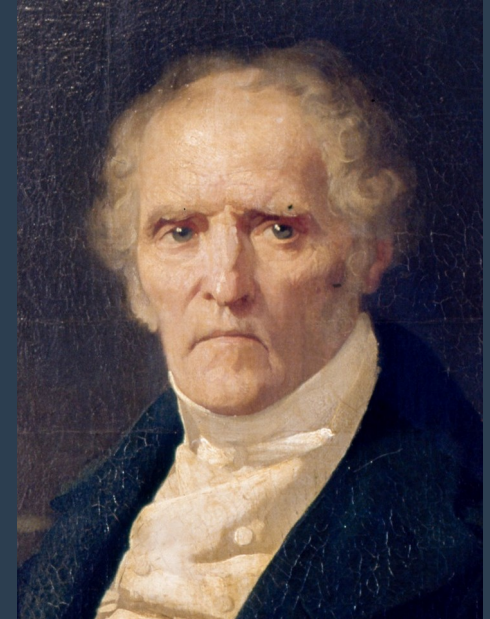


Procesamiento de señales. Fundamentos

Clase 3 - Euler + Fourier + DFT

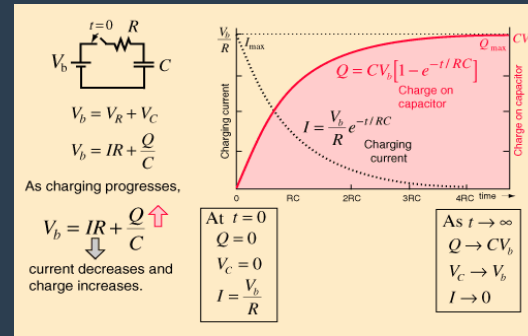
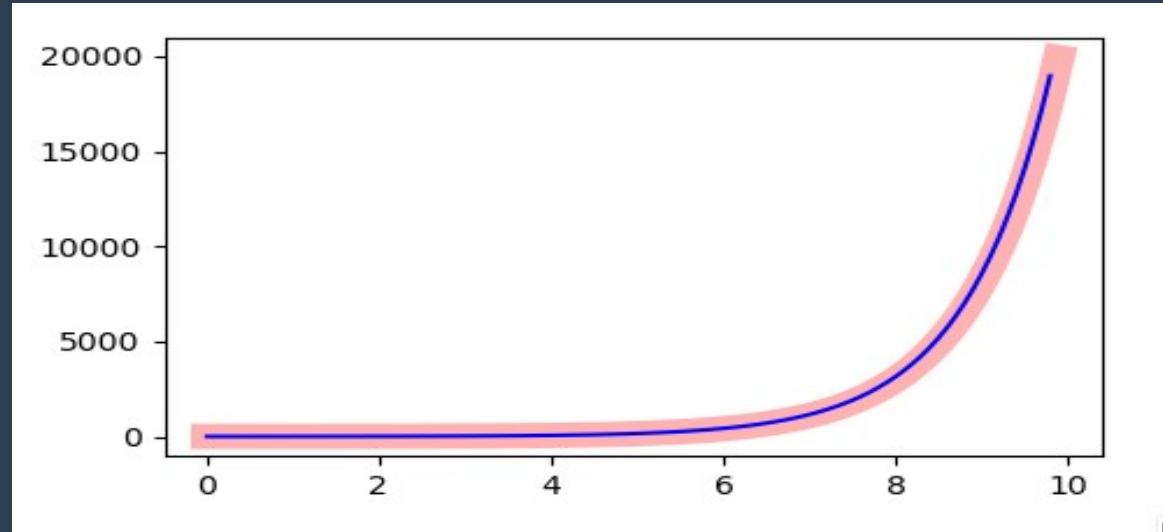
- El número e
- Derivada de e^{at} y $e^{j\omega t}$
- Círculo modulando una señal
- Máquina de Ruffini
- DFT - Transformada discreta de Fourier
- Propiedades
- Spectral Leakage + Zero padding
- Energía y relación de Parseval
- DFT (FFT) en la CAA



2.71828182845904509079559...

Numero e

- Que tiene de especial e?
- Porque un capacitor se carga de esa forma?
- Aplica en el calculo del interes continuo
- Ver código: derivadas.py



O en forma equivalente:

$$M = C \left(1 + \frac{r}{k} \right)^{nk}$$

C = Capital inicial

M = Monto o valor futuro

r = Tasa de interés anual, expresada como número decimal

t = Tiempo, expresado en años

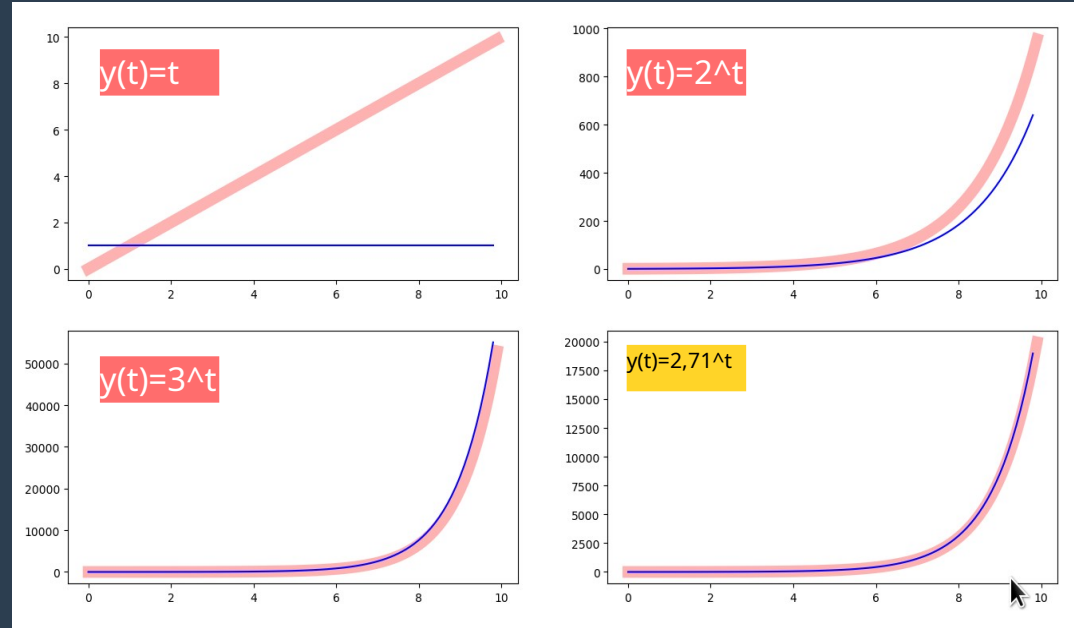
k = Número de capitalizaciones por año

Interés = Monto - Capital

Función derivada de e^t

- La única función en la cual la derivada es igual a si misma
- En la figuras generadas con el codigo el trazo azul es la derivada de la función
- Recordar la regla de la cadena o reemplazo de variables para derivar $e^{(kt)}$
- Ver codigo: derivadas.py

$$e = \lim_{n \rightarrow \infty} \left(1 + \frac{1}{n} \right)^n$$



La derivada es igual a la funcion

$$\begin{aligned} f(t) &= e^t \implies f'(t) = e^t \\ f(t) &= e^{kt} \implies f'(t) = ke^{kt} \end{aligned}$$

Función derivada de $e^{j\omega t}$

- Y que pasa con la derivada de la función compleja?

La derivada es igual a la función

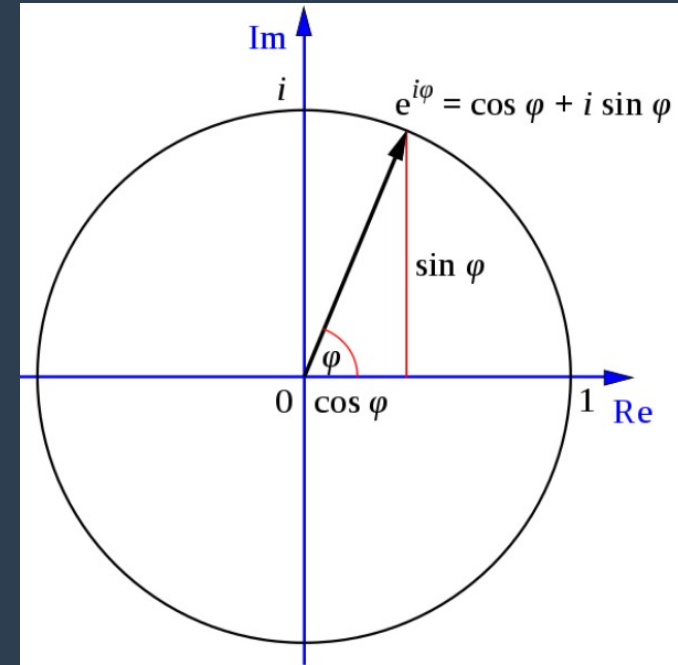
$$f(t) = e^{jt} \implies f'(t) = je^{jt}$$

$$e^{jt} = \cos(t) + j \sin(t)$$

$$e^{j\pi} = -1$$

$$e^{j\frac{\pi}{2}} = j$$

$$e^{j\frac{3\pi}{2}} = -j$$

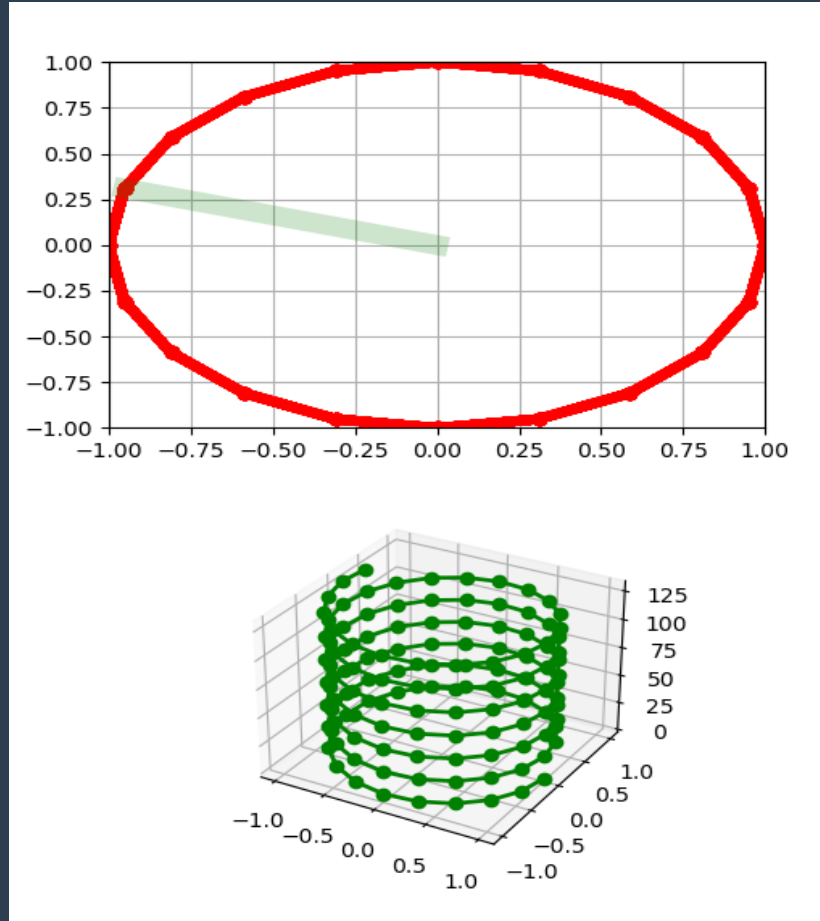


- Ver código: euler1.py

Simulando $e^{j \cdot t}$ con Python

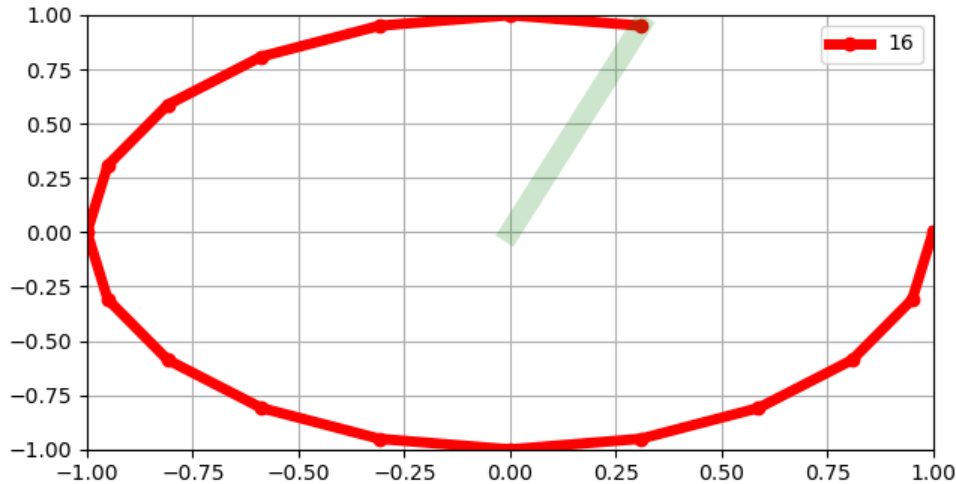
- $F_s=20$
- $N=20$
- CircleFrec=1Hz
- Traza un circulo sampleado cada $1/F_s$
- Ver código: euler1.py

```
5 def circle(f,n):  
4     return np.exp(-1j*2*np.pi*f*n*1/fs)  
3
```

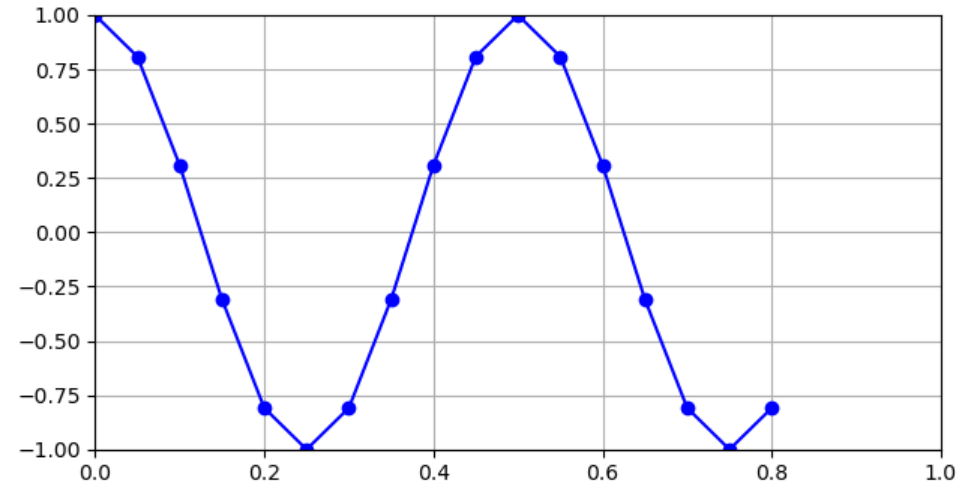


$e^{(j \cdot t)}$ y una senoïdal con Python

- $F_s=20 \Rightarrow t_s=0.05$
- $N=20$
- CircleFrec=1Hz
- Traza un círculo sampleado cada $1/F_s$



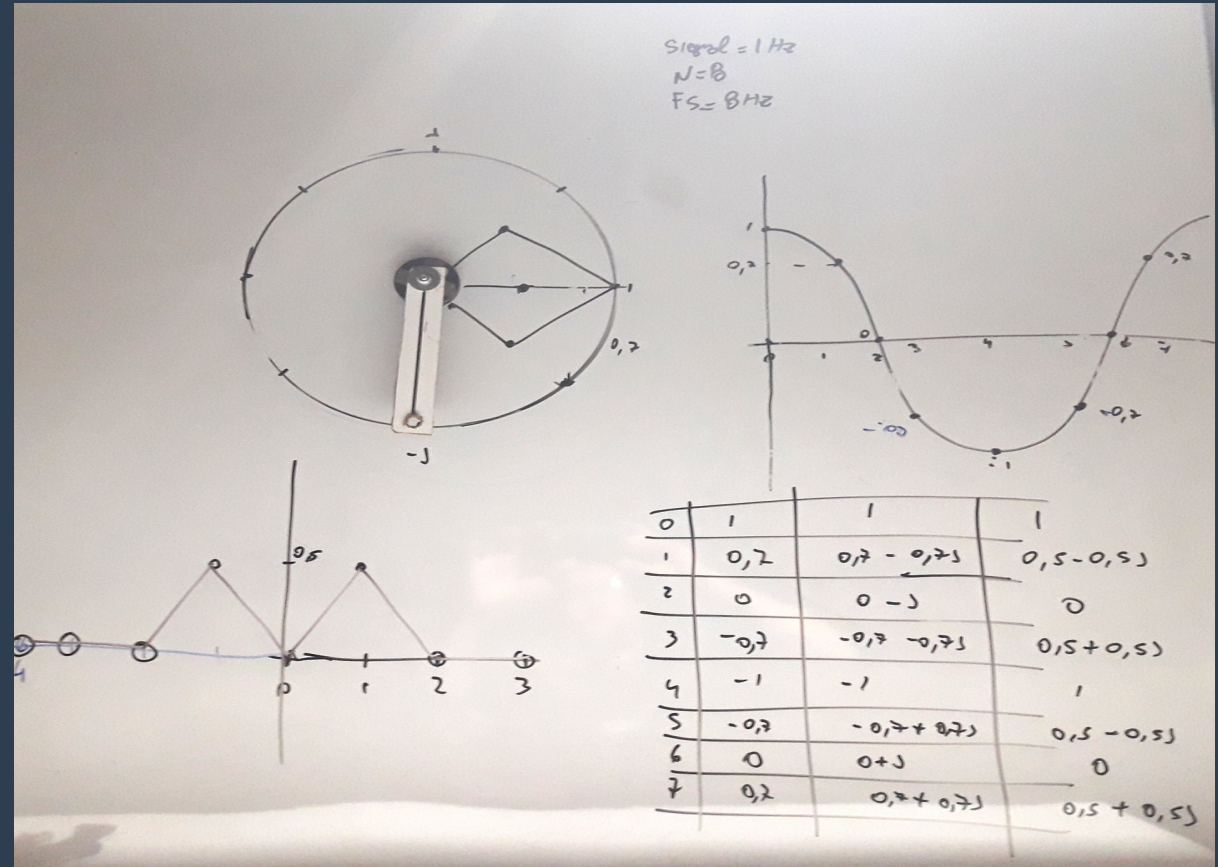
- $F_s=20 \Rightarrow t_s=0.05$
- $N=20$
- signalFrec=2Hz
- Traza una senoïde sampleada a $1/F_s$



• Ver código: euler2.py

La maquina de Rei-Ruof

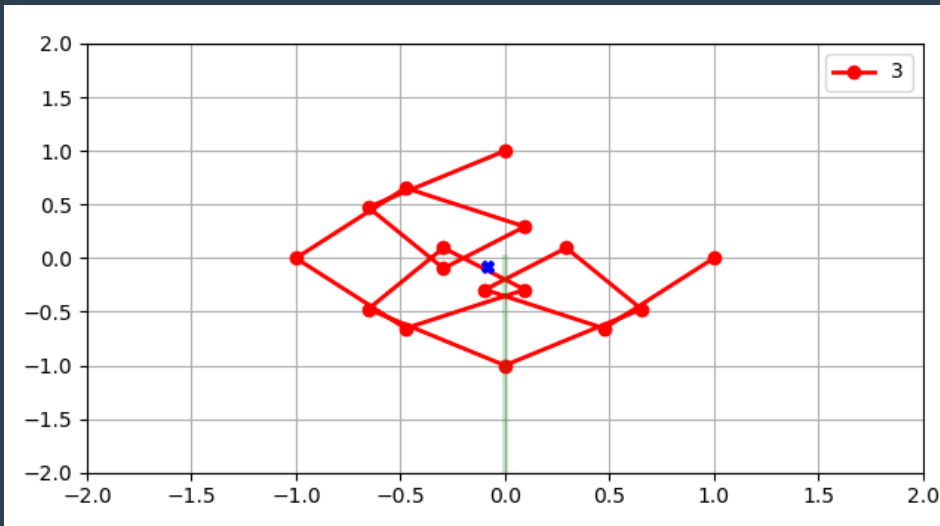
- $F_s=8$
- $N=8$
- $\text{Signal}=1\text{Hz}$
- Para cada frecuencia $F_s/N \cdot n$ se modula la señal con el vector que traza el círculo
- Luego se calcula el promedio y se lo asigna a la $F_s/N \cdot n$



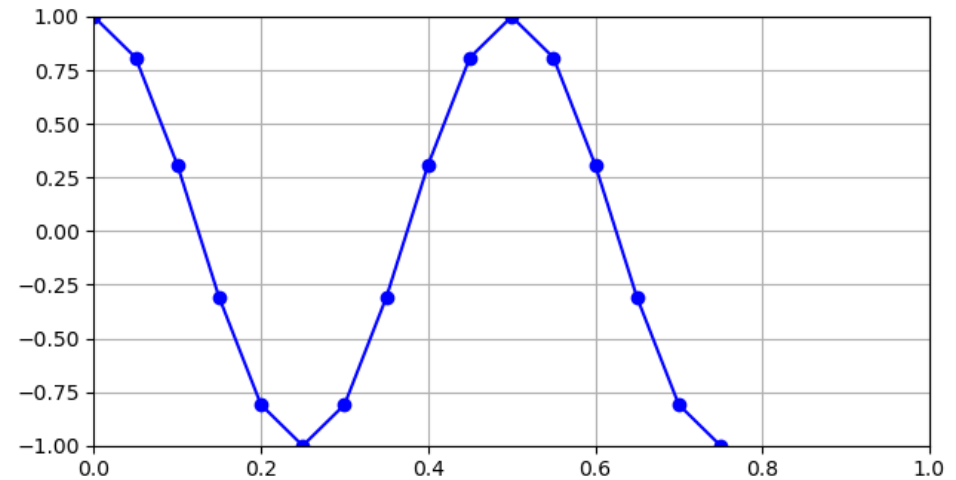
- Ver códigos: [maquina_reiruof.py](#)

La senoidal modulada con $e^{j \cdot t}$

- $F_s=20 \Rightarrow t_s=0.05$
- $N=20$
- CircleFrec=0 a (F_s-1)
- Traza un circulo modulado con la senoidal



- $F_s=20 \Rightarrow t_s=0.05$
- $N=20$
- signalFrec=2Hz
- Traza una senoide sampleada a $1/F_s$

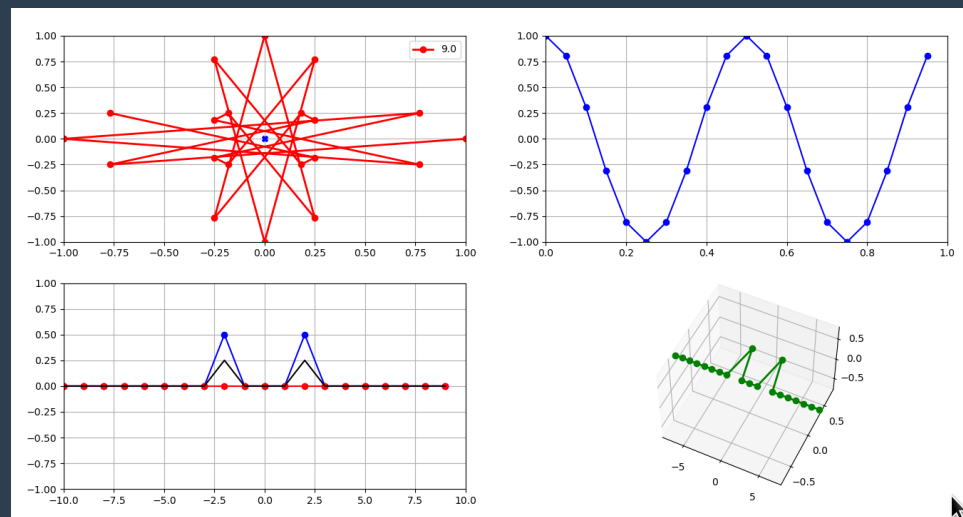


- Ver código: `dft1.py`

Centro de masas de la senoidal modulada

- $F_s=20 \Rightarrow t_s=0.05$
- $N=20$
- CircleFrec = $-F_s/2$ a $(F_s/2-F_s/N)$ cada f_s/N
- Grafico el centro de masas para cada CircleFrec
- El centro de masas es un numero complejo!

- $F_s=20 \Rightarrow t_s=0.05$
- $N=20$
- signalFrec=2Hz
- Traza una senoide sampleada a $1/F_s$



- Ver códigos: dft2.py y dft3.py

Centro de masas == Transformada discreta de Fourier??

$$Cm(k) = \frac{\sum_{n=0}^{N-1} x(n) * e^{-j2\pi k \frac{n}{N}}}{N}$$

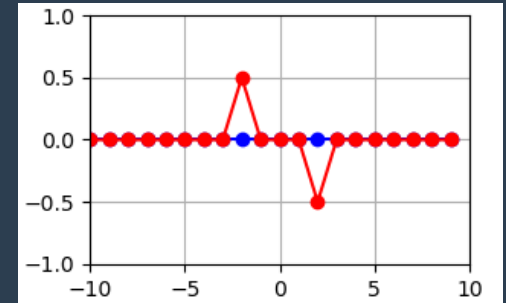
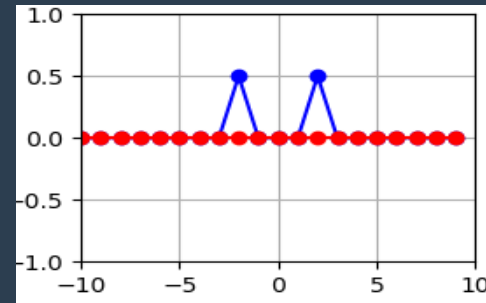
$$X[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}$$

$$X[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] \left(\cos(2\pi kn/N) - j \sin(2\pi kn/N) \right)$$

- Cm=Centro de masas en k
- $X(k)$ = DFT en k
 - Forma polar con e^{\wedge}
 - Forma rectangular con sin y cos
- La diferencia es $1/n$??
- Y si normalizo de otra manera ?
- Se puede normalizar la DFT con:
 - DFT / N => IDFT / 1
 - DFT / 1 => IDFT / N (default en numpy)
 - DFT / \sqrt{N} => IDFT / \sqrt{N}

Resumen de propiedades

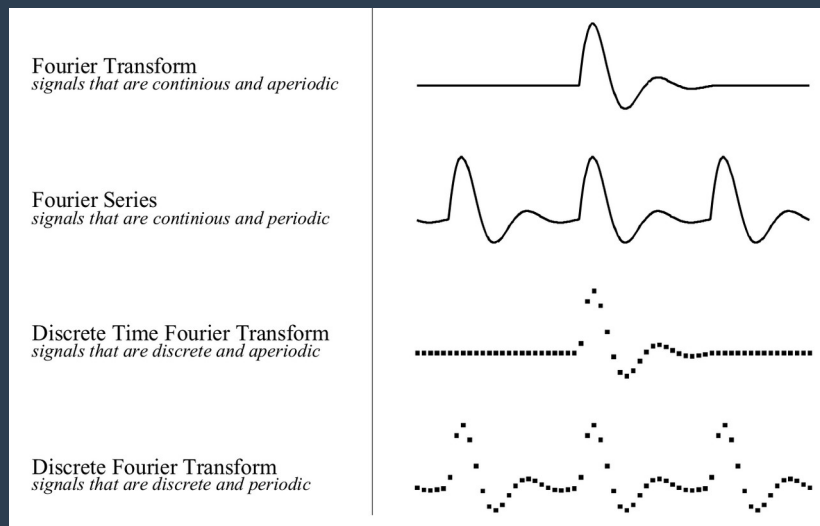
- $F_s=20 \Rightarrow t_s=0.05$
- $N=20$
- Resolución en tiempo = $1/f_s$
- Resolución en Frecuencia = f_s/N
- Muestras en tiempo = N
- Puntos en frecuencia = N
- Los valores de la señal en frecuencia son números complejos
- Los valores de la señal en tiempo son números reales??
- DFT en $n=0$ representa la componente de continua
- Si la entrada es real la DFT es compleja conjugada (hermitica)
- Si la señal es par (cos) la DFT es real pura
- Si la señal es impar (sin) la DFT es imaginaria pura



break

Familia de transformadas

- Existe una relación entre las transformadas de Fourier en tiempo continuo y en tiempo discreto
- De las dos opciones en tiempo discreto la DTFT no es realizable dado que implican infinitos puntos.
- **Solo utilizaremos la DFT**
- En general el calculo de la DFT lo hacemos utilizando el algoritmo FFT.
- No estudiaremos la implementación de la FFT, solo lo utilizamos para calcular la DFT de manera eficiente



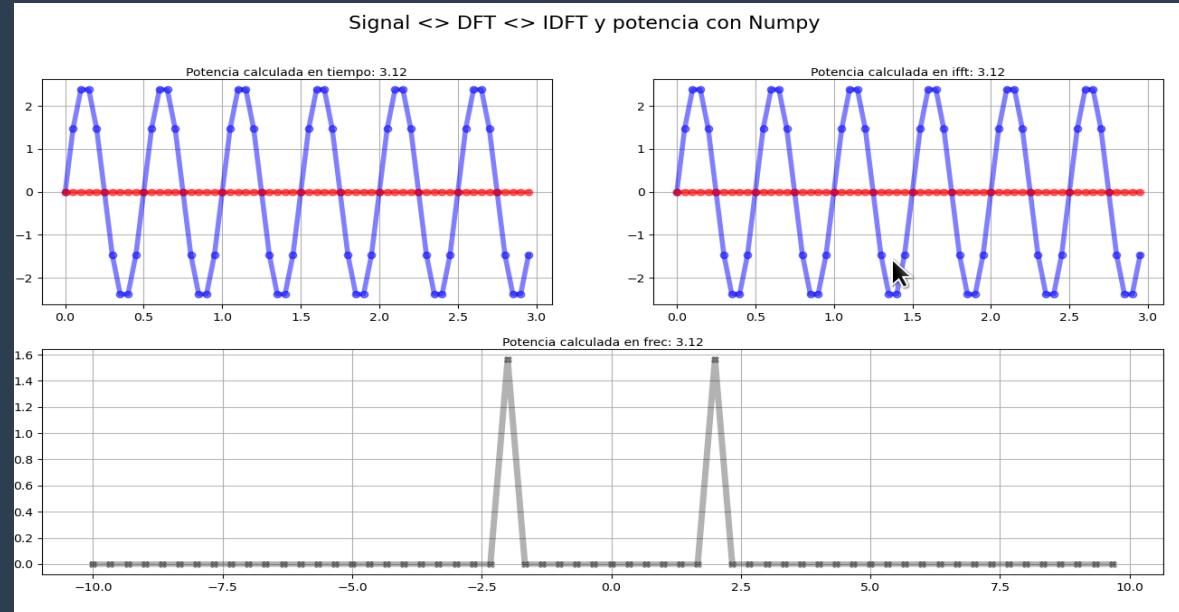
Time Duration		
Finite	Infinite	
Discrete FT (DFT) $X(k) = \sum_{n=0}^{N-1} x(n)e^{-j\omega_k n}$ $k = 0, 1, \dots, N-1$	Discrete Time FT (DTFT) $X(\omega) = \sum_{n=-\infty}^{+\infty} x(n)e^{-j\omega n}$ $\omega \in [-\pi, +\pi)$	discr. time n
Fourier Series (FS) $X(k) = \frac{1}{P} \int_0^P x(t)e^{-j\omega_k t} dt$ $k = -\infty, \dots, +\infty$	Fourier Transform (FT) $X(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t} dt$ $\omega \in (-\infty, +\infty)$	cont. time t
discrete freq. k	continuous freq. ω	

Potencia espectral – Parseval

- Como en tiempo y en frecuencia se representa la misma, ambos deberán tener la misma energía => Relación de Parseval
- La sumatoria de los samples al cuadrado dividido el numero de samples es la definicion de potencia promedio
- El modulo cuadrado de la DFT representa la densidad de potencia espectral
- Permite determinar en donde se concentra la energia de la señal.

$$P_{average}(x) = \frac{1}{N} \sum_{n=0}^{N-1} \|x[n]\|^2 = \sum_{n=0}^{N-1} \|F[n]\|^2$$

$$\begin{aligned} P_{sin} &= \frac{A^2}{2} \\ P_{sin} &= \frac{2,5^2}{2} \\ P_{sin} &= 3,125W \\ P &= 1,56 + 1,56 \\ P &= 3,125W \end{aligned}$$

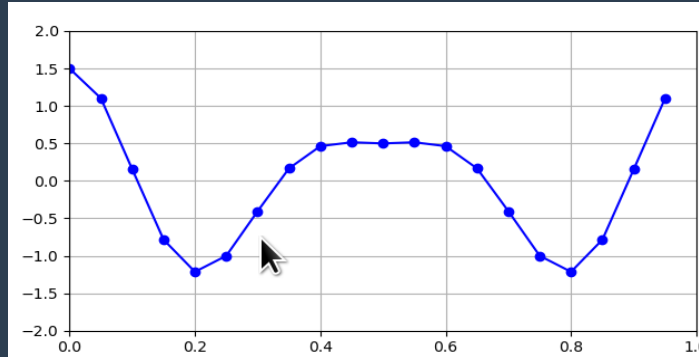


• Ver código: [utils/dft_idft_numpy.py](#)

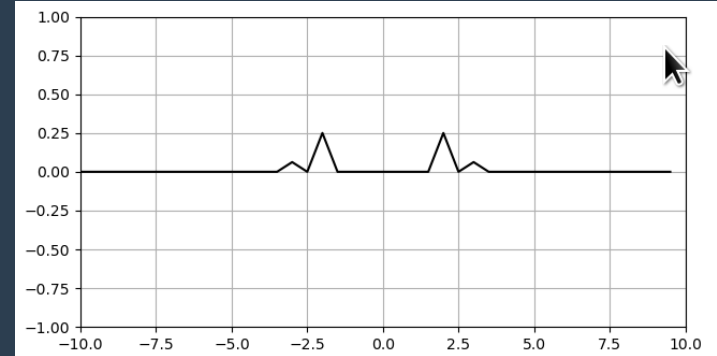
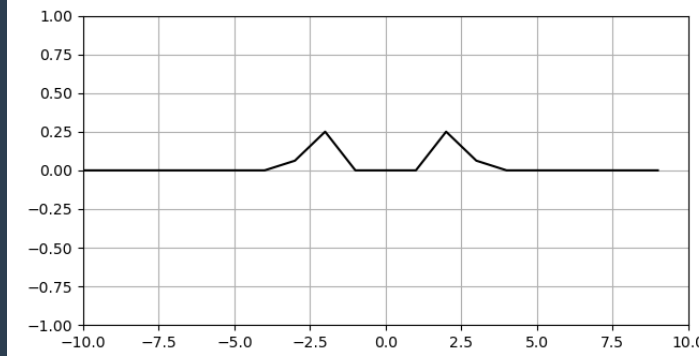
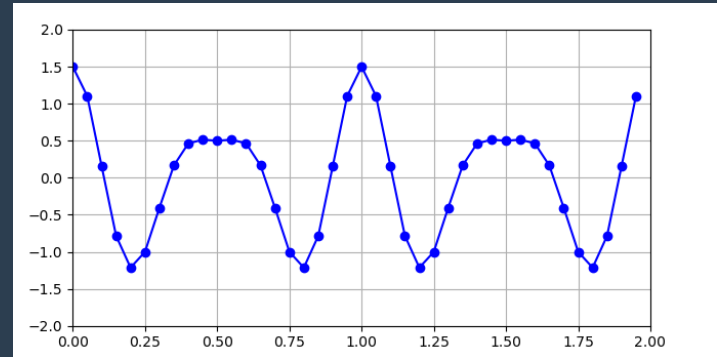
Resolución espectral F_s/N

- Es una medida de cuanto se puede discriminar en frecuencia
- Resolución espectral = f_s/N
- Para una determinada F_s , cuando mas grande N mejor resolución espectral
- Ej:
 - R deseada 5hz
 - $F_s=1000 \Rightarrow N \geq 200$
 - tiempo de adquisición $\geq N/f_s=0,2$ segs
- Ej
 - R deseada 0,01Hz
 - $F_s=1000 \Rightarrow N \geq 100000$
 - tiempo de adquisición $\geq N/f_s= 100$ segs

$F_s=200$ $N=20$



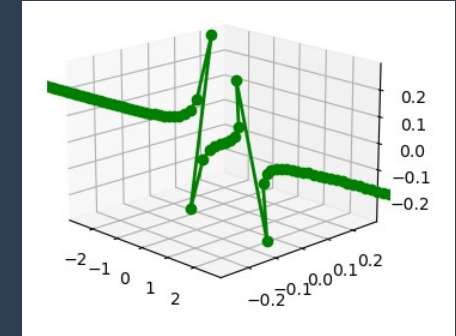
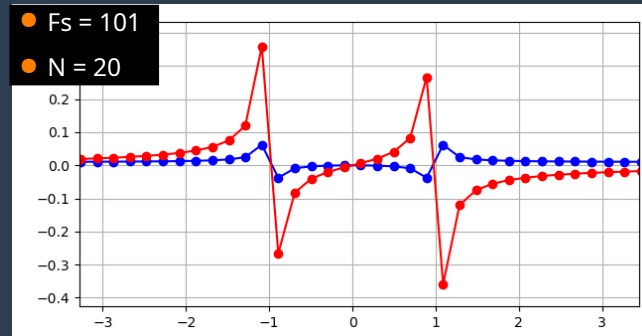
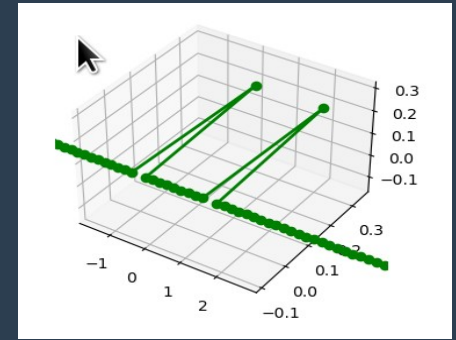
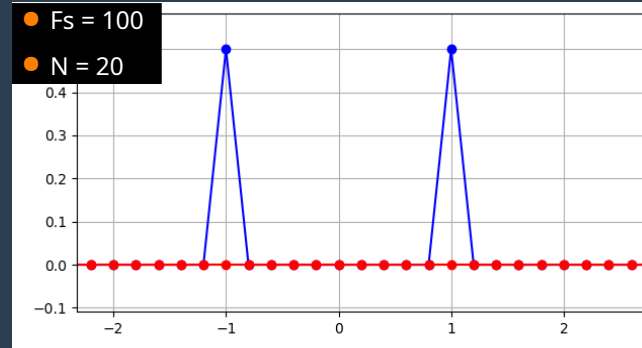
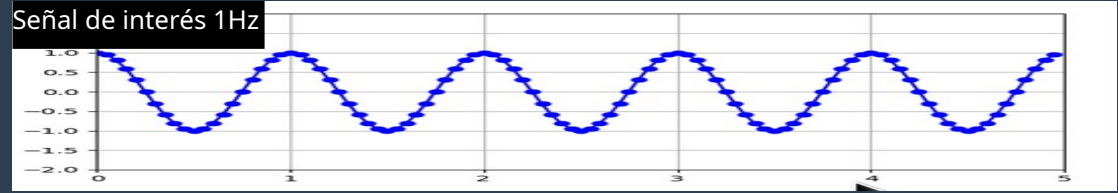
$F_s=200$ $N=40$



Ver código: `dft3.py`

Fuga espectral – Spectral leakage

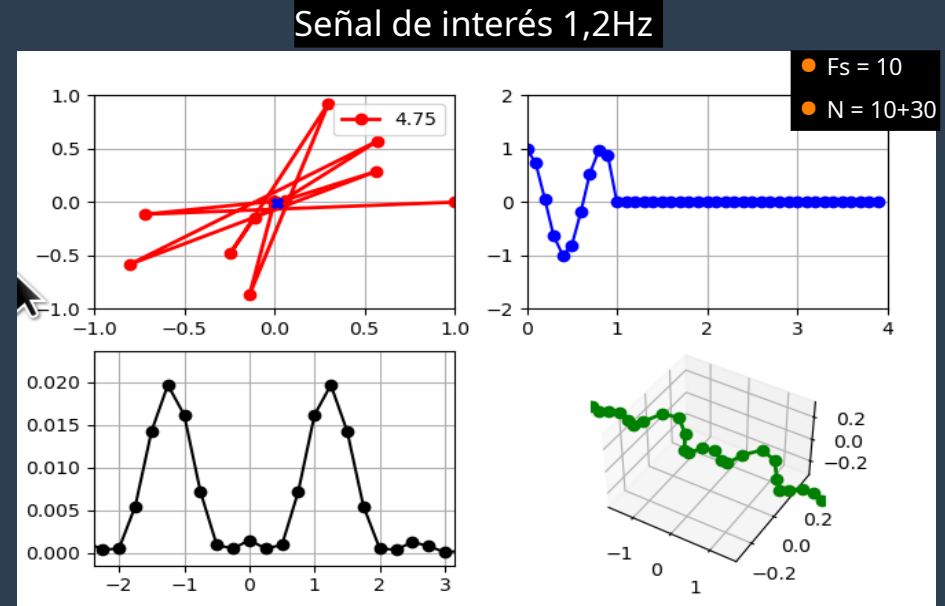
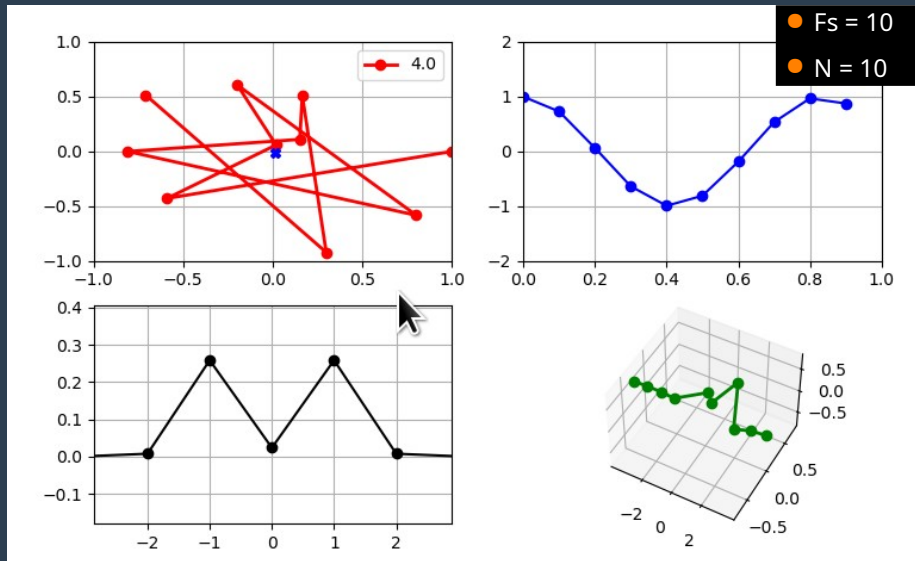
- Si la relación F_s/N (en el ejemplo $f_s/N=5$) es múltiplo de la señal de interés la DFT resuelve bien
- En otro caso, aparece un efecto no deseado.
- Como mitigarlo??
- Se necesitan mas puntos en la DFT, pero la señal ya fue capturada.
- Se puede rellenar con ceros! : zero padding



● Ver código: leakage.py

Fuga espectral – Zero padding

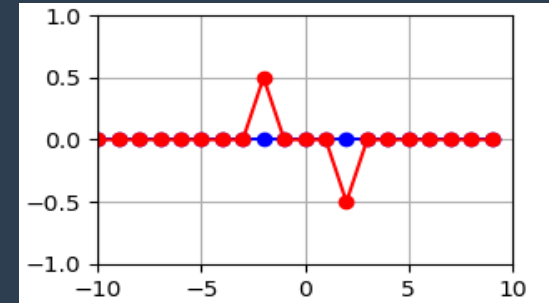
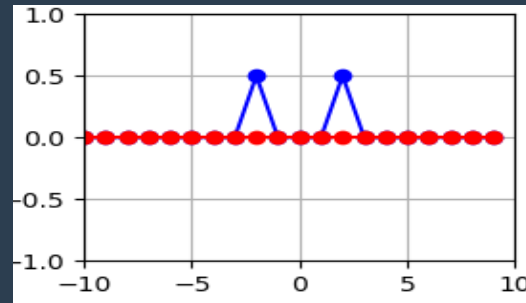
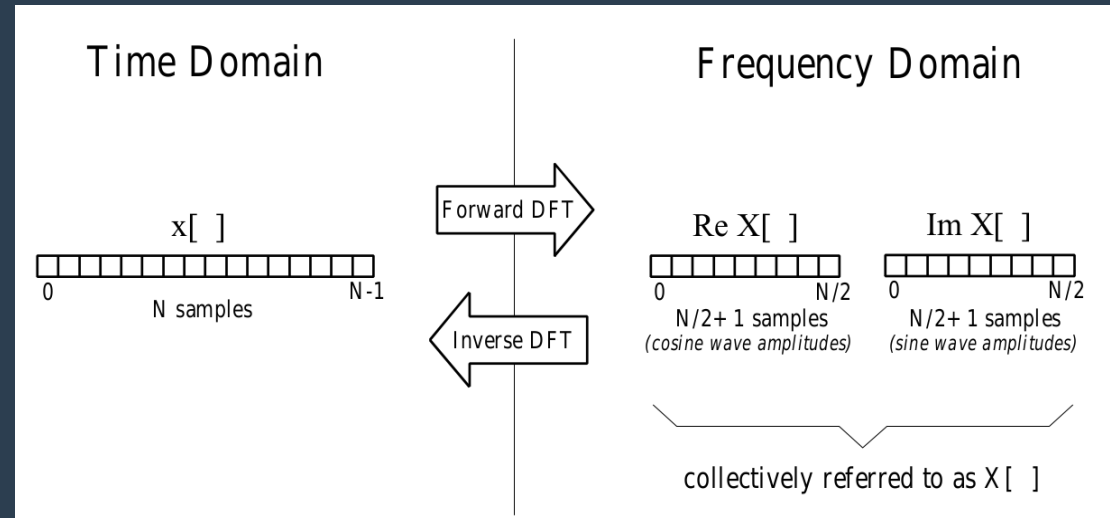
- La idea es tener mas frecuencias intermedias entre f_s/N
- Para eso incrementamos N rellenando con ceros la señal original $N+Z = N'$
- Un efecto adverso es que la amplitud de la DFT se ve disminuida
- La nueva resolución espectral es f_s/N'



• Ver código: `zero_padding.py`

Simetría en DFT con entrada Real pura

- Si la señal de entrada es real pura, la DFT tiene simetría alrededor del $n=0$
- La simetría muestra siempre valores complejos conjugados \Rightarrow función hermitica
- En muchas aplicaciones solo basta con la mitad de los datos y cálculos.

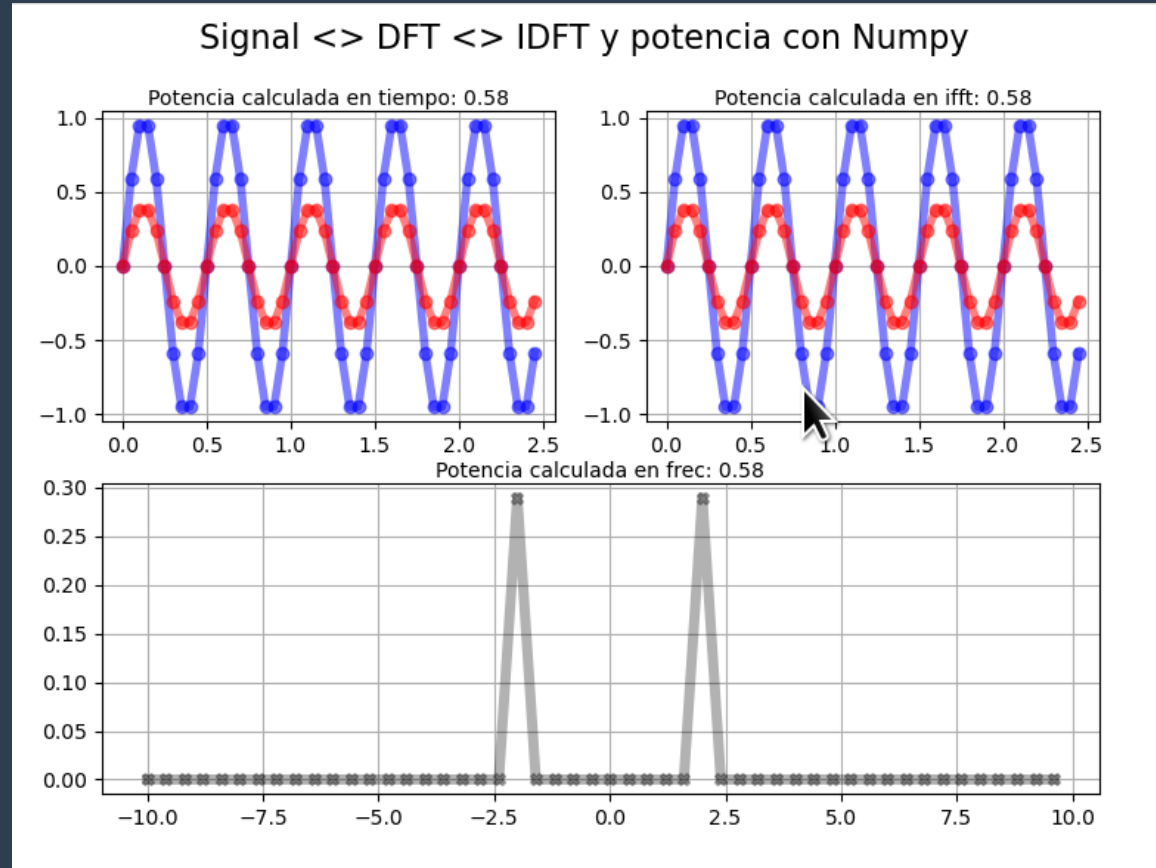


● Ver código: `dft3.py`

DFT con numpy

DFT con numpy

- Modulo np.fft
- Funcion np.fft.dft
- Evaluar np.fft.fftshift para centrar en cero
- Investigar el código para ver el calculo de potencia en tiempo y frecuencia.
- Tener en cuenta la normalizacion

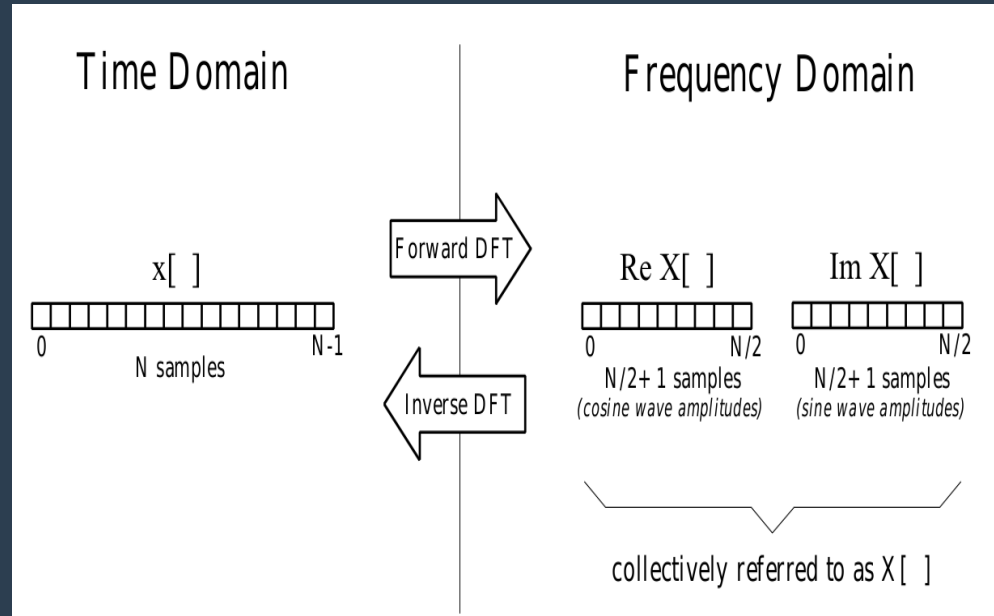
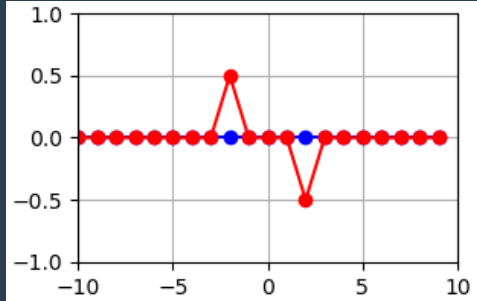
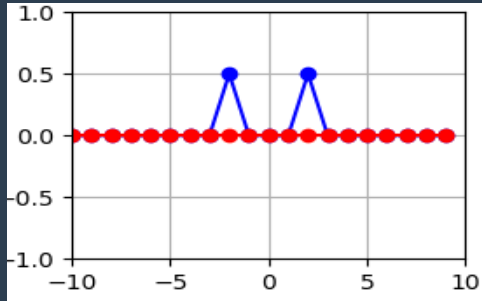


• Ver código: [dft_idft_numpy.py](#)

DFT con la CIAA

DFT Real con CMSIS-DSP

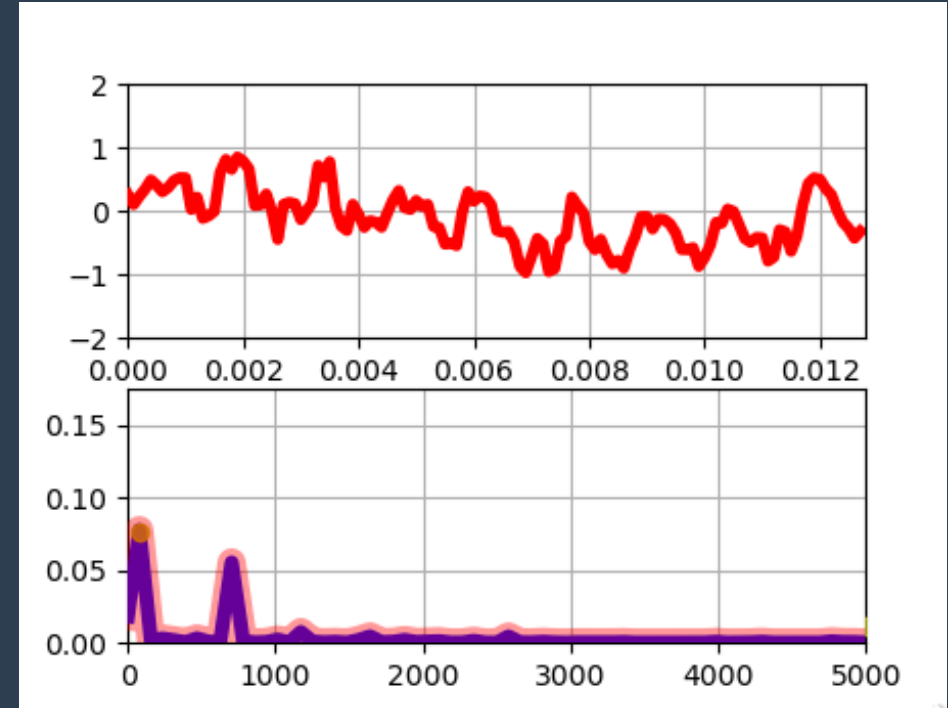
- Si la señal de entrada es real, la DFT tiene simetría alrededor del $n=0$
- Se puede hacer y usar solo la mitad de los cálculos
- NOTA: La salida de la función de CMSIS intercala la parte real y la imaginaria



$X = \{ \text{real}[0], \text{imag}[0], \text{real}[1], \text{imag}[1], \text{real}[2], \text{imag}[2] \dots \text{real}[(N/2)-1], \text{imag}[(N/2)-1] \}$

DFT con CMSIS-DSP en Q1.15

- En los ejemplos se utiliza Q1.15
- Leer la documentación!
- https://www.keil.com/pack/doc/CMSIS/DSP/html/group__RealFFT.html#ga053450cc600a55410ba5b5605e96245d
- Investigar otras opciones con float.
- Hay que inicializar la estructura antes de usar
- La fft MODIFICA los datos de entrada!!!
- Siempre verificar el tipo de datos de salida.
- Tener en cuenta los escalados.

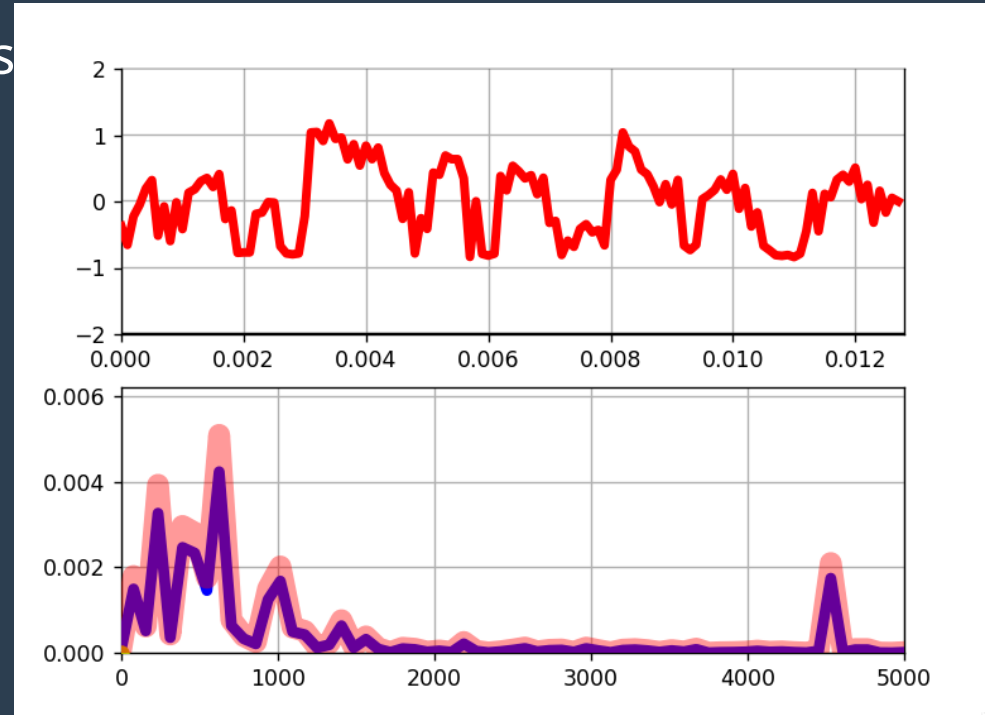


```
arm_rfft_init_q15( &S ,header.N ,0 ,1 );  
arm_rfft_q15( &S ,fftIn ,fftOut );  
arm_cmlx_mag_squared_q15( fftOut ,fftMag ,header.N/2+1 );
```

• Ver carpeta clase3/psf1

DFT con CMSIS-DSP en float32

- CMSIS-DSP también cuenta con opciones para calcular la FFT en float32
- Hay que inicializar la estructura antes de usar
- La fft MODIFICA los datos de entrada!!!
- Siempre verificar el tipo de datos de salida.
- Tener en cuenta los escalados.



```
arm_rfft_init_f32 ( &S ,&cS ,header.N ,0 ,1 );  
arm_rfft_f32      ( &S ,fftIn ,fftOut
```

• Ver carpeta clase3/psf2

TP1

TP1

- Link al TP1
- <https://forms.gle/g5QNTBFqi3WskGUC9>
- Entrega el 25/7 23:59
- Entrega fuera de termino implica una penalizacion de 0.9
- Se puede completar parcialmente, y cuando el TP esta listo, se indica en la ultima pregunta del test para que sea corregido
- Las consultas del TP se pueden realizar directamente en el mismo formulario en la ultima seccion

Procesamiento de señales. Fundamentos



Section 1 of 9

TP1 Procesamiento de señales,
Fundamentos 2022 - ms07

Fecha limite de entrega 25/07 23:59

Email *