TP2 Procesamiento de señales:

- 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.

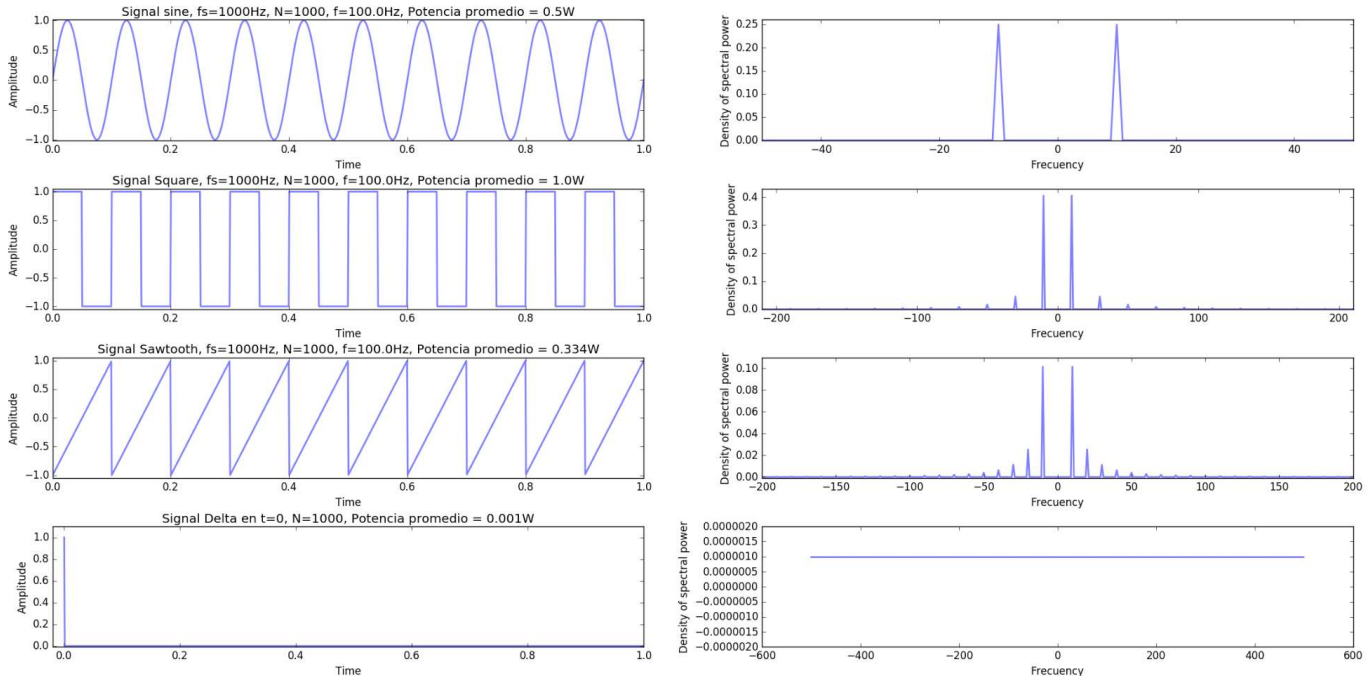
```
#generators con funciones
def signalSine(fs,f0, amp, N, fase):
    i = 0
   s = np.zeros((N ,), dtype=float)
    m = np.empty((2,N), float)
    t = np.linspace(0, 1, N, endpoint=True)
    for  i in range(N):
        s[i] = amp * np.sin(2*np.pi*fs*t[i]*1/f0 + fase)
        i+=1
    m[0] = np.transpose(s)
    m[1] = t
    return m

def signalSquare(fs,f0, amp, N):
    i = 0
    s = np.zeros((N ,), dtype=float)
    m = np.empty((2,N), float)
    t = np.linspace(0, 1, N, endpoint=True)
    for  i in range(N):
        s[i] = amp * signal.square(2*np.pi*fs*t[i]*1/f0)
        i+=1
    m[0] = np.transpose(s)
    m[1] = t
    return m

def signalSawtooth(fs,f0, amp, N):
    i = 0
   s = np.zeros((N ,), dtype=float)
    m = np.empty((2,N), float)
    t = np.linspace(0, 1, N, endpoint=True)
    for  i in range(N):
        s[i] = amp * signal.sawtooth(2*np.pi*fs*t[i]*1/f0)
        i+=1
    m[0] = np.transpose(s)
    m[1] = t
    return m

def signalDelta(N):
     s = np.zeros((N ,), dtype=float)
    m = np.empty((2,N), float)
    t = np.linspace(0, 1, N, endpoint=True)
    s[0] = 1
    m[0] = np.transpose(s)
    m[1] = t
    return m

def potenciaPromedio(s, N):
    i = 0
    sum = 0
```

```
    for  i in range(N):
        sum = sum + s[i]**2
    p = round(sum/N, 3)
    return p

def espectro(s, fs, N):
    m = np.empty((2,N), float)
    nData  = np.arange(0,N,1)
    fData  = nData*(fs/N)-fs/2
    arrayS = np.array(s)
    fftS = np.fft.fft(arrayS )
    fftS = np.fft.fftshift(fftS)
    spS = np.abs(fftS/N)**2
    m[0] = spS
    m[1] = fData
    return m
```

#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):Frecuencia, Amplitud, Potencia promedio, Fs, N.

```
fs = 1000
N = 1000
f01 = 0.1 * fs
fase = 0
amp = 1

plt.subplot(4,2,1)
m1 = signalSine(fs,f01, amp, N, fase)
p1 = potenciaPromedio(m1[0], N)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Signal sine, fs='+str(fs)+'Hz, N='+str(N)+', f='+str(f01)+'Hz, Potencia promedio = '+str(p1)+'W')
plt.ylim(-1.01, 1.01)
plt.plot(m1[1],m1[0],linewidth=2,alpha=0.5)

plt. subplot(4,2,2)
m1s = espectro(m1[0], fs, N)
plt.xlabel('Frecuency')
plt.ylabel('Density of spectral power')
plt.xlim(-50, 50)
plt.ylim(0, 0.26)
plt.plot(m1s[1], m1s[0],linewidth=2,alpha=0.5)

plt.subplot(4,2,3)
m2 = signalSquare(fs,f01, amp, N)
p2 = potenciaPromedio(m2[0], N)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Signal Square, fs='+str(fs)+'Hz, N='+str(N)+', f='+str(f01)+'Hz, Potencia promedio = '+str(p2)+'W')
plt.ylim(-1.05, 1.05)
plt.plot(m2[1],m2[0],linewidth=2,alpha=0.5)

plt. subplot(4,2,4)
m2s = espectro(m2[0], fs, N)
plt.xlabel('Frecuency')
plt.ylabel('Density of spectral power')
plt.xlim(-210, 210)
plt.ylim(-0.0001, 0.43)
plt.plot(m2s[1], m2s[0],linewidth=2,alpha=0.5)

plt.subplot(4,2,5)
m3 = signalSawtooth(fs,f01, amp, N)
p3 = potenciaPromedio(m3[0], N)
plt.xlabel('Time')
plt.ylabel('Amplitude')
```

```
plt.title('Signal Sawtooth, fs='+str(fs)+'Hz, N='+str(N)+', f='+str(f01)+'Hz, Potencia promedio = '+str(p3)+'W')
plt.ylim(-1.05, 1.05)
plt.plot(m3[1],m3[0], linewidth=2,alpha=0.5)

plt. subplot(4,2,6)
m3s = espectro(m3[0], fs, N)
plt.xlabel('Frecuency')
plt.ylabel('Density of spectral power')
plt.xlim(-200, 200)
plt.ylim(-0.001, 0.11)
plt.plot(m3s[1], m3s[0],linewidth=2,alpha=0.5)

plt.subplot(4,2,7)
m4 = signalDelta(N)
p4 = potenciaPromedio(m4[0], N)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Signal Delta en t=0, N='+str(N)+', Potencia promedio = '+str(p4)+'W')
plt.xlim(-0.02, 1)
plt.ylim(-0.01, 1.1)
plt.plot(m4[1],m4[0],linewidth=2,alpha=0.5)

plt. subplot(4,2,8)
m4s = espectro(m4[0], fs, N)
plt.xlabel('Frecuency')
plt.ylabel('Density of spectral power')
plt.plot(m4s[1], m4s[0],linewidth=2,alpha=0.5)
plt.ylim(-0.000002, 0.000002)
plt.show()
```



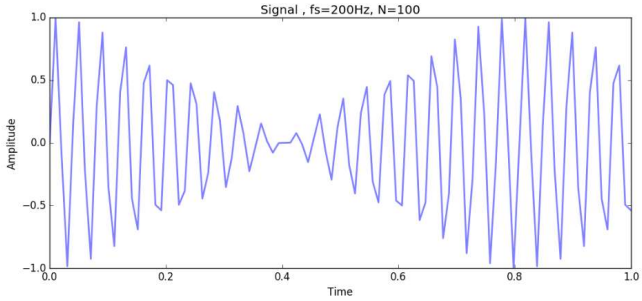
- Dado el archivo clases/tp2/resolucion_espectral.txt que contiene 100 valores reales sampleados a fs=200Hz, indique:
  1) Resolución espectral.
  2) Espectro en frecuencia de la señal.
  3) A simple inspección que frecuencia(s) distingue.
  4) Aplique alguna técnica que le permita mejorar la resolución espectral y tome nuevamente el espectro.
  5) Indique si ahora los resultados difieren del punto 3 y argumente su respuesta.

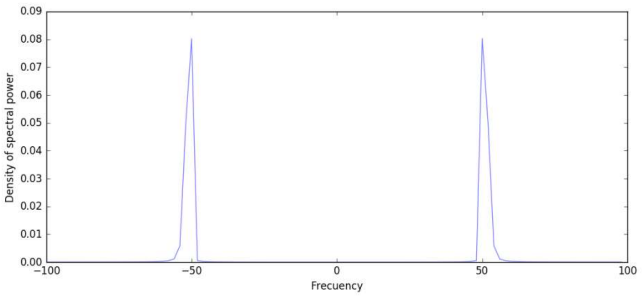
```
fs = 200
N = 100

with open('resolucion_espectral.txt', 'r') as G:
    stxt = G.read()
    stxt = stxt.lstrip('[')
    stxt = stxt.rstrip(']\n')
    stxt = stxt.split(',')

x = np.array(stxt)
s = x.astype(np.float)

t = np.linspace(0, 1, N, endpoint=True)

plt.subplot(2,2,1)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Signal , fs='+str(fs)+'Hz, N='+str(N))
plt.plot(t,s,linewidth=2,alpha=0.5)

# 1) Resolución espectral.
resolucionEspectral = fs/N
print("Resolucion espectral:")
print(resolucionEspectral)

# 2) Espectro en frecuencia de la señal.
m1s = espectro(s, fs, N)
plt.subplot(2,2,2)
plt.xlabel('Frecuency')
plt.ylabel('Density of spectral power')
plt.plot(m1s[1], m1s[0],linewidth=1,alpha=0.5)

# 3) A simple inspección que frecuencia(s) distingue.

# Se observan dos picos en frecuencia en 50HZ y -50 Hz.

# 4) Aplique alguna técnica que le permita mejorar la resolución espectral y tome nuevamente el espectro.

# Para una determinada fs, cuanto más grande N mejor resolución espectral, elijo N = 200.

N2 = 200
resolucionEspectral2 = fs/N2
print("Resolucion espectral mejorada:")
print(resolucionEspectral2)

s0 = np.zeros((N2-N ,), dtype=float)
s2 = np.concatenate((s, s0))

t2 = np.linspace(0, 1, N2, endpoint=True)

plt.subplot(2,2,3)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Signal , fs='+str(fs)+'Hz, N='+str(N2))
```

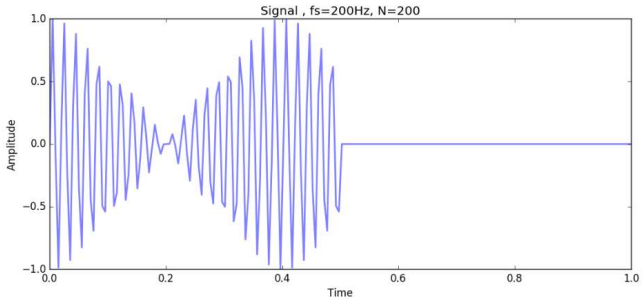
```
plt.plot(t2,s2,linewidth=2,alpha=0.5)

ms2 = espectro(s2, fs, N2)

plt.subplot(2,2,4)
plt.xlabel('Frecuency')
plt.ylabel('Density of spectral power,zero padding ')
plt.plot(ms2[1], ms2[0],linewidth=1,alpha=0.5)
plt.show()
```

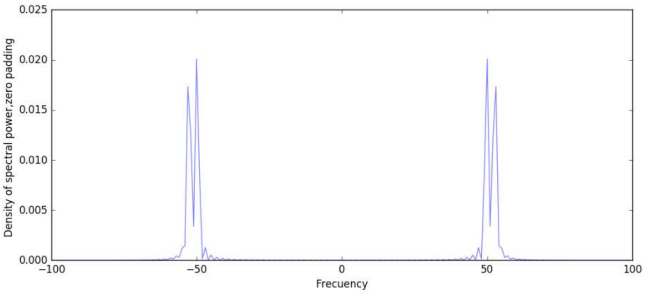
 # 5) Indique si ahora los resultados difieren del punto 3 y argumente su respuesta

 # Se necesitaron más puntos en la DFT, pero la señal ya fue capturada, por lo que se rellena con ceros:zero padding
 # Los resultados difieren, se mejoró la resolución espectral, obteniendo cuatro picos en frecuencia, esto no se veía anteriormente.
 # Un efecto adverso es que la amplitud de la DFT se ve disminuida, esto se observa en la gráfica









- 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 qué punto podría limitar el ancho de banda del espectro dado en el archivo y que aún se logre interpretar la señal?

```
p = np.load("fft_hjs.npy")

# 1) ¿Puede estimar que representa esta señal? (tip: grafique en 2d la idft)

fftDataShifted = np.fft.fftshift(p)
ifftData= np.fft.ifft(p)
N = len(ifftData)
nData  = np.arange(-N/2,N/2,1)
t = np.linspace(0,1,N)

plt.subplot(3,3,1)
plt.title('Señal FFT shifted')
plt.plot(nData,fftDataShifted,'b-o',linewidth=1,alpha=0.5)

plt.subplot(3,3,2)
plt.title('Señal ')
plt.plot(t,np.real(ifftData),'b-o',linewidth=4,alpha=0.5,label="real")
plt.plot(t,np.imag(ifftData),'r-o',linewidth=4,alpha=0.5,label="imag")

plt.subplot(3,3,3)
plt.title('Antitrasnformada IDFT, N='+str(N))
plt.xlim(-0.7, 1.1)
plt.plot(np.imag(ifftData), np.real(ifftData),'b-o',linewidth=4,alpha=0.5)

N2 = 500
fftDataShifted2 = fftDataShifted[N2/2:N-N2/2]
ifftData2= np.fft.ifft(np.fft.ifftshift(fftDataShifted2))
nData2 = np.arange(-N2/2,N2/2,1)
t2 = np.linspace(0,1,len(ifftData2))

plt.subplot(3,3,4)
plt.title('Señal FFT shitfed')
plt.plot(nData2,fftDataShifted2 ,'b-o',linewidth=1,alpha=0.5)

plt.subplot(3,3,5)
plt.title('Señal ')
plt.plot(t2,np.real(ifftData2),'b-o',linewidth=4,alpha=0.5,label="real")
plt.plot(t2,np.imag(ifftData2),'r-o',linewidth=4,alpha=0.5,label="imag")

plt.subplot(3,3,6)
plt.title('Antitrasnformada IDFT, N='+str(len(ifftData2)))
plt.plot(np.imag(ifftData2), np.real(ifftData2),'b-o',linewidth=4,alpha=0.5)

N3 = 499
fftDataShifted3 = fftDataShifted[N2/2:N-N2/2-1]
ifftData3= np.fft.ifft(np.fft.ifftshift(fftDataShifted3))
nData3 = np.arange(-N2/2,N2/2-1,1)
t3 = np.linspace(0,1,len(ifftData3))

plt.subplot(3,3,7)
```

```
plt.title('Señal FFT shitfed')
plt.plot(nData3,fftDataShifted3 ,'b-o',linewidth=1,alpha=0.5)

plt.subplot(3,3,8)
plt.title('Señal ')
plt.plot(t3,np.real(ifftData3),'b-o',linewidth=4,alpha=0.5,label="real")
plt.plot(t3,np.imag(ifftData3),'r-o',linewidth=4,alpha=0.5,label="imag")

plt.subplot(3,3,9)
plt.title('Antitrasnformada IDFT, N='+str(len(ifftData3)))
plt.plot(np.imag(ifftData3), np.real(ifftData3),'b-o',linewidth=4,alpha=0.5)
plt.show()
```

 # 2) Hasta qué punto podría limitar el ancho de banda del espectro dado en el archivo y que aún se logre interpretar la señal?

 # Se puede observar de la gráfica de la fft shifted que la información se encuentra concentrada en las bajas frecuencias.
 # Entonces eliminando las muestras correspondientes con las altas frecuencias podemos seguir distinguiendo la imágen de Homero.
 # Como se observa el límite está en tomar 500 muestras correspondientes al rango de frecuencias bajas.