TPNº2- Transformada Discreta de Fourier

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

Enunciado:
Grafique las siguientes señales lado a lado con su respectivo espectro en frecuencias: 1) Senoidal. 2) Cuadrada. 3) Triangular 4) Delta en t=0. Indicando en cada caso los siguientes parámetros (si corresponde) : 1) Frecuencia. B) Amplitud. C) Potencia promedio. D) Fs. E) N. 5) Pegue el link a un pdf con los códigos, gráficos y comentarios.

Se realiza un programa(***señales_TTF_folino.py***) en donde posee funciones que generan señales cuadradas triangulares, senoidales e impulso.

Importante: a las funciones se le agregan algunos parámetros extra para tener mayor flexibilidad .

El programa posee una sección de definiciones de los valores a probar:

```
8   #Valores a probar
9   f    =   2       # frecuencia de la señal a realizar la transformada
10  Fs   =   40      # frecuencia de muestreo
11  Ts   =   1/Fs
12  fase =   0
13  N    =   40
14  amp  =   2.5
15
16  Pu   =   100     # Cantidad de puntos para generar la continua
17
```

Y mediante módulos se calculan para cada caso dos funciones una seudo-contínua, y otra en donde se muestrea la primera. Por otro lado en el mismo módulo se calcula la transformada rápida de Fourier, y se calcula la potencia promedio en las señales periódicas.

Para la función senoidal, se obtuvo el siguiente código:

```
def senoidal(fs,f,amp,muestras,fase):
    #----------Discreto-----------------
    ts = Ts*np.arange(0, fs,1)
    w1 =amp*np.sin(2*np.pi*f*ts+fase)
    #----------Contínuo-----------------
    tsC = Ts*np.arange(0, fs,1/Pu)
    w1C = amp*np.sin(2*np.pi*f*tsC+fase)
    #----------Tranformada--------------
    fft = np.fft.fft(w1)/muestras
    M_fft = np.fft.fftshift(abs(fft))
    #F = Fs*np.arange(0, len(w1))/len(w1)
    F=np.fft.fftfreq(fs,1/muestras)
    F=np.fft.fftshift(F)
    #----------Potencia promedio---------
    pot_promedio=np.sum(M_fft**2)
    return ts,w1,tsC,w1C,F,M_fft,pot_promedio
```

Para la función cuadrática el código es :

```
65  def cuadrada(fs,f,amp,muestras,fase):
66      #----------Discreto-----------------
67      ts = Ts*np.arange(0, fs,1)
68      w1 =amp*sc.square(2*np.pi*ts*f+fase)
69      #----------Contínuo-----------------
70      tsC = Ts*np.arange(0, fs,1/Pu)
71      w1C = amp*sc.square(2*np.pi*tsC*f+fase)
72      #----------Tranformada--------------
73      fft = np.fft.fft(w1)/muestras
74      M_fft = np.fft.fftshift(abs(fft))
75      #F = Fs*np.arange(0, len(w1))/len(w1)
76      F=np.fft.fftfreq(fs,1/muestras)
77      F=np.fft.fftshift(F)
78      #----------Potencia promedio---------
79      pot_promedio=np.sum(M_fft**2)
80      return ts,w1,tsC,w1C,F,M_fft,pot_promedio
```

La función triangular:

```
96   def triangular(fs,f,amp,muestras,fase):
97       #----------Discreto-----------------
98       ts = Ts*np.arange(0, fs,1)
99       w1 =amp*sc.sawtooth(2*np.pi*f*ts+fase,1)
100      #----------Contínuo-----------------
101      tsC = Ts*np.arange(0, fs,1/Pu)
102      w1C = amp*sc.sawtooth(2*np.pi*f*tsC+fase,1)
103      #----------Tranformada--------------
104      fft = np.fft.fft(w1)/muestras
105      M_fft = np.fft.fftshift(abs(fft))
106      #F = Fs*np.arange(0, len(w1))/len(w1)
107      F=np.fft.fftfreq(fs,1/muestras)
108      F=np.fft.fftshift(F)
109      #----------Potencia promedio---------
110      pot_promedio=np.sum(M_fft**2)
111      return ts,w1,tsC,w1C,F,M_fft,pot_promedio
```

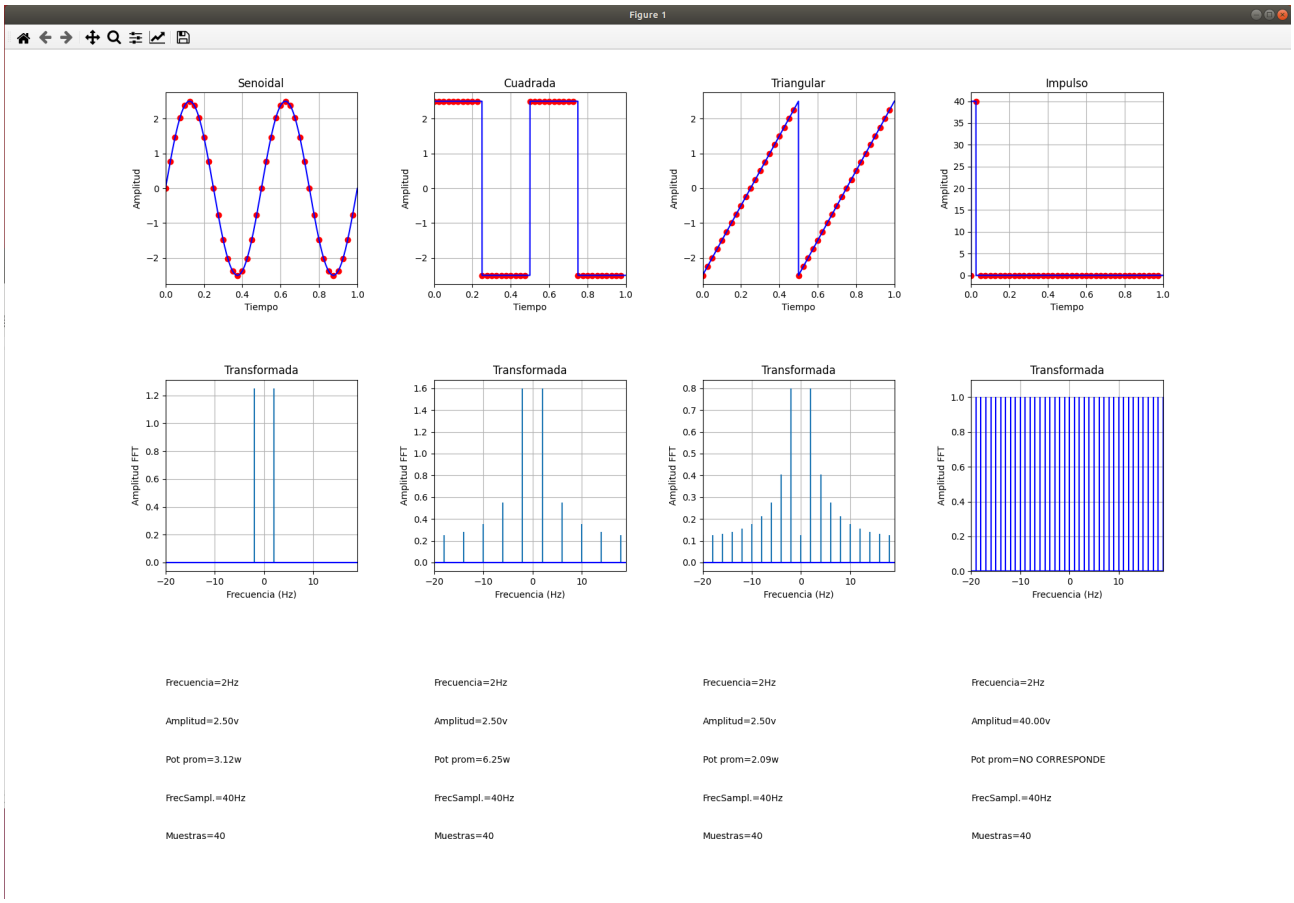
El delta de Dirac(función impulso), se codificó de la siguiente manera:

```
125  def impulso(fs,f,amp,muestras,fase):
126      #----------Discreto-----------------
127      ts = Ts*np.arange(0, fs,1)
128      w1=np.zeros(len(ts))
129      for i in range(1,len(ts)):
130          if( ts[i] < 1/amp ):
131              w1[i]=amp
132          else:
133              w1[i]=0
134      #----------Contínuo-----------------
135      tsC = Ts*np.arange(0, fs,1/Pu)
136      w1C=np.zeros(len(tsC))
137      for i in range(1,len(tsC)):
138          if( tsC[i] < 1/amp ):
139              w1C[i]=amp
140          else:
141              w1C[i]=0
142      #----------Tranformada--------------
143      fft = np.fft.fft(w1)/muestras
144      M_fft = np.fft.fftshift(abs(fft))
145      #F = Fs*np.arange(0, len(w1))/len(w1)
146      F=np.fft.fftfreq(fs,1/muestras)
147      F=np.fft.fftshift(F)
148      #----------Potencia promedio---------
149      pot_promedio=0
150      #pot_promedio=np.sum(M_fft**2)
151      return ts,w1,tsC,w1C,F,M_fft,pot_promedio
152
```

Para cada tipo de señal se realizaron tres gráficos, el primero indicando la señal en contínua(azul) y con puntos rojos los valores de muestreo. El segundo gráfico corresponde a la transformada y el tercero muestra los valores solicitados. Se muestra a continuación el código de uno de esos juegos de gráficos.

```
196  #--------------------Cuadrada--------------------------------------
197  s2 = fig.add_subplot(3,4,2)
198  plt.title("Cuadrada")
199  plt.xlabel("Tiempo")
200  plt.ylabel("Amplitud")
201  s2.grid(True)
202  ts,w1,tsC,w1C,F,M_fft,pot_promedio=cuadrada(Fs,f,amp,N,fase)
203  plt.xlim(0,N/Fs)
204  s2.plot(ts,w1,'ro')
205  s2.plot(tsC,w1C,'b-')
206
207
208  s6 = fig.add_subplot(3,4,6)
209  plt.title("Transformada")
210  plt.xlabel("Frecuencia (Hz)")
211  plt.ylabel("Amplitud FFT")
212  s6.grid(True)
213  plt.xlim(-Fs/2,Fs/2-Fs/N)
214  #s6.plot(F, M_fft)
215  s6.stem(F, M_fft, markerfmt=" ", basefmt="-b")
216
217  s10 = fig.add_subplot(3,4,10)
218  plt.xlim(0,10)
219  plt.ylim(0,10)
220  plt.axis('off')
221  s10.spines['right'].set_visible(False)
222  s10.spines['top'].set_visible(False)
223  s10.spines['bottom'].set_visible(False)
224  s10.spines['left'].set_visible(False)
225  plt.text(0,9,"Frecuencia="+str(f)+"Hz",fontsize=10)
226  plt.text(0,7,"Amplitud="+str(f'{amp:.{2}f}')+"v",fontsize=10)
227  plt.text(0,5,"Pot prom="+str(f'{pot_promedio:.{2}f}')+"w",fontsize=10)
228  plt.text(0,3,"FrecSampl.="+str(Fs)+"Hz",fontsize=10)
229  plt.text(0,1,"Muestras="+str(N),fontsize=10)
```

A continuación se muestran la gráficas obtenidas con el programa ***señales_TTF_folino.py***:



Las gráficas se validaron con la teoría probándolas con distintos juegos de valores. Es interesante ver que cuando el delta de **Dirac** se va transformando en un escalón comienza a aparecer la función ***sinc***.