FACULTAD DE INGENIERIA

Universidad de Buenos Aires

Procesamiento de señnales, fundamentos

Maestría en sistemas embebidos MSE2020

Universidad de Buenos Aires

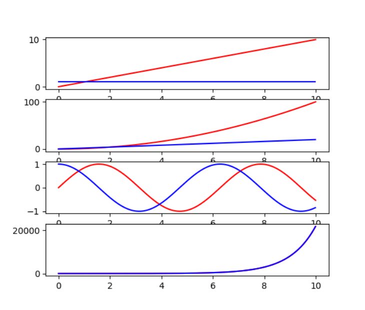
Clase 2 - Euler | Fourier

Ing. Pablo Slavkin

2.7182818284590450907955982984276488423347473144

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

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



La derivada es igual a la funcion

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

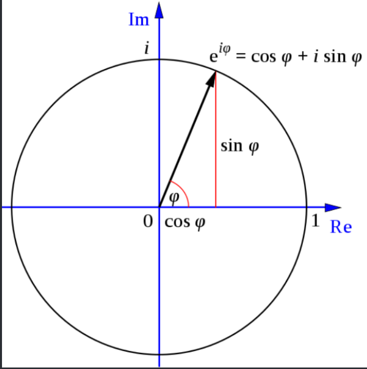
$e^{j2\pi t}$

Euler

Pero que pasa con e^{jt}?

La derivada es igual a la funcion

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



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

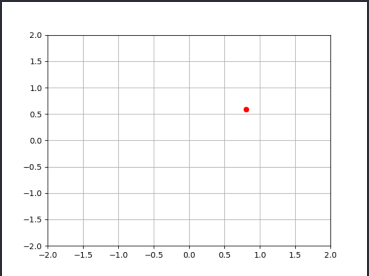
$$e^{j\pi} = -1$$

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

$$e^{\frac{j3\pi}{2}} = i$$

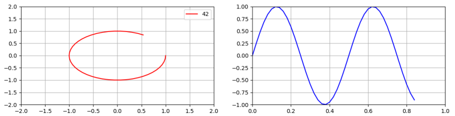
$e^{j2\pi t}$ animado

```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=100 ,blit=False,repeat=True)
plt.show()
```



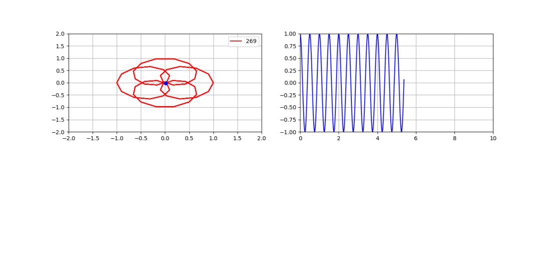
$e^{j2\pi t}$ y $\sin(t)$ *animados independientemente*

```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()
```



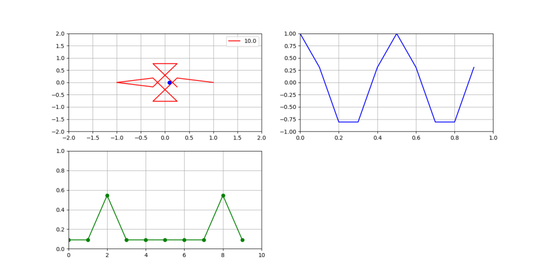
$e^{j2\pi t}$ modulado por $\sin(t)$ y centro de masas

```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 = 3
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()
```



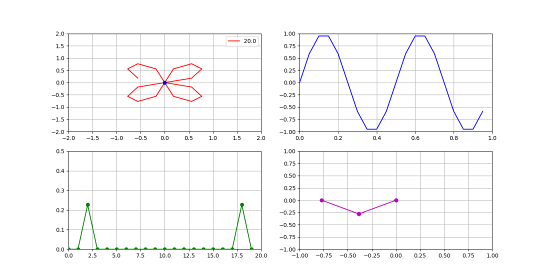
$e^{j2\pi t}$ modulado por $\sin(t)$ y centro de masas en f

```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*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)
#------------------------------------
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()
```



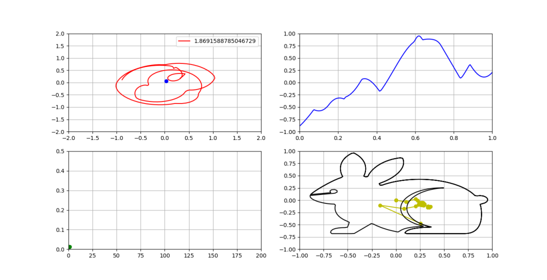
$e^{j2\pi t}$ del centro de masas de vuelta a $\sin(t)$

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
#------------------------------------
fig        = plt.figure()
fs         = 20
N          = 20
#------------------------------------
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*1/fs)
circleInv  = 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.sin(2*np.pi*f*n*1/fs)#+0.5*np.cos(2*np.pi*
                    f*2*n*1/fs)
#------------------------------------
fourierAxe  = fig.add_subplot(2,2,3)
fourierLn,  = plt.plot([],[],'g-o')
fourierAxe.grid(True)
fourierAxe.set_xlim(0,fs)
fourierAxe.set_ylim(0,0.5)
fourierData=[]
#------------------------------------
inversaAxe         = fig.add_subplot(2,2,4)
inversaLn,vectorLn = plt.plot([],[],'m-o',[],[],'b-o')
inversaAxe.grid(True)
inversaAxe.set_xlim(-1,1)
inversaAxe.set_ylim(-1,1)
inversaData = []
vectorData  = []
#------------------------------------
tData=[]
fData=[]

def init():
    return circleln,
def updateF(n):
    global fourierData,fData,vectorData
    if aniT.repeat==True:
        return
    vectorData=[0]
    for f in range(N):
        vectorData.append(vectorData[-1]+circleInv(np.abs(
                        fourierData[f]),f*fs/N,n))
    inversaLn.set_data(np.real(vectorData),np.imag(vectorData))
    return inversaLn,

def updateT(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(prom)
        fData.append(circleFrec)
        fourierLn.set_data(fData,np.abs(fourierData)**2)
        prom       = 0
        frecIter+=1
        if frecIter == N:
            aniT.repeat=False
        circleFrec = frecIter*fs/N
        circleLn.set_label(circleFrec)
        circleAxe.legend()
    return circleLn,circleAxe,signalLn,promLn,fourierLn

aniT=FuncAnimation(fig,updateT,N,init,interval=10  ,blit=True,
                    repeat=True)
aniF=FuncAnimation(fig,updateF,N,init,interval=300 ,blit=True,
                    repeat=True)
plt.show()
```



Señales complejas, conejo

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
#------------------------------------
fig        = plt.figure()
fs         = 10
N          = 200
#------------------------------------
conejo=np.load("conejo.npy")[::10]
N=len(conejo)
#signal=lambda f,n: conejo[n]
#------------------------------------
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*1/fs)
circleInv  = 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 = 0.1
signalData=[]
signal  = lambda f,n: 0.5*np.sin(2*np.pi*f*n*1/fs)+0.5j*np.sin(2*np
                    .pi*f*2*n*1/fs)
#------------------------------------
fourierAxe  = fig.add_subplot(2,2,3)
fourierLn,  = plt.plot([],[],'g-o')
fourierAxe.grid(True)
fourierAxe.set_xlim(0,fs)
fourierAxe.set_ylim(0,0.5)
fourierData=[]
#------------------------------------
inversaAxe         = fig.add_subplot(2,2,4)
inversaLn,vectorLn = plt.plot([],[],'y-o',[],[],'k-')
inversaAxe.grid(True)
inversaAxe.set_xlim(-1,1)
inversaAxe.set_ylim(-1,1)
inversaData = []
vectorData  = []
penData     = []
#------------------------------------
tData=[]
fData=[]

#time=np.arange(0,N,1)
#frec=np.arange(0,fs,fs/N)
#fftData=np.fft.fft(signal(signalFrec,time))/N

def init():
    return circleln,
def updateF(n):
    global fftData,vectorData,penData,fourierData
    if aniT.repeat==True:
        return
    vectorData=[0]
    for f in range(N):
        vectorData.append(vectorData[-1]+circleInv(fourierData[f],f
                          *fs/N,n))
    inversaLn.set_data(np.real(vectorData),np.imag(vectorData))
    penData.append(vectorData[-1])
    vectorLn.set_data(np.real(penData),np.imag(penData))
    return inversaLn,vectorLn

def updateT(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,np.real(signalData))

    if n==N-1:
        circleData = []
        signalData = []
        tData      = []
        fourierData.append(prom)
        fData.append(circleFrec)
        fourierLn.set_data(fData,np.abs(fourierData)**2)
        prom       = 0
        frecIter+=1
        if frecIter == N:
            aniT.repeat=False
        circleFrec = frecIter*fs/N
        circleLn.set_label(circleFrec)
        circleAxe.legend()
    return circleLn,circleAxe,signalLn,promLn,fourierLn

aniT=FuncAnimation(fig,updateT,N,init,interval=10  ,blit=True,
                   repeat=True)
aniF=FuncAnimation(fig,updateF,N,init,interval=200 ,blit=True,
                   repeat=True)
plt.show()
```



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*