Matplotlib para gráficas 2D

Matplotlib es el módulo de dibujo de gráficas 2D y 3D que vamos a utilizar, aunque no es el único existente. Matplotlib tiene multitud de librerías de las cuales nosotros, por semejanza a Matlab, utilizaremos pyplot. Recomendamos al lector visitar http://matplotlib.sourceforge.net/, donde puede encontrar multitud de programas y ejemplos de cómo hacer dibujos con Matplotlib. La documentación oficial se encuentra en http://matplotlib.sourceforge.net/contents.htm no obstante, a lo largo de esta sección veremos algunas de las funciones más interesantes

Para poder utilizar el módulo Matplotlib en nuestros programas primero debemos importarlo. Por comodidad los módulos de nombre más extenso suelen renombrarse para optimizar su importación. En nuestro caso vamos a importarlo como **plt** incluyendo en el inicio de nuestro programa la siguiente línea de código:

```
1  import matplotlib.pyplot as plt
2
3  import numpy as np
```

Hay tres formas de usar la librería Matplotlib:

- La podemos usar desde Python usando el módulo pylab. El módulo pylab pretende mostrar un entorno de trabajo parecido al de Matlab mezclando las librerías numpy y Matplotlib.
  Es la forma menos complicada de usar Matplotlib y se obtiene usando

```
1  from pylab import *
```

- Una segunda forma, que es la que veremos en este tutorial, es usando el módulo pyplot.

```
1  import matplotlib.pyplot as plt
```

- Por último, la forma más recomendable pero más compleja, sería usar Matplotlib mediante la interfaz orientada a objetos. Cuando se programa con Matplotlib, no mientras se trabaja interactivamente, esta es la forma que permite tener más control sobre el código.

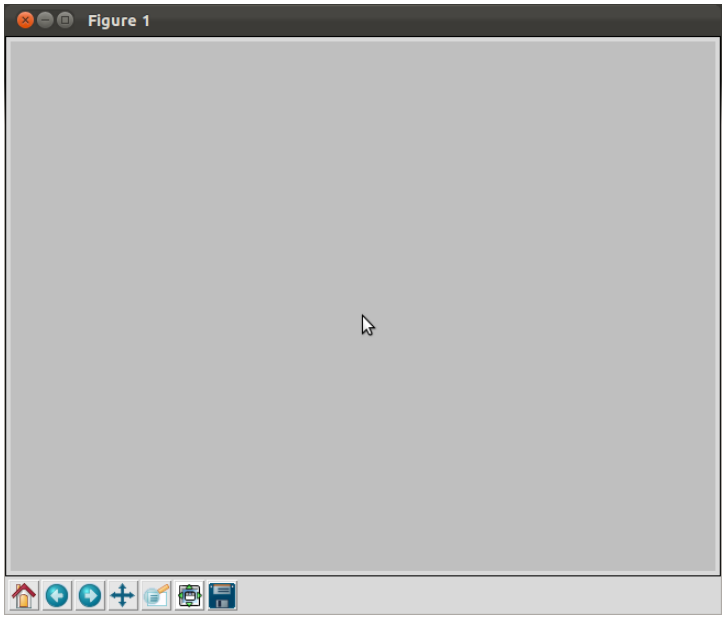
- `from matplotlib import pyplot`

  donde para ejecutar la función debemos usar
- `pyplot.funcion()`

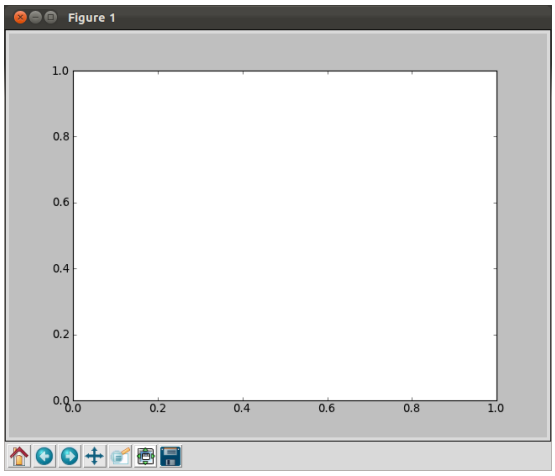
  o cargar todas las funciones,
- `from matplotlib import *`

Para que quede claro desde un principio, las dos zonas principales donde se dibujaran cosas o sobre las que se interactuará serán:

- E es la ventana donde irá el o los gráficos en sí:



- axes, que es el gráfico en sí donde se dibujará todo lo que le digamos y está localizada dentro de una figure.



Para lo primero (figure) usaremos la palabra 'ventana' mientras que para lo segundo (axes) usaremos la palabra 'gráfico'.

Funciones principales

El módulo Matplotlib (al igual que la mayoría de los módulos) es enorme y explicar todas y cada una de las funciones nos llevaría demasiado tiempo así que analizaremos las más interesantes

figure(num, figsize, dpi, facecolor, edgecolor, frameon) –> Crea una nueva figura. Se puede utilizar sin argumentos. Devuelve un identificador a la figura.

num = numeración de la figura, si num = None, las figuras se numeran automáticamente.

figsize = (ancho, alto) . Tamaño de la figura

dpi = Resolución de la imagen en puntos por pulgada.

facecolor = Color del rectángulo de la figura.

edgecolor = Color del perímetro de la figura.

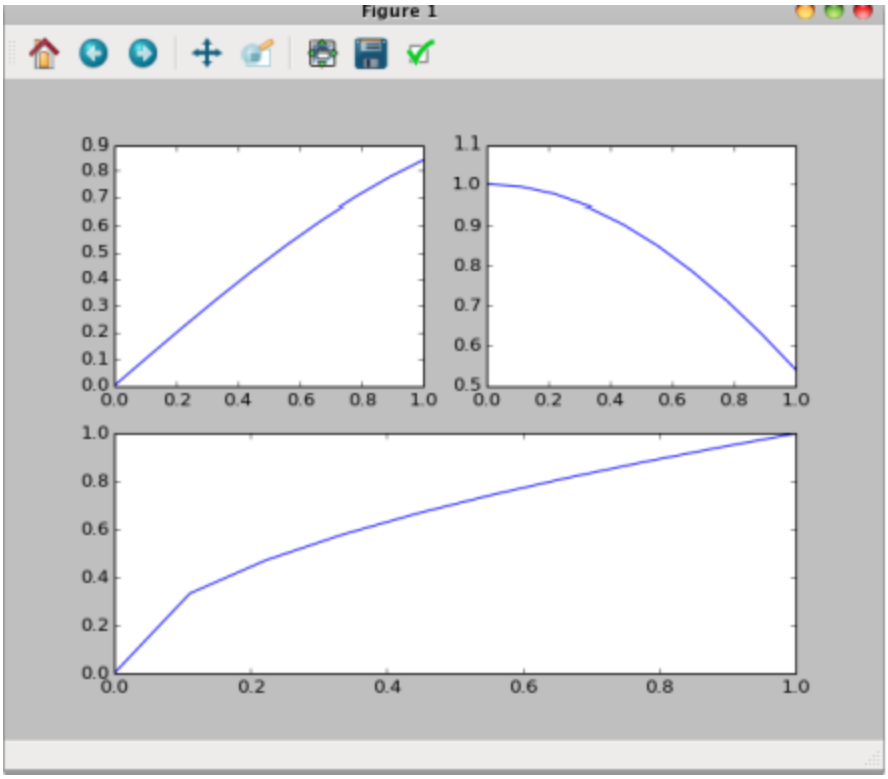
frameon = Si es falso, elimina el marco de la figura.

figure(num = None, figsize = (8, 6), dpi = 80, facecolor = 'w', edgecolor = 'k')

subplot(numfilas, numColumnas, numerografico) –> Permite incluir varias gráficas en una única figura.

subplot(211)

```
subplot(221)
subplot(222)
subplot(212)
x=linspace(0,1,10)
y=sin(x)
z=cos(x)
w=sqrt(x)
plot(x,w) # dibuja sobre el eje activo (212)
subplot(221) # nuevo eje activo (221)
plot(x,y) # dibuja sobre el eje activo (221)
subplot(222) # cambiamos de eje activo al (222)
plot(x,z) # dibuja sobre el eje activo (222)
```



plot(x, y, linestyle, linewidth, marker) –> Permite incluir varias gráficas en una única figura.

x = Abscisas.

y = Ordenadas. Tanto x como y pueden ser abscisas tuplas, listas o arrays. La única condición es que el tamaño de ambas debe ser el mismo ya que en caso contrario Python nos devolverá un fallo de tipo dimensión. También se puede hacer una gráfica sin especificar la coordenada x.

linestyle = color y tipo de dibujar la gráfica. Por ejemplo 'k- -'

linewidth = ancho de línea.

marker = Marcador.

```
plt.plot(x, y, 'k--', linewidth = 2)
```

Carácter	Estilo de línea
-	línea continua
--	línea discontinua
:	línea punteada
-.	línea semipunteada

Carácter	Color	Carácter	Marcador
b	azul	.	punto
g	verde	o	círculo
r	rojo	^	triángulo
y	amarillo	*	estrella
m	magenta	x	cruz
k	negro	s	cuadrado
w	blanco	+	signo más

También se puede escribir el color de las siguientes formas: nombres ('green'); cadenas hexadecimales ('#008000'); tuplas con convención RGB (0,1,0); intensidades de escala de grises ('0.8').

Marcadores:

Los tipos principales son:

['+' | '*' | ',' | '.' | '1' | '2' | '3' | '4' | '<' | '>' | 'D' | 'H' | '^' | '_' | 'd' | 'h' | 'o' | 'p' | 's' | 'v' | 'x'].

Fillstyle Relleno de símbolo elegido para representar cada dato.

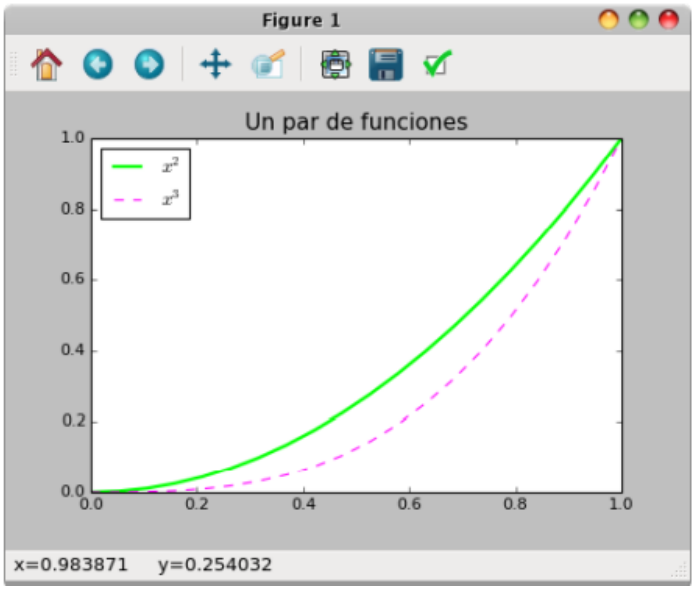
['full' | 'left' | 'right' | 'bottom' | 'top']

show() Presenta las figuras en pantalla mediante una ventana en la que podemos interactuar. Este comando se debe añadir siempre que necesitemos observar nuestra gráfica ya que sino python

realizará el cálculo, pero no presentará la imagen. Lo usual es ponerlo al final del programa posibilitando así la llamada a todas las figuras que se encuentren en el mismo.

legend(labels, loc) Coloca una leyenda. Cuando se presentan varías curvas simultáneamente este comando identifica cada una. Puede utilizarse sin argumentos si en cada función plot (o equivalente) se ha incluido el argumento 'label', que es un texto para identificar la curva.
legend(('Etiqueta1', 'Etiqueta2', 'Etiqueta3'), loc = 'upper left')

```
x=linspace(0,1,20)
y=x**2
z=x**3
plot(x,y,linewidth=2,label='$x^2$',color=(0,1,0))
plot(x,z,linestyle='dashed',color=(1,0,1),label='$x^3$')
title('Un par de funciones',fontsize=14)
legend(loc=0)
```



xlabel('s', comandos_optativos) Etiqueta el eje de abscisas de la gráfica actual

plt.ylabel('s', comandos_optativos)
Etiqueta el eje de abscisas de la gráfica actual.
title('s', comandos_optativos)
Etiqueta el eje de abscisas de la gráfica actual.

- s = Texto que aparecerá en el título

- comandos_optativos = En esta etiqueta englobamos todos los modificadores de la fuente etc.

```
xlabel("Tiempo (s)", fontsize = 20)
ylabel("Distancia (m)", fontsize = 20)
title("velocidad (m/s)", fontsize = 20)
```

text(x, y, s, comandos_optativos) Añade el texto s en las coordenadas espaciales x, y. El texto se puede modificar como otro texto (tamaño, color, etc.).

- x, y = Coordenadas espaciales horizontal y vertical.
- s = Texto que queremos añadir

```
text(5, 7, "Más texto", fontsize = 12)
```

axis() Establece u obtiene las propiedades de los ejes. Podemos establecer el rango de coordenadas en *x* e *y* que queremos mostrar en el gráfico. Del mismo modo, podemos seleccionar la relación de aspecto entre las coordenadas x e y.

- axis(), devuelve los límites de los ejes ([xmin, xmax, ymin, ymax])
- axis(v), establece los valores límites de los ejes a v = [xmin, xmax, ymin, ymax]
- axis('off'), elimina líneas de ejes y etiquetas
- axis('equal'), cambia los límites de x e y para que los incrementos de x e y tengan la misma longitud (un círculo parece un círculo)
- axis('scaled'), cambia las dimensiones del plot para conseguir la misma longitud de intervalo en x e y.
- axis('tight'), cambia los ejes para que se muestren todos los datos.

axhline(y, xmin, xmax) Con esos valores de los parámetros predeterminados establecidos, la línea se ajusta al rango representado por los correspondientes ejes.

- y, array con los datos
- xmin = 0, valor mínimo
- xmax = 1, valor máximo

```
axhline(y = .5, xmin = 0.25, xmax = 0.75
```

hold() Si el parámetro es *True*, podemos poner más curvas en la misma gráfica. Si el parámetro es *False*, entonces al escribir una nueva curva, las anteriores se borran.

grid() Establece la malla a True (visible) o False (oculta).

```
grid(True)
grid(color = 'r', linestyle = '', linewidth = 2)
```

savefig(ruta) Guarda la gráfica en un archivo.

- ruta: ruta y nombre de archivo. Los tipos de datos pueden ser .png, .eps, .pdf, .ps, .svg.
- dpi = None: resolución de la imagen en puntos por pulgada
- facecolor = 'w': color del rectángulo de la figura
- edgecolor = 'w': color del perímetro de la figura
- orientation = 'portrait': orientación del papel (landscape)
- format = None : (png, pdf, ps, eps y svg).
- transparent = False, si es True, creará una gráfica de fondo transparente.

```
savefig('figura3.eps', dpi = 300) #guarda la gráfica con 300dpi (puntos por pulgada)
```

close() Cierra la gráfica

Asimismo, podemos usar la sintaxis de Latex para hacer más precisas y elegantes nuestras etiquetas. Lo único que debemos hacer es poner nuestra cadena de texto de la siguiente forma: r'cadena latex'. Por ejemplo:

xlabel(r'$₩lambda (₩AA)$') se muestra como $\lambda(\AA)$

Matplotlib para gráficas 3D

Para usar gráficos 3D con Matplotlib precisamos realizar la siguiente importación

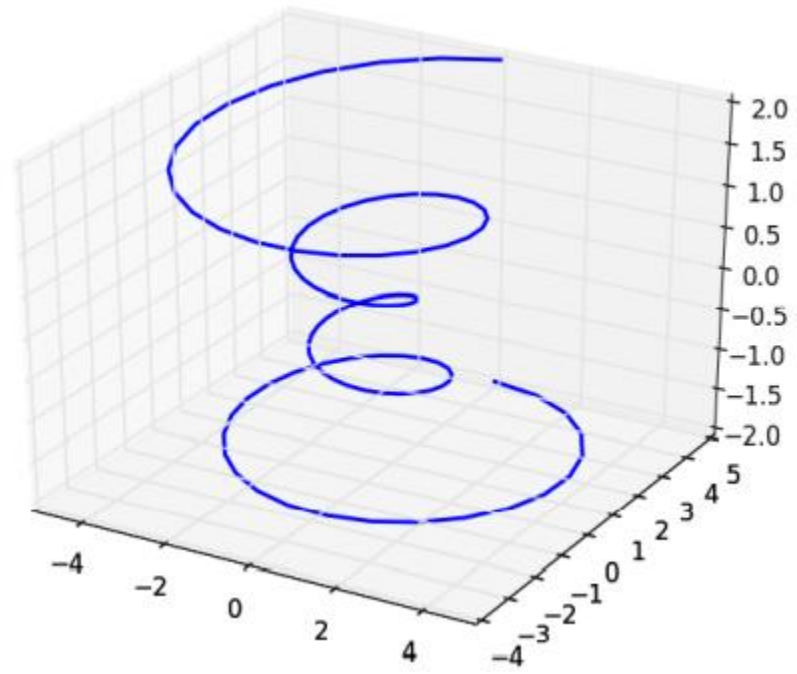
```
from mpl_toolkits.mplot3d import*
```

y a continuación, usamos la opción **Axes3D** a la hora de crear unos ejes:

```
from mpl_toolkits.mplot3d import*
from matplotlib.pyplot import*
from numpy import*

fig=figure(figsize=(8,6));
axes3d=Axes3D(fig);
t = linspace(-4*pi ,4*pi ,100)
z = linspace(-2,2,100)
r = z**2+1
```

```
x = r*sin(t)
y = r*cos(t)
axes3d.plot(x,y,z,linewidth=2)
```



Para dibujar superficies se emplea la misma técnica que en MATLAB, esto es, es necesario crear dos matrices de datos que generan los puntos de una malla bidimensional sobre la que se define la función a dibujar. Por ejemplo, si queremos dibujar el grafo de la función

$$f(x, y) = \sin\left(2\pi\sqrt{x^2 + y^2}\right)$$

en el dominio [−1, 1] × [−1, 1] hemos de preparar los datos de la siguiente forma:

```
from mpl_toolkits.mplot3d import*
from matplotlib.pyplot import*
from numpy import*

fig=figure();
axes3d=Axes3D(fig);
x = linspace(-1,1,150)
X1,Y1=meshgrid(x,x) # mallado fino
Z1=sin(2*pi*sqrt(X1**2+Y1**2))
y = linspace(-1,1,20)
X2,Y2=meshgrid(y,y) # mallado grueso
Z2=sin(2*pi*sqrt(X2**2+Y2**2))
fig = figure(figsize=(12,6))
ax = fig.add_subplot(121,projection='3d')
bx = fig.add_subplot(122,projection='3d')
surf = ax.plot_surface(X1,Y1,Z1)
wire = bx.plot_wireframe(X2,Y2,Z2)
axes3d.set_title('grafica 3D')
axes3d.set_xlabel('Eje x')
axes3d.set_ylabel('Eje y')
axes3d.set_zlabel('Eje z')
show();
```