

Tarea 2

Alejandro José, Luncey Contreras, 202112396

Escuela de Mecánica Eléctrica, Facultad de Ingeniería, Universidad de San Carlos de Guatemala

En esta tarea se elaboró un filtro aplicado a una onda senoidal dentro del lenguaje científico de programación Octave, a través del uso de diversos elementos matemáticos y computacionales, se codificó el modelo y se presentó en forma gráfica. Adquiriendo así, conocimientos sobre el lenguaje Octave y la matemática involucrada.

I. CÓDIGO

```
fs=1000;
t=0:1/fs:1;
f=100;
x=sin(2*pi*f*t);
xf=fft(x);
n=length(x);
fcutoff=50;
h=ones(n,1);
h(round(n*fcutoff/fs)+1:end)=0;
xf_filtered=ifft(xf_filtered);
figure;
subplot(2,1,1);
plot(t,x);
title('Señal original');
xlabel('Tiempo (s)');
ylabel('Amplitud');
subplot(2,1,2);
plot(t,real(x_filtered));
title('Señal filtrada');
xlabel('Tiempo (s)');
ylabel('Amplitud');
```

```
ale@ALEKS:~$ octave
GNU Octave, version 6.4.0
Copyright (C) 2021 The Octave Project Developers.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.
Octave was configured for "x86_64-pc-linux-gnu".
Additional information about Octave is available at https://www.octave.org.
Please contribute if you find this software useful.
For more information, visit https://www.octave.org/get-involved.html
Read https://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

octave:1> fs=1000;
octave:2> t=0:1/fs:1;
octave:3> f=100;
octave:4> x=sin(2*pi*f*t);
octave:5> xf=fft(x);
octave:6> n=length(x);
octave:7> fcutoff=50;
octave:8> h=ones(n,1);
octave:9> h(round(n*fcutoff/fs)+1:end)=0;
octave:10> xf_filtered=ifft(xf_filtered);
octave:11> x_filtered=ifft(xf_filtered);
octave:12> figure;
octave:13> subplot(2,1,1);
octave:14> plot(t,x);
octave:15> title('Señal original');
octave:16> xlabel('Tiempo (s)');
octave:17> ylabel('Amplitud');
octave:18> subplot(2,1,2);
octave:19> plot(t,real(x_filtered));
octave:20> title('Señal filtrada');
octave:21> xlabel('Tiempo (s)');
octave:22> ylabel('Amplitud');
```

Figura 1: Código en terminal, elaboración propia

El código fue ejecutado directamente en la terminal de Ubuntu en su versión 22.04 para el subsistema de linux que dispone Windows (WSL).

En el código se declararon tres constantes (fs , t , x y $fcutoff$) que representan la frecuencia en la señal original, el conjunto de valores desde 0 a 1 espaciados por $1/fs = 1/1000$, el seno de $2\pi * f * t = 200\pi$ y la frecuencia de corte, respectivamente. Esto en su conjunto es representado gráficamente en la primera imagen (superior) que resulta del programa en su totalidad. Mientras que para la segunda gráfica se convirtió la señal original (x) en su transformada rápida de Fourier $fft()$ para aproximarla a cero a medida que se acercaba a la mitad de la misma, es decir el 50 por ciento.

II. RESULTADOS

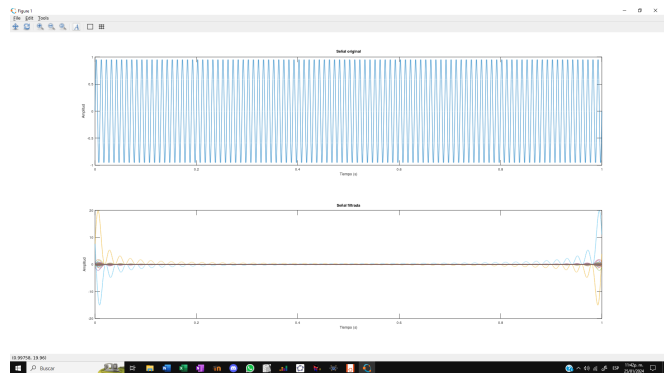


Figura 2: Gráfica senoidal filtrada, elaboración propia

En la parte superior de la imagen podemos observar la gráfica de una función senoidal de amplitud 1, frecuencia 200π y dentro del rango positivo de 0 a 1; tal y como se declaró a la variable t , x y f . Por el otro lado, en la segunda gráfica se muestra una señal que disminuye a medida que se acerca a la bisectriz de la misma y mientras se alejaba volvía a su amplitud original. Además, también se pueden observar distintas aproximaciones de la gráfica diferenciadas por color.