# Operaciones básicas

APELLIDOS, NOMBRE: DANIEL QUEIJO SEOANE

## Objetivo

En esta práctica se realizará el diezmado de una señal y su posterior reconstrucción utilizando distintos métodos que permiten estimar el valor de una muestra utilizando los valores de las muestras vecinas. Se considerarán los siguientes métodos para reconstruir la señal:

- Método 1: muestra anterior
- Método 2: promedio de las muestras vecinas.

## EJERCICIO 1: DIEZMADO Y RECONSTRUCCIÓN

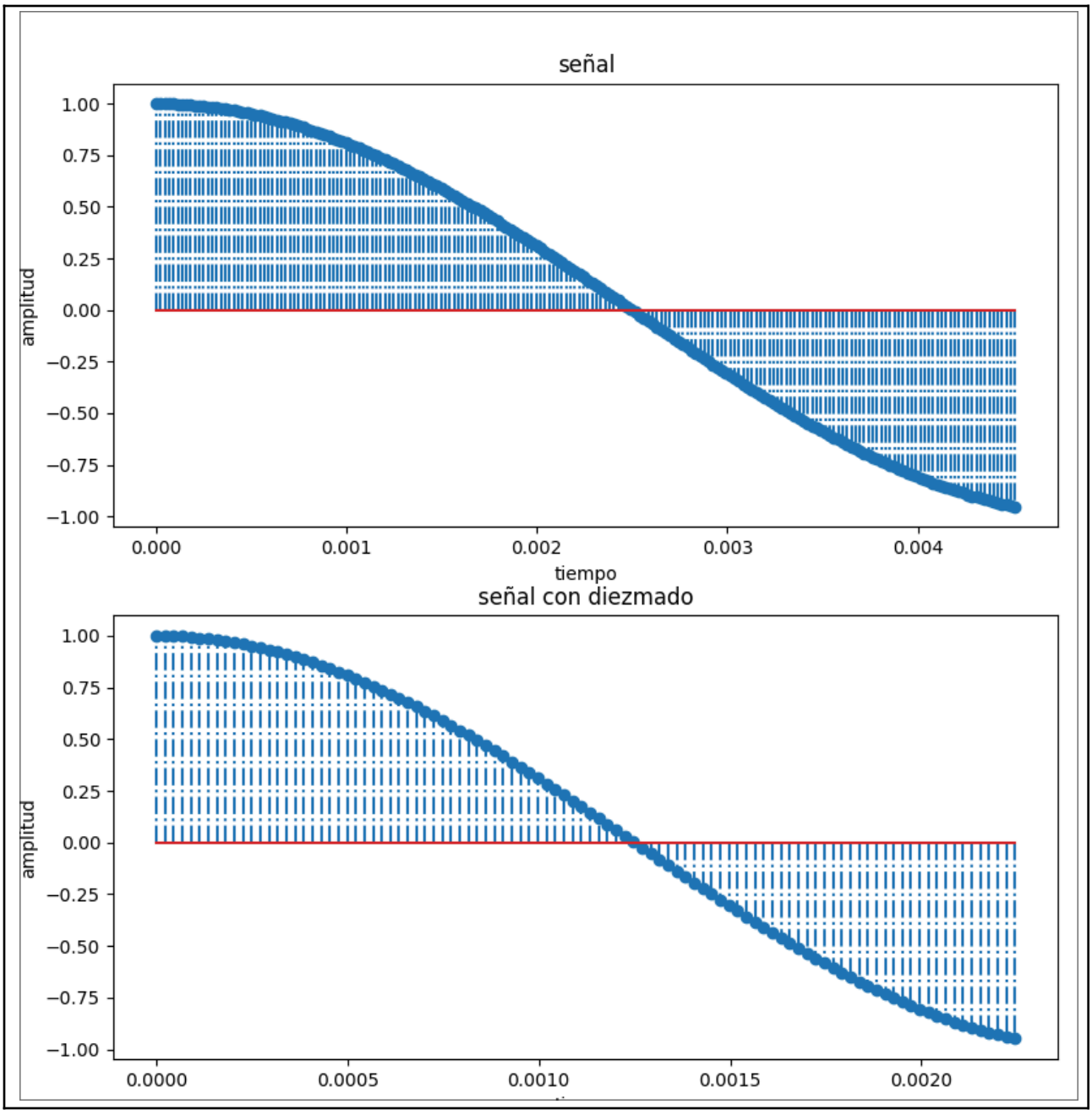
El siguiente código genera una señal senoidal. Posteriormente, hace un diezmado x(N n) utilizando xcosd = xcos[0:ln:N] donde ln es la longitud de la señal original. Aunque N puede tomar cualquier número entero, pero **utilizaremos N = 2**.

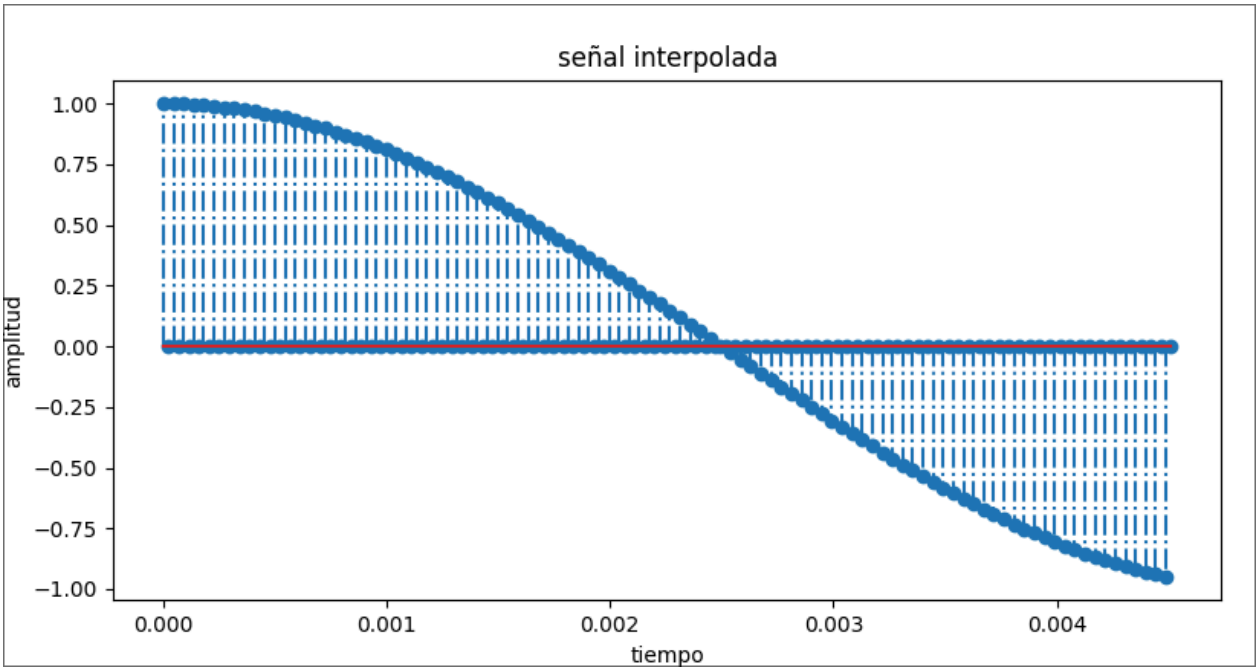
```
import matplotlib.pyplot as plt
import numpy as np
import math
import matplotlib.pyplot as plt

"================================================="
" Diezmado "
valini = 0 # Valor inicial
duracion = 200  # Num muestras
fs = 44100  # Hz
f = 100
N = 2

n = np.arange(valini,duracion)/fs
xcos = np.cos(2 * np.pi * f * n)

ln = len(n)
xcosd = xcos[0:ln:N]
nd = np.arange(valini,duracion/N)/fs

plt.subplot(211)
```

```
plt.stem(n, xcos, '-.')
plt.xlabel('tiempo')
plt.ylabel('amplitud')
plt.title('señal')

plt.subplot(212)
plt.stem(nd, xcosd, '-.')
plt.xlabel('tiempo')
plt.ylabel('amplitud')
plt.title('señal con diezmado')
plt.show()
"================================================="
" Interpolación             "

ln = len(n)
xcosi = np.zeros(ln)
xcosi[0:ln:N] = xcosd

plt.subplot(211)
plt.stem(n, xcosi, '-.')
plt.xlabel('tiempo')
plt.ylabel('amplitud')
plt.title('señal interpolada')
plt.show()
```

Ejecute el código anterior y fíjese en los valores de la señal que resulta de hacer a interpolación.

**Copie aquí la figura que obtiene paras xcos, xcosd y xcosi.**


señal
amplitud
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.000
0.001
0.002
0.003
0.004
tiempo
señal con diezmado
amplitud
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0000
0.0005
0.0010
0.0015
0.0020




**¿Cuál es la relación entre la duración de xcos, xcosd y xcosi?**

xcos es la señal original, a la cual después le aplicamos un diezmado, q guardamos en xcosd. xcosd tiene la mitad elementos que xcos, y para "reconstruir" xcos utilizamos una interpolación sobre xcosd que guardamos en xcosi. Esta vuelve a tener el mismo tamaño q xcos, pero los valores intermedios que conseguimos son 0(porque xcosd es una señan discreta), por lo que no obtenemos realmente una reconstrucción de la señal original.

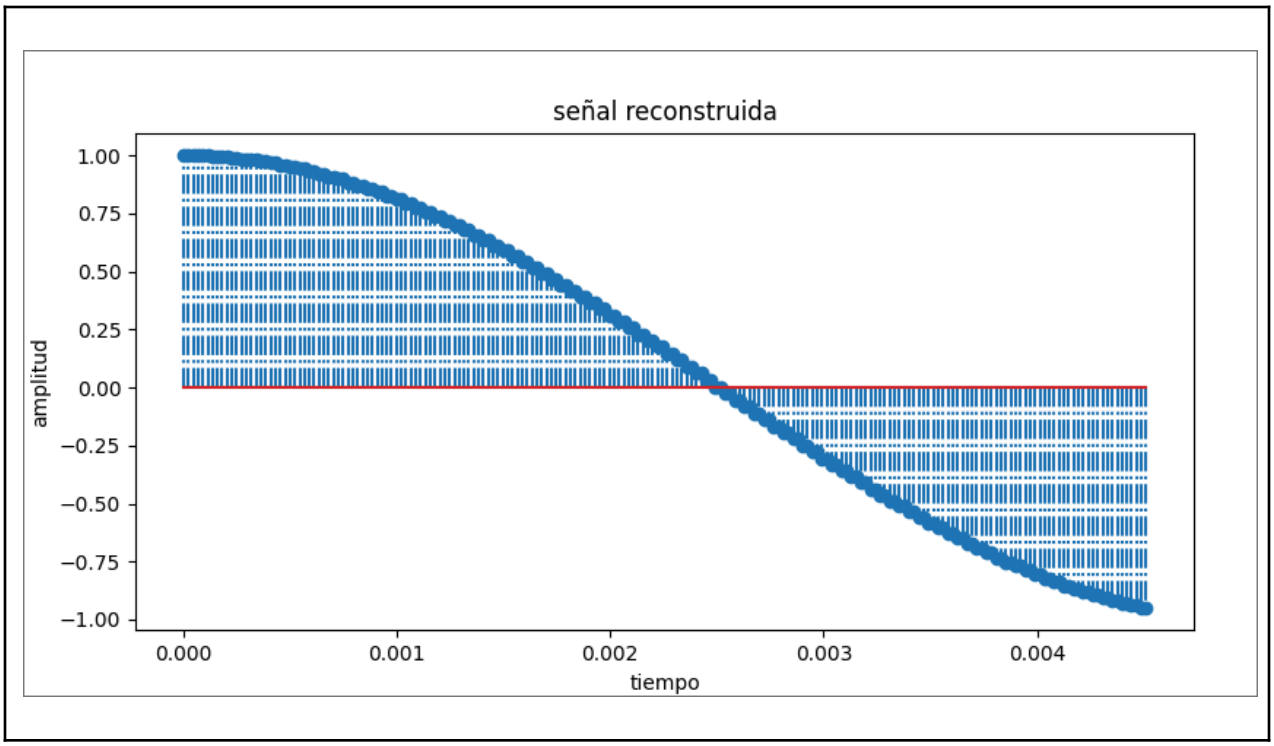
## EJERCICIO 2: RECONSTRUCCIÓN CON LA MUESTRA ANTERIOR

Como se pudo ver en el ejercicio anterior, **la interpolación no es la inversa del diezmado** porque no permite reconstruir exactamente la señal original. De hecho, la señal vale cero en aquellos instantes que fueron eliminados en el diezmado. Por ello, los sistemas de reconstrucción deben realizar una "estimación" de las muestras eliminadas.

Una forma de obtener una mejor reconstrucción es **asignar el valor anterior distinto de cero** a los valores donde la señal no tiene valor (es decir, por la interpolación se ha asignado 0). Por ejemplo, si los valores son 0.5, **0**, 1, **0**, 0, **0**, 1 la señal reconstruida serán 0.5, **0.5**, 1, **1**, 0, **0**, 1.

Realice un código que haga esa operación. Una forma sencilla es utilizar asignaciones del tipo: `xcosr[valini:valfin:N] = xcosd[valini2:valfin2]`

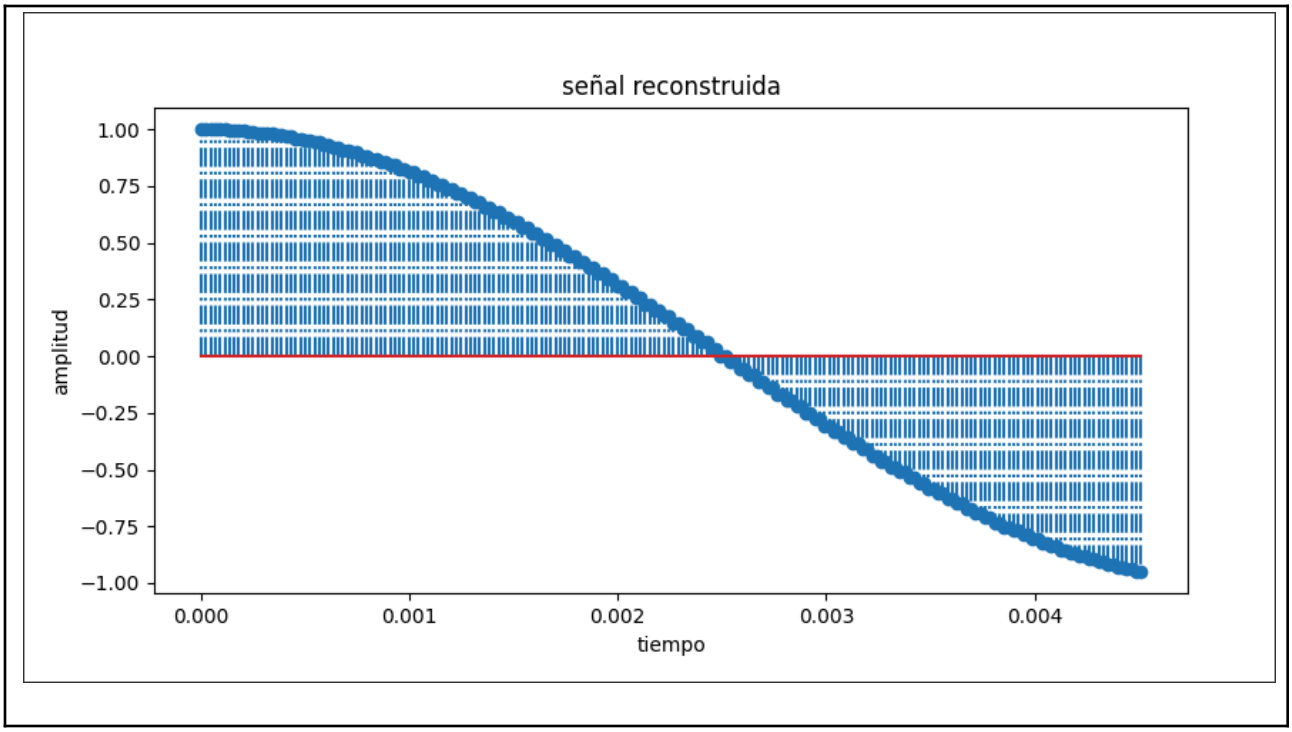
**Copie aquí la figura de xcosi que obtiene.**




## EJERCICIO 3: RECONSTRUCCIÓN CON LA MEDIA DE MUESTRAS VECINAS

Escriba el código de un método de reconstrucción donde a cada muestra no recuperada se le **asigna el promedio de las muestras contiguas**. Por ejemplo, si los valores originales son 0.5, **0**, 1, **0**, 0, **0**, 1 la señal reconstruida será 0.5, **0.75**, 1, **0.5**, 0, **0.5**, 1

| **Copie aquí la figura de xcosi que obtiene.** |
|---|




## EJERCICIO 4: RECONSTRUCCIÓN DE AUDIO

Cambie la señal senoidal utilizada en los ejercicios anteriores por el registro de audio. Necesitará incluir lo siguiente:

```
from scipy.io.wavfile import read
import sounddevice as sd
```

Para leer un fichero de audio utilice el siguiente código.

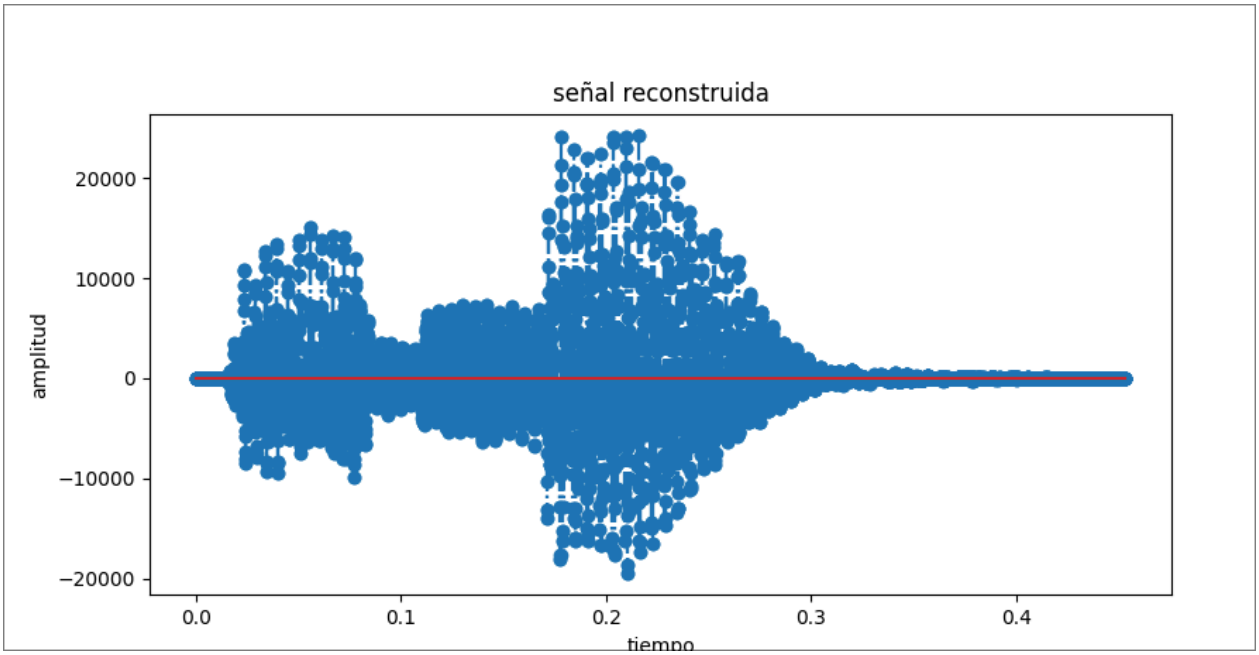
```
from scipy.io.wavfile import read
import sounddevice as sd

fs, xcos  = read('hola_22050.wav')
valini = 0 # Valor inicial
duracion = len(xcos)
n = np.arange(valini,duracion)/fs
```

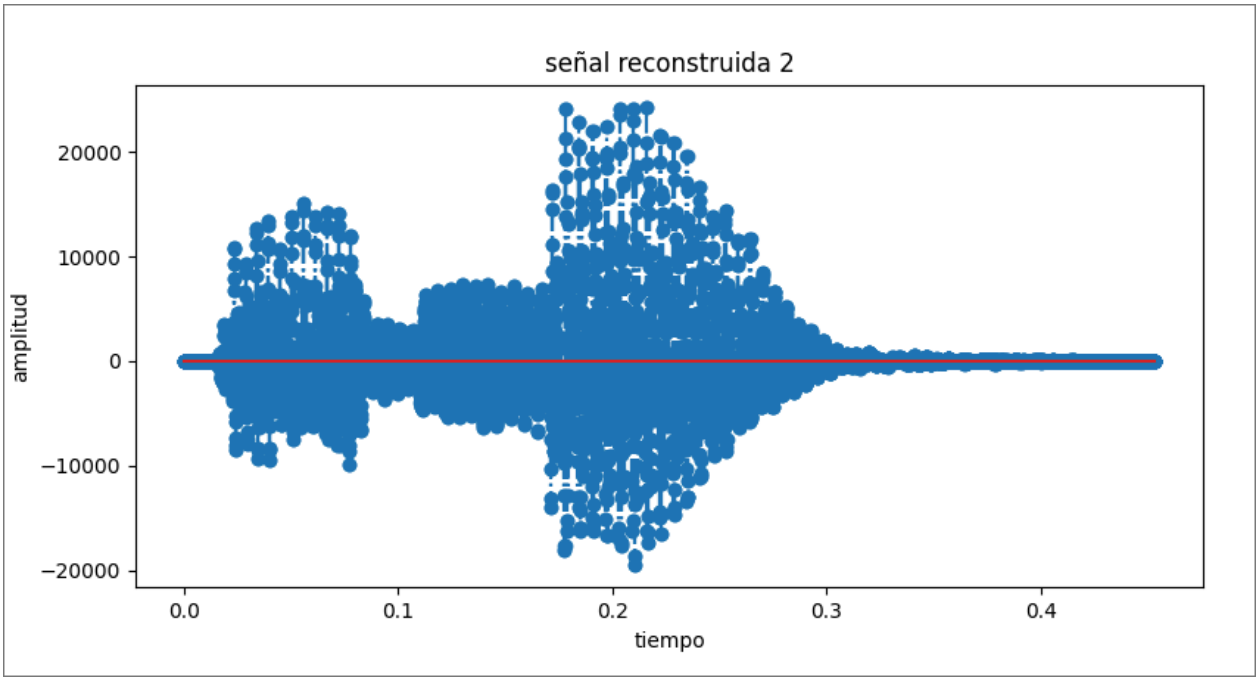
Puede escuchar las señales (original, diezmado, interpolada y reconstruida) utilizando lo siguiente:

```
sd.play(x, fs)  #x es la señal que quiere escuchar
```

Observe las gráficas y, si puede, escuche el sonido.

**Copie aquí la figura de xcosi que obtiene con los dos métodos de reconstrucción.**




**¿Qué opina del sonido?**

No encuentro unas diferencias apreciables entre las dos formas de reconstruir la señal. En cambio, con xcosi si que puedo percibir un sonido más estridente y "roto"

## EJERCICIO 5: INTERPOLACIÓN DE IMÁGENES

En el procesamiento de imágenes es muy común utilizar el diezmado para reducir el tamaño de la imagen que se quiere procesar o almacenar. A continuación, se muestrea un patrón para hacer ese diezmado.

Para la reconstrucción se emplea la media de los 4 valores vecinos (salvo en la primera y última fila o columna) como se muestra a continuación.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1,2,5 | 2 | 2,3,6 | 3 | 3,4,7 | 4 | 4,8 |
| 1,5,9 | 5 | 2,5,6,10 | 6 | 3,6,7,11 | 7 | 4,7,8,12 | 8 |
| 9 | 5,9,10,13 | 10 | 6,10,11,14 | 11 | 7,11,12,15 | 12 | 8,12,16 |
| 9,13 | 13 | 10,13,14 | 14 | 11,14,15 | 15 | 12,15,16 | 16 |

Haga un programa que lea una imagen, realice el diezmado y la reconstrucción. Fíjese en el tamaño de las imágenes obtenidas. Suba el código a Campus Virtual y copie en el recuadro las imágenes que obtuvo.

**Copie la imagen original. ¿Qué tamaño tiene?**

**Copie la imagen después de realizar el diezmado. ¿Qué tamaño tiene?**

**Copie la imagen después de realizar la reconstrucción. ¿Qué tamaño tiene?**