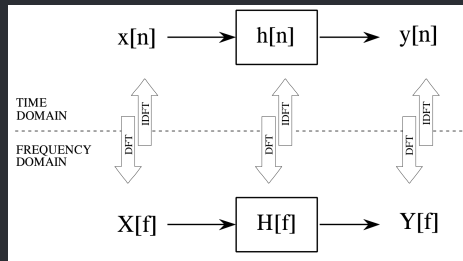


# Procesamiento de señales, fundamentos

Maestría en sistemas embebidos  
Universidad de Buenos Aires  
MSE 5Co2020

## Clase 5 - Aplicaciones de DFT

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



2020-06-02

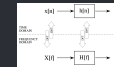
Procesamiento de señales, fundamentos

## Procesamiento de señales, fundamentos

Maestría en sistemas embebidos  
Universidad de Buenos Aires  
MSE 5Co2020

### Clase 5 - Aplicaciones de DFT

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



# Repaso Convolución

## Multiplicacion?!

## Algoritmo de Multiplicacion de 2do grado

	1	2
	3	4
<hr/>		
	4	8
3	6	0
<hr/>		
3	10	8

## Procesamiento de señales, fundamentos

2020-06-02

## Repaso Convolución

## Repaso Convolución

Multiplicacion?!

Algoritmo de Multiplicación de 2do grado

	1	2
	3	4
<hr/>		
	4	8
3	6	0
<hr/>		
3	10	8

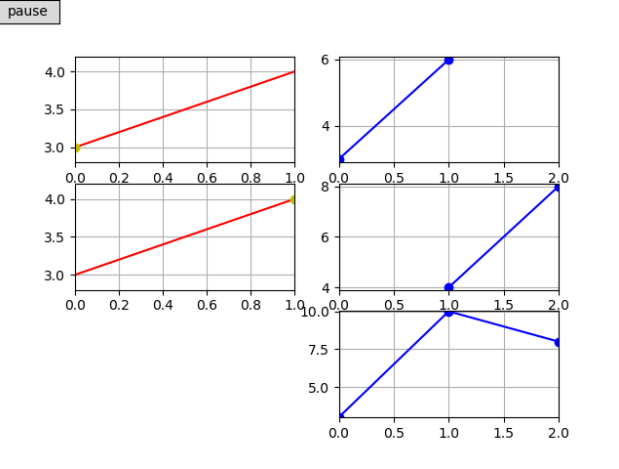
- no hay que lanzar nada
- explicar 3 manera de multiplicar un numero
- darle forma de respuesta al impulso y senial

# Repaso Convolución

## Descomposición delta

SUma deltas desplazadas

1	2	0
3	0	0
<hr/>		
3	6	0
<hr/>		
0	1	2
0	4	0
<hr/>		
3	6	0
0	4	8
<hr/>		
3	10	8

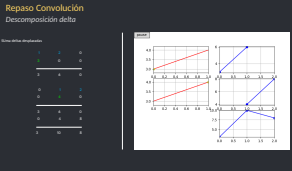


2020-06-02

Procesamiento de señales, fundamentos

Repaso Convolución

- lanzar conv\_as\_multiply1
- muestro la misma cuenta con señales



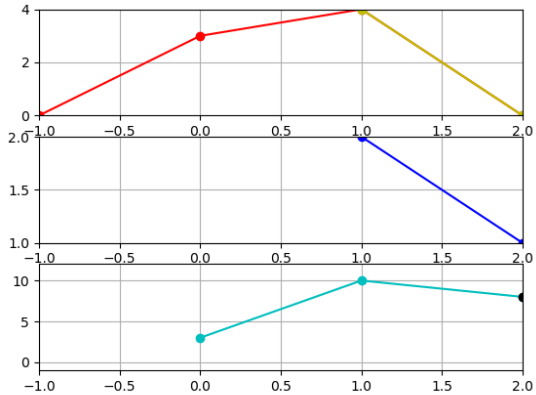
# Repaso Convolución

## Convolucion formal

Convolucion

2	1	0	0
0	3	4	0
<hr/>			
	3	0	0
<hr/>			
0	2	1	0
0	3	4	0
<hr/>			
	3	10	0
<hr/>			
0	0	2	1
0	3	4	0
<hr/>			
	3	10	8

pause

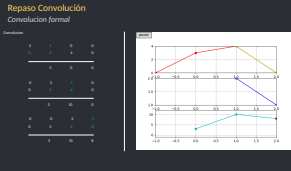


2020-06-02

Procesamiento de señales, fundamentos

Repaso Convolución

- lanzar conv\_as\_multiply2
- mostro la misma cuenta con señales



# Repaso Convolución

## Multiplicación?!

Algoritmo de Multiplicacion

	1	2
	3	4
<hr/>		
	4	8
3	6	0
<hr/>		
3	10	8

SUma deltas desplazadas

1	2	0
3	0	0
<hr/>		
3	6	0
0	4	8
<hr/>		
3	10	8

Convolucion

2	1	0	0
0	3	4	0
<hr/>			
	3	0	0
0	2	1	0
0	3	4	0
<hr/>			
	3	10	0
0	0	2	1
0	3	4	0
<hr/>			
	3	10	8

2020-06-02

Procesamiento de señales, fundamentos

Repaso Convolución

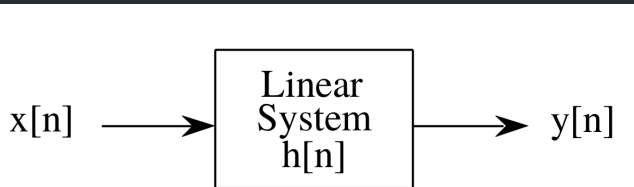
- no hay que lanzar nada
- explicar 3 manera de multiplicar un numero
- darle forma de respuesta al impulso y senial



# Repaso Convolucion

## Propiedades

- Conmutativa
- Distributiva
- Asociativa



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

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

2020-06-02

Procesamiento de señales, fundamentos

Repaso Convolucion

Repaso Convolucion

Propiedades

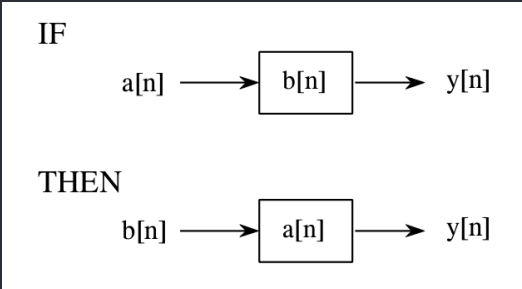
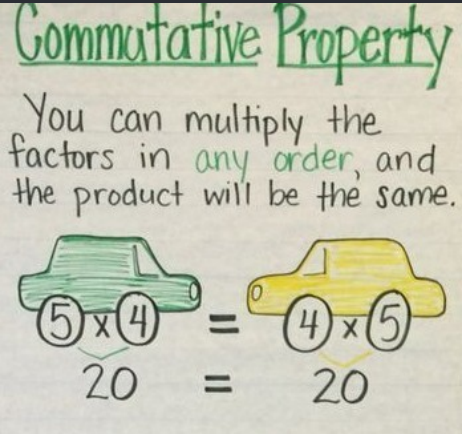
- Conmutativa
- Distributiva
- Asociativa

```
graph LR; x[n] --> LS[Linear System h[n]]; LS --> y[n]
```

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

# Repaso Multiplicacion

## Propiedad conmutativa



2020-06-02

Procesamiento de señales, fundamentos

Repaso Multiplicacion



# Repaso Multiplicacion

## Propiedad asociativa

Associative Property

You can group the factors in different ways, and the product will be the same.

$(3 \times 4) \times 2 = 24$

$3 \times (4 \times 2) = 24$

2020-06-02

## Repaso Multiplicacion

Repaso Multiplicacion  
Propiedad asociativa

Associative Property

You can group the factors in different ways, and the product will be the same.

$(3 \times 4) \times 2 = 24$

$3 \times (4 \times 2) = 24$

IF

$x[n] \rightarrow h_1[n] \rightarrow h_2[n] \rightarrow y[n]$

THEN

$x[n] \rightarrow h_2[n] \rightarrow h_1[n] \rightarrow y[n]$

ALSO

$x[n] \rightarrow h_1[n] * h_2[n] \rightarrow y[n]$



# Repaso Multiplicacion

## Propiedad distributiva

Distributive Property

A multiplication fact can be broken up into the sum of two other multiplication facts.

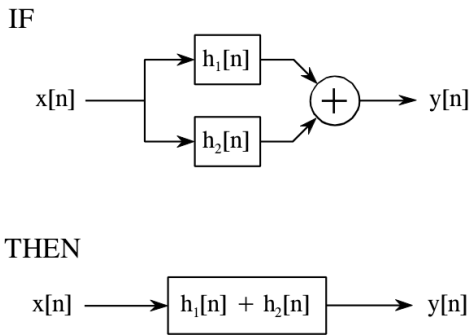
$23 \times 2 = ?$

$(20 + 3) \times 2 =$

$(20 \times 2) + (3 \times 2) =$

$40 + 6 = 46$

Break it into numbers that are easier to multiply with mental math.



2020-06-02

Repaso Multiplicacion

Repaso Multiplicacion

Propiedad distributiva

Distributive Property

A multiplication fact can be broken up into the sum of two other multiplication facts.

$23 \times 2 = ?$

$(20 + 3) \times 2 =$

$(20 \times 2) + (3 \times 2) =$

$40 + 6 = 46$

Break it into numbers that are easier to multiply with mental math.

IF

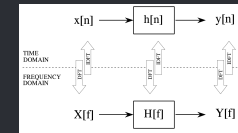
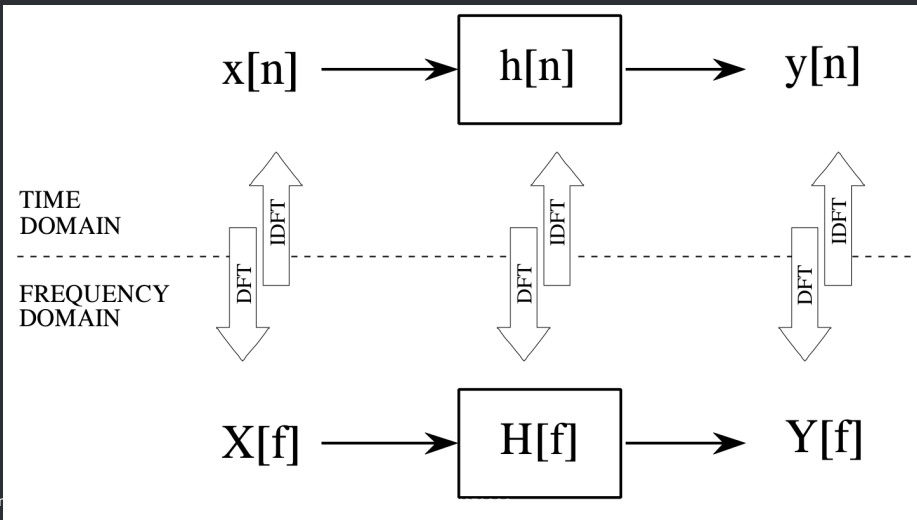
$x[n] \rightarrow \begin{cases} h_1[n] \\ h_2[n] \end{cases} \rightarrow \oplus \rightarrow y[n]$

THEN

$x[n] \rightarrow h_1[n] + h_2[n] \rightarrow y[n]$

# Convolución vs Multiplicación

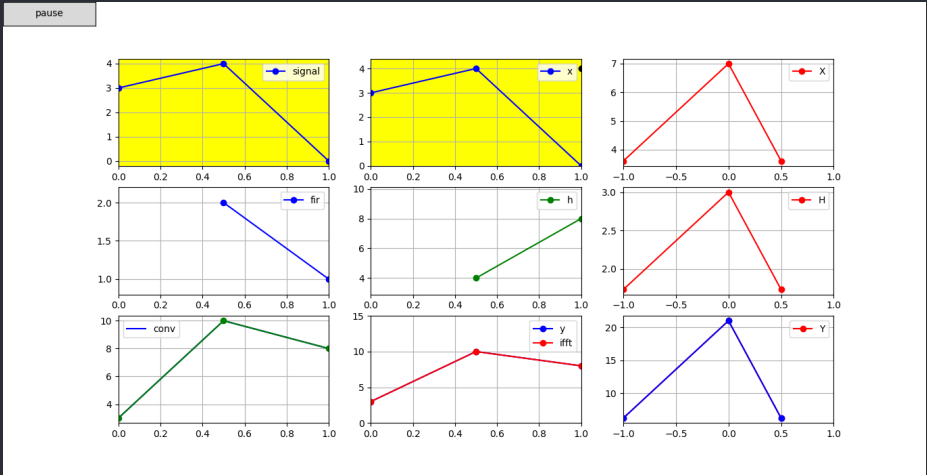
## Teorema de la convolución



- explicar que cuando la entrada en t es la delta estamos simulando barrido en f, porque la delta tiene todas las f
- luego en F la entrada son círculos rotando multiplicados por círculos rotando, amplitud se escala y fase(exponente) se suman, se corre la fase
- no hacemos la demostración del teorema, sino que lo probamos prácticamente
- explicar la conclusion y el teorema de la convolución

# Multiplicación con DFT

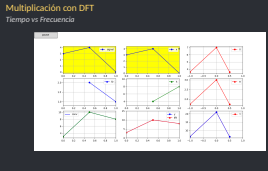
Tiempo vs Frecuencia



2020-06-02

Procesamiento de señales, fundamentos

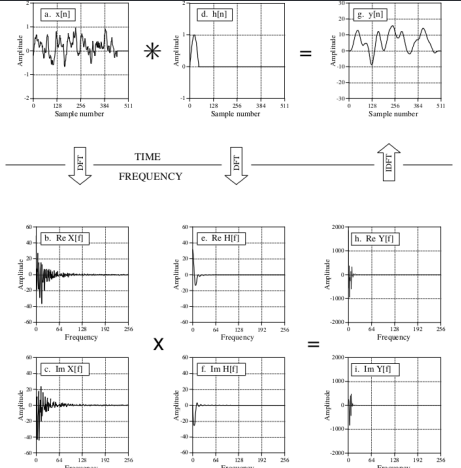
- Convolución vs Multiplicación
  - Multiplicación con DFT



- explicar multiplicación usando DFT

# Convolución vs Multiplicación

## Teorema de la convolución



2020-06-02

### Procesamiento de señales, fundamentos

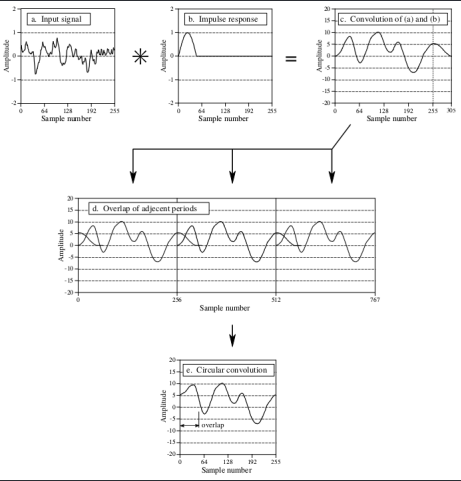
- Convolución vs Multiplicación
- Convolución vs Multiplicación



- explicar la conclusion y el teorema de la convolución
- explicar que dado h r y(t) podemos dividir en frec y obtener x()

# Convolución vs Multiplicación

## Convolución circular

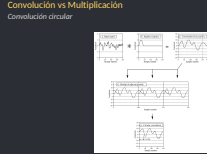


2020-06-02

Procesamiento de señales, fundamentos

└─ Convolución vs Multiplicación

└─ Convolución vs Multiplicación



- explicar el efecto de la convolución circular

# Convolución vs Multiplicación

## Teorema de la convolución

$$x * y = \text{DTFT}^{-1} \left[ \text{DTFT}\{x\} \cdot \text{DTFT}\{y\} \right]$$

2020-06-02

Procesamiento de señales, fundamentos

└─ Convolución vs Multiplicación

└─ Convolución vs Multiplicación

Convolución vs Multiplicación  
Teorema de la convolución

$$x * y = \text{DTFT}^{-1} \left[ \text{DTFT}\{x\} \cdot \text{DTFT}\{y\} \right]$$

- explicar el tema de la DTFT y la transformada circular

# Filtrado

## Pasabajos

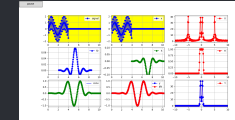
2020-06-02

Procesamiento de señales, fundamentos

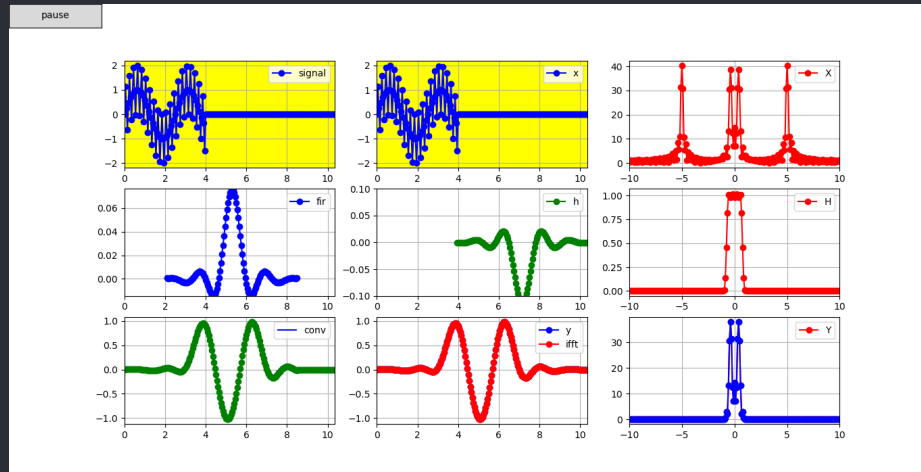
└─ Convolución vs Multiplicación

└─ Filtrado

Filtrado  
Pasabajos



- explicar ahora el uso de la convolucion en el filtrado
- a partir de 64 puntos de fir conviene FFT, por menos conviene convolution en tiempo



# Filtrado

## Pasaaltos

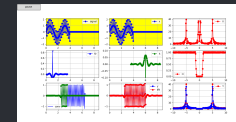
2020-06-02

Procesamiento de señales, fundamentos

└─ Convolución vs Multiplicación

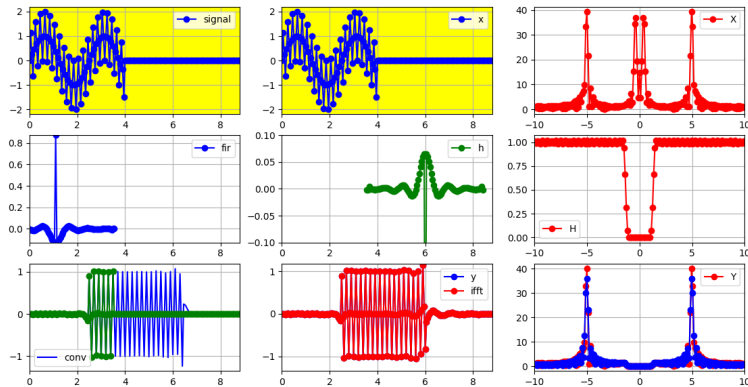
└─ Filtrado

Filtrado  
Pasaaltos



- explicar ahora el uso de la convolucion en el filtrado
- a partir de 64 puntos de fir conviene FFT, por menos conviene convolution en tiempo

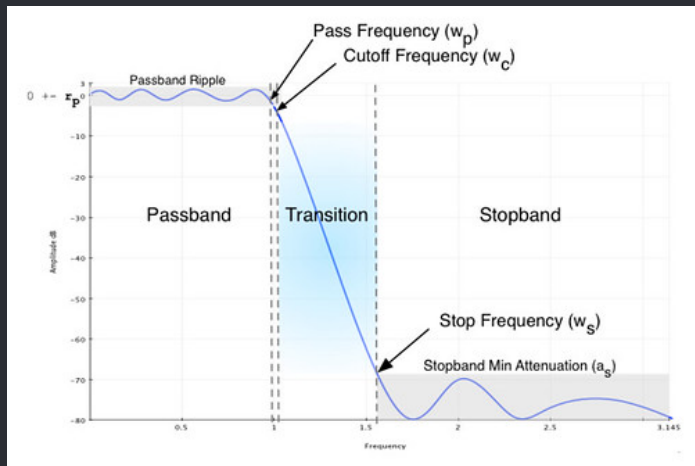
pause





# Filtrado

## Definición



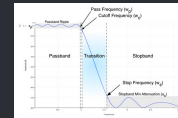
2020-06-02

Procesamiento de señales, fundamentos

└─ Convolución vs Multiplicación

└─ Filtrado

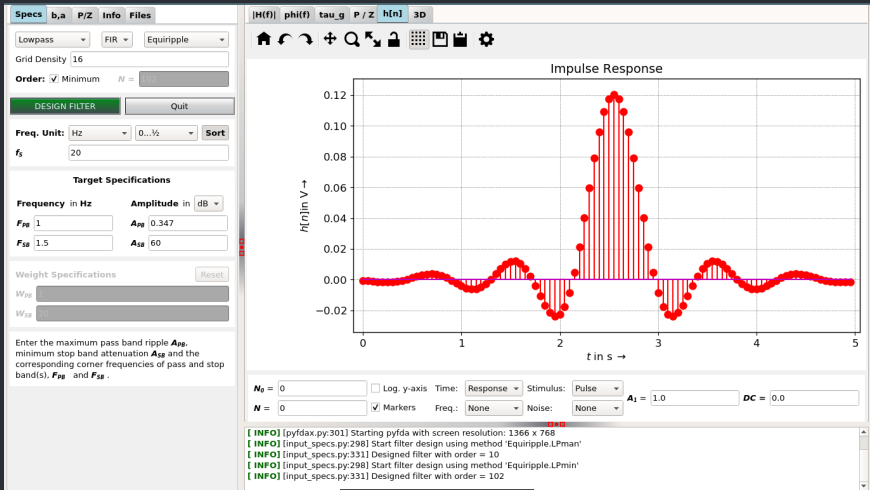
Filtrado  
Definición



- explicar las zonas de los filtros, tipos de filtro
- relación de compromiso entre ripple y bandas, etc

# Filtrado

PyFDA `/opt/anaconda3/bin/pyfdax`

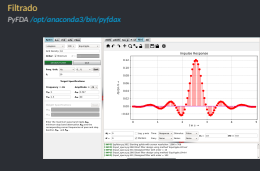


2020-06-02

## Procesamiento de señales, fundamentos

└─ Convolución vs Multiplicación

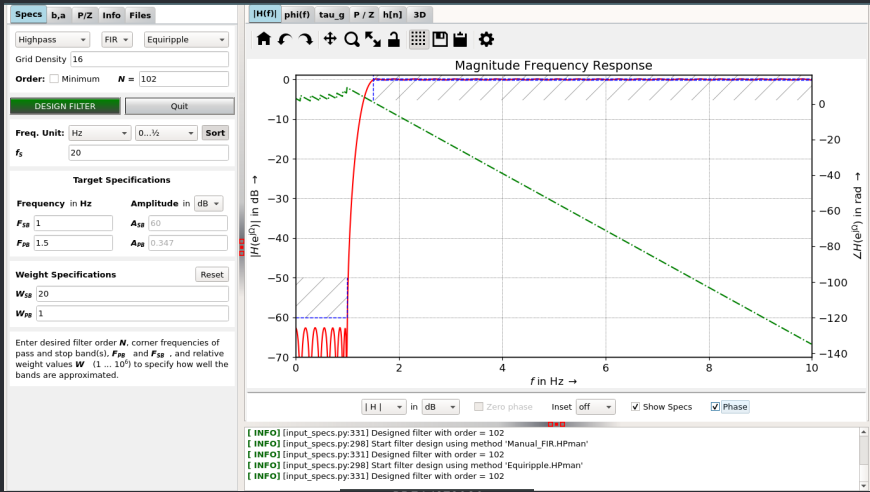
└─ Filtrado



- explicar ahora el uso de la convolution en el filtrado
- a partir de 64 puntos de fir conviene FFT, por menos conviene convolution en tiempo

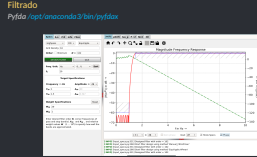
# Filtrado

Pyfda /opt/anaconda3/bin/pyfdax



2020-06-02

- Procesamiento de señales, fundamentos
  - Convolución vs Multiplicación
    - Filtrado



- explicar ahora el uso de la convolution en el filtrado
- a partir de 64 puntos de fir conviene FFT, por menos conviene convolucion en tiempo

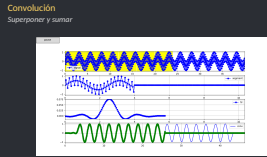
# Convolución

Superponer y sumar

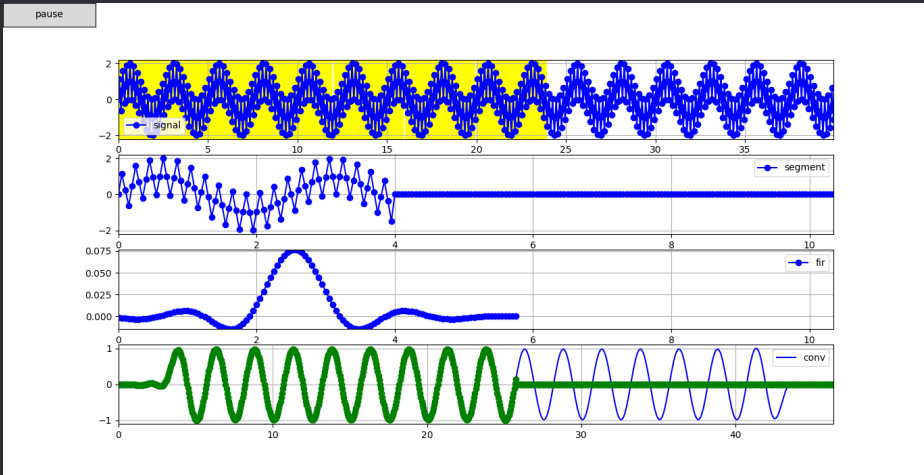
2020-06-02

Procesamiento de señales, fundamentos

- Convolución vs Multiplicación
  - Convolución



- explicar el detalle de overlap para sumar

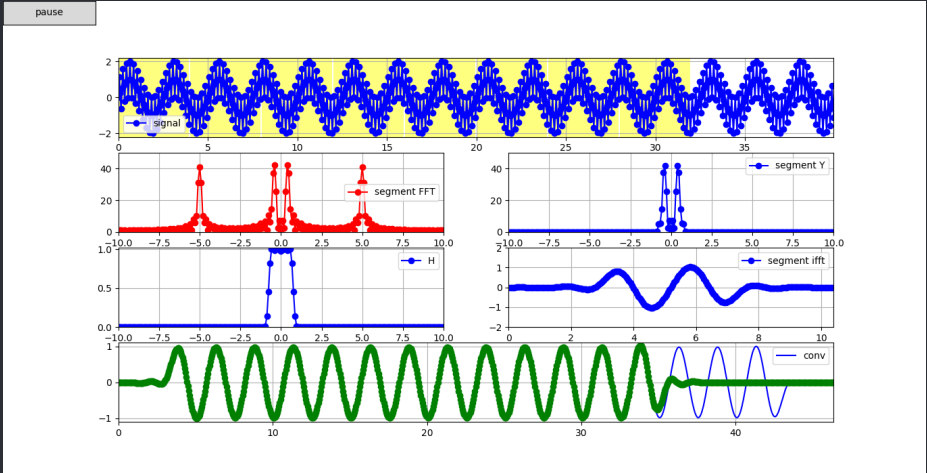
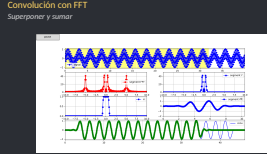


# Convolución con FFT

Superponer y sumar

2020-06-02

- Procesamiento de señales, fundamentos
  - Convolución vs Multiplicación
    - Convolución con FFT



- explicar el detalle de overlap para sumar

# Filtrado con CIAA

## Conversor PyFDA a fir.h para C

### Código en Python para convertir los coeficientes del fir extendidos en Q1.15 en C

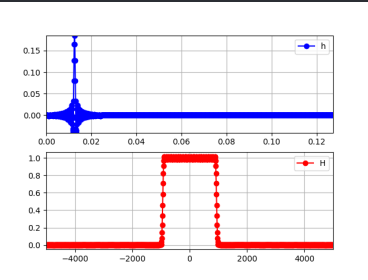
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sc

#-----
fig = plt.figure()
fs = 10000
N = 1024
firData=np.load("5_clase/low_pass_1k.npy").astype(float)

firData=np.insert(firData,0,firData[-1]) #ojo que pydfa
me guarda 1 dato menos...
M = len(firData)

firExtendedData=np.concatenate((firData,np.zeros(N-1)))
impar=((N+M-1)%2)
#-----
tData=np.linspace(0,(N+M-1)/fs,N+M-1,endpoint=False)
fData=np.concatenate((np.linspace(-fs/2,0,(N+M-1)//2,
endpoint=False),\
np.linspace(0,fs/2,(N+M-1)//2+impar,endpoint=
False)))
#-----
firAxe = fig.add_subplot(2,1,1)
firLn, = plt.plot(tData,firExtendedData,'b-o',label="h"
)
firAxe.legend()
firAxe.grid(True)
firAxe.set_xlim(0,(N+M-2)/fs)
firAxe.set_ylim(np.min(firData),np.max(firData))
#-----
HData=np.fft.fft(firExtendedData)
circularHData=np.concatenate((HData[len(HData)//2+impar
:],HData[0:len(HData)//2+impar]))

HAXe = fig.add_subplot(2,1,2)
HLn, = plt.plot(fData,np.abs(circularHData),'r-o',label
="H")
HAXe.legend()
HAXe.grid(True)
HAXe.set_xlim(-fs/2,fs/2)
#-----
def convertToC(h,H,fileName):
    cFile = open(fileName,"w+")
    cFile.write("#define h_LENGTH {}\n".format(len(
        firData)))
    cFile.write("#define h_PADD_LENGTH {}\n".format(len(
        h)))
    cFile.write("#define H_PADD_LENGTH {}\n".format(len(
        H)))
    h*=2**15
    h=h.astype(np.int16)
    H*=2**15
    cFile.write("q15_t h[]={\n")
    for i in h:
        cFile.write("{},\n".format(i))
    cFile.write("};\n")
    cFile.write("q15_t H[]={\n")
    for i in H:
        cFile.write("{}},{},\n".format(np.real(i).astype
(np.int16),np.imag(i).astype(np.int16)))
    cFile.write("};\n")
    convertToC(firExtendedData,HData,"5_clase/ciaa/psf2/src/
        fir.h")
    plt.get_current_fig_manager().window.showMaximized()
    plt.show()
```



2020-06-02

### Procesamiento de señales, fundamentos

CIAA

Filtrado con CIAA

#### Filtrado con CIAA

Conversor PyFDA a fir.h para C

Código en Python para convertir los coeficientes del fir extendidos en Q1.15 en C

- mostrar como pasar de pyfda a C
- lanzar psf1
- probar distintos filtros y ver resultado
- hacer notar el efecto del padding

# Filtrado con CIAA

## Con padding y convolución



## Convolución en tiempo con padding en CIAA para filtrado

```
#include "sapi.h"
#include "arm_math.h"
#include "arm_const_structs.h"
#include "fir.h"

#define MAX_FFT_LENGTH 2048
#define BITS 10
int16_t fftLength = 512;
int16_t hLength = h LENGTH;
int16_t adc [ MAX_FFT_LENGTH];
q15_t x [ MAX_FFT_LENGTH];
q15_t fftOut [ ( MAX_FFT_LENGTH)*2 ];
q15_t fftMag [ ( MAX_FFT_LENGTH)/2+1 ];

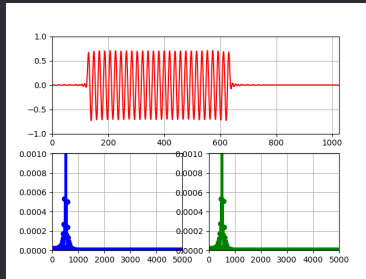
uint32_t maxIndex = 0;
q15_t maxValue = 0;
arm_rfft_instance_q15 S;
uint16_t convLength = 0;
uint16_t sample = 0;

int calcFftLength(int N,int M) {
    int convLength=N+M-1,i;
    for(i=MAX_FFT_LENGTH;i>=convLength;i+=1)
        ;
    return i<<1;
}

int sendStr(char A[],int N) { uartWriteByteArray ( UART_USB ,A ,N ); }
int sendBlock(q15_t A[],int N) { uartWriteByteArray ( UART_USB ,(uint8_t*)A ,2*N ); }
```

```
int main ( void ) {
    boardConfig ( );
}
```

```
adcConfig ( ADC_ENABLE );
cyclesCounterInit ( EDU_CIAA_NXP_CLOCK_SPEED );
while(1) {
    convLength=calcFftLength(fftLength,hLength);
    for(sample=0;sample<fftLength;sample++) {
        cyclesCounterReset();
        adc[sample] = (((int16_t)adcRead(CH1)-512)>>(10-BITS))<<(6+10-BITS);
        gpioToggle( LEDB);
        while(cyclesCounterRead()< 20400)
            ;
    }
    for(sample=fftLength;sample<convLength;sample++)
        adc[sample]=0;
    sendStr ( "header" ,6 );
    sendBlock ( &fftLength ,1 );
    sendBlock ( &convLength ,1 );
    arm_conv_fast_q15 ( adc,fftLength,h,convLength-fftLength+1,x );
    sendBlock ( x ,convLength );
    arm_rfft_init_q15 ( &S ,convLength ,0 ,1 );
    arm_rfft_q15 ( &S ,x ,fftOut );
    arm_cmplx_mag_squared_q15 ( fftOut ,fftMag ,convLength );
    arm_max_q15 ( fftMag ,convLength/2+1 ,&maxValue ,&maxIndex );
    sendBlock ( fftOut ,convLength );
    sendBlock ( &maxValue ,1 );
    sendBlock ( (q15_t*)&maxIndex ,1 );
    gpioToggle( LEDR);
}
```

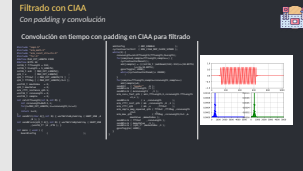


## Procesamiento de señales, fundamentos

CIAA

Filtrado con CIAA

2020-06-02



- mostrar como pasar de pyfda a C
- lanzar psf1
- probar distintos filtros y ver resultado
- hacer notar el efecto del padding

# Filtrado con CIAA

Con padding y FFT



## Convolución en tiempo con padding en CIAA para filtrado

```
#include "sapi.h"
#include "arm_math.h"
#include "arm_const_structs.h"
#include "fir.h"

#define MAX_FFT_LENGTH 1024
int16_t fftLength = 128;
int16_t hLength = h LENGTH;
int16_t adc [ MAX_FFT_LENGTH ];
q15_t x [ MAX_FFT_LENGTH ];
q15_t fftOut[ 2* MAX_FFT_LENGTH ];
q15_t H [ 2* MAX_FFT_LENGTH ];
q15_t hTemp [ 2* MAX_FFT_LENGTH ];

uint32_t maxIndex = 0;
q15_t maxValue = 0;
arm_rfft_instance_q15 S;
uint16_t convLength = 0;
uint16_t sample = 0;

int calcFftLength(int N,int M) {
    int convLength=N*M-1,i;
    for(i=MAX_FFT_LENGTH;i>=convLength;i>=>1)
        ;
    return i<=1;
}

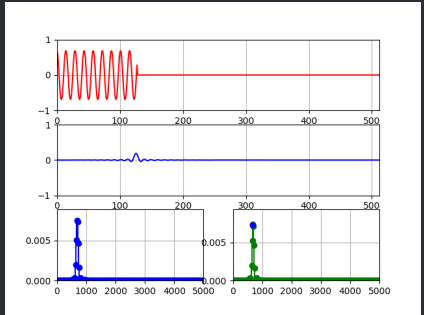
int sendStr(char A[],int N) { uartWriteByteArray ( UART_USB ,A
,N); }

int sendBlock(q15_t A[],int N) { uartWriteByteArray ( UART_USB
,(uint8_t*) A ,2*N ); }

int main ( void ) {
    boardConfig ( );
    uartConfig ( UART_USB, 460800 );
    adcConfig ( ADC_ENABLE );
    cyclesCounterInit ( EDU_CIAA_NXP_CLOCK_SPEED );
```

```
while(1) {
    convLength=calcFftLength(fftLength,hLength);
    for(sample=0;sample<fftLength;sample++) {
        cyclesCounterReset();
        adc[sample] = ((int16_t )adcRead(CH1)-512)<<6;
        gpioToggle( LEDB);
        while(cyclesCounterRead())< 20400)
            ;
    }
    for(sample=fftLength;sample<convLength;sample++)
        adc[sample]=0;

    sendStr ( "header" ,6 );
    sendBlock ( &fftLength ,1 );
    sendBlock ( &convLength ,1 );
    sendBlock ( adc ,convLength );
    sendBlock ( h ,convLength );
    arm_rfft_init_q15 ( &S ,convLength ,0 ,1 );
    arm_rfft_q15 ( &S ,adc ,fftOut );
    for(int i=0;i<convLength;i++)
        hTemp[i]=h[i];
    arm_rfft_init_q15 ( &S ,convLength ,0 ,1 );
    arm_rfft_q15 ( &S ,hTemp ,H );
    for(int i=0;i<convLength;i++)
        H[i]=H[i]*convLength;
    arm_cmplx_mult_cmplx_q15(fftOut,H,H,convLength);
    sendBlock ( H ,convLength );
    arm_cmplx_mag_squared_q15 ( H ,H ,convLength/2+1
    );
    arm_max_q15 ( H ,convLength/2+1 ,&maxValue
    ,&maxIndex );
    sendBlock ( &maxValue ,1 );
    sendBlock ( (q15_t*) &maxIndex ,1 );
    gpioToggle( LEDR);
}
}
```



2020-06-02

## Procesamiento de señales, fundamentos

- └ CIAA
- └ Filtrado con CIAA

Filtrado con CIAA

Con padding y FFT

Convolución en tiempo con padding en CIAA para filtrado

- mostrar como pasar de pyfda a C
- lanzar psf1
- probar distintos filtros y ver resultado
- hacer notar el efecto del padding



# Bibliografía

*Libros, links y otro material*

[1] *ARM CMSIS DSP.*

[https://arm-software.github.io/CMSIS\\_5/DSP/html/index.html](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] *Wikipedia.*

[https://en.wikipedia.org/wiki/Convolution\\_theorem](https://en.wikipedia.org/wiki/Convolution_theorem)

2020-06-02

Procesamiento de señales, fundamentos

└─ CIAA

└─ Bibliografía

Bibliografía

Libros, links y otro material

[1] ARM CMSIS DSP.  
[https://arm-software.github.io/CMSIS\\_5/DSP/html/index.html](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] Wikipedia.  
[https://en.wikipedia.org/wiki/Convolution\\_theorem](https://en.wikipedia.org/wiki/Convolution_theorem)