5.3. Dibujando funciones y nubes de puntos

El módulo matplotlib.pyplot define clases y funciones para dibujar diferentes gráficos matemáticos, entre ellos nubes de puntos (x,y) a través de su función plot(). Si sólo indicamos las ordenadas y usará para x los valores 0, 1... necesarios para hacer el gráfico, uniendo los puntos consecutivos con líneas. Podemos indicar las coordenadas x.

```
import matplotlib.pyplot as plt
plt.plot((2,-1,3,5,2))
```

```
plt.plot((1,2,3,4,5), (2,-1,3,5,2))
```

El formato de la nube de puntos se puede modificar mediante una cadena con determinados caracteres. El color puede indicarse con el carácter r, g, b, c, m, y, k o w. Los marcas de los puntos se indica con -, --, -., :, ., ,, o, v, ^... (ver la ayuda interactiva de matplotlib.pyplot.plot para completar la lista). Si se añade un '-' al formato se unirán los puntos consecutivos con rectas.

```
plt.plot(range(1,5), (2,-1,3,5,2)), 'r^')
plt.plot(range(1,5), (2,-1,3,5,2)), 'bo-')
```

Esta función reconoce muchas opciones indicadas con clave=valor, donde la clave puede ser alpha, animated, color, drawstyle, figure, fillstyle, label, linestyle, linewidth, marker, url y visible entre otras.

Para dibujar varios plots en el mismo gráfico basta con crearlos, los ejes de coordenadas se ajustarán automáticamente para representarlos a todos. En estos casos se aconseja asignar una etiqueta a cada plot y mostrar la leyenda del gráfico con plt.legend().

Si queremos crear varios gráficos añadimos a cada uno sus plots y lo mostramos con plt.show(). También podemos crearlos usando plt.subplot(), plt.subplots() o plt.subplot2grid().

matplotlib.pyplot define muchas funciones para configurar diferentes aspectos del gráfico, como draw(), imsave(), ion(), ioff(), savefig(), title(), vlines(), hlines(), xlabel(), xscale(), xlim(), xticks(), ylabel()...

Aprendimos a representar funciones matemáticas eligiendo muchos valores x y dibujando los puntos (x, f(x)), lo que podemos emular fácilmente con plt.plot(x, f(x)). La potencia del módulo numpy hace más fácil la escritura del código.

```
import matplotlib.pyplot as plt
import numpy as np
f = plt.figure()
g = f.add_subplot(211)
g2 = f.add_subplot(212)
x = np.linspace(-10, 10)
g.plot(x, x**2)
g2.plot(x, np.sin(x))
g2.plot(x, np.cos(x))
```

`matplotlib.figure.Figure` es una clase que facilita la gestión de los gráficos creados con `matplotlib.pyplot`.

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(4,5), dpi=90)
grf = fig.add_subplot(111)
grf.plot((3,7,9,2,11,7), 'o-')
```

Crea objetos figura a los que podemos añadir una cuadrícula en la que dibujar gráficos

Gráficos en tkinter

Si ejecutamos desde consola una aplicación que muestre un gráfico con la función `plt.show()` se mostrará una ventana con el gráfico y una barra con botones que permite interactuar con el gráfico e incluso guardarlo como imagen.

matplotlib contiene módulos con los que se pueden usar interfaces gráficas definidas en otros módulos y paquetes de Python, entre ellos tkinter. Este preámbulo nos da una idea de lo que podemos hacer: crear figuras y usar clases que nos permiten colocarlas en nuestras aplicaciones visuales como si se tratara del widget Canvas, añadiendo una barra de herramientas.

```
import matplotlib
matplotlib.use('TkAgg')
from matplotlib.backends.backend_tkagg import \
    FigureCanvasTkAgg, NavigationToolbar2TkAgg
import matplotlib.pyplot as plt
```

```
import numpy as np
import time
from tkinter import Tk
```

Usaremos `np.random` y `time.time()` para generar una serie de números pseudoaleatorios.

```
np.random.seed(int(time.time()))
z = np.random.randint(0, 100, 1000)
fg, grf = plt.subplots()
grf.plot(z, '.')
```

Creamos el gráfico, con `plt.subplots()` que devuelve una tupla. A continuación creamos la ventana con un Frame para agrupar la figura y la barra de herramientas, que creamos con las dos clases importadas de backends, asociando la primera a la figura y al frame y la segunda a la primera y al frame. Colocamos el widget creado con `FigureCanvasTkAgg` en su padre, el frame, y el frame en la ventana principal.

```
from tkinter import Tk, Frame
v = Tk()
fr = Frame(v)
canvas = FigureCanvasTkAgg(fg, fr)
NavigationToolbar2TkAgg(canvas, fr)
canvas.get_tk_widget().pack(fill='both', expand=True)
fr.pack()
v.mainloop()
```