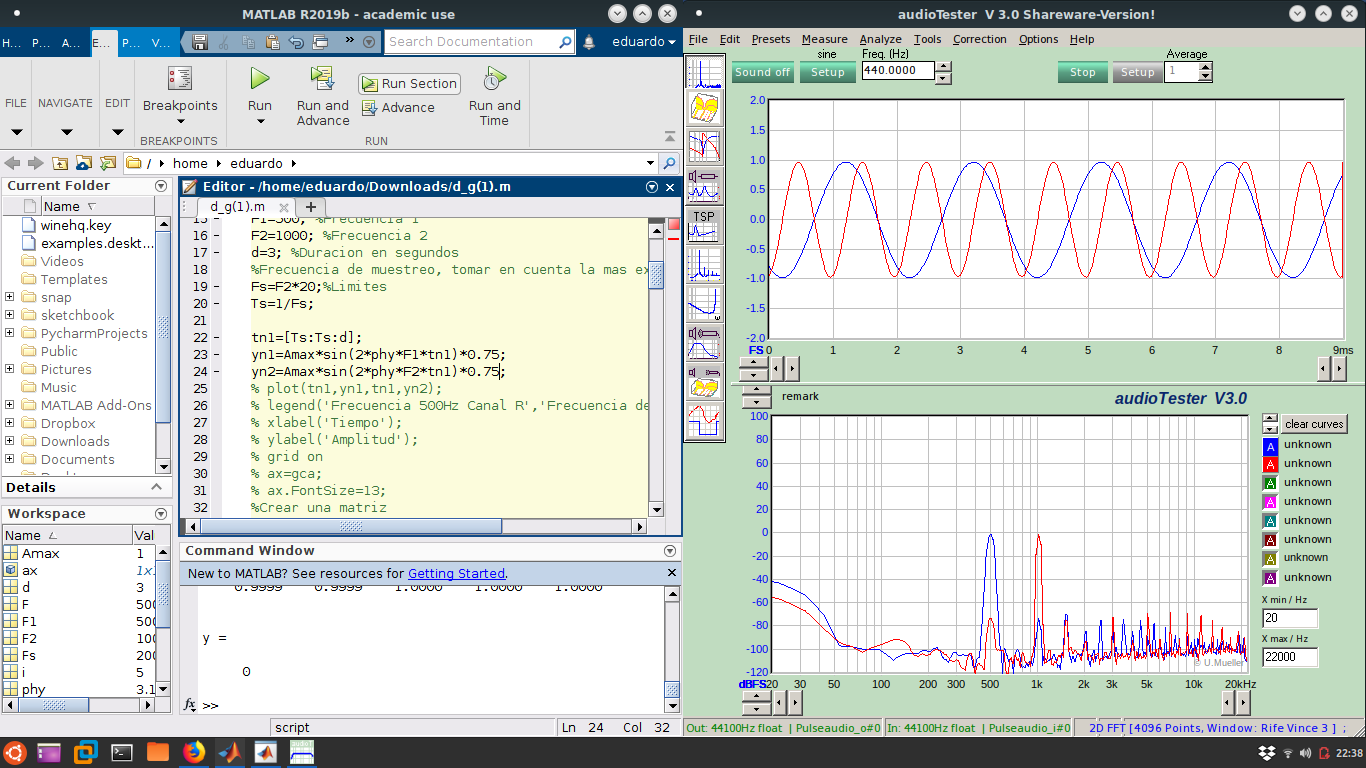
grid on

%Crear la matriz: Columna 1 corresponde al canal 1 y Columna 2 corresponde al canal 2

yn=[yn1' yn2'];

sound(yn,Fs)

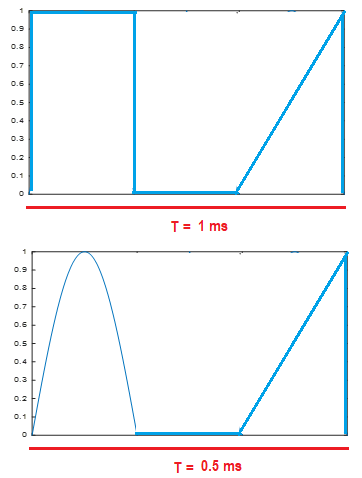
Una vez ejecutado el programa, en el audio tester se puede observar que las dos señales aparecen diferenciadas por colores tanto en el dominio de tiempo como de la frecuencia. Se observan los picos en las frecuencias de cada señal, en 500 Hz para la azul y en 1000 Hz en el caso de la señal roja (Figura 1.2.26). El aparecimiento de armónicos se puede impedir si el usuario baja el volumen.



**Figura 1.2.26.** Mediciones tomadas con el instrumento virtual, canal 1 (1000 Hz en rojo), canal 2 (500Hz en azul).

1KHz Y 2 KHz

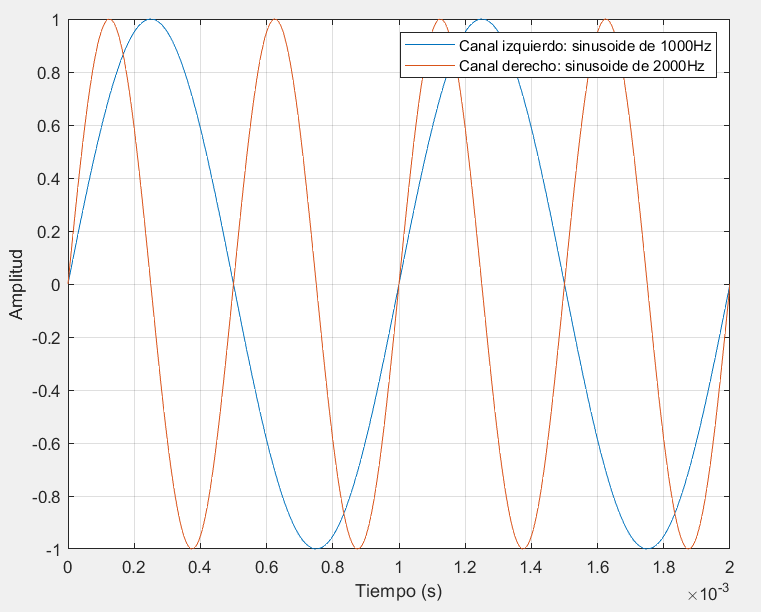
DEBER: Generar las siguientes dos ondas (Figura 1.2.27) de modo que la una vaya hacia el canal 1 y la otra hacia el canal 2 del puerto de audio y verifíquelo mediante audio tester. Explique cuál es el comportamiento de los armónicos obtenidos.



Canal 2

Canal 1

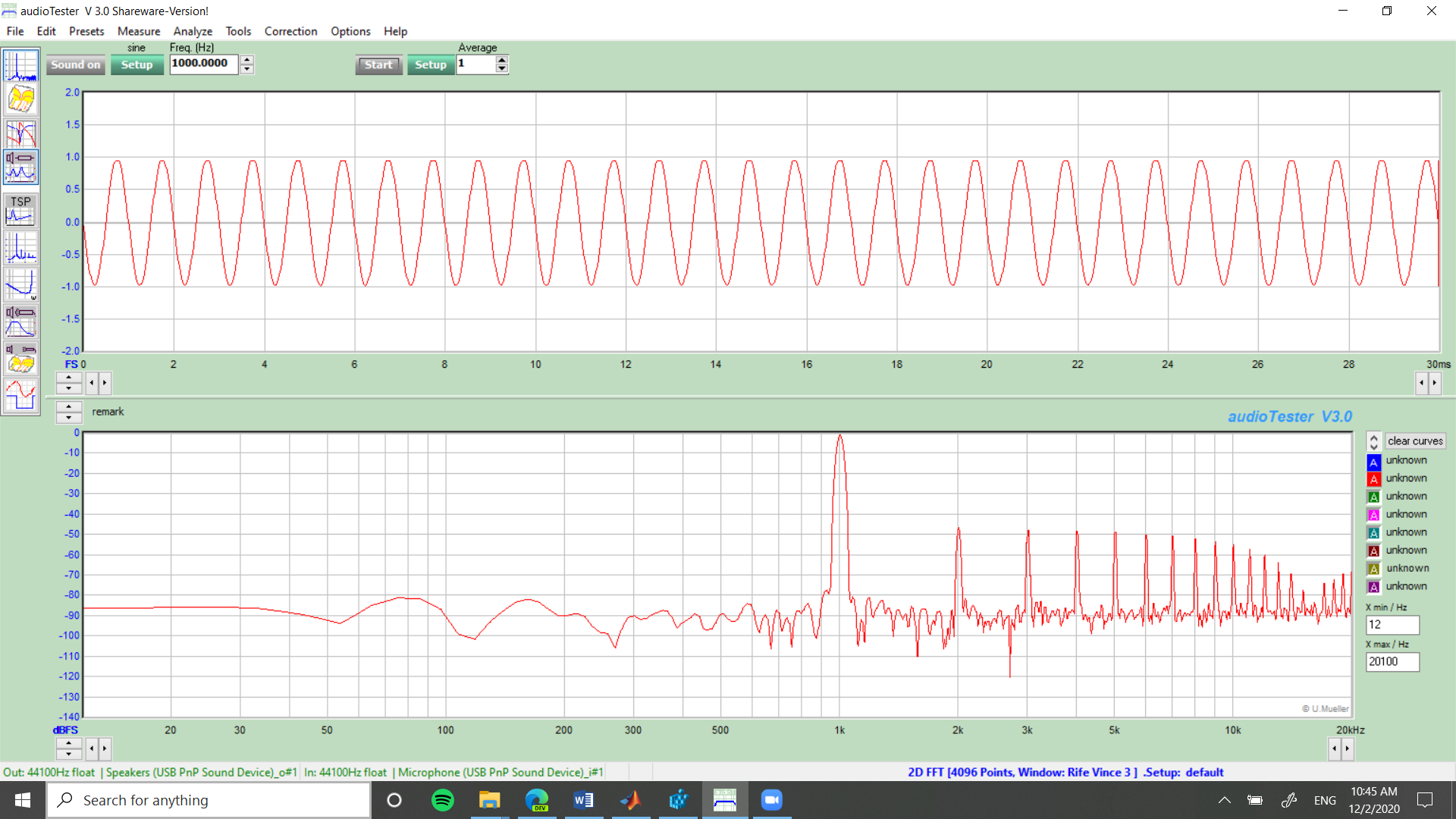
**Figura 1.2.27.** Ondas que deben salir hacia los canales del puerto de audio.



Debido a problemas con la tarjeta de audio las 2 señales sinusoidales se suman, es por esto que se tiene como resultado la siguiente forma de onda, donde se puede apreciar que el armónico principal es en 1[kHz] y los siguientes armónicos disminuyen en amplitud paulatinamente.

**Resultado obtenido en audiotester de las ondas en dos canales**

La primera onda que se dirige al canal izquierdo es una sinusoidal de frecuencia 1[kHz], a continuación se presenta el resultado obtenido en audiotester de dicha onda, donde nuevamente se aprecia que su armonio principal se encuentra en 1[kHz] y se tiene tanto armónicos impares como pares los cuales disminuyen en amplitud.



De igual manera, se presenta la segunda onda sinusoidal en la cual su amplitud se encuentra en el rango de 1 a -1 [V] al momento de hacer uso de audiotester. Su armónico principal se encuentra en 2[kHz] y el resto de armónicos secundarios son solo armónicos pares y disminuyen paulatinamente.

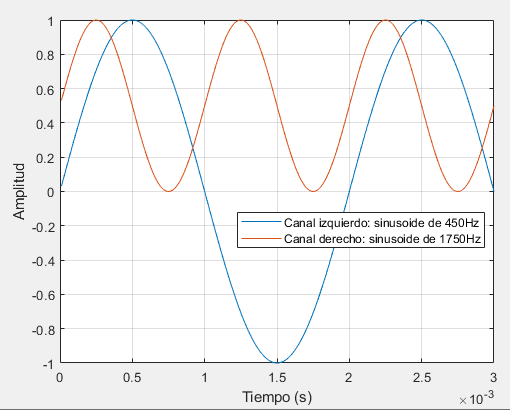


**1.2.7 SENSIBILIDAD DEL PUERTO DE AUDIO A COMPONENTES DC (ACOPLAMIENTO EN AC).**

Queremos saber si el puerto de audio sensible a valores de componentes DC, es decir, si a una de las señales del ejemplo anterior añadimos una componente DC, esta aparece o no en la señal final. Para investigarlo, a una de las señales del apartado anterior se le reduce su amplitud a la mitad y se le suma una componente DC de 0.5. El resto del programa queda sin alteraciones:

|  |
| --- |
| yn1=Amax\*sin(2\*phy\*F1\*tn); % queda sin cambios  % A yn2 se añade una componente DC de 0.5 V  yn2=(Amax/2)\*sin(2\*phy\*F2\*tn)+0.5; |

El resultado obtenido en Matlab se lo puede ver en la Figura 1.2.28.



**Figura 1.2.27** Señales de distinta frecuencia, una de ellas con componente DC generadas a través de Matlab.

Al observar en el audio tester tanto en el dominio del tiempo como de la frecuencia, el acoplamiento en AC se comprueba ya que la componente CD de la señal modificada ha sido eliminada (Figura 1.2.29).



**Figura 1.2.28.** La componente CD añadida a una de la señales ha sido eliminada por lo que el puerto de audio está acoplado en AC. 1 Y 2 KHz… CUIDAR DE NO ACERCARSE A +/- 1 V PARA EVITAR ARMONICOS POR CORTE

DEBER: realice un **generador secuencial** de modo que primero se genere una onda que contenga una frecuencia fundamental 1 KHz y que solo su contenga armónicos pares, esta debe durar 3 segundos. A continuación genere otra onda que contenga una frecuencia fundamental 2 KHz y que solo su contenga armónicos impares, esta debe durar 2 segundos. Finalmente, genere otra onda que contenga una frecuencia fundamental 3 KHz y que contenga todos los armónicos, esta debe durar 3 segundos. Luego debe repetirse indefinidamente hasta que el usuario decida pararlo. Realice un video de su ejecución que súbalo como parte del deber.

Todo el procedimiento debe estar debidamente explicado y comentado el código, además debe realizar una discusión de los resultados obtenidos. Siempre nombre las Figuras y explíquelas tal como se lo hace en este documento.

SUBIR AL TEAMS SIN COMPRIMIR, 1 SOLO VIDEO POR GRUPO (SOLO UN INTEGRANTE)

function varargout = Generador(varargin)

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @Generador\_OpeningFcn, ...

'gui\_OutputFcn', @Generador\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

function Generador\_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);

function varargout = Generador\_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

function pushbutton1\_Callback(hObject, eventdata, handles)

closereq();

function pushbutton2\_Callback(hObject, eventdata, handles)

Amax=1; %Amplitud de la señal

Fo=1000; %Frecuencia 1

T=1/Fo; % Periodo

dur=3; % Duracion de 3 seg

Ts=T/100; % Muestreo

fs=1/Ts; % Frecuencia de Muestreo

tiempo=0:Ts:dur; %Tiempo de Muestreo

Fx1=Amax\*sin(2\*pi\*Fo\*tiempo); %Funcion Armonicos Pares

for i=1:length(Fx1)

if Fx1(i)<=0

Fx1(i)=-Fx1(i); %Correccion de Onda

end

end

dur1=2; %Duracion 2 seg

tiempo1=0:Ts:dur1; %Tiempo de Muestreo para 2 seg

F2=2000; %Frecuencia 2

Fx2=Amax\*square(2\*pi\*F2\*tiempo1,50); %Funcion Armonicos Impares

F3=3000; %Frecuencia 3

Fx3=Amax\*sawtooth(2\*pi\*F3\*tiempo,100/100); %Funcion Armonicos Pares e Impares

Fx=[Fx1 Fx2 Fx3]; % Union de Señales

Fxt=[Fx Fx Fx Fx Fx Fx];% Ampliacion de Señales

FxT=[Fxt Fxt Fxt];

axes(handles.axes1); %Figura 1

plot(tiempo,Fx1);

title('Armonicos Pares');

xlabel('Tiempo');

ylabel('Voltaje');

axis([0 0.001 -0.2 1.2]);

grid minor;

axes(handles.axes2); %Figura 2

plot(tiempo1,Fx2);

title('Armonicos Impares');

xlabel('Tiempo');

ylabel('Voltaje');

axis([0 0.001 -1.2 1.2]);

grid minor;

axes(handles.axes3); %Figura 3

plot(tiempo,Fx3);

title('Todos los Armonicos');

xlabel('Tiempo');

ylabel('Voltaje');

axis([0 0.001 -1.2 1.2]);

grid minor;

sound(Fxt,fs); %Salida de audio