

PYTHON的繪圖庫

MATPLOTLIB快速入門

Matplotlib

—繪製精美的圖表

matplotlib 是python最著名的繪圖庫，它提供了一整套和matlab相似的命令API，十分適合互動式地進行製圖。而且也可以方便地將它作為繪圖控制項，嵌入GUI應用程式中。

它的文檔相當完備，並且Gallery頁面中有上百幅縮略圖，打開之後都有來源程式。因此如果你需要繪製某種類型的圖，只需要在這個頁面中流覽/複製/粘貼一下，基本上都能搞定。

展示頁面的位址：

<http://matplotlib.sourceforge.net/gallery.html>

快速繪圖

快速繪圖

matplotlib的pyplot子庫提供了和matlab類似的繪圖API，方便用戶快速繪製2D圖表。(matplotlib_simple_plot.py)

pylab 模組

matplotlib還提供了名為pylab的模組，其中包括了許多numpy和pyplot中常用的函數，方便使用者快速進行計算和繪圖，可以用於IPython中的快速互動式使用。

快速繪圖

matplotlib中的快速繪圖的函式程式庫可以通過如下語句載入：

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

接下來調用figure創建一個繪圖對象，並且使它成為當前的繪圖對象。

```
plt.figure(figsize=(8,4))
```

通過figsize參數可以指定繪圖物件的寬度和高度，單位為英寸；dpi參數指定繪圖物件的解析度，即每英寸多少個圖元，缺省值為80。因此本例中所創建的圖表視窗的寬度為 $8 \times 80 = 640$ 圖元。

快速繪圖

也可以不創建繪圖物件直接調用接下來的plot函數直接繪圖，matplotlib會自動創建一個繪圖物件。

如果需要同時繪製多幅圖表的話，可以是給figure傳遞一個整數參數指定圖示的序號，如果所指定序號的繪圖物件已經存在的話，將不創建新的對象，而只是讓它成為當前繪圖對象。

下面的兩行程式通過調用plot函數在當前的繪圖物件中進行繪圖：

```
plt.plot(x,y,label="$sin(x)$",color="red",linewidth=2)  
plt.plot(x,z,"b--",label="$cos(x^2)$")
```

快速繪圖

```
plt.plot(x,y,label="$sin(x)$",color="red",linewidth=2)  
plt.plot(x,z,"b--",label="$cos(x^2)$")
```

plot函數的調用方式很靈活，第一句將x,y陣列傳遞給plot之後，用關鍵字參數指定各種屬性：

- label：給所繪製的曲線一個名字，此名字在圖示(legend)中顯示。只要在字串前後添加"\$"符號，matplotlib就會使用其內嵌的latex引擎繪製的數學公式。
- color：指定曲線的顏色
- linewidth：指定曲線的寬度

第三個參數“b--”指定曲線的顏色和線型

快速繪圖

接下來通過一系列函數設置繪圖物件的各個屬性：

```
plt.xlabel("Time(s)")  
plt.ylabel("Volt")  
plt.title("PyPlot First Example")  
plt.ylim(-1.2,1.2)  
plt.legend()
```

- • xlabel / ylabel : 設置X軸/Y軸的文字
- • title : 設置圖表的標題
- • ylim : 設置Y軸的範圍
- • legend : 顯示圖示

最後調用`plt.show()`顯示出創建的所有繪圖物件。

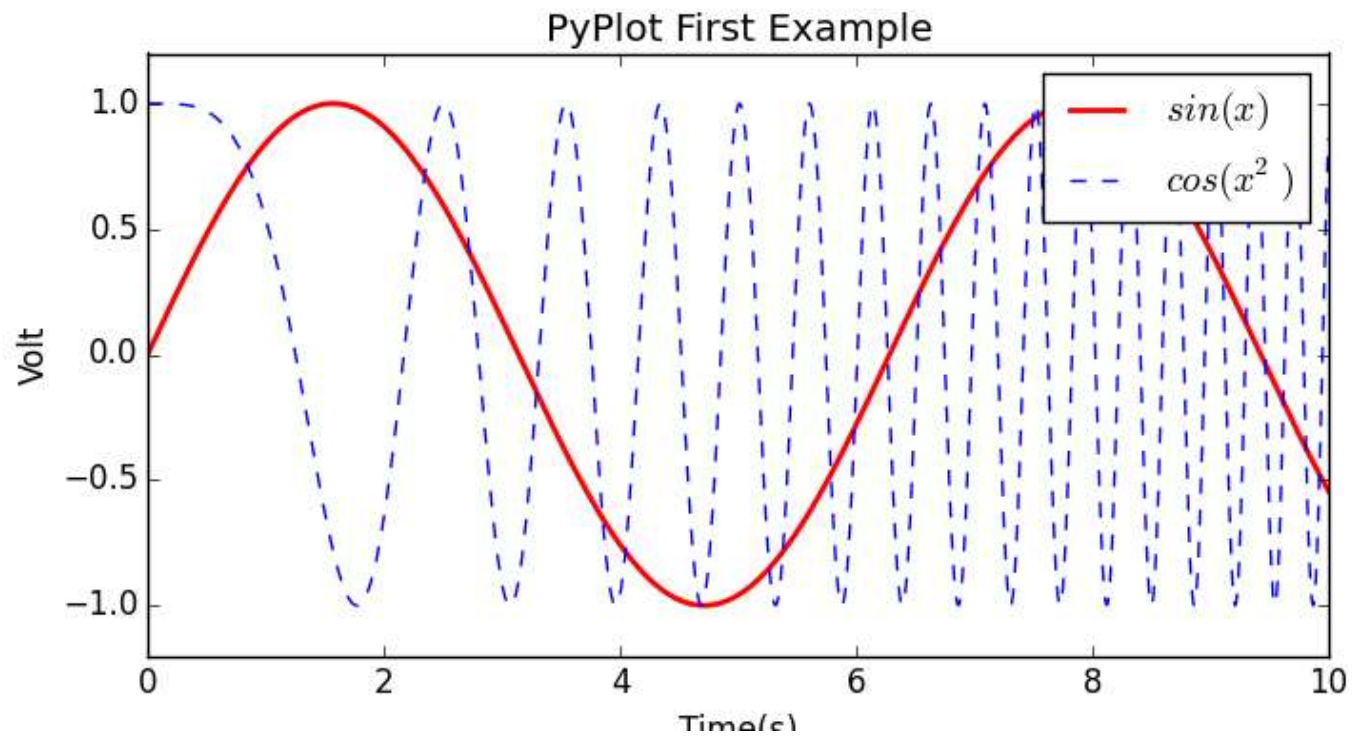
快速繪圖

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 1000)
y = np.sin(x)
z = np.cos(x**2)

plt.figure(figsize=(8,4))
plt.plot(x,y,label="$sin(x)$",color="red",linewidth=2)
plt.plot(x,z,"b--",label="$cos(x^2)$")
plt.xlabel("Time(s)")
plt.ylabel("Volt")
plt.title("PyPlot First Example")
plt.ylim(-1.2,1.2)
plt.legend()
plt.show()
```

快速繪圖



快速繪圖

還可以調用`plt.savefig()`將當前的Figure對象保存成影像檔，圖像格式由影像檔的副檔名決定。下麵的程式將當前的圖表保存為“test.png”，並且通過dpi參數指定圖像的解析度為 120，因此輸出圖像的寬度為“ $8 \times 120 = 960$ ”個圖元。

```
run matplotlib_simple_plot.py  
plt.savefig("test.png",dpi=120)
```

直接用`savefig()`將圖表保存成影像檔.使用這種方法可以很容易編寫出 批量輸出圖表的程式.

快速繪圖

繪製多軸圖

一個繪圖物件(`figure`)可以包含多個軸(`axis`)，在Matplotlib中用軸表示一個繪圖區域，可以將其理解為子圖。上面的第一個例子中，繪圖物件只包括一個軸，因此只顯示了一個軸(子圖(`Axes`))。可以使用`subplot`函數快速繪製有多個軸的圖表。`subplot`函數的調用形式如下：

```
subplot(numRows, numCols, plotNum)
```

快速繪圖

subplot將整個繪圖區域等分為numRows行和numCols列個子區域，然後按照從左到右，從上到下的順序對每個子區域進行編號，左上的子區域的編號為1。如果numRows，numCols和plotNum這三個數都小於10的話，可以把它們縮寫為一個整數，例如subplot(323)和subplot(3,2,3)是相同的。subplot在plotNum指定的區域中創建一個軸物件。如果新創建的軸和之前創建的軸重疊的話，之前的軸將被刪除。

快速繪圖

下麵的程式創建3行2列共6個軸，通過axisbg參數給每個軸設置不同的背景顏色。

```
for idx, color in enumerate("rgbk"):
    plt.subplot(320+idx+1, axisbg=color)
plt.show()
```

如果希望某個軸佔據整個行或者列的話，可以如下調用subplot：

```
plt.subplot(221) # 第一行的左圖
plt.subplot(222) # 第一行的右圖
plt.subplot(212) # 第二整行
plt.show()
```

快速繪圖

當繪圖物件中有多個軸的時候，可以通過工具列中的Configure Subplots按鈕，互動式地調節軸之間的間距和軸與邊框之間的距離。如果希望在程式中調節的話，可以調用subplots_adjust函數，它有left, right, bottom, top, wspace, hspace等幾個關鍵字參數，這些參數的值都是0到1之間的小數，它們是以繪圖區域的寬高為1進行正規化之後的座標或者長度。

快速繪圖

`subplot()`返回它所創建的Axes物件，可以將它用變數保存起來，然後用`sca()`交替讓它們成為當前Axes物件，並調用`plot()`在其中繪圖。如果需要同時繪製多幅圖表，可以給`figure()`傳遞一個整數參數指定Figure物件的序號，如果序號所指定的figure物件已經存在，將不創建新的對象，而只是讓它成為當前的Figure對象。下面的程式演示了如何依次在不同圖表的不同子圖中繪製曲線。

(matplotlib_multi_figure.py)

快速繪圖

首先通過figure()創建了兩個圖表，它們的序號分別為1和2。然後在圖表2中創建了上下並排的兩個子圖，並用變數ax1和ax2保存。

```
import numpy as np
import matplotlib.pyplot as plt

plt.figure(1) # 創建圖表1
plt.figure(2) # 創建圖表2
ax1 = plt.subplot(211) # 在圖表2中創建子圖1
ax2 = plt.subplot(212) # 在圖表2中創建子圖2

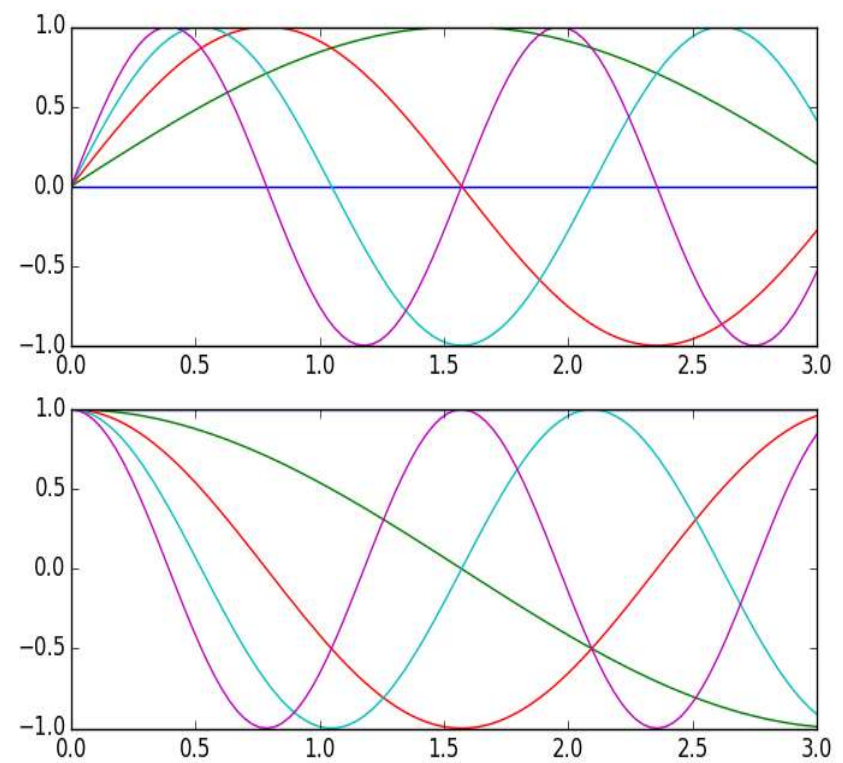
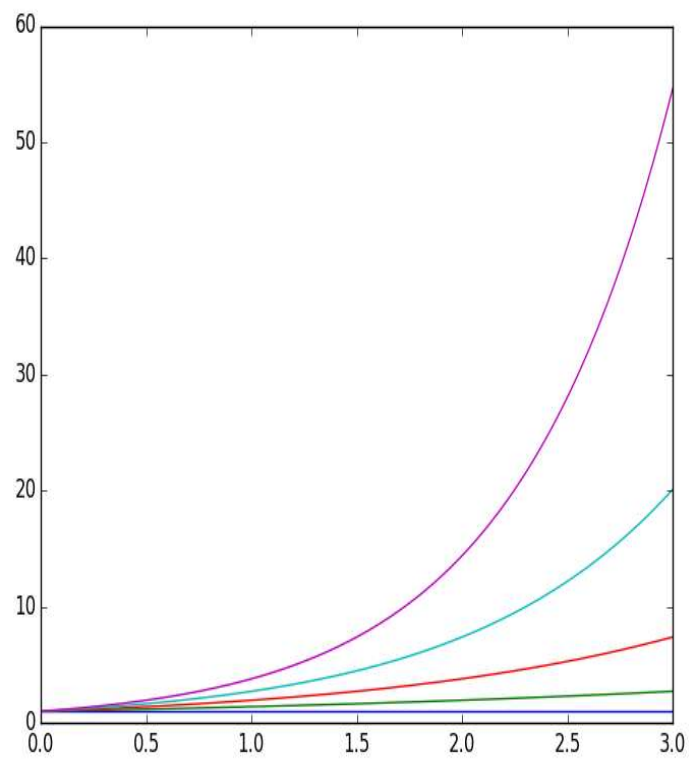
x = np.linspace(0, 3, 100)
```

快速繪圖

在迴圈中，先調用`figure(1)`讓圖表1成為當前圖表，並在其中繪圖。然後調用`sca(ax1)`和`sca(ax2)`分別讓子圖`ax1`和`ax2`成為當前子圖，並在其中繪圖。當它們成為當前子圖時，包含它們的圖表2也自動成為當前圖表，因此不需要調用`figure(2)`依次在圖表1和圖表2的兩個子圖之間切換，逐步在其中添加新的曲線

```
for i in xrange(5):  
    plt.figure(1) # 選擇圖表1  
    plt.plot(x, np.exp(i*x/3))  
    plt.sca(ax1) # 選擇圖表2的子圖1  
    plt.plot(x, np.sin(i*x))  
    plt.sca(ax2) # 選擇圖表2的子圖2  
    plt.plot(x, np.cos(i*x))  
plt.show()
```

快速繪圖



快速繪圖

坐標軸設定

Axis容器包括坐標軸的刻度線、刻度標籤、座標網格以及坐標軸標題等內容。刻度包括主刻度和副刻度，分別通過`get_major_ticks()`和`get_minor_ticks()`方法獲得。每個刻度線都是一個`XTick`或`YTick`物件，它包括實際的刻度線和刻度標籤。為了方便訪問刻度線和文本，Axis物件提供了`get_ticklabels()`和`get_ticklines()`方法，可以直接獲得刻度標籤和刻度線。下面例子進行繪圖並得到當前子圖的X軸對象axis:

```
>>> plt.plot([1,2,3],[4,5,6])
>>> plt.show()
>>> axis = plt.gca().xaxis
```

快速繪圖

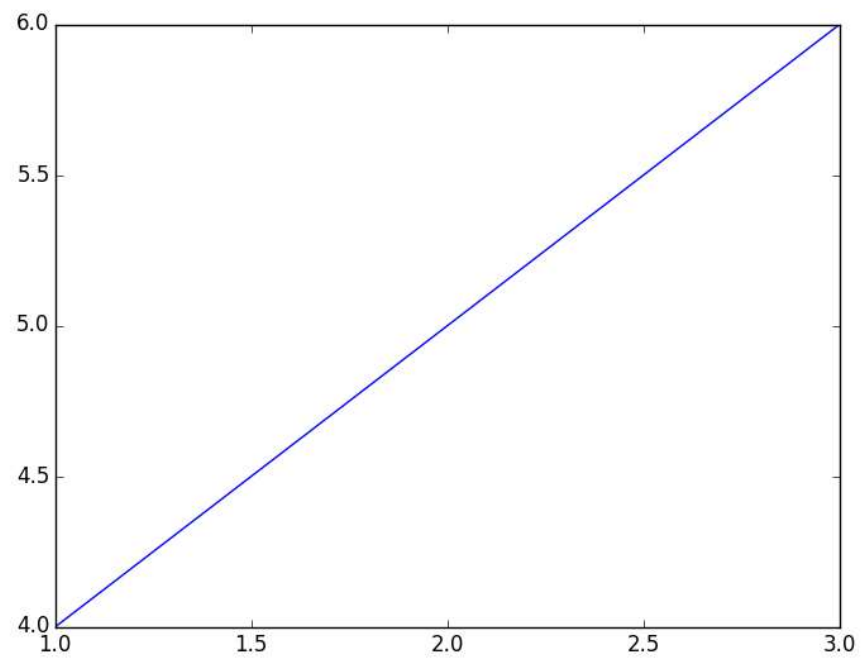
獲得axis對象的刻度位置列表：

```
>>> axis.get_ticklocs()  
array([ 1. , 1.5, 2. , 2.5, 3. ])
```

下面獲得axis物件的刻度標籤以及標籤中的文字：

```
>>> axis.get_ticklabels() # 獲得刻度標籤列表  
<a list of 5 Text major ticklabel objects>  
>>> [x.get_text() for x in axis.get_ticklabels()]  
# 獲得刻度的文本字串  
[u'1.0', u'1.5', u'2.0', u'2.5', u'3.0']
```

快速繪圖



快速繪圖

下面獲得X軸上表示主刻度線的列表，可看到X軸上共有10條刻度線

```
>>> axis.get_ticklines()  
<a list of 10 Line2D ticklines objects>
```

由於沒有副刻度線，因此副刻度線列表的長度為0：

```
>>> axis.get_ticklines(minor=True) # 獲得副刻度線列表  
<a list of 0 Line2D ticklines objects>
```

```
>>> plt.xticks(fontsize=16, color="red", rotation=45)
```

快速繪圖

上面的例子中副刻度線列表為空，這是因為用於計算副刻度位置的物件預設為 `NullLocator`, 它不產生任何刻度線。而計算主刻度位置的物件為 `AutoLocator`, 它會根據當前的縮放等配置自動計算刻度的位置。

`matplotlib` 提供了多種配置刻度線位置的 `Locator` 類，以及控制刻度標籤顯示的 `Formatter` 類。下麵的程式設置 x 軸的主刻度為 $\pi/4$, 副刻度為 $\pi/20$, 並且主刻度上的標籤用數學符號顯示 π 。

(`matplotlib_axis_text.py` 自訂坐標軸的刻度和文字)

快速繪圖

與刻度定位和文本格式化相關的類都在 `matplotlib.ticker` 模組中定義，程式從中載入了兩個類：`MultipleLocator`, `FuncFormatter`.

```
from matplotlib.ticker import MultipleLocator, FuncFormatter
```

```
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator, FuncFormatter
import numpy as np
x = np.arange(0, 4*np.pi, 0.01)
y = np.sin(x)
plt.figure(figsize=(8,4))
plt.plot(x, y)
ax = plt.gca()
```

快速繪圖

程式中通過pi_formatter()計算出刻度值對應的刻度文本. (很繁瑣)

```
def pi_formatter(x, pos):  
    m = np.round(x / (np.pi/4))  
    n = 4  
    while m!=0 and m%2==0: m, n = m//2, n//2  
    if m == 0:  
        return "0"  
    if m == 1 and n == 1:  
        return "$\pi$"  
    if n == 1:  
        return r"$%d \pi$" % m  
    if m == 1:  
        return r"$\frac{\pi}{%d}$" % n  
    return r"$\frac{%d \pi}{%d}$" % (m,n)
```

快速繪圖

```
>>>X = np.linspace(0, 4*np.pi, 17, endpoint=True)
>>>X
array([ 0.          ,  0.78539816,  1.57079633,  2.35619449,
        3.14159265,  3.92699082,  4.71238898,  5.49778714,
        6.28318531,  7.06858347,  7.85398163,  8.6393798 ,
        9.42477796, 10.21017612, 10.99557429, 11.78097245,
       12.56637061])

>>>plt.xticks([ 0.          ,  0.78539816,  1.57079633,
                2.35619449,
                3.14159265,  3.92699082,  4.71238898,  5.49778714,
                6.28318531,  7.06858347,  7.85398163,  8.6393798 ,
                9.42477796, 10.21017612, 10.99557429, 11.78097245,
                12.56637061],
               [r'$0$', r'$\pi/4$', r'$\pi/2$', r'$3\pi/4$',
                r'$\pi$', r'$5\pi/4$', r'$3\pi/2$', r'$7\pi/4$',
                r'$2\pi$', r'$9\pi/4$', r'$5\pi/2$', r'$11\pi/4$',
                r'$3\pi$', r'$13\pi/4$', r'$7\pi/2$', r'$15\pi/4$', r'$4\pi$'])
# r'$ \frac{2\pi}{3} $',
```

快速繪圖

以指定值的整數倍為刻度放置主、副刻度線。

```
ax.xaxis.set_major_locator( MultipleLocator(np.pi/4) )  
ax.xaxis.set_minor_locator( MultipleLocator(np.pi/20) )
```

使用指定的函數計算刻度文本，它會將刻度值和刻度的序號作為參數傳遞給計算刻度文本的函數。

```
ax.xaxis.set_major_formatter( FuncFormatter( pi_formatter ) )
```

```
# 設置兩個坐標軸的範圍  
pl.ylim(-1.5,1.5)  
pl.xlim(0, np.max(x))
```

快速繪圖

```
pl.subplots_adjust(bottom = 0.15) # 設置圖的底邊距

pl.grid() # 開啟網格

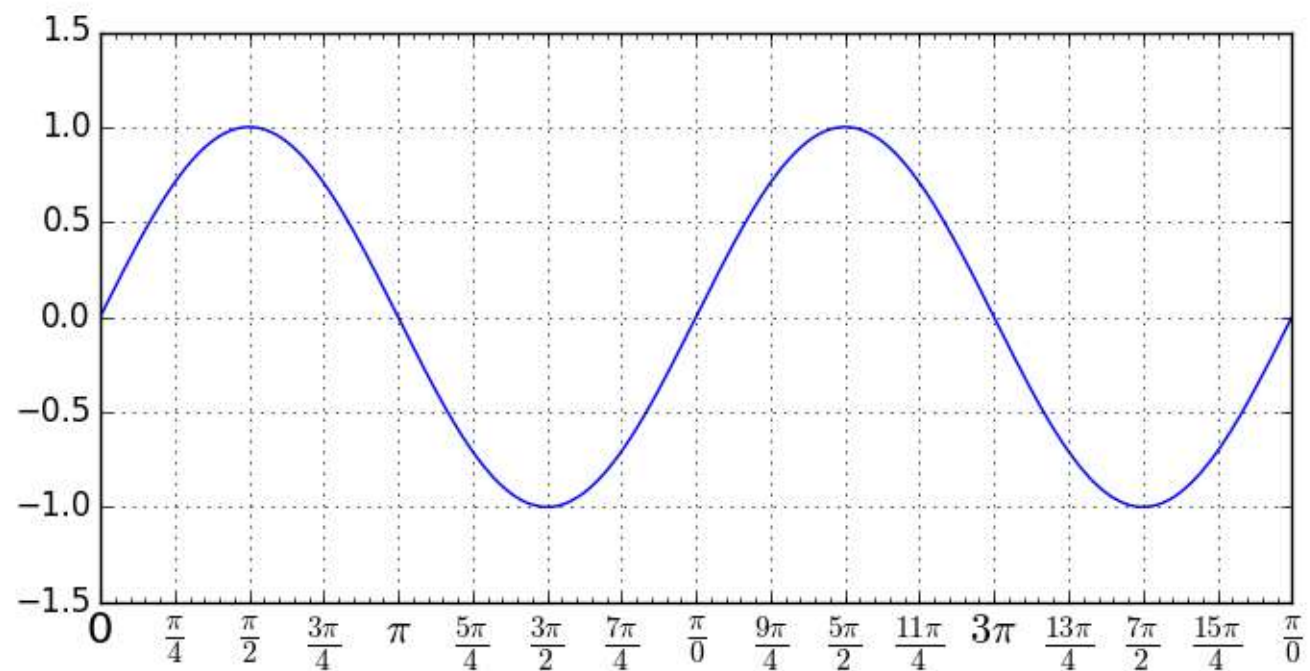
# 主刻度為 $\pi/4$ 
ax.xaxis.set_major_locator( MultipleLocator(np.pi/4) )

# 主刻度文本用pi_formatter函數計算
ax.xaxis.set_major_formatter( FuncFormatter( pi_formatter ) )

# 副刻度為 $\pi/20$ 
ax.xaxis.set_minor_locator( MultipleLocator(np.pi/20) )

# 設置刻度文本的大小
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(16)
pl.show()
```

快速繪圖



繪圖函數簡介

對數座標圖

前面介紹過如何使用`plot()`繪製曲線圖，所繪製圖表的X-Y軸座標都是算術座標。下面看看如何在對數坐標系中繪圖。

繪製對數座標圖的函數有三個：`semilogx()`、`semilogy()`和`loglog()`，它們分別繪製X軸為對數座標、Y軸為對數座標以及兩個軸都為對數座標時的圖表。

繪圖函數簡介

下面的程式使用4種不同的坐標系繪製低通濾波器的頻率回應曲線。其中，左上圖為`plot()`繪製的算術坐標系，右上圖為`semilogx()`繪製的X軸對數坐標系，左下圖為`semilogy()`繪製的Y軸對數坐標系，右下圖為`loglog()`繪製的雙對數坐標系。使用雙對數坐標系表示的頻率回應曲線通常被稱為波特圖。(matplotlib_log.py)

```
import numpy as np
import matplotlib.pyplot as plt

w = np.linspace(0.1, 1000, 1000)
p = np.abs(1/(1+0.1j*w)) # 計算低通濾波器的頻率回應
```


繪圖函數簡介

```
plt.subplot(221)  
plt.plot(w, p, linewidth=2)  
plt.ylim(0,1.5)
```

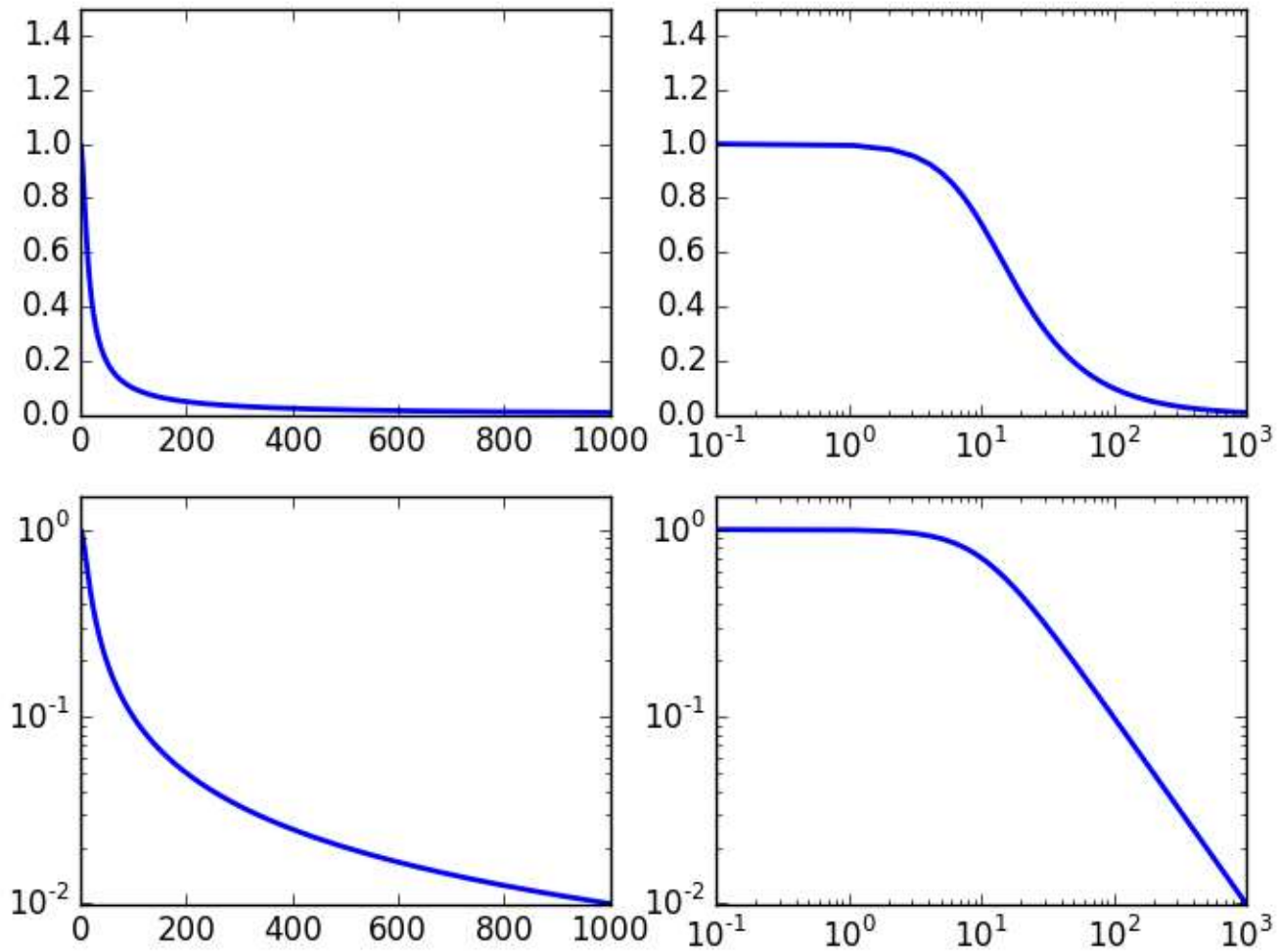
```
plt.subplot(222)  
plt.semilogx(w, p, linewidth=2)  
plt.ylim(0,1.5)
```

```
plt.subplot(223)  
plt.semilogy(w, p, linewidth=2)  
plt.ylim(0,1.5)
```

```
plt.subplot(224)  
plt.loglog(w, p, linewidth=2)  
plt.ylim(0,1.5)
```

```
plt.show()
```

繪圖函數簡介



繪圖函數簡介

極座標圖

極坐標系是和笛卡爾(X-Y)坐標系完全不同的坐標系，極坐標系中的點由一個夾角和一段相對中心點的距離來表示。下麵的程式繪製極座標圖，(matplotlib_polar.py)。

```
import numpy as np
import matplotlib.pyplot as plt

theta = np.arange(0, 2*np.pi, 0.02)
```

繪圖函數簡介

```
plt.subplot(121, polar=True)  
plt.plot(theta, 1.6*np.ones_like(theta), linewidth=2)  
plt.plot(3*theta, theta/3, "--", linewidth=2)
```

在代碼中我們用`subplot()`創建了一個子圖，並設置了`polar`參數為`True`，創建了一個極座標子圖。然後調用`plot()`在極座標子圖中繪圖。也可以使用`polar()`直接創建極座標子圖並在其中繪製曲線。

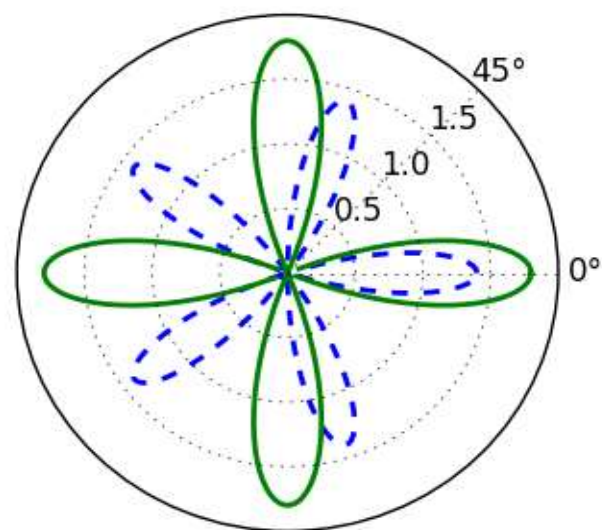
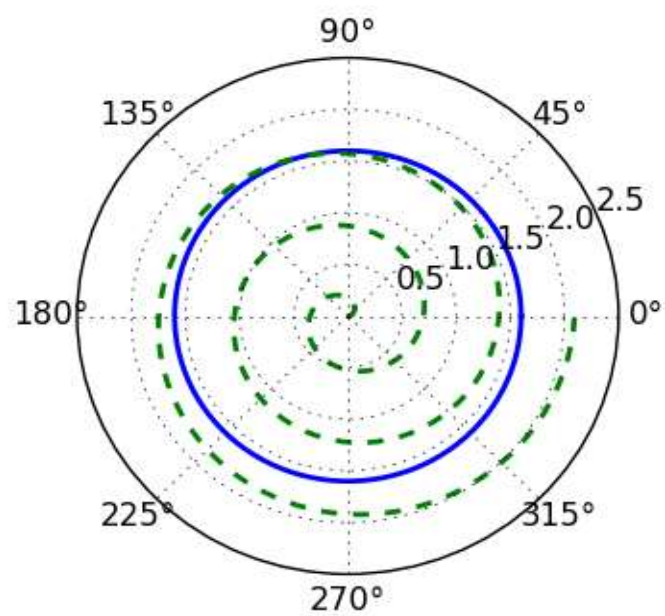
繪圖函數簡介

```
plt.subplot(122, polar=True)
plt.plot(theta, 1.4*np.cos(5*theta), "--", linewidth=2)
plt.plot(theta, 1.8*np.cos(4*theta), linewidth=2)
plt.rgrids(np.arange(0.5, 2, 0.5), angle=45)
plt.thetagrids([0, 45])

plt.show()
```

徑分別為0.5、1.0和1.5,這些文字沿著45°線排列。
Thetagrids()設置放射線柵格的角度，因此右圖
中只有兩條放射線，角度分別為0°和45°。

繪圖函數簡介



繪圖函數簡介

柱狀圖

柱狀圖用其每根柱子的長度表示值的大小，它們通常用來比較兩組或多組值。下面的程式從檔中讀入中國人口的年齡分佈資料，並使用柱狀圖比較男性和女性的年齡分佈。

(matplotlib_bar.py 繪製比較男女人口的年齡分佈圖)

```
import numpy as np
import matplotlib.pyplot as plt
```

繪圖函數簡介

```
data = np.loadtxt("china_population.txt")  
width = (data[1,0] - data[0,0])*0.4
```

讀入的數據中，第0列為年齡，它將作為柱狀圖的橫坐標。首先計算柱狀圖中每根柱子的寬度，因為要在每個年齡段上繪製兩根柱子，因此柱子的寬度應該小於年齡段的二分之一。這裡以年齡段的0.4倍作為柱子的寬度。

繪圖函數簡介

```
plt.figure(figsize=(8,5))
```

```
plt.bar(data[:,0]-width, data[:,1]/1e7, width,  
color="b", label=u"男")
```

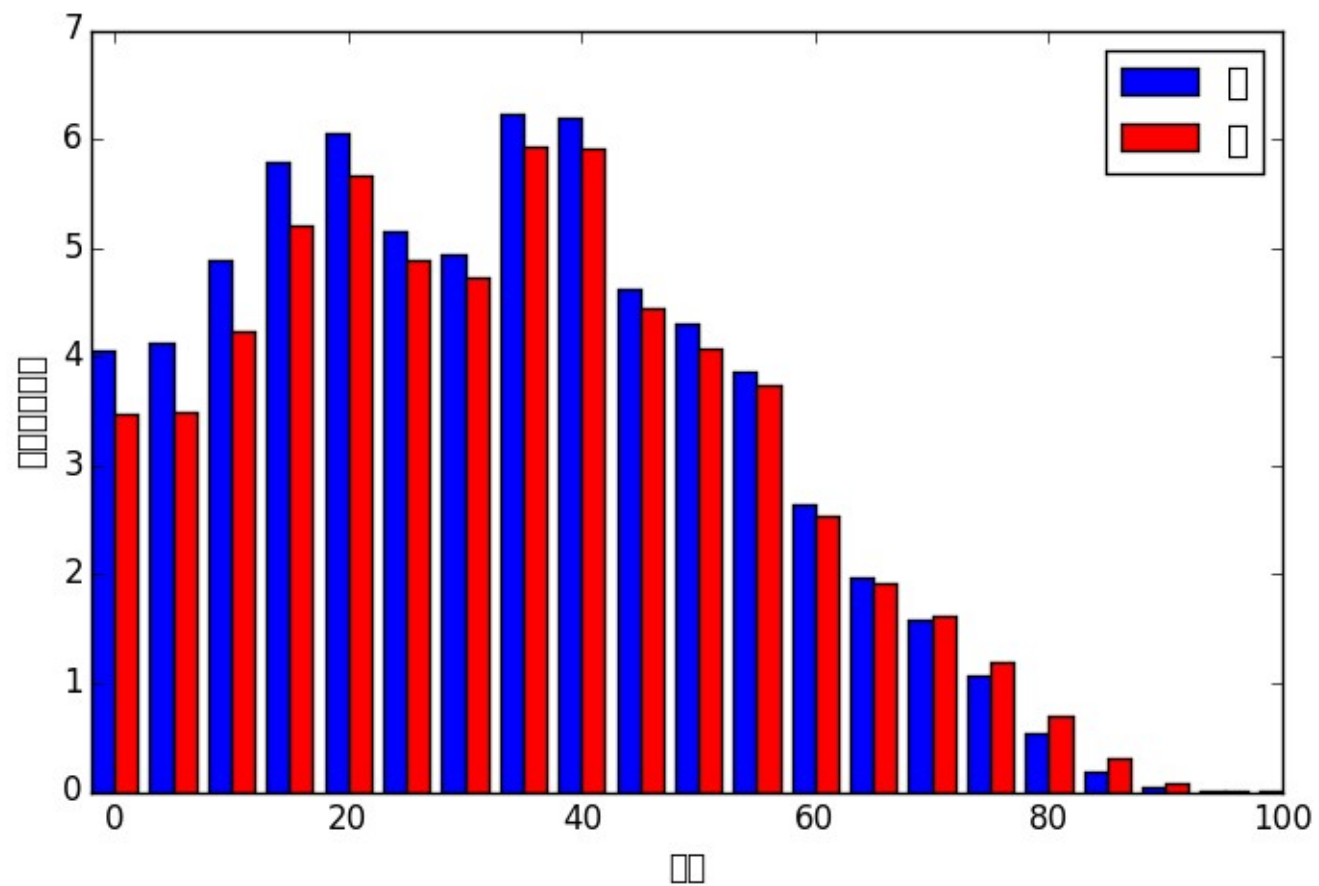
第一個參數為每根柱子左邊緣的橫坐標，為了讓男性和女性的柱子以年齡刻度為中心，這裡讓每根柱子左側的橫坐標為“年齡減去柱子的寬度”。Bar()的第二個參數為每根柱子的高度，第三個參數指定所有柱子的寬度。當第三個參數為序列時，可以為每根柱子指定寬度。

繪圖函數簡介

```
plt.bar(data[:,0], data[:,2]/1e7, width, color="r",  
label=u"女")  
plt.xlim(-width, 100)  
plt.xlabel(u"年齡")  
plt.ylabel(u"人口（千萬）")  
plt.legend()  
  
plt.show()
```

繪製女性人口分佈的柱狀圖，這裡以年齡為柱子的左邊緣橫坐標，因此女性和男性的人口分佈圖以年齡刻度為中心。由於bar()不自動修改顏色，因此程式中通過color參數設置兩個柱狀圖的顏色。

繪圖函數簡介



繪圖函數簡介

散列圖

使用plot()繪圖時，如果指定樣式參數為僅繪製資料點，那麼所繪製的就是一幅散列圖。例如：

```
>>>plt.plot(np.random.random(100),  
np.random.random(100), "o")
```

和scatter()繪製的散列圖類似，scatter()可以指定每個點的顏色和大小。下麵的程式演示scatter()的用法 (matplotlib_scatter.py).

色定

繪圖函數簡介

```
import numpy as np
import matplotlib.pyplot as plt

plt.figure(figsize=(8,4))
x = np.random.random(100)
y = np.random.random(100)
plt.scatter(x, y, s=x*1000, c=y, marker=(5, 1),
            alpha=0.8, lw=2, facecolors="none")
plt.xlim(0,1)
plt.ylim(0,1)

plt.show()
```

且和面積成正比。也可以定一個數，

繪圖函數簡介

指定所有點的大小；也可以是陣列，分別對每個點指定大小。

`c`參數指定每個點的顏色，可以是數值或陣列。這裡使用一維陣列為每個點指定了一個數值。通過顏色映射表，每個數值都會與一個顏色相對應。默認的顏色映射表中藍色與最小值對應，紅色與最大值對應。當`c`參數是形狀為 $(N,3)$ 或 $(N,4)$ 的二維陣列時，則直接表示每個點的RGB顏色。

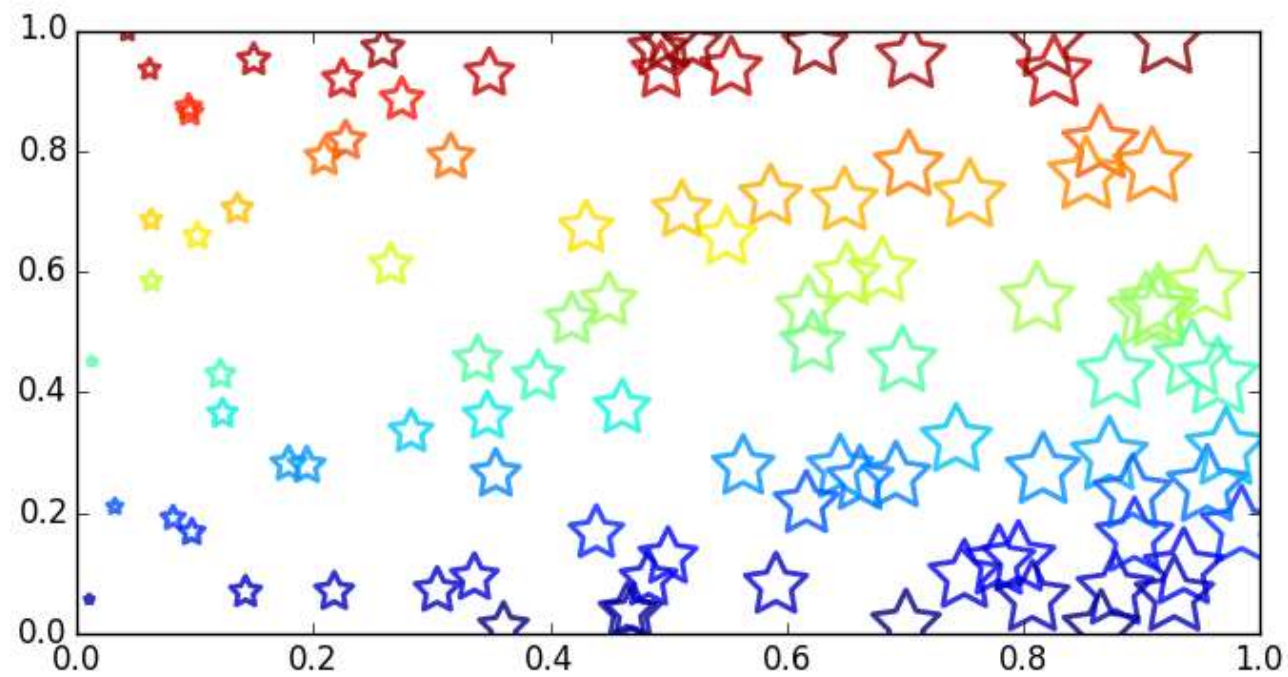
`marker`參數設置點的形狀，可以是個表示形狀的字串，也可以是表示多邊形的兩個元素的元組，第一個元素表示多邊形的邊數，

繪圖函數簡介

第二個元素表示多邊形的樣式，取值範圍為0、1、2、3。0表示多邊形，1表示星形，2表示放射形，3表示忽略邊數而顯示為圓形。

最後，通過alpha參數設置點的透明度，通過lw參數設置線寬，lw是line width的縮寫。facecolors參數為“none”時，表示散列點沒有填充色。

繪圖函數簡介



繪圖函數簡介


圖像

`imread()`和`imshow()`提供了簡單的圖像載入和顯示功能.

```
>>>img = plt.imread("lena.jpg")
```

`imread()`可以從圖像檔讀入資料，得到一個表示圖像的NumPy陣列。它的第一個參數是檔案名或檔物件，`format`參數指定圖像類型，如果省略，就由檔的副檔名決定圖像類型。對於灰度圖像，它返回一個形狀為 (M, N) 的陣列；對於彩色圖像，返回形狀為 (M, N, C) 的陣列。其中, M 為圖像的高度, N 為圖像的寬度, C 為3或4,表示圖像的通道數。

繪圖函數簡介

下麵的程式從“lena.jpg”中讀入圖像資料，得到的陣列是一個形狀為(393,512,3)的單字節不帶正負號的整數陣列。這是因為通常使用的圖像都是採用單字節分別保存每個圖元的紅、綠、藍三個通道的分量：

```
>>> img = plt.imread("lena.jpg")
```

```
>>> img.shape  
(393L, 512L, 3L)
```

```
>>> img.dtype  
dtype('uint8')
```

繪圖函數簡介

`imshow()`可以用來顯示`imread()`返回的陣列。如果陣列是表示多通道圖像的三維陣列，那麼每個圖元的顏色由各個通道的值決定：

```
>>> plt.imshow(img) #注意圖像是上下顛倒的
```

請注意，從JPG圖像中讀入的資料是上下顛倒的，為了正常顯示圖像，可以將陣列的第0軸反轉，或者設置`imshow()`的`origin`參數為“lower”，從而讓所顯示圖表的原點在左下角：

```
>>> plt.imshow(img[::-1]) #反轉圖像陣列的第0軸  
#or  
>>> plt.imshow(img, origin="lower") # 讓圖表的原點在左下角
```

繪圖函數簡介

如果三維陣列的元素類型為浮點數，那麼元素的取值範圍為0.0到1.0,與顏色值0到255 對應。超出這個範圍可能會出現顏色異常的圖元。下面的例子將陣列img轉換為浮點陣列並用 `imshow()` 進行顯示：

```
>>> img = img[: :-1]
>>> plt.imshow(img*1.0) #取值範圍為0.0到255.0的浮點陣列，不能正確顯示顏色
>>> plt.imshow(img/255.0) #取值範圍為0.0到1.0的浮點陣列，能正確顯示顏色
>>> plt.imshow(np.clip(img/200.0, 0, 1)) # 使用clip()限制取值範圍，整個圖像變亮
```

繪圖函數簡介

如果`imshow()`的參數是二維陣列，就使用顏色映射表決定每個圖元的顏色。下面顯示圖像中的紅色通道：

```
>>> plt.imshow(img[:, :, 0])
```

顯示效果比較嚇人，因為默認的影像對應將最小值映射為藍色、將最大值映射為紅色。可以使用`colorbar()`將顏色映射表在圖表中顯示出來：

```
>>> plt.colorbar()
```

繪圖函數簡介

通過imshow()的cmap參數可以修改顯示圖像時所採用的顏色映射表。顏色映射表是一個ColorMap對象，matplotlib中已經預先定義好了很多顏色映射表，可通過下面的語句找到這些顏色映射表的名字：(matplotlib_imshow.py)

```
>>> import matplotlib.cm as cm  
>>> cm._cmapnames  
['Spectral', 'copper', 'RdYlGn', 'Set2', 'sumner', 'spring', 'gist_ncar', ...]
```

下面使用名為copper的顏色映射表顯示圖像的紅色通道，很有老照片的味道：

```
>>> plt.imshow(img[:, :, 0], cmap=cm.copper)
```

繪圖函數簡介

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm

plt.subplots_adjust(0,0,1,1,0.05,0.05)
plt.subplot(331)
img = plt.imread("lena.jpg")
plt.imshow(img)

plt.subplot(332)
plt.imshow(img[::-1])

plt.subplot(333)
plt.imshow(img, origin="lower")

img = img[::-1]
plt.subplot(334)
plt.imshow(img*1.0)
```

繪圖函數簡介

```
plt.subplot(335)  
plt.imshow(img/255.0)
```

```
plt.subplot(336)  
plt.imshow(np.clip(img/200.0, 0, 1))
```

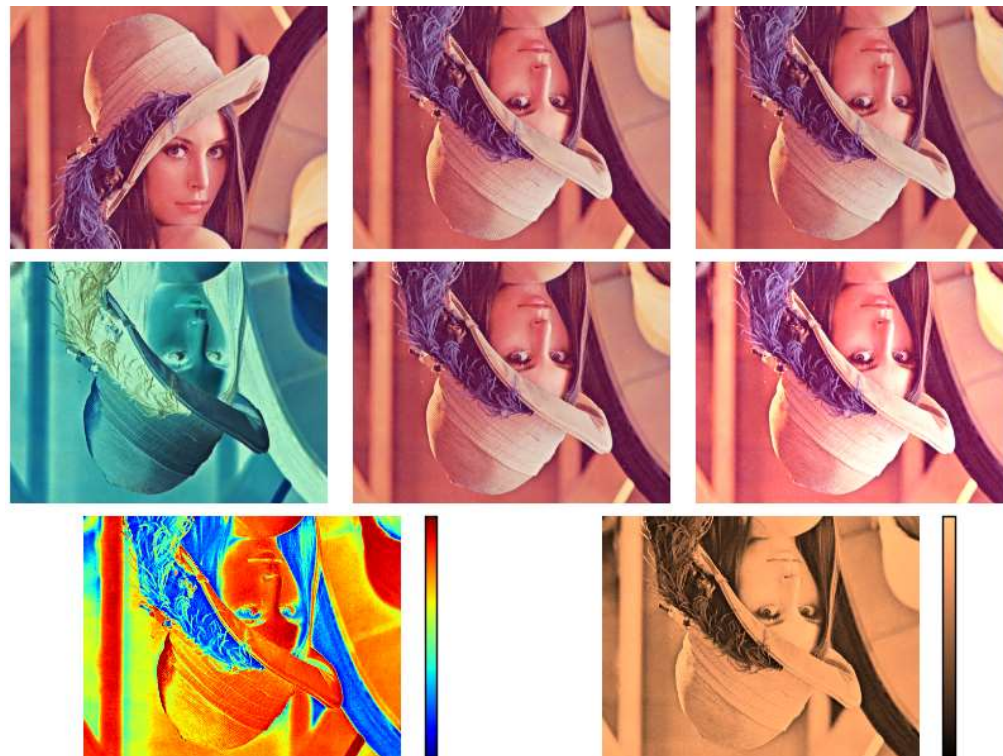
```
plt.subplot(325)  
plt.imshow(img[:, :, 0])  
plt.colorbar()
```

```
plt.subplot(326)  
plt.imshow(img[:, :, 0], cmap=cm.copper)  
plt.colorbar()
```

```
for ax in plt.gcf().axes:  
    ax.set_axis_off()  
    ax.set_axis_off()
```

```
plt.show()
```


繪圖函數簡介



繪圖函數簡介

還可以使用`imshow()`顯示任意的二維資料，
例如下面的程式使用圖像直觀地顯示了二元函
數 $f(x, y) = x^2 - y^2$ (matplotlib_2dfunc.py 使用
`imshow()`視覺化二元函數)

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm

y, x = np.ogrid[-2:2:200j, -2:2:200j]
z = x * np.exp(- x**2 - y**2)

extent = [np.min(x), np.max(x), np.min(y), np.max(y)]
```

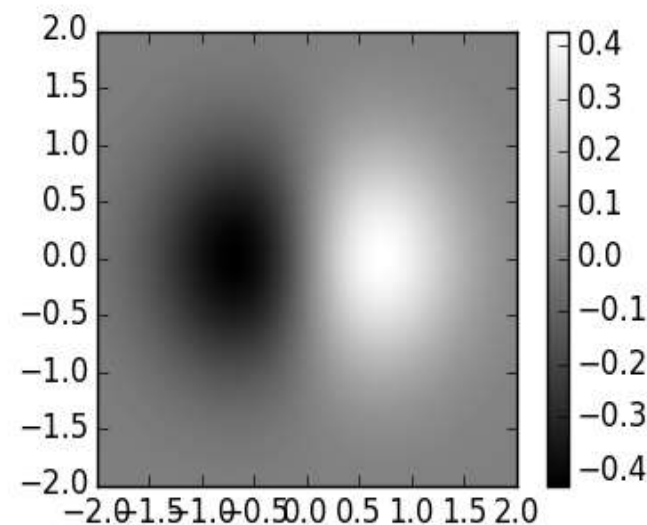
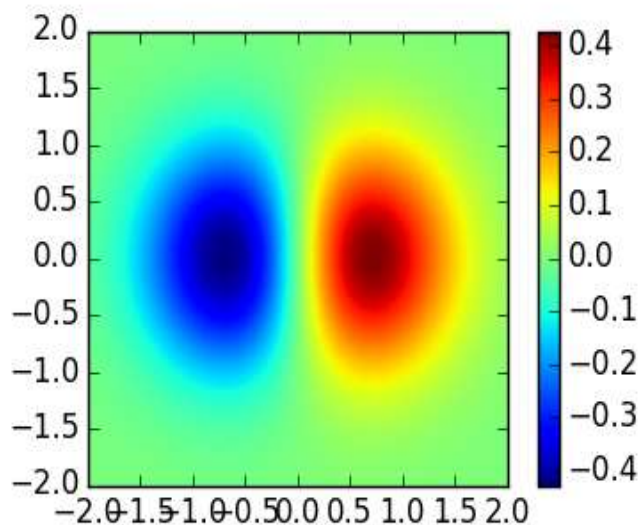
繪圖函數簡介

```
plt.figure(figsize=(10,3))  
plt.subplot(121)  
plt.imshow(z, extent=extent, origin="lower")  
plt.colorbar()  
plt.subplot(122)  
plt.imshow(z, extent=extent, cmap=cm.gray,  
origin="lower")  
plt.colorbar()  
  
plt.show()
```

首先通過陣列的廣播功能計算出表示函數值的二維陣列 z ，注意它的第0軸表示Y軸、第1軸表示X軸。然後將X、Y軸的取值範圍保存到`extent`列表中。

繪圖函數簡介

將extent列表傳遞給 `imshow()` 的 `extent` 參數，這樣一來，圖表的X、Y軸的刻度標籤將使用extent列表所指定的範圍。



繪圖函數簡介

等值線圖

還可以使用等值線圖表示二元函數。所謂等值線，是指由函數值相等的各點連成的平滑曲線。等值線可以直觀地表示二元函數值的變化趨勢，例如等值線密集的地方表示函數值在此處的變化較大。matplotlib中可以使用`contour()`和`contourf()`描繪等值線，它們的區別是：`contourf()`所得到的是帶填充效果的等值線。(matplotlib_contour.py用`contour`和`contourf`描繪等值線圖)

繪圖函數簡介

```
import numpy as np
import matplotlib.pyplot as plt

y, x = np.ogrid[-2:2:200j, -3:3:300j]
z = x * np.exp(- x**2 - y**2)

extent = [np.min(x), np.max(x), np.min(y), np.max(y)]

plt.figure(figsize=(10,4))
plt.subplot(121)
cs = plt.contour(z, 10, extent=extent)
plt.clabel(cs)
plt.subplot(122)
plt.contourf(x.reshape(-1), y.reshape(-1), z, 20)
plt.show()
```

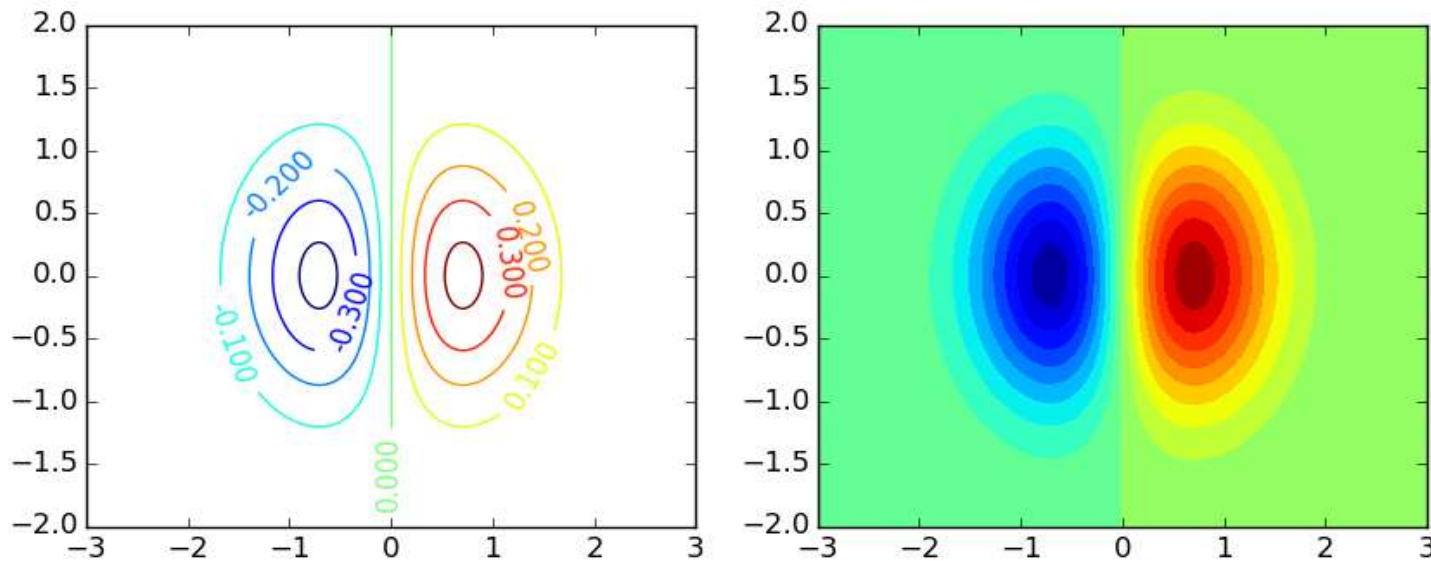
繪圖函數簡介

為了更清楚地區分X軸和Y軸，這裡讓它們的取值範圍和等分次數均不相同。這樣得到的陣列z的形狀為(200, 300)，它的第0軸對應Y軸、第1軸對應X軸。

調用`contour()`繪製陣列z的等值線圖，第二個參數為10，表示將整個函數的取值範圍等分為10個區間，即顯示的等值線圖中將有9條等值線。和`imshow()`一樣，可以使用`extent`參數指定等值線圖的X軸和Y軸的資料範圍。`contour()`所返回的是一個`QuadContourSet`物件，將它傳遞給`clabel()`，為其中的等值線標上對應的值。

繪圖函數簡介

調用`contourf()`,繪製將取值範圍等分為20份、帶填充效果的等值線圖。這裡演示了另外一種設置X、Y軸取值範圍的方法。它的前兩個參數分別是計算陣列`z`時所使用的X軸和Y軸上的取樣點,這兩個陣列必須是一維的。



繪圖函數簡介

還可以使用等值線繪製隱函數曲線。顯然，無法像繪製一般函數那樣，先創建一個等差陣列表表示變量的取值點，然後計算出陣列中每個x所對應的y值。可以使用等值線解決這個問題，顯然隱函數的曲線就是值等於0的那條等值線。下面的程式繪製函數

$$f(x, y) = (x^2 + y^2)^4 - (x^2 - y^2)^2$$

在 $f(x,y)=0$ 和 $f(x,y)-0.1 = 0$ 時的曲線。

(matplotlib_implicit_func.py)

```
import numpy as np
import matplotlib.pyplot as plt

y, x = np.ogrid[-1.5:1.5:200j, -1.5:1.5:200j]
f = (x**2 + y**2)**4 - (x**2 - y**2)**2
```

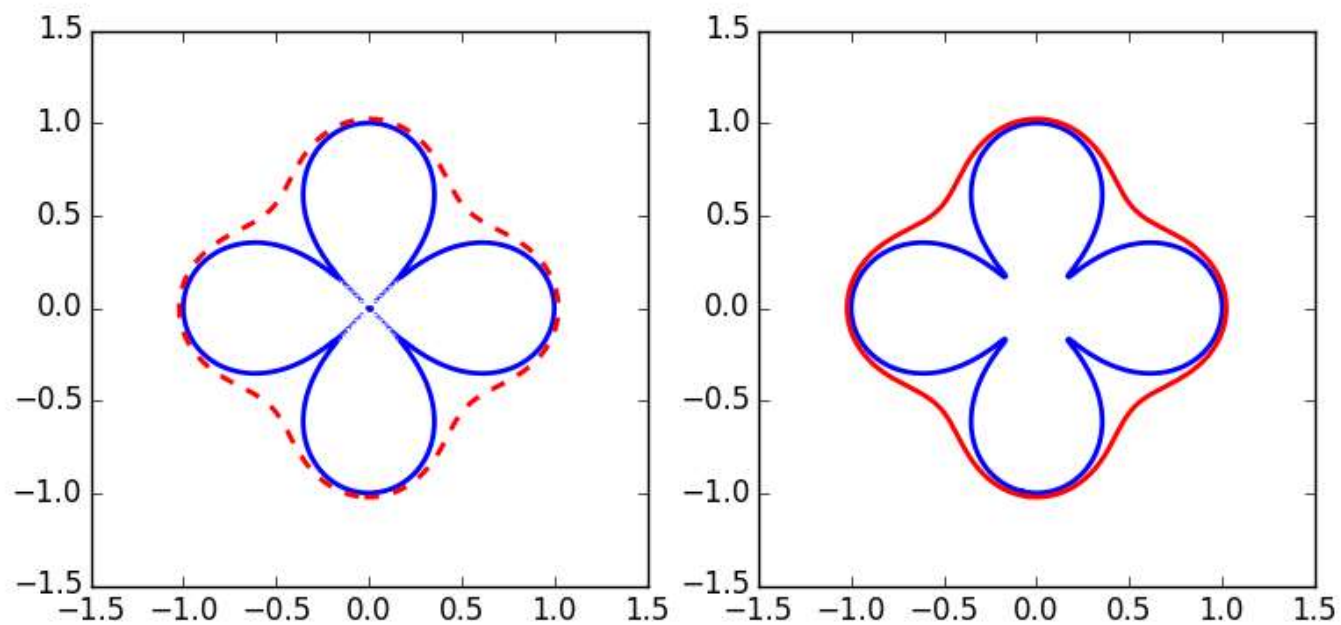
繪圖函數簡介

```
plt.figure(figsize=(9,4))
plt.subplot(121)
extent = [np.min(x), np.max(x), np.min(y), np.max(y)]
cs = plt.contour(f, extent=extent, levels=[0, 0.1],
colors=["b", "r"], linestyles=["solid", "dashed"],
linewidths=[2, 2])

plt.subplot(122)
for c in cs.collections:
    data = c.get_paths()[0].vertices
    plt.plot(data[:,0], data[:,1],
color=c.get_color()[0], linewidth=c.get_linewidth()[0])

plt.show()
```

繪圖函數簡介



繪圖函數簡介

在調用`contour()`繪製等值線時，可以通過`levels`參數指定所繪製等值線對應的函數值，這裡設置`levels`參數為`[0, 0.1]`，因此最終將繪製兩條等值線。

觀察圖會發現，表示隱函數 $f(x)=0$ 藍色實線並不是完全連續的，在圖的中間部分它由許多孤立的小段構成。因為等值線在原點附近無限靠近，因此無論對函數 f 的取值空間如何進行細分，總是會有無法分開的地方，最終造成了圖中的那些孤立的細小區域。而表示隱函數 $f(x,y)-0.1=0$ 的紅色虛線則是閉合且連續的。

繪圖函數簡介

可以通過`contour()`返回的物件獲得等值線上每點的資料，下面在IPython中觀察變量`cs`，它是一個 `QuadContourSet` 對象：

```
>>> run matplotlib_implicit_func.py
>>> cs
<matplotlib.contour.QuadContourSet instance at 0x0A348E90>
```

`cs`對象的`collections`屬性是一個等值線清單，每條等值線用一個`LineCollection`對象表示：

```
>>> cs.collections
<a list of 2 mcoll.LineCollection objects>
```

繪圖函數簡介

每個LineCollection物件都有它自己的顏色、線型、線寬等屬性，注意這些屬性所獲得的結果外面還有一層封裝，要獲得其第0個元素才是真正的配置：

```
>>> c.get_color()[0]
array([ 1.,  0.,  0.,  1.])
>>> c.get_linewidth()[0]
2
```

由類名可知，LineCollection物件是一組曲線的集合，因此它可以表示像藍色實線那樣由多條線構成的等值線。它的get_paths()方法獲得構成等值線的所有路徑，本例中藍色實線

繪圖函數簡介

所表示的等值線由42條路徑構成：

```
>>> len(cs.collections[0].get_paths())  
42
```

路徑是一個Path物件，通過它的vertices屬性可以獲得路徑上所有點的座標：

```
>>> path = cs.collections[0].get_paths()[0]  
>>> type(path)  
<class 'matplotlib.path.Path'  
>>> path.vertices  
array([[ -0.08291457, -0.98938936],  
       [ -0.09039269, -0.98743719],  
       ...,  
       [ -0.08291457, -0.98938936]])
```

繪圖函數簡介

下麵的程式從等值線集合cs中找到表示等值線的路徑，並使用plot()將其繪製出來。

```
plt.subplot(122)
for c in cs.collections:
    data = c.get_paths()[0].vertices
    plt.plot(data[:,0], data[:,1],
             color=c.get_color()[0], linewidth=c.get_linewidth()[0])
```


繪圖函數簡介

三維繪圖

`mpl_toolkits.mplot3d`模組在`matplotlib`基礎上提供了三維繪圖的功能。由於它使用`matplotlib`的二維繪圖功能來實現三維圖形的繪製工作，因此繪圖速度有限，不適合用於大規模資料的三維繪圖。如果需要更複雜的三維資料視覺化功能，可使用Mayavi。(matplotlib_surface.py 使用matplotlib繪製三維曲面)

繪圖函數簡介

```
"""
```

演示matplotlib的三維繪圖功能。

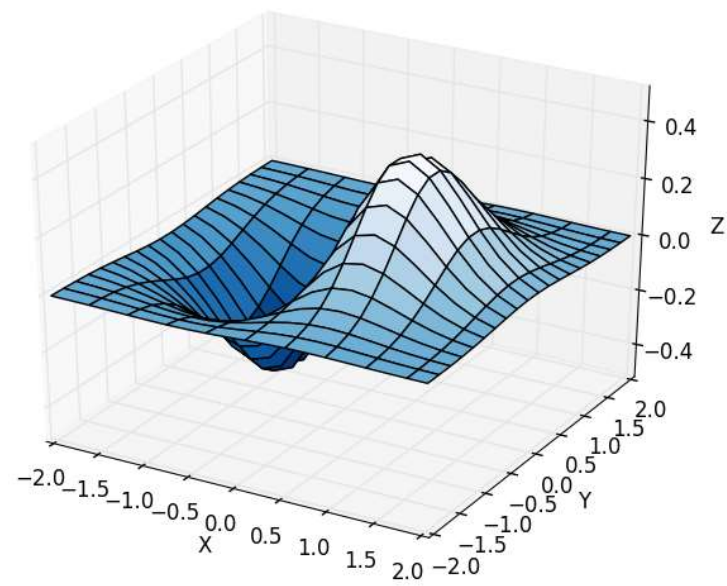
```
"""
```

```
import numpy as np
import mpl_toolkits.mplot3d
import matplotlib.pyplot as plt
```

```
x, y = np.mgrid[-2:2:20j, -2:2:20j]
z = x * np.exp( - x**2 - y**2)
```

```
ax = plt.subplot(111, projection='3d')
ax.plot_surface(x, y, z, rstride=2, cstride=1, cmap =
plt.cm.Blues_r)
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")
plt.show()
```

繪圖函數簡介



繪圖函數簡介

首先載入mplot3d模組，matplotlib中與三維繪圖相關的功能均在此模組中定義。使用 mgrid創建X-Y平面的網格並計算網格上每點的高度z。由於繪製三維曲面的函數要求X、Y 和Z軸的資料都用相同形狀的二維陣列表示，因此這裡不能使用ogrid創建。和之前的imshow() 不同.陣列的第0軸可以表示X和Y軸中的任意一個，在本例中第0軸表示X軸、第1軸表示Y軸。

在當前圖表中創建一個子圖，通過projection參數指定子圖的投影模式為“3d”，這樣 subplot()將返回一個用於三維繪圖的Axes3D子圖對象。

繪圖函數簡介

投影模式: 投影模式決定了點從資料座標轉換為屏幕坐標的方式.可以通過下面的語句獲得當前有效的投影模式的名稱：

```
>>> from matplotlib import projections  
>>> projections.get_projection_names()  
['3d', 'aitoff', 'hammer', 'lambert', 'mollweide', 'polar',  
'rectilinear']
```

只有在載入mplot3d模組之後，此清單中才會出現'3d'投影模式. 'aitoff'、'hammer', 'lambert', 'mollweide'等均為地圖投影，'polar' 為極座標投影, 'rectilinear'則是預設的直線投影模式.

繪圖函數簡介

調用Axes3D對象的plot_surface()繪製三維曲面。其中：參數x、y、z都是形狀為(20,20)的二維陣列，陣列x和y構成了X-Y平面上的網格，而陣列z則是網格上各點在曲面上的取值。通過cmap參數來指定值和顏色之間的映射，即曲面上各點的高度值與其顏色的對應關係。rstride和cstride參數分別是陣列的第0軸和第1軸的下標間隔。對於很大的陣列，使用較大的間隔可以提高曲面的繪製速度。程式中，plot_surfece()調用和下麵的語句是等價的：

```
ax.plot_surface(x[:,::2,:], y[:,::2,:], z[:,::2,:],  
               rstride=1, cstride=1)
```

繪圖函數簡介

除了繪製三維曲面之外，Axes3D物件還提供了許多其他的三維繪圖方法。可以通過下面的連結位址找到各種三維繪圖的演示程式：

<http://matplotlib.sourceforge.net/examples/mplot3d/index.html>

```
from mpl_toolkits.mplot3d import Axes3D
```

```
fig = plt.figure()  
ax = Axes3D(fig)  
X = np.arange(-4, 4, 0.25)  
Y = np.arange(-4, 4, 0.25)  
X, Y = np.meshgrid(X, Y)  
R = np.sqrt(X**2 + Y**2)  
Z = np.sin(R)
```

```
ax.plot_surface(X, Y, Z, rstride=1, cstride=1,  
cmap='hot')
```

