FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Procesamiento de señales, fundamentos

Maestría en sistemas embebidos
Universidad de Buenos Aires
MSE 5Co2020

Clase 4 - Euler | Fourier - IDFT

Convolucion

Ing. Pablo Slavkin
slavkin.pablo@gmail.com
wapp:011-62433453

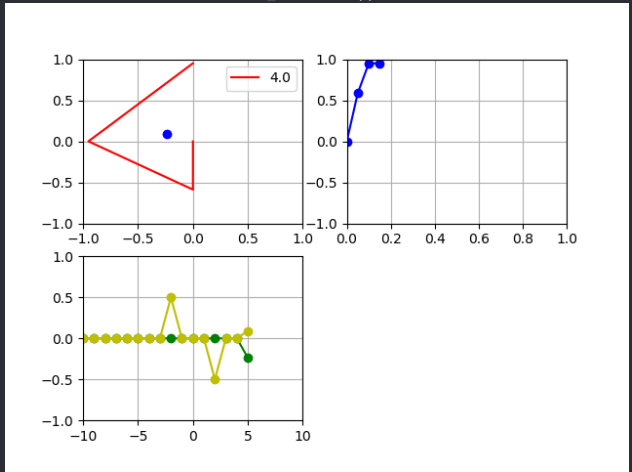
repaso DFT

$e^{j2\pi ft}$ modulado por $\sin(t)$ y centro de masas en f, DFT?

04m50s

- Correr el cofigo y notar como el promedio del circulo que gira multiplicado por la señal para diferentes frecuencias nos da la DFT
- Las frecuencias van de -Fs/2 inclusive a Fs/2 NO inclusive
- Para una señal de N puntos separados 1/fs segundos se obtienen N puntos en la DFT separados fs/N hz
- El modulo de cada bin de la DFT al cuadrado es la potencia de la señal en esa frecuencia

3_clase/euler4.py



Repaso DFT

Analisis, Transformada discreta de Fourier

07m48s

- Definicion formal de la DFT
- Notar que hay un faltor de escala que puede ser 1/N, $1/\sqrt{N}$ o 1 dependiendo de cada implementacion
- Tambien podria optarse por cambiar el signo del exponente, lo importante es mantener luego la dualidad entre el factor de escala y el signo con la IDFT

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1$$

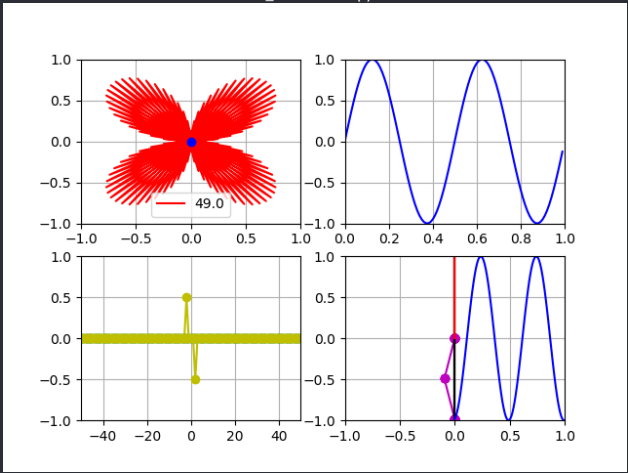
Síntesis

Como reconstruyo la señal en el tiempo

10m45s

- La IDFT toma como entrada un vector de N numeros complejos
- Se le llama sintesis a la reconstruccion de la señal en el tiempo
- DFT es completamente dual con la IDFT
- Se puede convertir una señal de t a f y f a t usando DFT e IDFT sin perder informacion

4_clase/euler6.py



Síntesis, Transformada inversa discreta de Fourier

IDFT

23m42s

- La IDFT es la suma de todos los bines en f evalueados en cada n
- La IDFT toma N numeros complejos y devuelve N numeros complejos
- Si el espectro es hermitico, la salida de la IDFT son N numeros reales
- El espectro es hermitico si los valores X(f) para f positivos son complejos conjugados de los valores de X(f) para f negativos
- Notar que para la DFT el exponente es negativo y para la IDFT es positivo

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N} kn} \qquad n = 0, \ldots, N-1$$

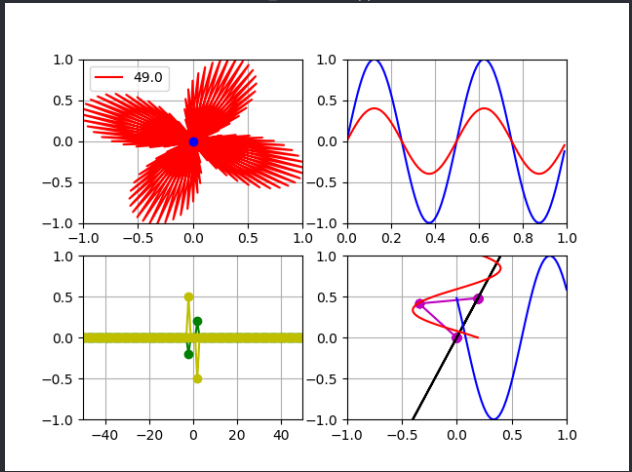
IDFT

Señales complejas como entrada?

48m40s

- La DFT puede recibir datos complejos
- La interpretacion en el tiempo de los datos complejos es arbitraria
- EL campo real podria ser audio y el imaginario temperatura por ej.
- La salida de la DFT que recibe N datos complejos, tambien devuelve N datos complejos
- Podria analizar 2 señales en un mismo calculo

4_clase/euler7.py

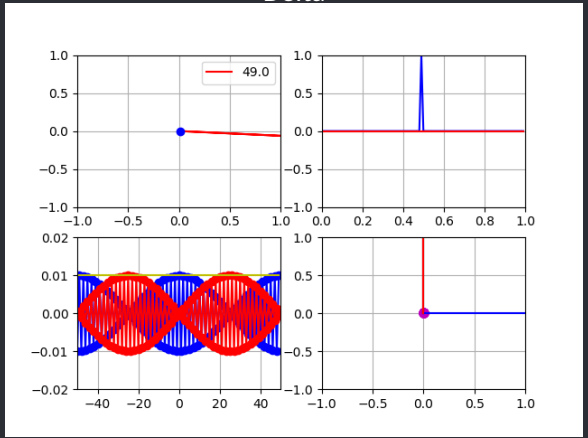


DFT<>IDFT

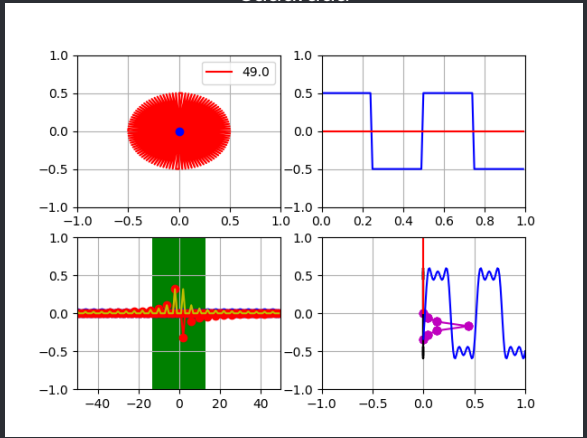
Transformadas relevantes

56m48s

Delta



Cuadrada

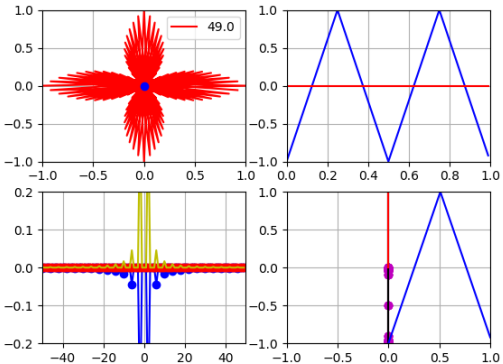


DFT<>IDFT

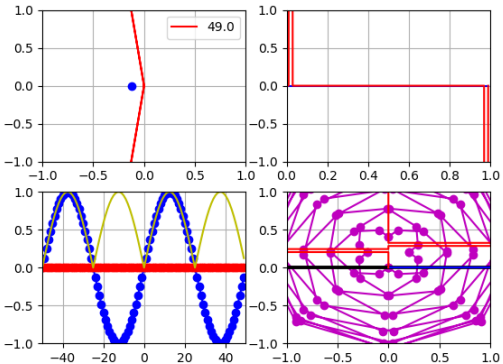
Transformadas relevantes

1h08m10s

Triangular



Conjugado

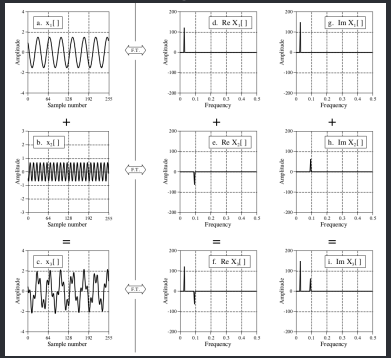


DFT<>IDFT

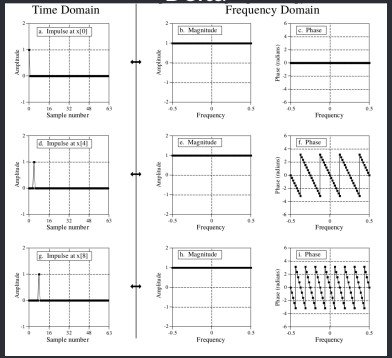
Transformadas relevantes

1h22m55s

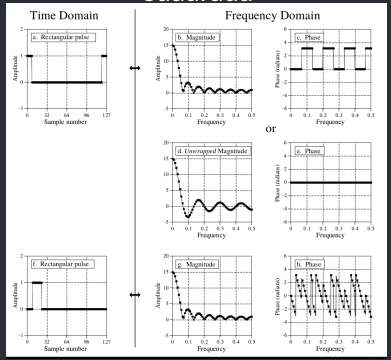
Sin



Delta



Cuadrada



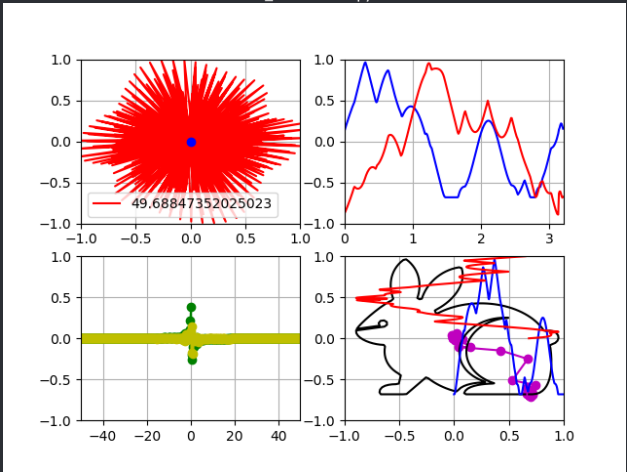
IDFT

Un conejo como entrada?

1h16m40s

- Ejemplo en donde la entrada a la DFT es compleja y representa los puntos de una figura en 2 dimensiones

4_clase/euler8.py



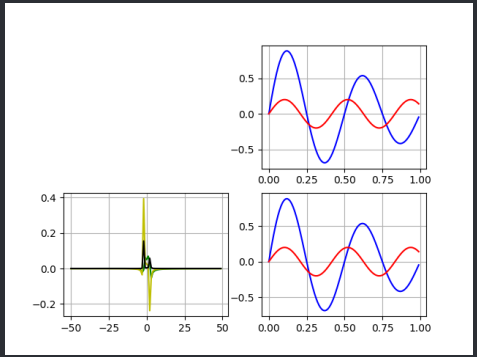
FFT-IFFT

Transformadas usando FFT en Python

1h26m00s

Ejemplo de codigo de como calcular la FFT e IFFT usando la biblioteca numpy

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
#-------------------------------------
fig         = plt.figure()
fs          = 100
N           = 100
frecIter    = 0
signalFrec  = 2
#-------------------------------------
tData       = np.arange(0,N/fs,1/fs)
nData       = np.arange(0,N,1)
circleFrec  = np.arange(-fs/2,fs/2,fs/N)
#-----------SIGNAL-------------------------
signalData = np.exp(-nData/fs)*np.sin(2*np.pi*signalFrec*nData*1/fs)+0.2j*np.sin(2*np.pi*1.2*
        signalFrec*nData*1/fs)
signalAxe  = fig.add_subplot(2,2,2)
signalRLn,signalILn,= plt.plot(tData,np.real(signalData),'b-',tData,np.imag(signalData),'r-')
signalAxe.grid(True)
#-----------FFT IFFT-----------------------
fftData  = np.fft.fft(signalData)
ifftData = np.fft.ifft(fftData)
fftData  = np.concatenate((fftData[N//2:N],fftData[0:N//2]))/N
#-----------FFT----------------------------
fftAxe          = fig.add_subplot(2,2,3)
fftRLn,fftILn,fftAbsLn = plt.plot(circleFrec,np.real(fftData),'g-' ,circleFrec,np.imag(
        fftData),'y-' ,circleFrec,np.abs(fftData)**2,'k-')
fftAxe.grid(True)
#----------IFFT---------------------------
ifftAxe       = fig.add_subplot(2,2,4)
penRLn,penILn = plt.plot(tData,np.real(ifftData),'b-',tData,np.imag(ifftData),'r-')
ifftAxe.grid(True)
#-------------------------------------
plt.show()
```



Cambio de tema?

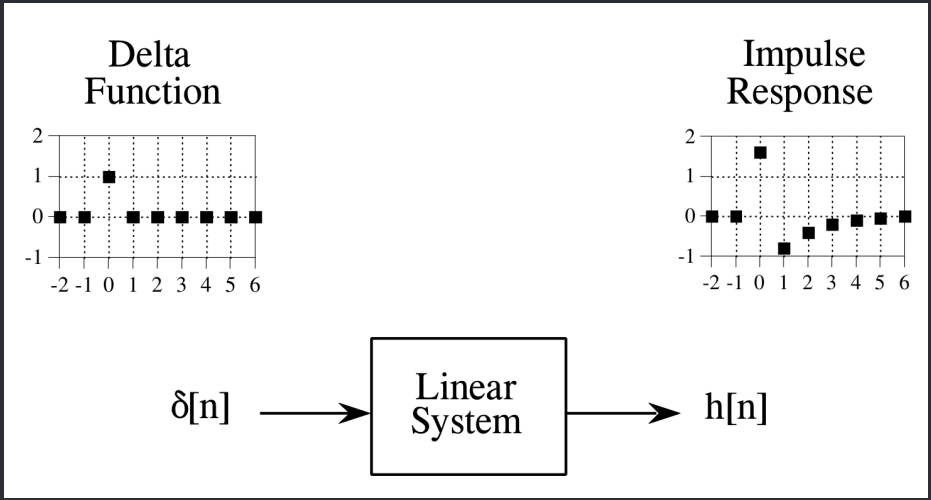
1h50m40s

1. Función delta
2. Respuesta al impulso
3. Convolución

Función delta

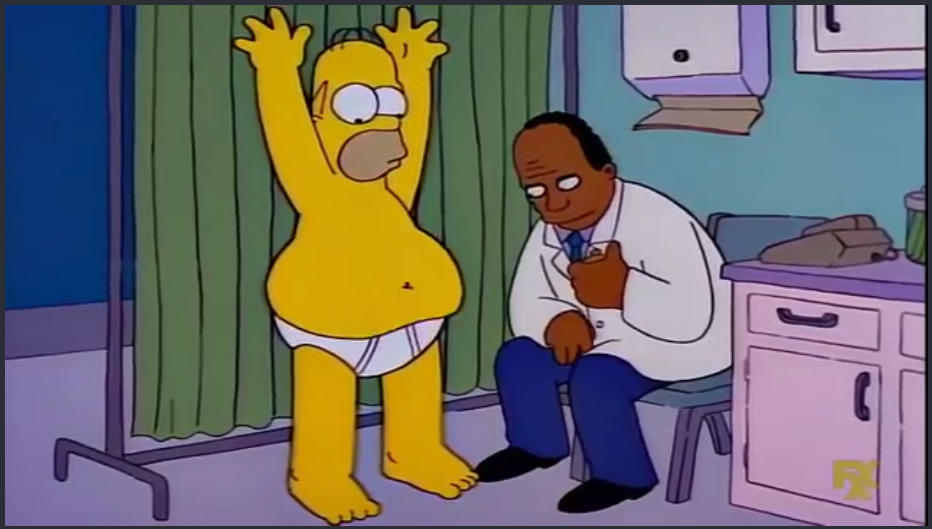
Respuesta al impulso

1h50m50s



Todo esta en la respuesta al impulso

1h53m50s



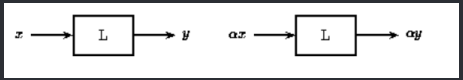
Repaso Sistemas

1h54m58s

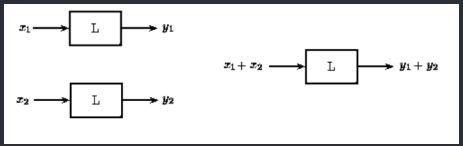
Linealidad

Un sistema es lineal cuando su salida depende linealmente de la entrada. Satisface el principio de superposición.

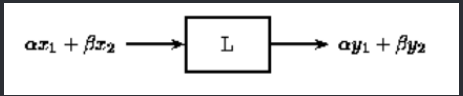
escalado



adición



superposición



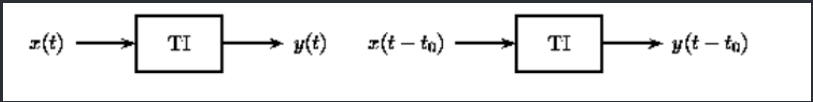
$y(t) = e^{x(t)}$

$y(t) = \frac{1}{2}x(t)$

1h57m52s

Invariantes en el tiempo

Un sistema es invariante en el tiempo cuando la salida para una determinada entrada es la misma sin importar el tiempo en el cual se aplica la entrada



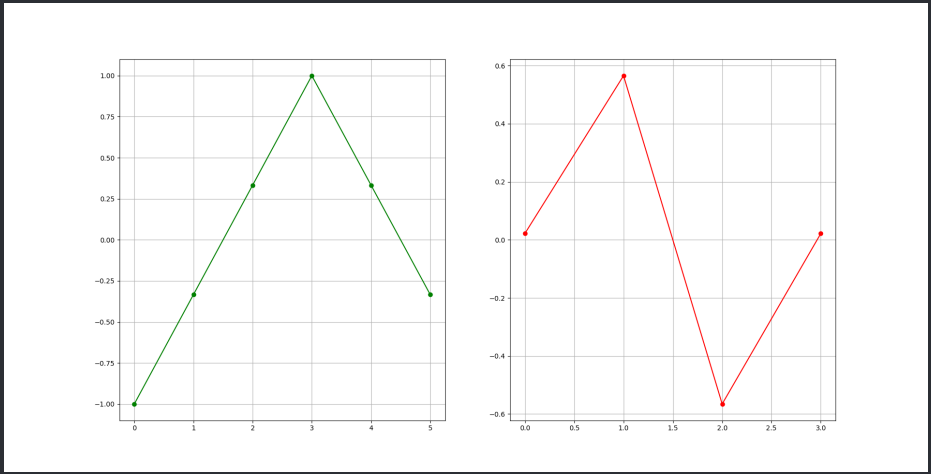
$$y(t) = x(t) * \cos(t)$$

$$y(t) = \cos(x(t))$$

Convolución

Señal vs h(n)

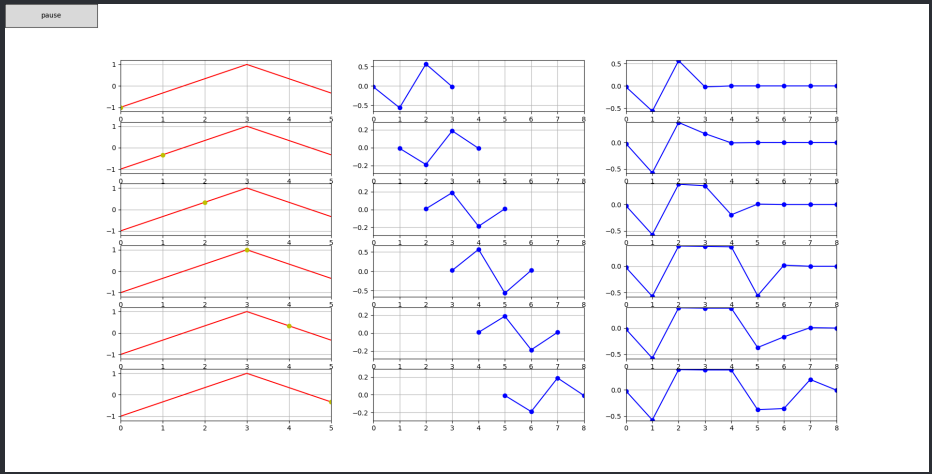
1h58m57s



Convolución

Descomposición delta

2h00m00s



Convolución

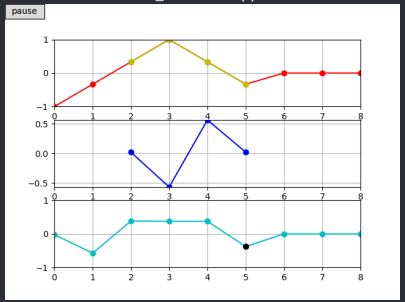
Respuesta al impulso

2h18m00s

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sc
from matplotlib.animation import FuncAnimation
from buttons import buttonOnFigure
#-------------------------------------
fig = plt.figure()
fs  = 6
N   = 6
xFrec = 1
hData,=np.load("4_clase/hi_pass_short.npy").astype(
        float)
hData=np.insert(hData,0,hData[-1]) #ojo que pydfa
        me guarda 1 dato menos...
hData[2]=1
M=len(hData)
#-------------------------------------
def x(f,n):
    return 1*sc.sawtooth(2*np.pi*xFrec*n/fs,0.5)

tData             = np.arange(-M+1,N+M-1,1)
xData             = np.zeros(N+2*(M-1))
xData[M-1:N+M-1] = x(xFrec,tData[M-1:N+M-1])
xAxe              = fig.add_subplot(3,1,1)
xLn,xHighLn       = plt.plot(tData,xData,'r-o'
        ,[],[],'y-o')
xAxe.grid(True)
xAxe.set_xlim(0,M+N-2)
xAxe.set_ylim(np.min(xData),np.max(xData))
#-------------------------------------
```

```
hAxe = fig.add_subplot(3,1,2)
hLn, = plt.plot([],[],'b-o')
hAxe.grid(True)
hAxe.set_xlim(0,M+N-2)
hAxe.set_ylim(np.min(hData),np.max(hData))
#-------------------------------------
yAxe      = fig.add_subplot(3,1,3)
yLn,yDotLn = plt.plot([],[],'c-o',[],[],'ko')
yAxe.grid(True)
yAxe.set_xlim(0,M+N-2)
yAxe.set_ylim(np.min(xData),np.max(xData))
yData=[]
#-------------------------------------
def init():
    global yData
    yData=np.zeros(N+M-1)
    return hLn,xLn,xHighLn,yLn,yDotLn

def update(i):
    global yData
    t=np.linspace(-M+1+i,i,M,endpoint=True)
    yData[i]=np.sum(xData[i:i+M]*hData[::-1])
    xHighLn.set_data(t,xData[i:i+M])
    hLn.set_data(t,hData[::-1])
    yLn.set_data(tData[M-1:],yData)
    yDotLn.set_data(tData[M-1+i],yData[i])
    return hLn,xLn,xHighLn,yLn,yDotLn

ani=FuncAnimation(fig,update,M+N-1,init,interval
        =1000 ,blit=True,repeat=True)
plt.get_current_fig_manager().window.showMaximized
        ()
b=buttonOnFigure(fig,ani)
plt.show()
```

4_clase/conv1.py



Funcion delta

Respuesta al impulso

2h16m40s

$$x[n] * h[n] = y[n]$$

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k]$$

Filtrado

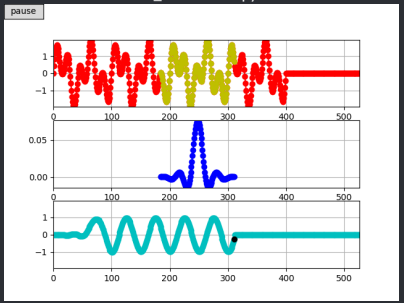
Pasa bajos

2h23m00s

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sc
from matplotlib.animation import FuncAnimation
from buttons import buttonOnFigure
#-------------------------------------
fig = plt.figure()
fs  = 100
N   = 400
xFrec = 3
hData,=np.load("4_clase/low_pass.npy").astype(float
      )
#hData,=np.load("4_clase/diferenciador.npy").astype
       (float)
M=len(hData)
#-------------------------------------
def x(f,n):
    #return 1*sc.sawtooth(2*np.pi*n/fs,0.5)
    return np.sin(2*np.pi*2*n*1/fs)+np.sin(2*np.pi
           *5*n*1/fs)
#-------------------------------------
tData            = np.arange(-M+1,N+M-1,1)
xData            = np.zeros(N+2*(M-1))
xData[M-1:N+M-1] = x(xFrec,tData[M-1:N+M-1])
xAxe             = fig.add_subplot(3,1,1)
xLn,xHighLn      = plt.plot(tData,xData,'r-o'
       ,[],[],'y-o')
xAxe.grid(True)
xAxe.set_xlim(0,M+N-2)
xAxe.set_ylim(np.min(xData),np.max(xData))
#-------------------------------------
```

```
hAxe = fig.add_subplot(3,1,2)
hLn, = plt.plot([],[],'b-o')
hAxe.grid(True)
hAxe.set_xlim(0,M+N-2)
hAxe.set_ylim(np.min(hData),np.max(hData))
#-------------------------------------
yAxe       = fig.add_subplot(3,1,3)
yLn,yDotLn = plt.plot([],[],'c-o',[],[],'ko')
yAxe.grid(True)
yAxe.set_xlim(0,M+N-2)
yAxe.set_ylim(np.min(xData),np.max(xData))
yData=[]
#-------------------------------------
def init():
    global yData
    yData=np.zeros(N+M-1)
    return hLn,xLn,xHighLn,yLn,yDotLn

def update(i):
    global yData
    t=np.linspace(-M+1+i,i,M,endpoint=True)
    yData[i]=np.sum(xData[i:i+M]*hData[::-1])
    xHighLn.set_data(t,xData[i:i+M])
    hLn.set_data(t,hData[::-1])
    yLn.set_data(tData[M-1:],yData)
    yDotLn.set_data(tData[M-1+i],yData[i])
    return hLn,xLn,xHighLn,yLn,yDotLn

ani=FuncAnimation(fig,update,M+N-1,init,interval=10
        ,blit=True,repeat=True)
plt.get_current_fig_manager().window.showMaximized
       ()
b=buttonOnFigure(fig,ani)
plt.show()
```

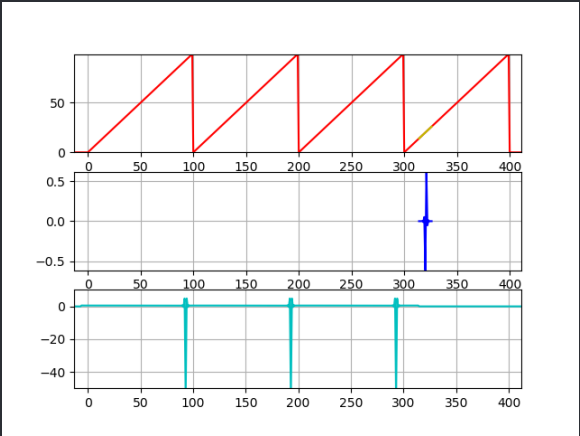
4_clase/conv2.py



Filtrado

diferenciador

2h37m50s



Convolución

Análisis en la CIAA

2h41m20s

```
#include "sapi.h"
#include "arm_math.h"
#include "arm_const_structs.h"
#define MAX_FFT_LENGTH 2048         // maxima longitud para la fft y chunk de samples
#define BITS 10                     // cantidad de bits usado para cuantizar
#define FIR_LENGTH 162
int16_t fftLength = 32;             // longitud de la fft y samples variable
int16_t adc [ MAX_FFT_LENGTH ];     // guarda los samples
q15_t fftIn [ MAX_FFT_LENGTH ];     // guarda copia de samples en Q15 como in para la fft.La fft corrompe
      los datos de la entrada!
q15_t fftOut[ MAX_FFT_LENGTH*2 ];   // salida de la fft
q15_t fftMag[ MAX_FFT_LENGTH/2+1 ]; // magnitud de la FFT
//low_pass 1000hz 127
//q15_t fir[ FIR_LENGTH]={ 22, 21, 24, 19, 6, -14, -40, -66, -83, -86, -73, -43, -3, 36, 65, 73, 56, 16,
          -35, -84, -115, -114, -77, -13, 63, 128, 161, 147, 84, -13, -119, -200, -227, -186, -81, 62,
          204, 299, 310, 224, 56, -152, -341, -445, -422, -250, 11, 320, 575, 687, 593, 287, -176, -683,
          -1083, -1220, -977, -305, 755, 2074, 3450, 4656, 5480, 5772, 5480, 4656, 3450, 2074, 755, -305,
          -977, -1220, -1083, -683, -176, 287, 593, 687, 575, 320, 11, -259, -422, -445, -341, -152, 56,
          224, 310, 299, 204, 62, -81, -186, -227, -200, -119, -13, 84, 147, 161, 128, 63, -13, -77, -114,
          -115, -84, -35, 16, 56, 73, 65, 36, -3, -43, -73, -86, -83, -66, -40, -14, 6, 19, 24, 21, 22};
//bandpass 440hz 162
q15_t fir[ FIR_LENGTH]={ 1, 0, 0, -0, -2, -5, -9, -14, -19, -25, -29, -32, -33, -30, -24, -14, 0, 17, 37,
          57, 76, 91, 101, 104, 98, 83, 60, 31, -1, -36, -68, -95, -113, -121, -117, -103, -81, -54, -26,
          -3, 11, 13, 2, -22, -58, -99, -140, -172, -186, -175, -135, -62, 41, 170, 315, 461, 593, 694,
          748, 742, 667, 522, 312, 46, -254, -566, -864, -1118, -1303, -1396, -1383, -1260, -1028, -704,
          -310, 123, 561, 967, 1307, 1551, 1679, 1679, 1551, 1307, 967, 561, 123, -310, -704, -1028,
          -1260, -1383, -1396, -1303, -1118, -864, -566, -254, 46, 312, 522, 667, 742, 748, 694, 593, 461,
           315, 170, 41, -62, -135, -175, -186, -172, -140, -99, -58, -22, 2, 13, 11, -3, -26, -54, -81,
          -103, -117, -121, -113, -95, -68, -36, -1, 31, 60, 83, 98, 104, 101, 91, 76, 57, 37, 17, 0, -14,
           -24, -30, -33, -32, -29, -25, -19, -14, -9, -5, -2, -0, 0, 0, 1};
q15_t firOut [ MAX_FFT_LENGTH+FIR_LENGTH+1 ];
uint32_t maxIndex = 0;              // indexador de maxima energia por cada fft
q15_t maxValue    = 0;              // maximo valor de energia del bin por cada fft
arm_rfft_instance_q15 S;
uint16_t sample   = 0;              // contador para samples
int main ( void ) {
   boardConfig        (                    );
   uartConfig         ( UART_USB, 460800   );
   adcConfig          ( ADC_ENABLE         );
   cyclesCounterInit  ( EDU_CIAA_NXP_CLOCK_SPEED );
   while(1) {
```

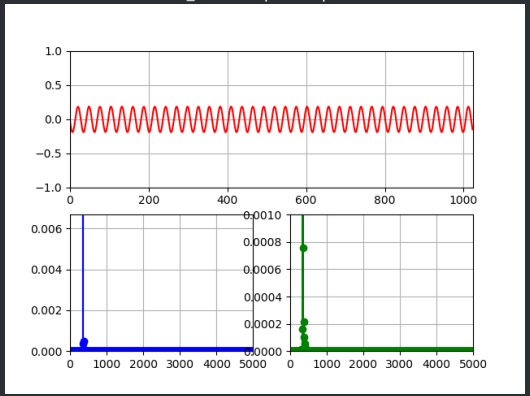
```
cyclesCounterReset();                                          // inicializa el
        conteo de ciclos de reloj
uartWriteByteArray ( UART_USB ,(uint8_t* )&adc[sample]    ,sizeof(adc[0]) );   // envia el sample
        ANTERIOR
uartWriteByteArray ( UART_USB ,(uint8_t* )&fftOut[sample] ,sizeof(fftOut[0])); // envia la fft del
        sample ANTERIO
//TODO hay que mandar fftLength/2 "+1" y solo estoy mandando fftLength/2. revisar
adc[sample]   =(((int16_t )adcRead(CH1)-512)>>(10-BITS))<<(6+10-BITS);         // PISA el sample que
        se acaba de mandar con una nueva muestra
fftIn[sample] = adc[sample];                                   // copia del adc
        porque la fft corrompe el arreglo de entrada
if ( ++sample==fftLength ) {                                   // si es el ultimo
   sample = 0;                                                 // arranca de nuevo
   uartWriteByteArray ( UART_USB ,(uint8_t* )&maxValue ,2);
   uartWriteByteArray ( UART_USB ,(uint8_t* )&maxIndex ,2);
   uartWriteByteArray ( UART_USB ,"header" ,6 );               // manda el header
           que casualmente se llama "header" con lo que arranca una nueva trama
   uartWriteByteArray ( UART_USB ,(uint8_t* )&fftLength ,sizeof(fftLength));  // manda el largo de
           la fft que es variable
   arm_conv_fast_q15(fftIn,fftLength,fir,FIR_LENGTH,firOut);
   arm_rfft_init_q15 ( &S ,fftLength ,0 ,1 );                  // inicializa una
           estructira que usa la funcion fft para procesar los datos. Notar el /2 para el largo
   arm_rfft_q15      ( &S ,&firOut[FIR_LENGTH]     ,fftOut    );                  // por
           fin.. ejecuta la rfft REAL rfft
   arm_cmplx_mag_squared_q15 ( fftOut ,fftMag ,fftLength/2+1 );
   arm_max_q15 ( fftMag ,fftLength/2+1 ,&maxValue ,&maxIndex );
   gpioToggle( LEDR);
   if ( gpioRead(TEC1 )==0) {
      gpioToggle(LEDB);
```

Convolución

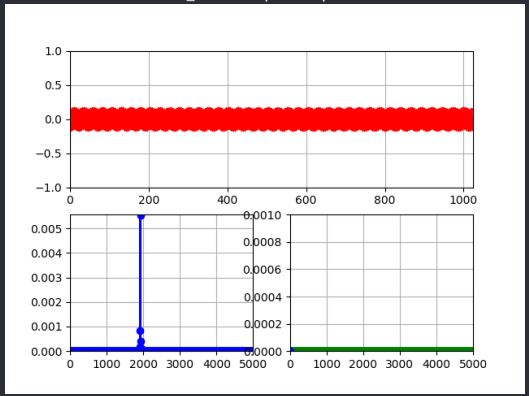
Análisis en la CIAA

2h21m11s

4_clase/ciaa/psf2/src/psf.c



4_clase/ciaa/psf2/src/psf.c



Bibliografía

Libros, links y otro material

[1] *ARM CMSIS DSP*.
https://arm-software.github.io/CMSIS_5/DSP/html/index.html

[2] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. Second Edition, 1999.

[3] *Grant Sanderson*
https://youtu.be/spUNpyF58BY

[4] *Interactive Mathematics Site Info*.
https://www.intmath.com/fourier-series/fourier-intro.php