TPNº2- AntiTransformada Discreta de Fourier

Autor: Pablo D. Folino
Link del repositorio: https://github.com/PabloFolino/MSE_PSF_TP2.git

Enunciado:

En el archivo clases/tp2/fft_hjs.npy se almacenaron los valores de un espectro en frecuencia correspondientes a una señal desconocida. Indique: 1) Puede estimar que representa esta señal? (tip: grafique en 2d la idft) 2) Hasta que punto podría limitar el ancho de banda del espectro dado en el archivo y que aun se logre interpretar la señal? 3) Pegue el link a un pdf con los códigos y los gráficos utilizados.

Se realiza un programa(***señales_iTTF_folino_v3.py***) en donde se lee el archivo de referencia, se gráfica la señal de entrada su espectro y características.

Como no se especifica ni el tiempo se sampling ni la frecuencia, se adoptan algunos valores al comienzo del programa:

señales_iTTF_folino_v3.py > X_ifft_conFiltro

```python
# Autor : Pablo D. Folino
# Ejercitación lee información de una archivo "señales_iTTF_folino_v3.py"

import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sc
import os

# Valores supuestos
Fs  =   5       # frecuencia de muestreo, desconozco con que
Ts  =   1/Fs    # frecuencia se obtuvieron los datos

```

Se lee el archivo a evaluar, y se verifica alguna característica como la cantidad de datos que posee el archivo:

```python
# Se lee el archivo
X_fft=np.load("../Informe/Archivos de enunciados/fft_hjs.npy")[::1]
# with open("../Informe/Archivos de enunciados/fft_hjs.npy","r") as ins:
#     cont = ins.read()  # Esto devuelve el contenido completo, no linea por linea
#     arr = eval(cont)
#     señal=arr

os.system("clear")

N=len(X_fft)

# Se verifica que se leyeron los N elementos
print("\t Se pudieron leer los N={} elementos del archivo".format(N))
print("\t El formato de los elementos es={}".format(type(X_fft[0])))
```

Se calcula la transformada inversa de la señal de entrada:

```
#--------------------Señal en frecuencia-------------------------------------
F=np.arange(-N/2,N/2)
#X_fft=np.fft.ifftshift(X_fft)
M_fft= (abs(X_fft)/N)
#----------Potencia promedio---------
pot_promedio=np.sum(M_fft**2)
#----------Discreto-----------------
ts = Ts*np.arange(0, N,1)           # Ojo al desconocer Fs --> Ts es ficticio
#-------Anti Tranformada-------------
señal = np.fft.ifft(X_fft)

```

Como para probar la señal de entrada con distintos filtros se utiliza una función en la cual se pasa como parámetro la cantidad de muestras (centradas en el origen) de dicho filtro:

```
def X_ifft_conFiltro(X_fft,filtro):
    #--------------------Señal filtrada en frecuencia--------------------------
    X_fft_filtro=np.copy(X_fft)
    F_filtro=np.arange(-len(X_fft)/2,len(X_fft)/2-Fs/len(X_fft))
    centro=int(len(X_fft)/2)                                    # Acordrse que N=len(X_fft)
    for i in range(0,centro-filtro):
        X_fft_filtro[i]=0
        X_fft_filtro[i+centro+filtro]=0
    M_fft_filtro= abs(X_fft_filtro)**2/len(X_fft)
    #----------Discreto-----------------
    ts_filtro = Ts*np.arange(0,len(X_fft),1)
    #-------Anti Tranformada-------------
    señal_filtro = np.fft.ifft(X_fft_filtro)
    return ts_filtro,señal_filtro,F_filtro,X_fft_filtro,M_fft_filtro
```

Y luego se grafica la señal en el espectro, la señal en antitransformada original y con varios filtros eliminando algunas muestras:

```
#--------------------Señal original-------------------------------------
s1 = fig.add_subplot(3,2,1)
plt.title("Señal de entrada |X_fft|")
plt.xlabel("Frecuencia (Hz)")
plt.ylabel("Amplitud FFT")
s1.grid(True)
s1.plot(F, M_fft,"g-")

s9 = fig.add_subplot(3,2,2)
plt.title("Valores")
plt.xlim(0,10)
plt.ylim(0,10)
plt.axis('off')
s9.spines['right'].set_visible(False)
s9.spines['top'].set_visible(False)
s9.spines['bottom'].set_visible(False)
s9.spines['left'].set_visible(False)
plt.text(0,9,"Se supone que FrecSampl.="+str(Fs)+"Hz",fontsize=10)
plt.text(0,7,"Resolución temporal="+str(f'{Ts:.{2}f}')+"seg",fontsize=10)
plt.text(0,5,"Resolución espectral="+str(f'{Fs/N:.{4}f}')+"Hz",fontsize=10)
plt.text(0,3,"Pot prom="+str(f'{pot_promedio:.{2}f}')+"w",fontsize=10)
plt.text(0,1,"Muestras="+str(N),fontsize=10)


s1 = fig.add_subplot(3,2,3)
plt.title("Antitransformada en 2D de la señal de entrada "+str(N)+"muestras")
plt.xlabel("Im(señal)")
plt.ylabel("Real(Señal)")
s1.grid(True)
s1.plot(np.imag(señal),np.real(señal),'r-')

```

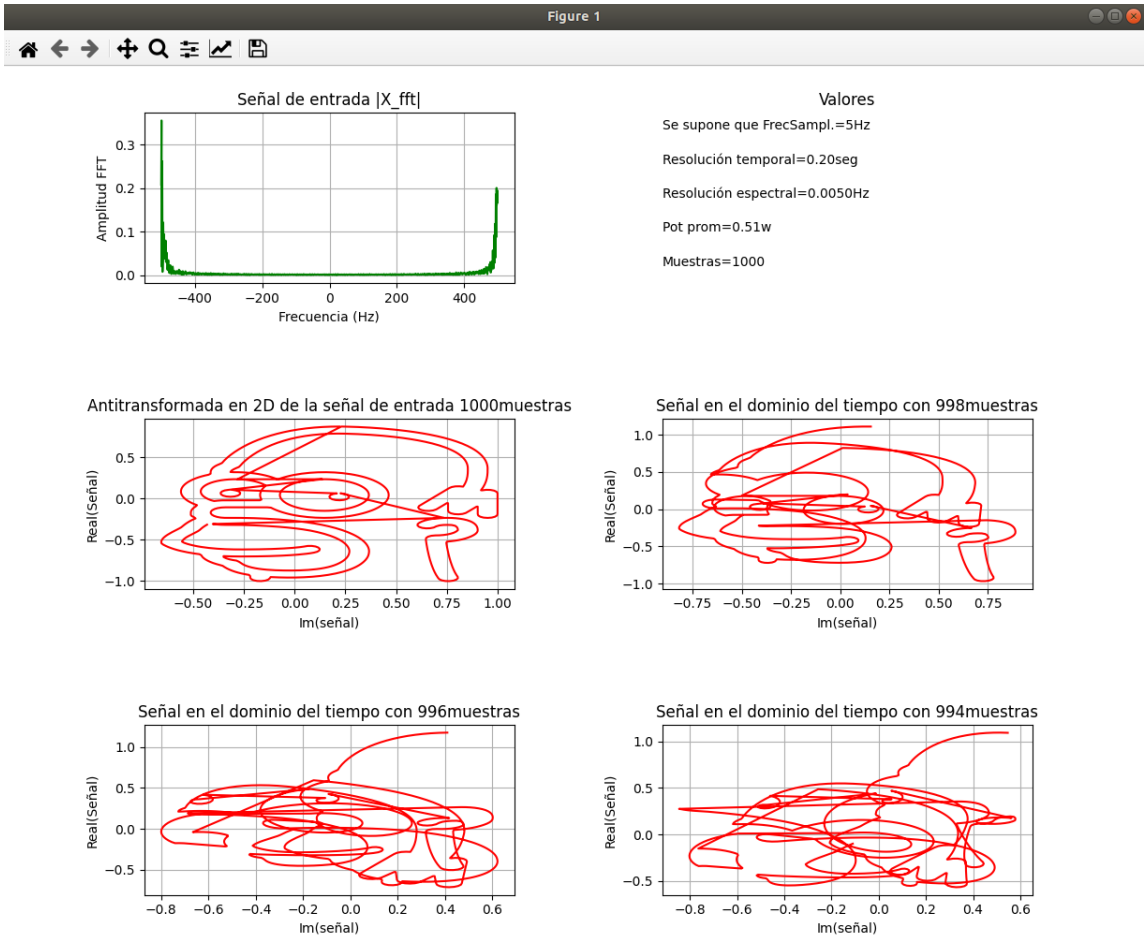
```python
# --------------------Señal filtrada en frecuencia  -------------------------
for i in range(1,4,1):
    delta=1             # Muestras que se restan a cada lado el espectro
    filtro=int((N-2*delta*i)/2)
    ts_filtro,señal_filtro,F_filtro,X_fft_filtro,M_fft_filtro=X_ifft_conFiltro(X_fft,filtro)

    s1 = fig.add_subplot(3,2,(3+i))
    plt.title("Señal en el dominio del tiempo con "+str(filtro*2)+"muestras")
    plt.xlabel("Im(señal)")
    plt.ylabel("Real(Señal)")
    s1.grid(True)
    s1.plot(np.imag(señal_filtro),np.real(señal_filtro),'r-')



plt.get_current_fig_manager().window.showMaximized() #para QT5
plt.show()
```

Los gráficos obtenidos son los siguientes:



Como se observa en el espectro la mayoría de la información se encuentra en los extremos del espectro, con lo cual al acotar aunque sea un poco el mismo se pierde información de importancia y la imagen ya no es clara.