Procesamiento de señales. Fundamentos

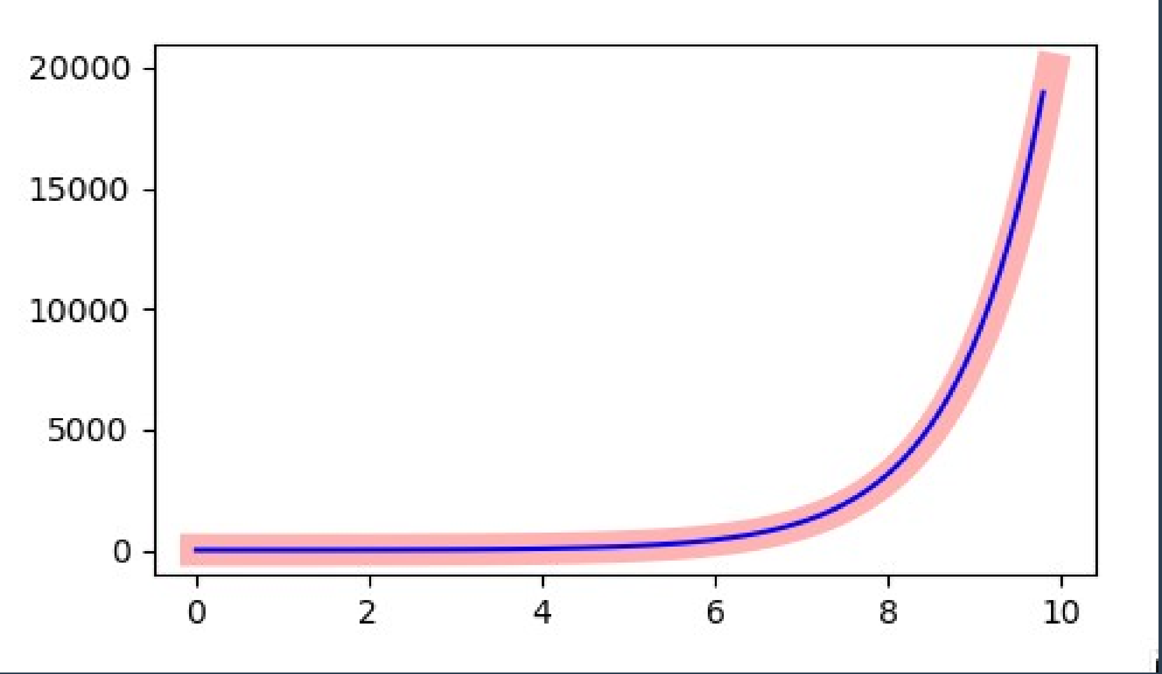
Clase 3 - Euler + Fourier + DFT

- El numero e
- Derivada de e^t y e^(jt)
- Circulo modulando una señal
- Maquina de Rei-Ruof
- DFT - Transformada discreta de Fourier
- Propiedades
- Spectral Leakage + Zero padding
- Energía y relación de Parseval
- DFT (FFT) en la CIAA

2.71828182845904509079559...

Numero e

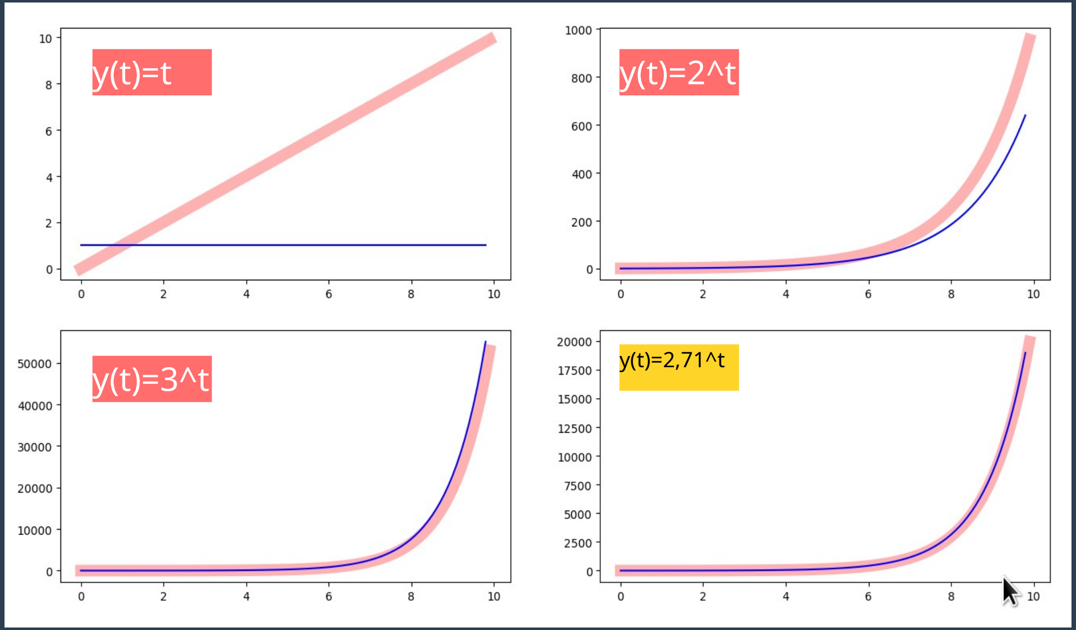
- Que tiene de especial e?
- Encuesta
- https://forms.gle/b9Gqr8sSehuJcdNQA
- Ver código: derivadas.py



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

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



La derivada es igual a la funcion

$$f(t) = e^t \implies f'(t) = e^t$$
$$f(t) = e^{kt} \implies f'(t) = ke^{kt}$$

Función derivada de e^(j*t)

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

La derivada es igual a la funcion

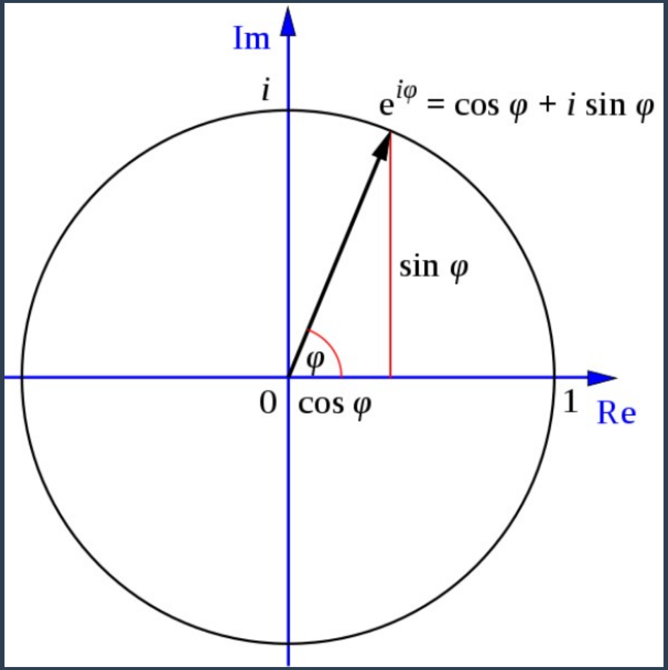
$$f(t) = e^{jt} \implies f'(t) = je^{jt}$$

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

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

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

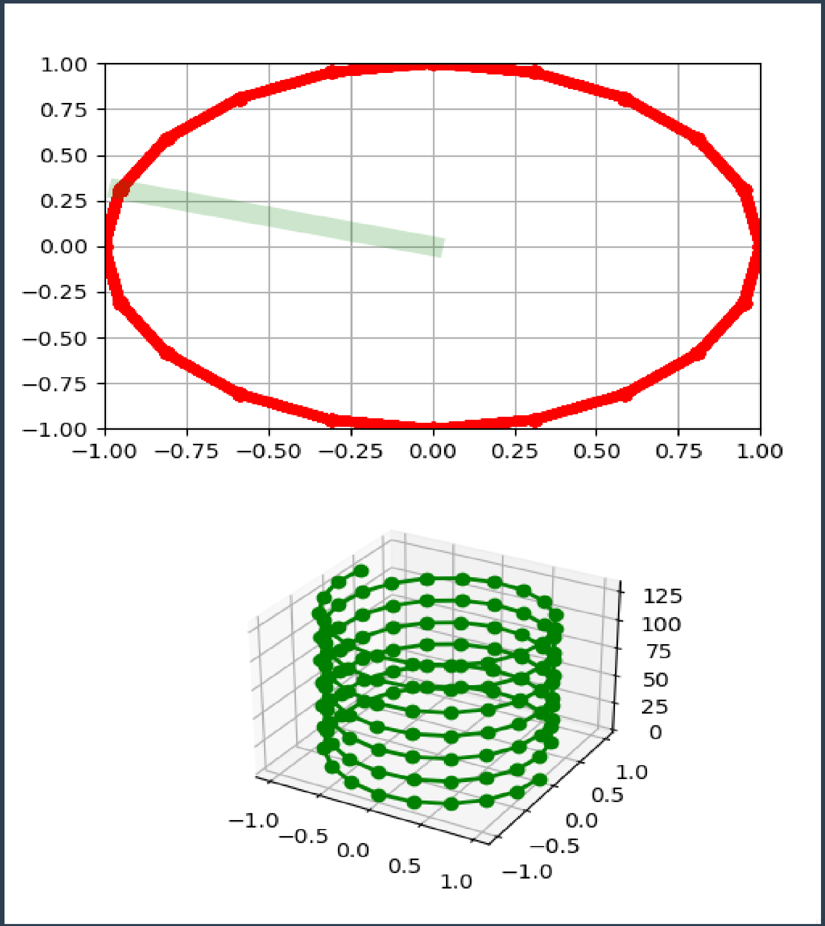
$$e^{\frac{j3\pi}{2}} = -j$$



- Ver código: euler1.py

Simulando e^(j*t) con Python

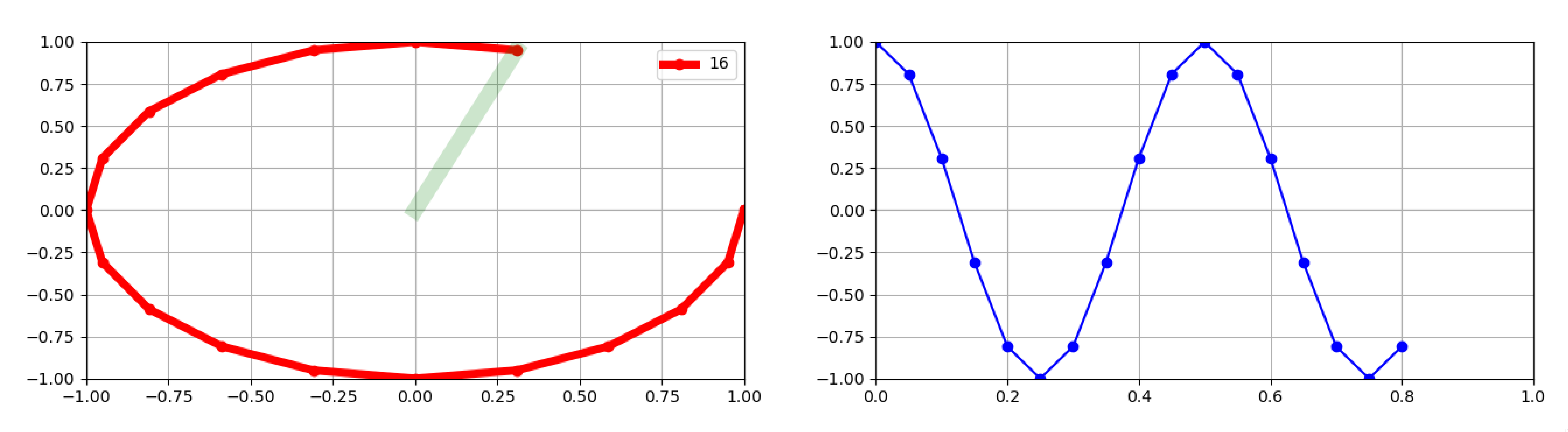
- Fs=20
- N=20
- CircleFrec=1Hz
- Traza un circulo sampleado cada 1/Fs
- Ver código: euler1.py



e^(j*t) y una senoidal con Python

- Fs=20 => ts=0.05
- N=20
- CircleFrec=1Hz
- Traza un circulo sampleado cada 1/Fs

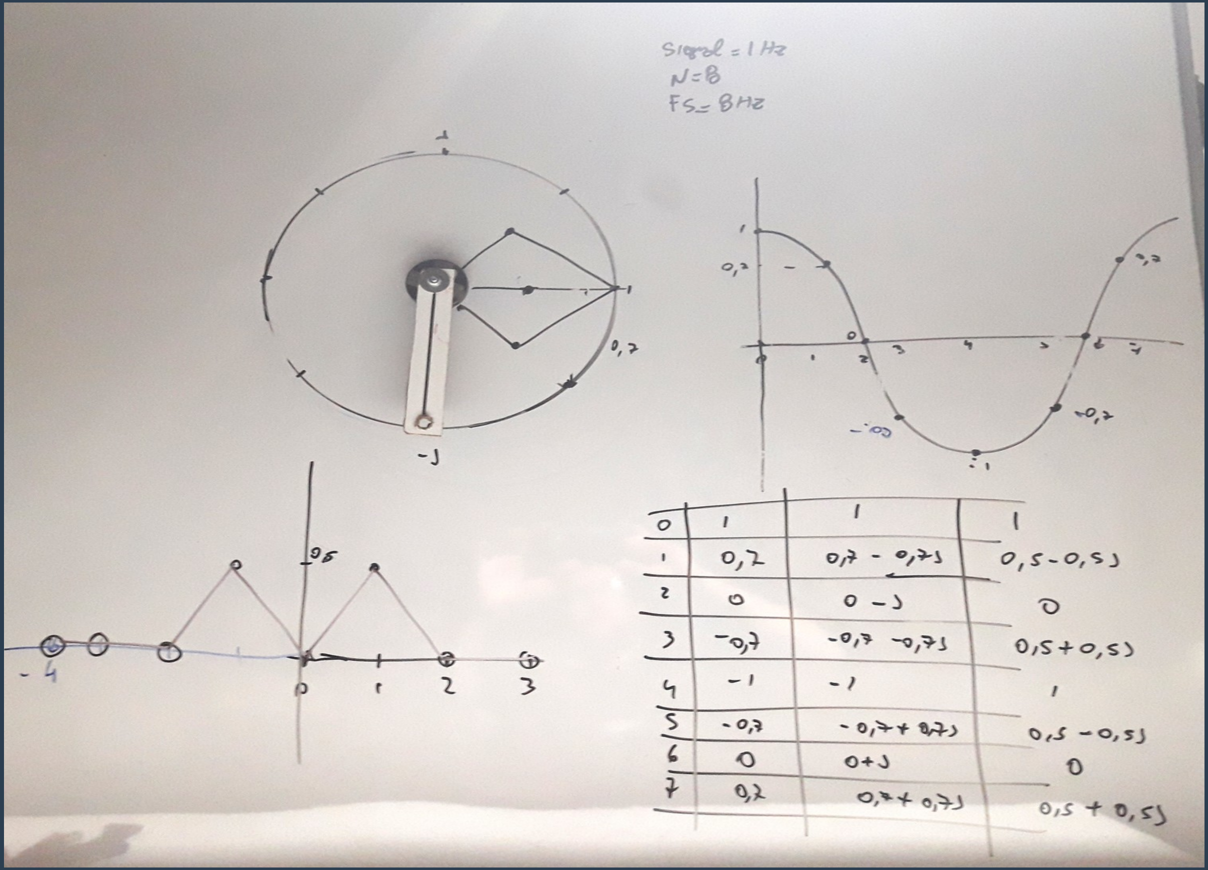
- Fs=20 => ts=0.05
- N=20
- signalFrec=2Hz
- Traza una senoide sampleada a 1/Fs



- Ver código: euler2.py

La maquina de Rei-Ruof

- Fs=8
- N=8
- Signal=1Hz
- Para cada frecuencia Fs/N*n se modula la señal con el vector que traza el circulo
- Luego se calcula el promedio y se lo asigna a la Fs/N*n

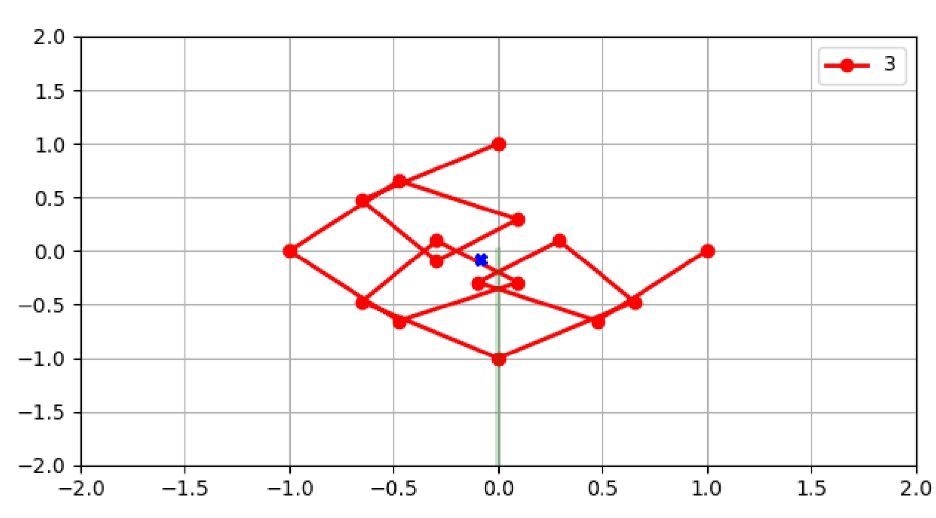


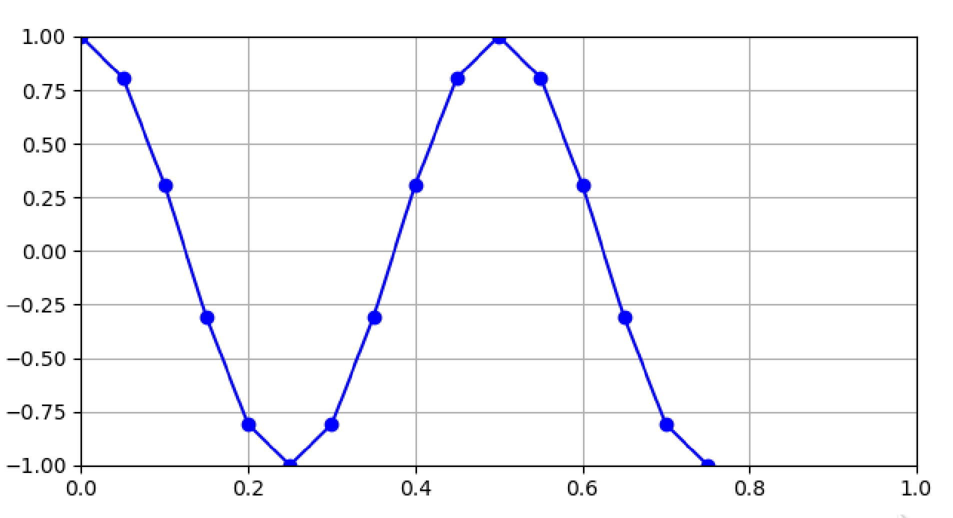
- Ver códigos: maquina_reiruof.py

La senoidal modulada con e^(j*t)

- Fs=20 => ts=0.05
- N=20
- CircleFrec=0 a (Fs-1)
- Traza un circulo modulado con la senoidal

- Fs=20 => ts=0.05
- N=20
- signalFrec=2Hz
- Traza una senoide sampleada a 1/Fs



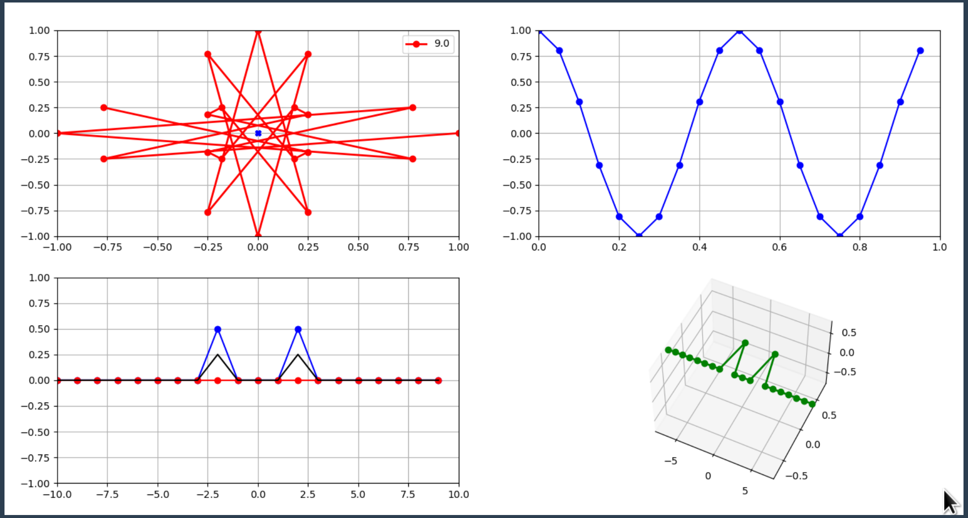


- Ver código: dft1.py

Centro de masas de la senoidal modulada

- Fs=20 => ts=0.05
- N=20
- CircleFrec=-Fs/2 a (Fs/2-Fs/N) cada fs/N
- Grafico el centro de masas para cada CircleFrec
- El centro de masas es un numero complejo!

- Fs=20 => ts=0.05
- N=20
- signalFrec=2Hz
- Traza una senoide sampleada a 1/Fs



- Ver códigos: dft2.py y dft3.py

Centro de masas == Transformada discreta de Fourier??

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

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

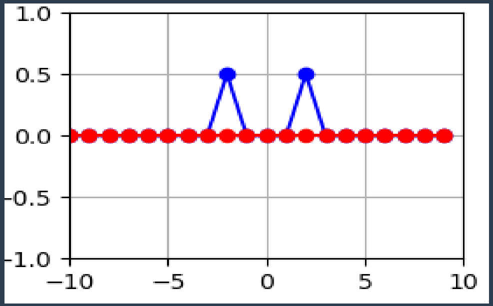
$$X[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] \left(\cos(2\pi kn/N) - j \sin(2\pi kn/N) \right)$$

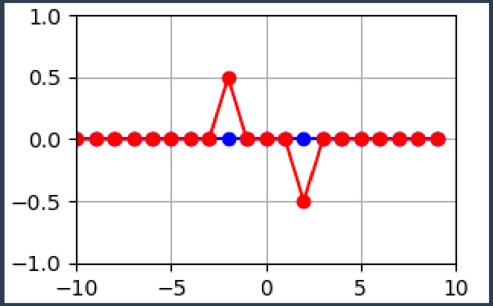
- Cm=Centro de masas en k
- X(k) = DFT en k
  - Forma polar con e^
  - Forma rectangular con sin y cos
- La diferencia es 1/n ??
- Y si normalizo de otra manera ?
- Se puede normalizar la DFT con:
  - DFT / N => IDFT / 1
  - DFT / 1 => IDFT / N (default en numpy)
  - DFT / $\sqrt{N}$ => IDFT / $\sqrt{N}$

Resumen de propiedades

- Fs=20 => ts=0.05
- N=20
- Resolución en tiempo= 1/fs
- Resolución en Frecuencia= fs/N
- Muestras en tiempo =N
- Puntos en frecuencia = N
- Números en frecuencia complejos
- Números en tiempo reales??

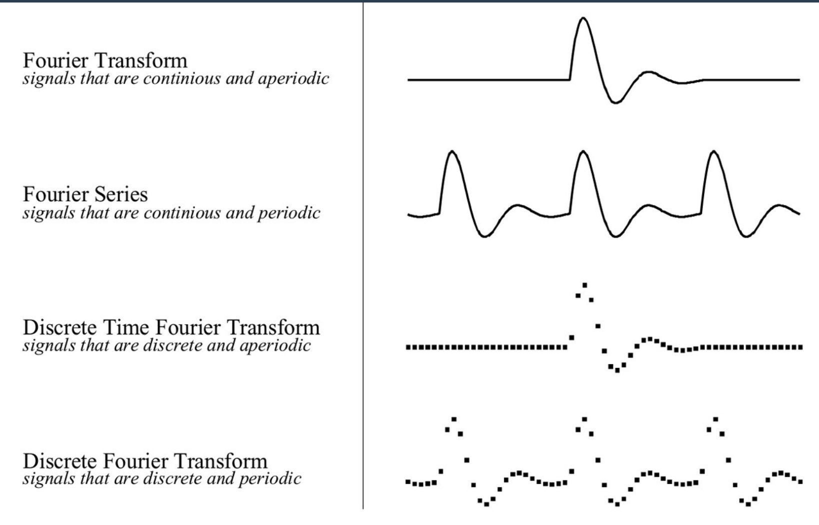
- DFT en n=0 representa la componente de continua
- Si la entrada es real la DFT es compleja conjugada (hermitica)
- Si la señal es par (cos) la DFT es real pura
- Si la señal es impar (sin) la DFT es imaginaria pura





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) $X(k)=\sum_{n=0}^{N-1} x(n)e^{-j\omega_k n}$ $k=0,1,\ldots,N-1$	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) $X(k)=\frac{1}{P}\int_0^P x(t)e^{-j\omega_k t}dt$ $k=-\infty,\ldots,+\infty$	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.

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

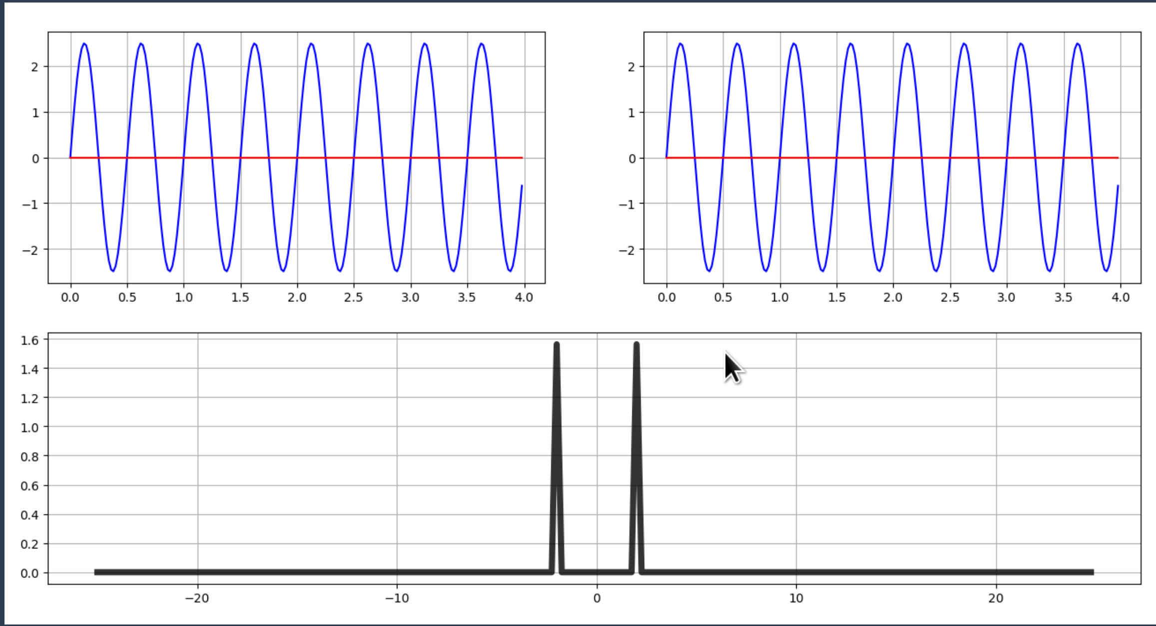
$$P_{sin} = \frac{A^2}{2}$$

$$P_{sin} = \frac{2{,}5^2}{2}$$

$$P_{sin} = 3{,}125W$$

$$P = 1{,}56 + 1{,}56$$

$$P = 3{,}125W$$

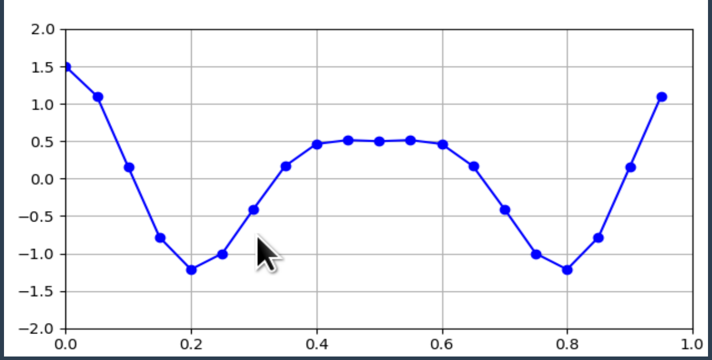


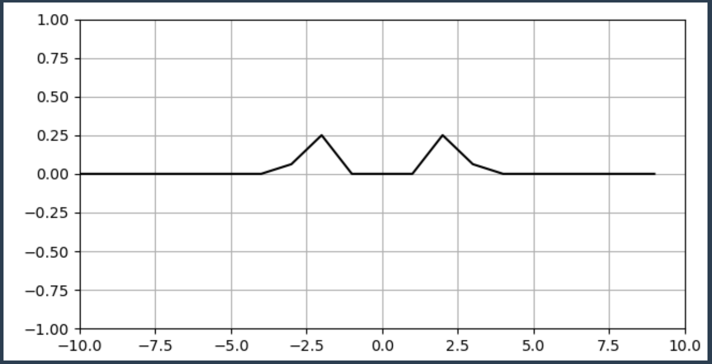
- Ver código: dft_idft_numpy.py

Resolución espectral Fs/N

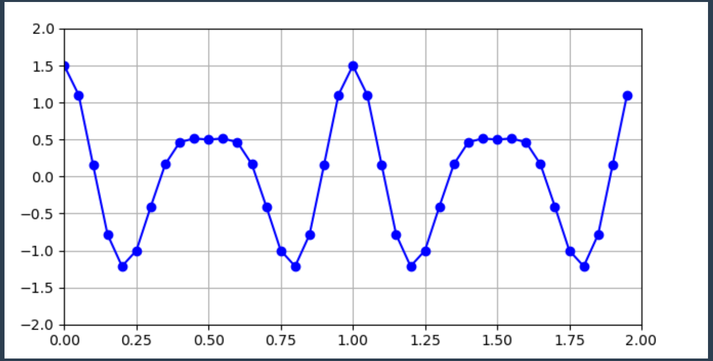
- Es una medida de cuanto se puede discriminar en frecuencia
- Resolución espectral = fs/N
- Para una determinada Fs, cuando mas grande N mejor resolución espectral
- Ej:
  - R deseada 5hz
  - Fs=1000 => N >= 200
  - tiempo de adquisición >= N/fs=0,2 segs
- Ej
  - R deseada 0,01Hz
  - Fs=1000 => N >=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

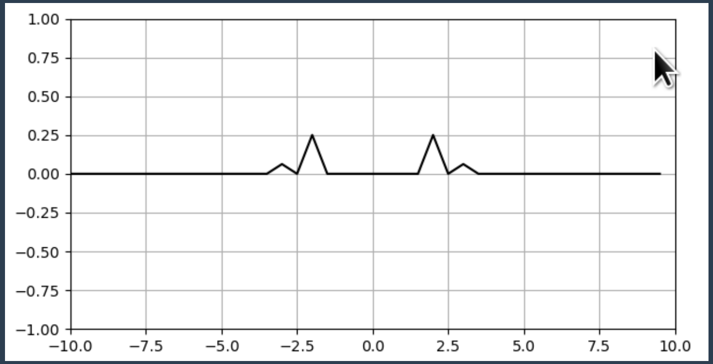
Fs=20 0 N=20





Fs=20 0 N=40

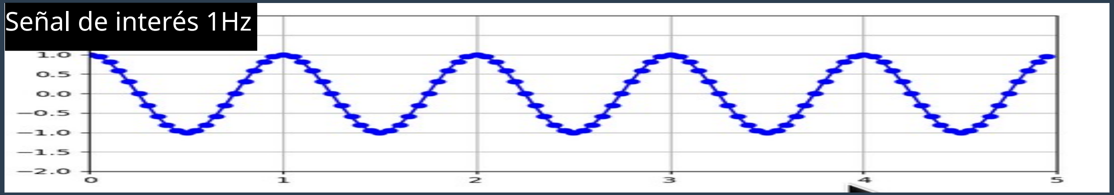


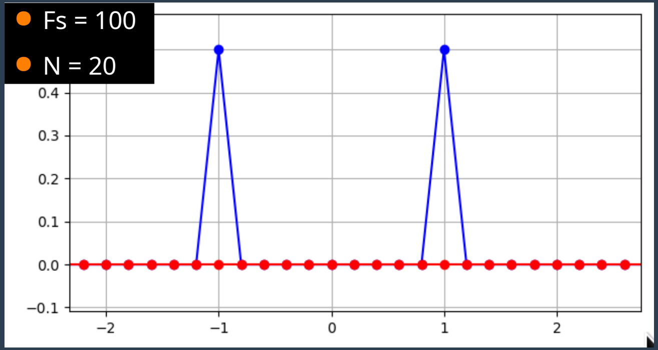


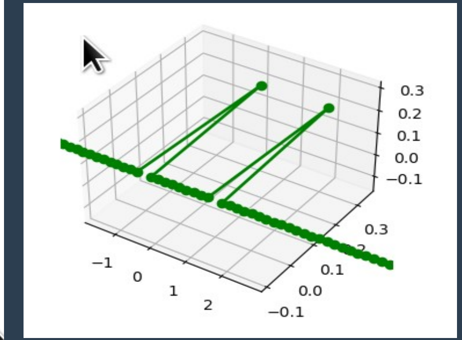
- Ver código: dft3.py

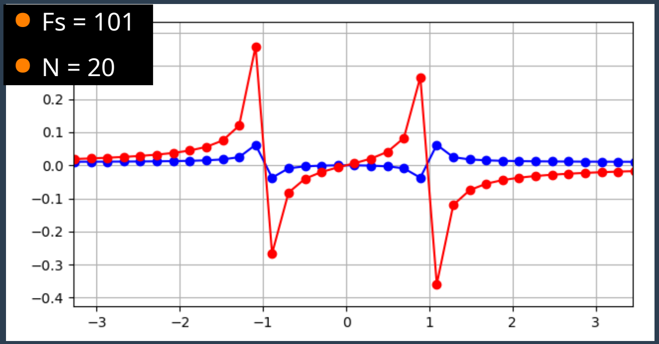
Fuga espectral – Spectral leakage

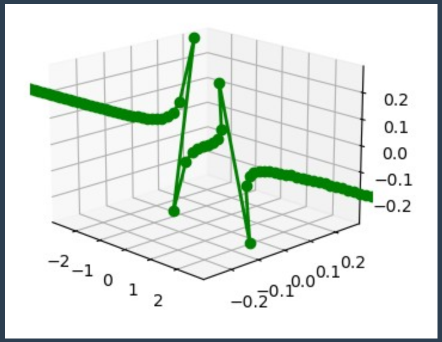
- Si la relación Fs/N=5 es múltiplo de la señal de interés la DFT resuelve bien
- En otro csdo, aparece un efecto no deseado.
- Como mitigarlo??
- Se necesitan mas puntos en la DFT, pero la señal ya fue capturada.
- Se puede rellenar con ceros!









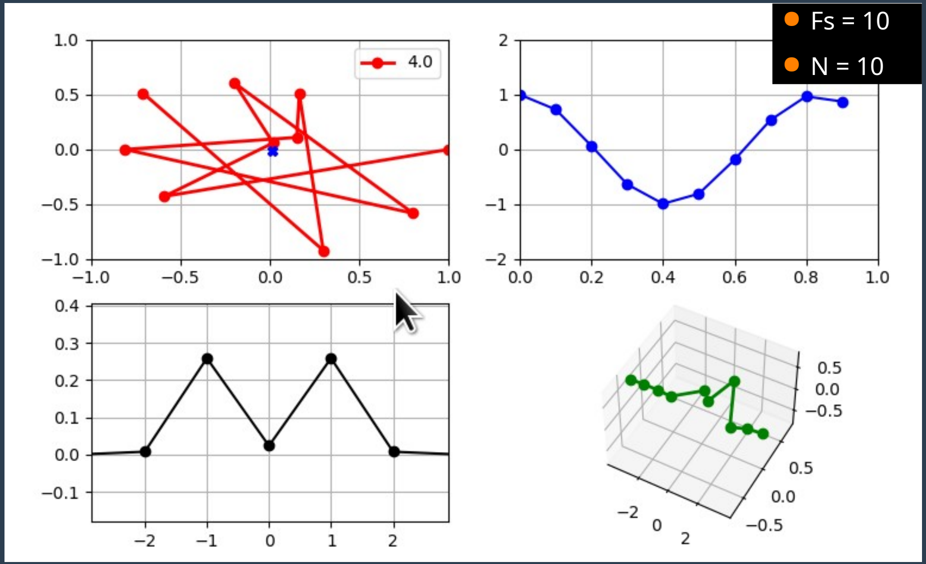


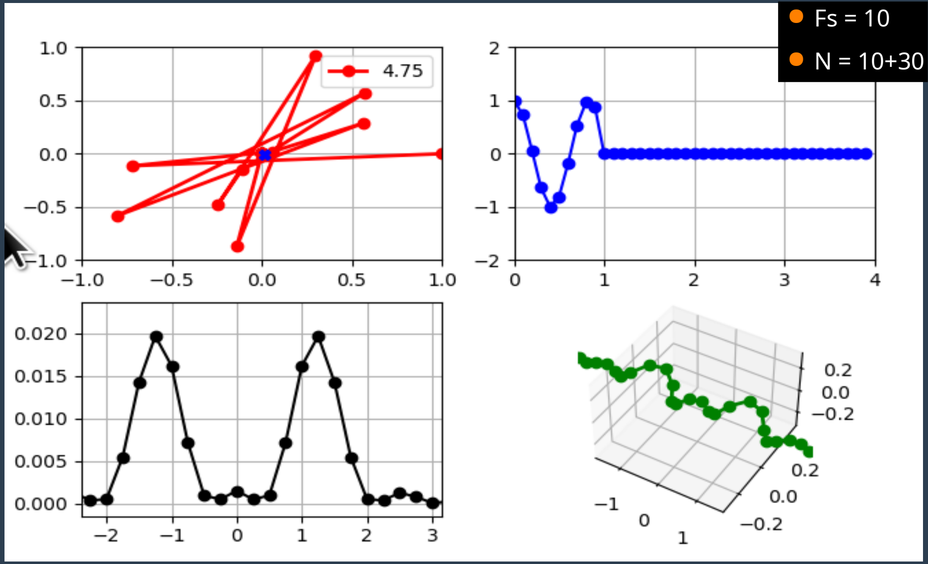
- Ver código: leakage.py

Fuga espectral – Zero padding

- La idea es tener mas frecuencias intermedias entre fs/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 fs/N’

Señal de interés 1,2Hz

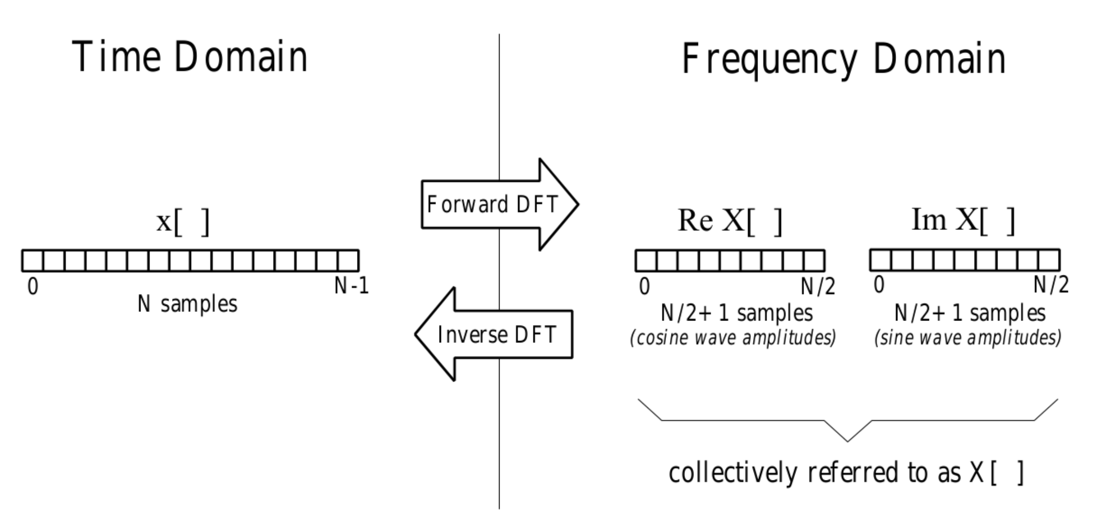


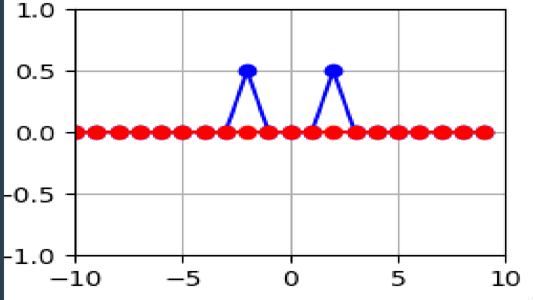


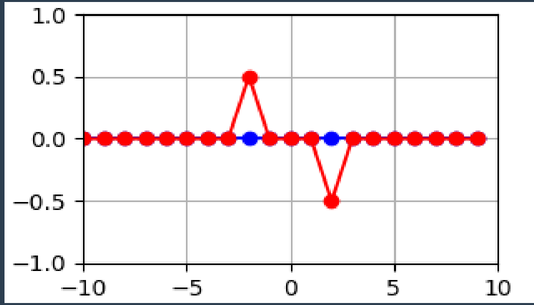
- Ver código: zero_padding.py

Simetria en DFT con entrada Real pura

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



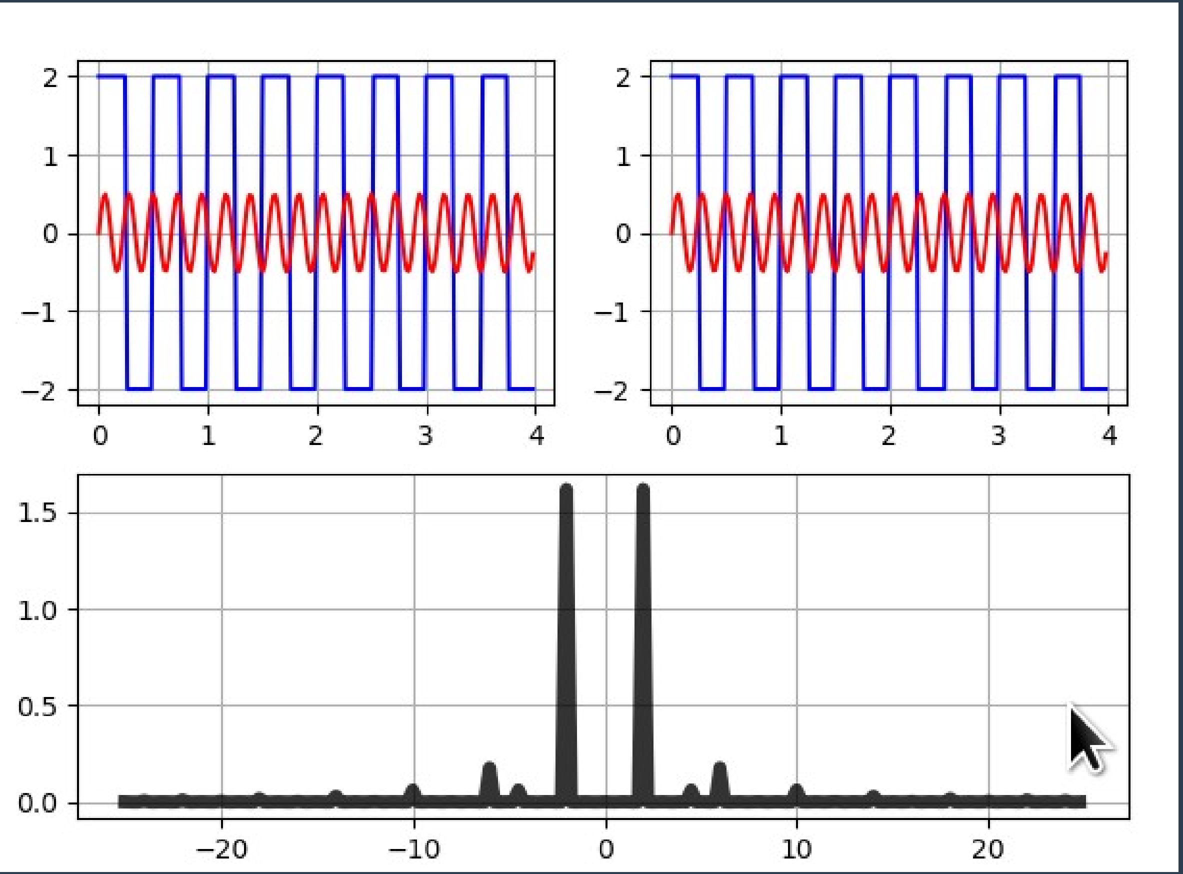




- Ver código: dft3.py

DFT con numpy

- Modulo np.fft
- Funcion np.fft.dft
- Evaluar np.fft.fftshift para centrar en cero
- Tener en cuenta la normalizacion

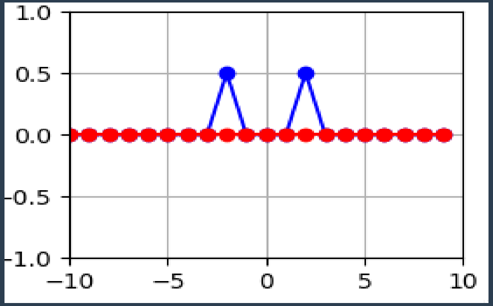


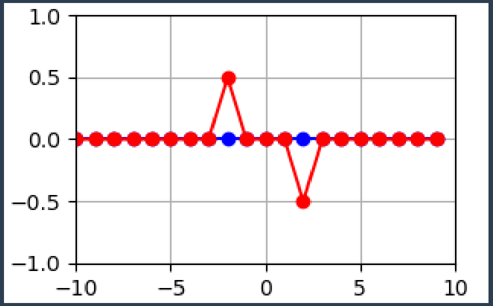
- Ver código: dft_idft_numpy.py

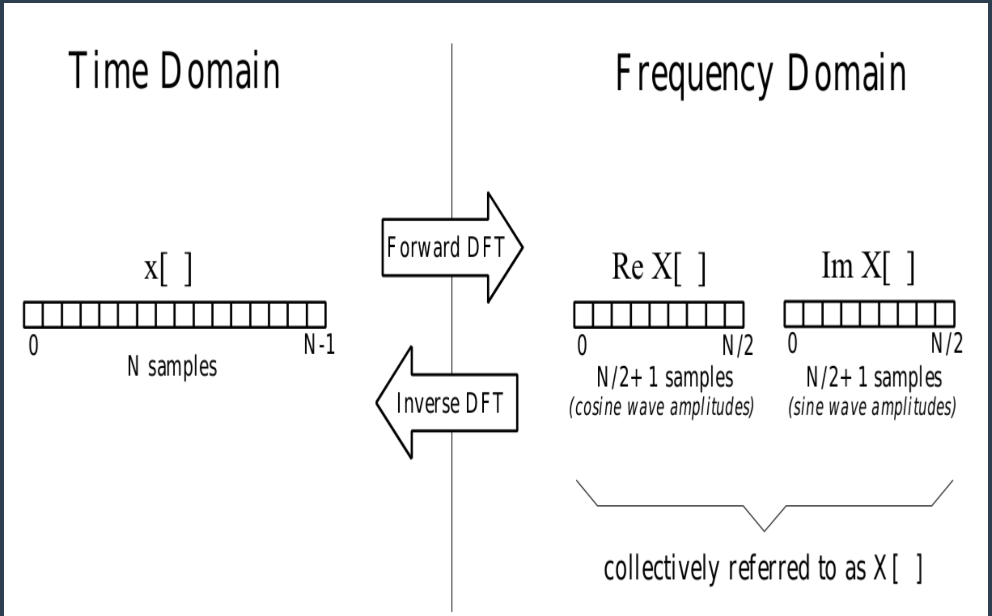
DFT con la CIAA

DFT Real con CMSIS-DSP

- Si la señal de entrada es reales, 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





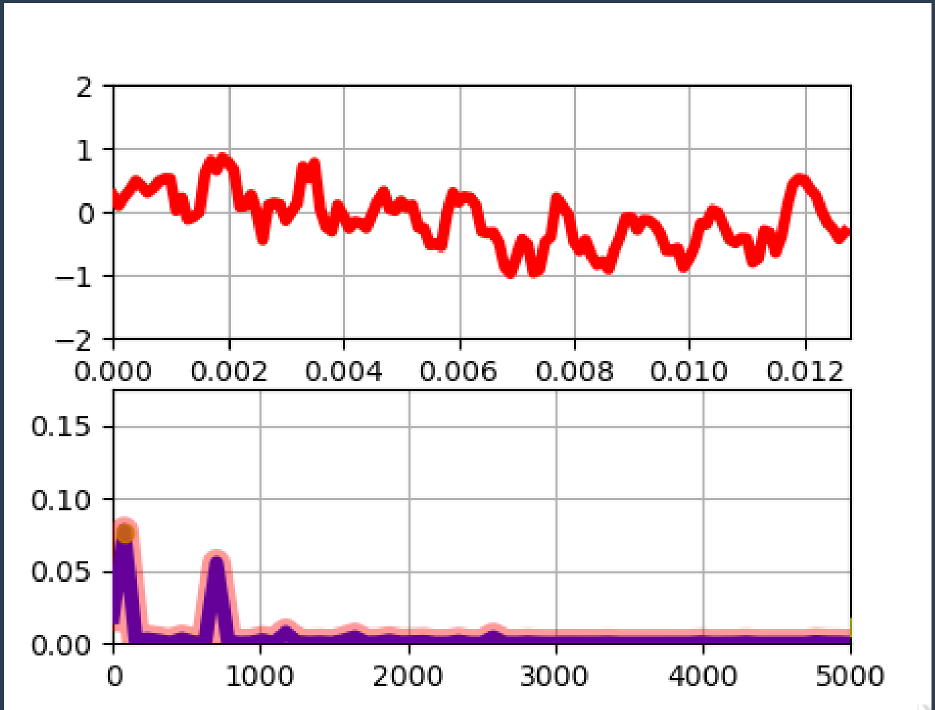


X = { real[0], imag[0], real[1], imag[1], real[2], imag[2] … real[(N/2)-1], imag[(N/2)-1 }

- Ver carpeta clase3/psf1

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.




```
arm_rfft_init_q15          ( &S ,header.N ,0 ,1          );
arm_rfft_q15               ( &S ,fftIn ,fftOut           );
arm_cmplx_mag_squared_q15 ( fftOut ,fftMag ,header.N/2+1 );
```

- Ver carpeta clase3/psf1