FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Procesamiento de señales, fundamentos

Maestría en sistemas embebidos MSE2020
Universidad de Buenos Aires

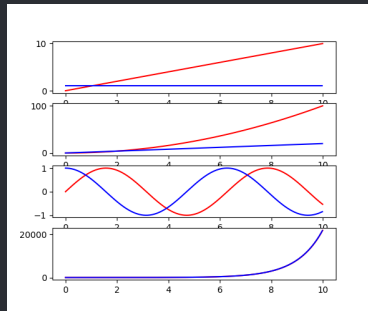## Clase 2 - Euler | Fourier

Ing. Pablo Slavkin

- $f(t) = t$
- $f(t) = t^2$
- $f(t) = sin(t)$

- $f'(t) = 1$
- $f'(t) = 2 * t$
- $f'(t) = cos(t)$

La derivada es igual a la funcion

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

$e^{j2\pi ft}$

Euler

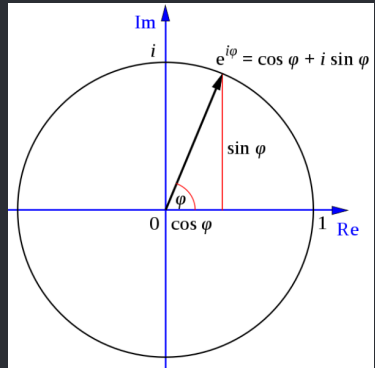Pero que pasa con e^{jt}?

La derivada es igual a la funcion

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

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

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

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

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

$e^{j2\pi ft}$

$e^{j2\pi ft}$ animado

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

fig       = plt.figure()
fs        = 10
N         = 10

circleAxe  = fig.add_subplot(1,1,1)
circleLn,  = plt.plot([],[],'ro')
circleAxe.grid(True)
circleAxe.set_xlim(-2,2)
circleAxe.set_ylim(-2,2)
circleFrec = 1

circle  = lambda c,f,n: c*np.exp(-1j*2*np.pi*f*n*1/fs)

def update(n):
    circleLn.set_data(np.real(circle(1,circleFrec,n)),
                      np.imag(circle(1,circleFrec,n)))
    return circleLn,

ani=FuncAnimation(fig,update,N,interval=1000 ,blit=False,repeat=True)
plt.show()
```

$e^{j2\pi ft}$

$e^{j2\pi ft}$ y $\sin(t)$ animados independientemente

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
#-------------------------------------
fig       = plt.figure()
fs        = 50
N         = 50
#-------------------------------------
circleAxe = fig.add_subplot(2,2,1)
circleLn, = plt.plot([],[],'r-')
circleAxe.grid(True)
circleAxe.set_xlim(-2,2)
circleAxe.set_ylim(-2,2)
circleFrec = 1
circleData=[]
circle = lambda c,f,n: c*np.exp(-1j*2*np.pi*f*n*1/fs)
#-------------------------------------
signalAxe = fig.add_subplot(2,2,2)
signalLn, = plt.plot([],[],'b-')
signalAxe.grid(True)
signalAxe.set_xlim(0,N/fs)
signalAxe.set_ylim(-1,1)
signalFrec = 2
signalData=[]
signal = lambda f,n: np.cos(2*np.pi*f*n*1/fs)
#-------------------------------------
tData=[]
def init():
    return circleLn,
def update(n):
    global circleData,signalData,tData
    circleData.append(circle(1,circleFrec,n))
    circleLn.set_data(np.real(circleData),
                      np.imag(circleData))
    signalData.append(signal(signalFrec,n))
    tData.append(n/fs)
    signalLn.set_data(tData,signalData)

    if n==N-1:
        circleData=[]
        signalData=[]
        tData=[]
    circleLn.set_label(n)
    circleAxe.legend()
    return circleLn,circleAxe,signalLn
ani=FuncAnimation(fig,update,N,init,interval=10 ,blit=True,repeat=True)
plt.show()
```
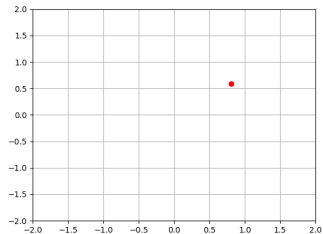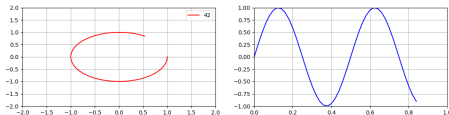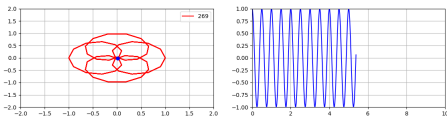
$e^{j2\pi ft}$ modulado por $\sin(t)$ y centro de masas

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
#--------------------------------------
fig       = plt.figure()
fs        = 50
N         = 500
#--------------------------------------
circleAxe = fig.add_subplot(2,2,1)
circleLn,promLn = plt.plot([],[],'r-',[],[],'bo')
circleAxe.grid(True)
circleAxe.set_xlim(-2,2)
circleAxe.set_ylim(-2,2)
circleFrec = 2
circleData = []
prom= 0
circle    = lambda c,f,n: c*np.exp(-1j*2*np.pi*f*n*1/fs)
#--------------------------------------
signalAxe = fig.add_subplot(2,2,2)
signalLn, = plt.plot([],[],'b-')
signalAxe.grid(True)
signalAxe.set_xlim(0,N/fs)
signalAxe.set_ylim(-1,1)
signalFrec = 2
signalData=[]
signal    = lambda f,n: np.cos(2*np.pi*f*n*1/fs)
#--------------------------------------
tData=[]
def init():
    return circleln,
def update(n):
    global circleData,signalData,tData,promData
    circleData.append(circle(1,circleFrec,n)*signal(signalFrec,n))
    prom=np.average(circleData)
    promLn.set_data(np.real(prom),
                    np.imag(prom))
    circleLn.set_data(np.real(circleData),
                      np.imag(circleData))
    signalData.append(signal(signalFrec,n))
    tData.append(n/fs)
    signalLn.set_data(tData,signalData)

    if n==N-1:
        circleData = []
        signalData = []
        tData      = []
        prom       = 0
    circleLn.set_label(n)
    circleAxe.legend()
    return circleLn,circleAxe,signalLn,promLn
ani=FuncAnimation(fig,update,N,init,interval=10 ,blit=True,repeat=True)
plt.show()
```
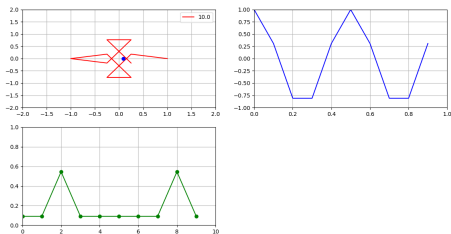
$e^{j2\pi ft}$

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

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
#-----------------------------------
fig     = plt.figure()
fs      = 10
N       = 10
#-----------------------------------
circleAxe = fig.add_subplot(2,2,1)
circleLn,promLn = plt.plot([],[],'r-',[],[],'bo')
circleAxe.grid(True)
circleAxe.set_xlim(-2,2)
circleAxe.set_ylim(-2,2)
circleFrec = 0
circleData = []
prom       = 0
frecIter   = 0
circle   = lambda c,f,n: c*np.exp(-1j*2*np.pi*f*n/fs)
#-----------------------------------
signalAxe = fig.add_subplot(2,2,2)
signalLn, = plt.plot([],[],'b-')
signalAxe.grid(True)
signalAxe.set_xlim(0,N/fs)
signalAxe.set_ylim(-1,1)
signalFrec = 2
signalData=[]
signal   = lambda f,x: np.cos(2*np.pi*f*n*x/fs)
#-----------------------------------
fourierAxe = fig.add_subplot(2,2,3)
fourierLn, = plt.plot([],[],'g-o')
fourierAxe.grid(True)
fourierAxe.set_xlim(0,fs)
fourierAxe.set_ylim(0,1)
fourierData=[]
#-----------------------------------
tData=[]
fData=[]
def init():
    return circleLn,
def update(n):
    global circleData,signalData,tData,promData,frecIter,circleFrec,fourierData,fData
    circleData.append(circle(1,circleFrec,n)*signal(signalFrec,n))
    prom=np.average(circleData)
    promLn.set_data(np.real(prom),
                    np.imag(prom))
    circleLn.set_data(np.real(circleData),
                      np.imag(circleData))
    signalData.append(signal(signalFrec,n))
    tData.append(n/fs)
    signalLn.set_data(tData,signalData)
    if n==N-1:
        circleData = []
        signalData = []
        tData      = []
        fourierData.append(np.real(prom))
        fData.append(circleFrec)
        fourierLn.set_data(fData,fourierData)
        prom       = 0
        frecIter+=1
        if frecIter == N:
            ani.repeat=False
        circleFrec = frecIter*fs/N
        circleLn.set_label(circleFrec)
        circleAxe.legend()
    return circleLn,circleAxe,signalLn,promLn,fourierLn
ani=FuncAnimation(fig,update,N,init_interval=10 ,blit=True,repeat=True)
plt.show()
```

$e^{j2\pi ft}$

$e^{j2\pi ft}$ *del centro de masas de vuelta a* $\sin(t)$

Señales complejas, conejo
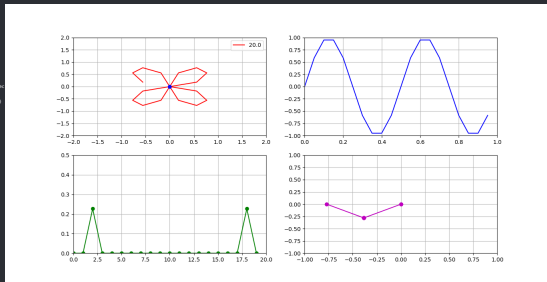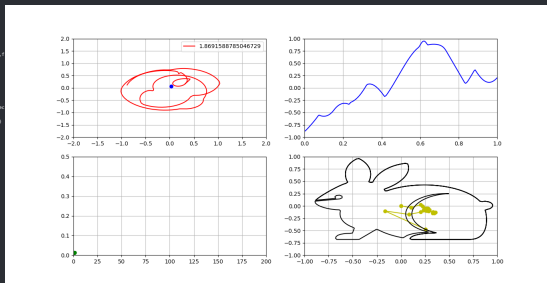
Bibliografia
Libros, links y otro material

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

[2] *Interactive Mathematics Site Info*.

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