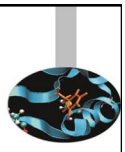
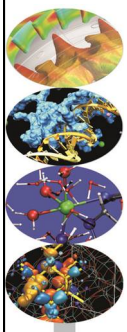
CINECA SCAI
SuperComputing Applications and Innovation



Introduzione a ***matplotlib***



Mario Rosati
CINECA – Roma
m.rosati@cineca.it

CINECA

CINECA SCAI
SuperComputing Applications and Innovation

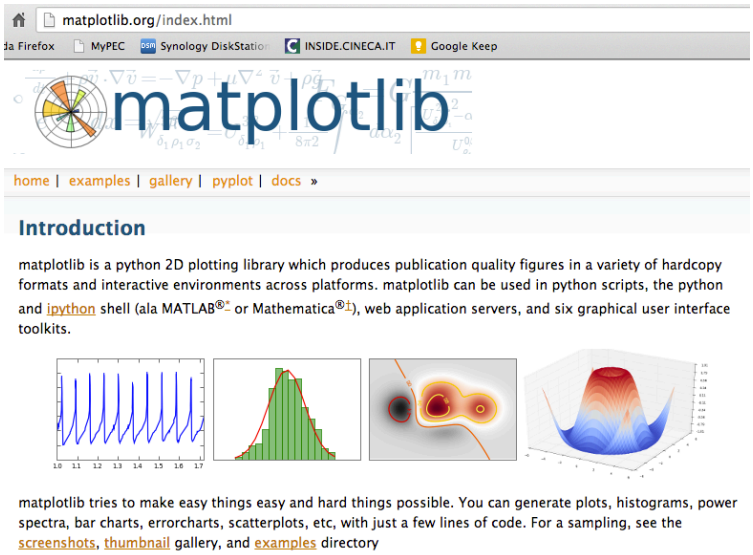
Cos'è *Matplotlib*

- è un modulo per la generazione di grafici 2D e, in piccola parte 3D
- è completamente sviluppata in Python e utilizza il modulo Numpy per la rappresentazione di grandi array.
- è divisa in tre componenti:
  - *Matplotlib API* (interfaccia a oggetti)
  - *Pylab interface*: interfaccia procedurale alla Matplotlib API, progettata, in origine, per emulare i comandi grafici di Matlab, in ambiente Python.
  - *Output back-end*: grafici per l'output su varie tipologie di GUI e su diversi formati di file
- Lcontiene una serie di funzionalità che la rendono adatta ad applicazioni di calcolo scientifico (tra le altre cose, è possibile utilizzare la sintassi LaTex per aggiungere formule sui grafici)

2

SCAI
SuperComputing Applications and Innovation

Documentazione di riferimento



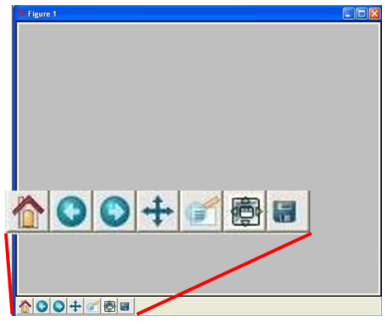
SCAI
SuperComputing Applications and Innovation

Introduzione a Pylab

- L'interfaccia *PyLab* costituisce il modo più semplice per lavorare con Matplotlib; le sue funzioni grafiche sono molto simili a quelle disponibili in ambiente Matlab.

Esempio

```
>>> from pylab import *
>>> figure()
>>> show()
```



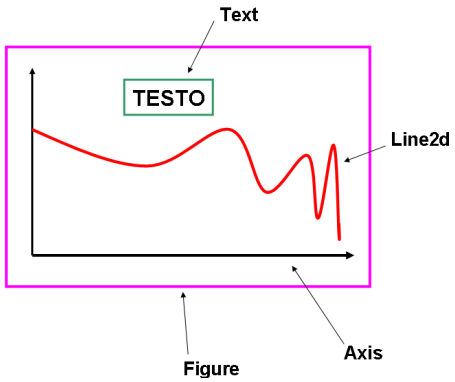
- La funzione *figure()* istanzia un oggetto figura
- La funzione *show()* visualizza tutte le figure create

SCAI
SuperComputing Applications and Innovation

Introduzione a Pylab

Le principali entità del modulo sono:

- *Figure:* l'oggetto figure ha i suoi attributi (risoluzione, dimensioni,..)
- *Line2d:* le linee2d possiedono diverse proprietà (marcatori, …)
- *Text:* per la gestione di box testuali (*plain* o *math*)
- *Axis:* per la gestione degli assi



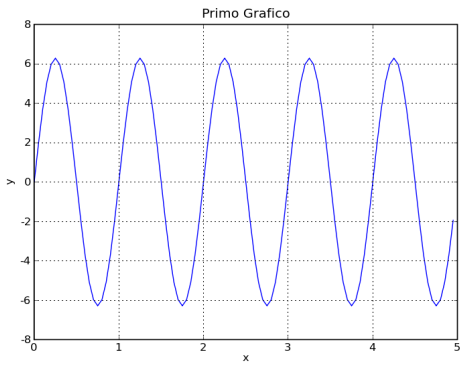
5

SCAI
SuperComputing Applications and Innovation

Il Primo grafico Pylab

Il primo Grafico

```
>>> from numpy import *
>>> from pylab import *
>>> t=arange(0,5,0.05)
>>> f=2*pi*sin(2*pi*t)
>>> plot(t,f)
>>> grid()
>>> xlabel('x')
>>> ylabel('y')
>>> title('Primo grafico')
>>> show()
```

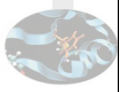


NB: Il grafico viene visualizzato solo alla chiamata della funzione show()

6

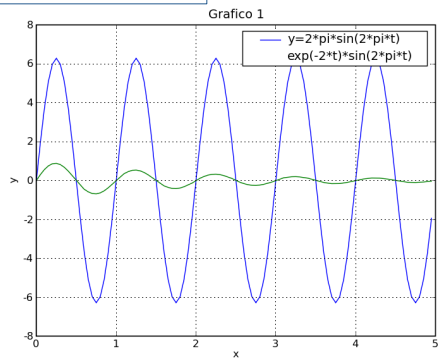
CINECA SCAI
SuperComputing Applications and Innovation

Comandi di base



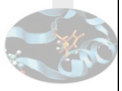
Sovrapposizione di 2 grafici nella stessa figure

```
>>> hold(True)
>>> f2 = sin(2*pi*t)*exp(-2*t)
>>> plot(t,f2)
>>> legend(('y=2*pi*sin(2*pi*x)','sin(2*pi*x)*exp(-2*x)'))
```



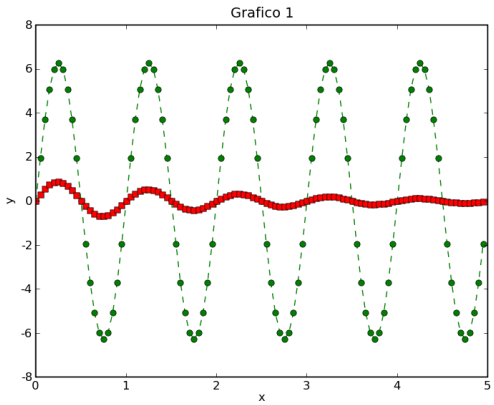
CINECA SCAI
SuperComputing Applications and Innovation

Comandi di base



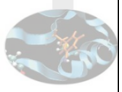
Sovrapposizione di 2 grafici nella stessa *figure*: un alternativa

```
>>> clf
>>> plot(t,f,'g--o',t,f2,'r:s')
>>> xlabel('x')
>>> ylabel('y')
>>> title('Grafico 1')
```



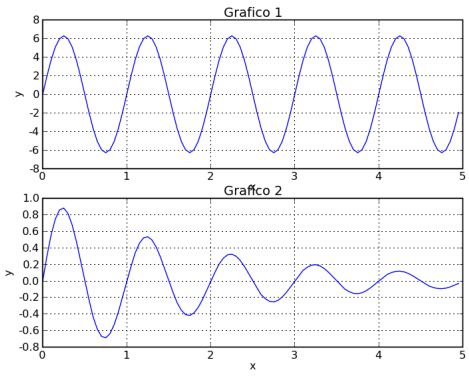
SCAI
SuperComputing Applications and Innovation

Comandi di base



Costruzione di sub-plot

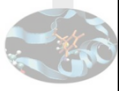
```
>>> subplot(211)
>>> plot(t,f)
>>> xlabel('x')
>>> ylabel('y')
>>> title('Grafico 1')
>>> subplot(212)
>>> plot(t,f2)
>>> xlabel('x')
>>> ylabel('y')
>>> title('Grafico 2')
```



9

SCAI
SuperComputing Applications and Innovation

Figure



- E' possibile gestire e creare un numero arbitrario di figure tramite il comando *figure().*
- Una figure possiede i seguenti attributi:
  - *figsize* dimensione in inches
  - *facecolor* colore di riempimento
  - *edgecolor* colore del bordo
  - *dpi* risoluzione
  - *frameon* per mantenere il background grigio alla figura.
- Per chiudere la figura si possono usare i comandi:
  - *close(num)*
  - *close(istance)*
  - *close('all')*

10

SCAI
SuperComputing Applications and Innovation

Plot e Subplot

- Il comando

  plot(line2d , [properties line2d])

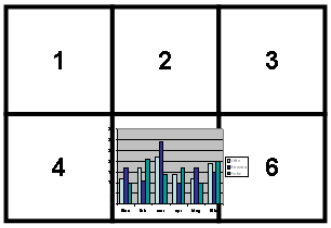
  è un comando versatile che consente di creare grafici multilinea specificando lo stile.

- Il comando

  subplot(nrows,ncol,index)

  permette di creare grafici multipli su una griglia con un numero specifico di righe

subplot(2,3,5)



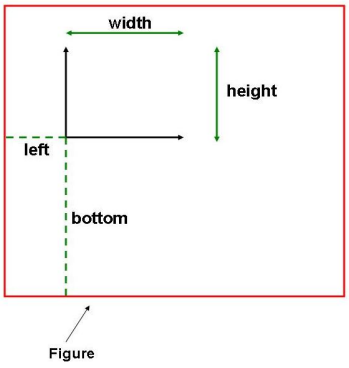
11

SCAI
SuperComputing Applications and Innovation

Axes

- L'oggetto *axes()* permette la gestione degli assi e si comporta in maniera simile a subplot
- *axes()* equivale a subplot(111)
- *axes([left,bottom, width, height])* posiziona e dimensiona il grafico secondo la lista di parametri passati come argomento.

  - Alcuni metodi:
    - *axis*([*xmin,xmax,ymin,ymax*])
    - *grid*()
    - *xticks*(*location,label*)
    - *legend*([*list_lines*],[*list_label*], *loc,* [*text_prop*])



12

SCAI
SuperComputing Applications and Innovation

Line2D Properties

- L'oggetto linea ha diversi attributi: è possibile modificare le dimensioni, lo stile, il colore etc. La funzione:

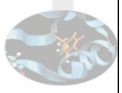
  *setp(*args, **kwargs)*

  permette di cambiare tali attributi.
- In alternativa è possibile modificare gli attributi tramite i metodi dell'oggetto line2D.
- Tra gli attributi ricordiamo:
  - *color* 'b', 'r', 'g', 'y', ' k', 'w , 'c', 'm'
  - *linewidth* float
  - *linestyle* '', '-', '- -', ':', '.-'
  - *label* stringa
  - *marker* '.', 'o', 'D', '^', 's', '*', '+', 'h'
  - *markersize float*
  - *markerfacecolor* color

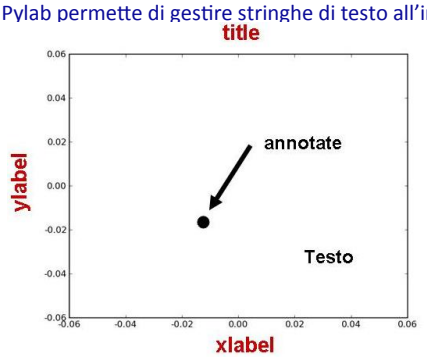
13

SCAI
SuperComputing Applications and Innovation

Gestione del testo



Pylab permette di gestire stringhe di testo all'interno di grafici.



xlabel (s, *args, **kwargs)

ylabel (s, *args, **kwargs)

title (s, *args, **kwargs)

annotate(s, xy, xytext=None, xycoords='data', textcoords='data', arrowprops=None,**props)

text(x, y, s, fontdict=None, **kwargs)

Inoltre Pylab è in grado di inglobare espressioni matematiche in espressioni di testo utilizzando la sintassi LaTex. Per esempio la sintassi:

xlabel(r'$y_i=2\pi \sin(2\pi x)$') equivale a $y_i = 2\pi \sin(2\pi x)$

E' necessario inoltre imporre: rcParams(text.usetex)=True

14

SCAI
SuperComputing Applications and Innovation

Text Properties

- L'oggetto testo possiede le seguenti proprietà:
  - *Fontsize:* xx-small, x-small, small, medium, large, x-large, xx-large
  - *Fontstyle:* *normal, italic, oblique*
  - *Color*
  - *Rotation: degree , 'vertical', 'horizontal'*
  - *Verticalalignment: 'top', 'center', 'bottom'*
  - *Horizontalalagnment: 'left', 'center', right'*
- Gli attributi possono essere modificati in tre modi:
  - tramite *keyword arguments,*
  - tramite la funzione *setp,*
  - tramite i metodi dell'oggetto testo

```
>>>xlabel('ciao', color ='r', fontsize='large')      #keyword arguments
>>>l=ylabel('asse y')
>>>setp(l,rotation=45)                                #setp()
>>>l.set_color('r')                                  #object method
```

15

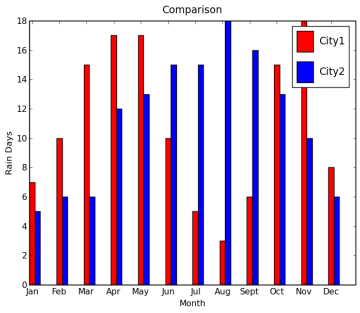
SCAI
SuperComputing Applications and Innovation

Diagrammi a barre

bar(left, height)

Esempio:

```
from pylab import *
n_day1=[7,10,15,17,17,10,5,3,6,15,18,8]
n_day2=[5,6,6,12,13,15,15,18,16,13,10,6]
m=['Jan','Feb','Mar','Apr','May','Jun'
,'Jul','Aug','Sept','Oct','Nov','Dec']
width=0.2
i=arange(len(n_day1))
r1=bar(i, n_day1,width, color='r',linewidth=1)
r2=bar(i+width,n_day2,width,color='b',linewidth=1)
xticks(i+width/2,m)
xlabel('Month'); ylabel('Rain Days'); title('Comparison')
legend((r1[0],r2[0]),('City1','City2'),loc=0,labelsep=0.06)
```



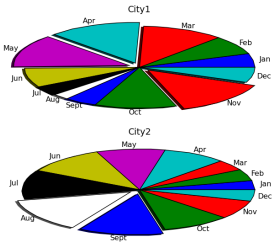
16

CINECA SCAI
SuperComputing Applications and Innovation

Torta

Oppure con gli stessi dati come creare una torta:

pie(x)




```
subplot(211)
pie(n_day1,labels=m,
explode=[0,0,0,0.1,0.1,0,0,0,0,0,0.1,0],
shadow=True)
title('City1')
subplot(212)
pie(n_day2,labels=m,
explode=[0,0,0,0,0,0,0,0.1,0.1,0,0,0],
shadow=True)
title('City2')
```

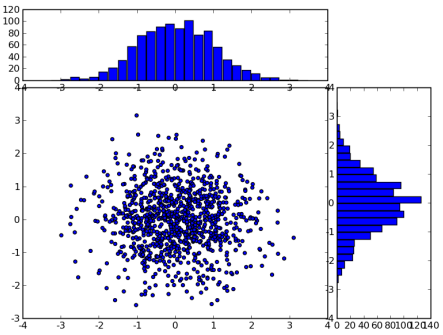

CINECA SCAI
SuperComputing Applications and Innovation

Scatter plot - Istogrammi

scatter(x,y) e hist(x)

```
x = numpy.random.randn(1000)
y = numpy.random.randn(1000)
axscatter = axes([0.1,0.1,0.65,0.65])
axhistx = axes([0.1,0.77,0.65,0.2])
axhisty = axes([0.77,0.1,0.2,0.65])

axscatter.scatter(x, y)
binwidth = 0.25
xymax = max([max(fabs(x)), max(fabs(y))])
lim = (int(xymax/binwidth)+1) * binwidth
bins = arange(-lim, lim+binwidth, binwidth)
axhistx.hist(x, bins=bins)
axhisty.hist(y, bins=bins, orientation='horizontal')
show()
```



SCAI
SuperComputing Applications and Innovation

Contour plot

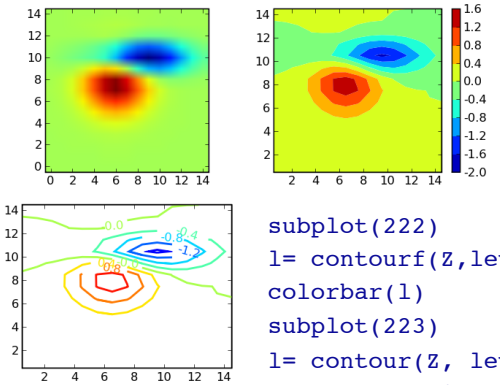
*contourf(*args, **kwargs)*
*contour(*args,**kwargs)*

```
from pylab import *
from matplotlib.mlab import bivariate_normal
delta = 0.5
x = arange(-3.0, 4.001, delta)
y = arange(-4.0, 3.001, delta)
X, Y = meshgrid(x, y)
Z1 = bivariate_normal(X, Y, 1.0, 1.0, 0.0, 0.0)
Z2 = bivariate_normal(X, Y, 1.5, 0.5, 1, 1)
Z = (Z1 - Z2) * 10
levels = arange(-2.0, 1.601, 0.4)

figure()
subplot(221)
imshow(Z,origin='lower')
```


SCAI
SuperComputing Applications and Innovation

Contour plot



```
subplot(222)
l= contourf(Z,levels,origin='lower')
colorbar(l)
subplot(223)
l= contour(Z, levels,origin='lower',
           linewidths=2)
clabel(l,inline=1,
   fmt='%1.1f',fontsize=14)
show()
```

CINECA SCAI
SuperComputing Applications and Innovation

Output

Matplotlib supporta diversi backend grafici. Possiamo dividere la tipologia di backend in due categorie:

- User interface backend: per l'assemblaggio in GUI. In Python esistono diverse librerie per la costruzione di interfaccie grafiche tra cui Tkinter, PyQt, pygtk che vengono supportate da matplotlib.
- Hardcopy backend: per la stampa su file. Vengono supportati i seguenti formati *.jpg, *png, *svg, *pdf, *rgba.

21