Appunti gnuplot

Autore: Matteo Lucarelli
ultima versione su: www.matteolucarelli.net
versione pdf

Guida rapida all'uso di gnuplot

Gnuplot è un potente strumento per il calcolo, la crezione di grafici e l'analisi di dati sperimentali. Permette sia l'uso interattivo, cioè impartendo i comandi in un terminale, che come un linguaggio di scripting. Può generare grafici e dati in moltissimi formati (testo, immagini, plotter, ecc.), in due o tre dimensioni, cartesiani, polari, in scala logaritmica, ecc. E' disponibile praticamente per ogni piattaforma.

Può essere facilmente integrato ed utilizzato da altri strumenti come terminale di visualizzazione. Da ogni linguaggio può essere utilizzato tramite il meccanismo delle pipe posix. Esistono inoltre diverse interfacce di programmazione dedicate (C,Java,Python,ecc.).

Link utili:

- homepage: www.gnuplot.info
- pagina ufficile delle demo: gnuplot.sourceforge.net/demo/
- altre demo interessanti: www.csse.uwa.edu.au/programming/gnuplot_demos/

Generalità

Per invocare il terminale digitare il comando

```
gnuplot
```

Al momento dell'invocazione possono essere modificate le caratteristiche del rendering:

```
gnuplot -gray   # avvia in modalità scala di grigi
gnuplot -mono   # avvia in modalità monocromatica
```

Per eseguire uno script senza invocare l'interprete è sufficiente passarlo come parametro al programma:

```
gnuplot script
```

All'interno dell'interprete è possibile eseguire i normali comandi di shell facendoli precedere dal segno '!' :

```
gnuplot> ! ls
gnuplot> ! mkdir output
```

Per abbandonare il programma:

```
gnuplot> exit
gnuplot> quit
```

Per creare uno script contenente le istruzioni necessarie a generare il grafico corrente è possibile in ogni momento impartire il comando

```
gnuplot> save "nomefile.plt"
```

Uno script precedentemente salvato può essere ricaricato tramite il comando

```
gnuplot> load "nomefile.plt"
```

Si noti che l'estensione utilizzata è arbitraria, infatti gli script sono semplici file di testo contenenti sequenze di comandi, e come tali possono essere modificati tramite un comune editor.

Negli script:

```
\ estende un comando sulla linea successiva
# identifica una linea di commento
```

Help

All'interno del terminale è disponibile un dettagliato help:

```
gnuplot> help                    # visualizza la schermata principale di help
gnuplot> help comando            # visualizza la schermata di help del comando
```

Mentre premendo 'h' durante la visualizzazione di un grafico si ottiene, sul terminale, l'help delle azioni real-time (zoom, rotazione, righelli, misure, ecc).

Inoltre il ritorno di un errore contiene quasi sempre la lista dei possibili valori, ad esempio:

```
gnuplot> plot sqrt(x) with

         expecting 'lines', 'points', 'linespoints', 'dots', 'impulses',
        'yerrorbars', 'xerrorbars', 'xyerrorbars', 'steps', 'fsteps',
        'histeps', 'filledcurves', 'boxes', 'boxerrorbars', 'boxxyerrorbars',
        'vectors', 'financebars', 'candlesticks', 'errorlines', 'xerrorlines',
        'yerrorlines', 'xyerrorlines', 'pm3d'
```

Opzioni

L'ambiente supporta moltissime opzioni.

```
gnuplot> show parameter      # mostra l'impostazione di parameter
gnuplot> show all            # mostra tutte le impostazioni
gnuplot> set parameter value # imposta il parametro
gnuplot> unset parameter     # disabilita il parametro
```

Alcune delle impostazioni di uso più frequente sono:

```
gnuplot> set xlabel "asse X"                # etichetta x
gnuplot> set title "Grafico 1"              # titolo
gnuplot> set xrange [0:2]                   # intervallo x
```

```
gnuplot> set xtics 1                            # spaziature tacche asse x
gnuplot> set autoscale y                        # calcolo automatico intervallo y
gnuplot> set hidden3d                           # grafico 3D non trasparente
gnuplot> set view angolo_h, angolo_v, zoom      # vista di partenza
gnuplot> set key default                        # abilita la legenda
gnuplot> unset key                              # disabilita la legenda
gnuplot> set key top left                       # riposiziona la legenda
gnuplot> set style function linespoint          # stile di tracciamento funzioni
```

Molte delle impostazioni viste (il range, lo stile, ecc) possono essere definite anche contestualmente al comando di plottaggio. La differenza è che tramite il comando set si imposta il default che verrà utilizzato se non diversamente specificato.

Alcune impostazioni utili a rendere i grafici più belli:

```
set ytics font "Helvetica,12"                   # font dell'asse y
set xtics 5 font "Helvetica,12"                 # font dell'asse y
unset key                                       # disabilita la legenda
set border 3                                    # abilita il bordo destro e inferiore
set grid                                        # disegna la griglia
set term x11 enhanced font "Helvetica,12"       # imposta il font default
```

Grafici

Per creare un grafico con le impostazioni di default è sufficiente impartire il comando **plot**:

```
gnuplot> plot sin(x)   # crea un grafico della funzione seno
```

Sono inoltre disponibili i comandi:

```
gnuplot> splot   # crea grafici 3D
gnuplot> replot  # senza parametri, ridisegna l'ultimo grafico
```

Per visualizzare più di un grafico contemporaneamente si utlizza la virgola:

```
gnuplot> plot f(x), sin(x), x
```

Il range utilizzato ed il numero di punti possono essere impostati tramite:

```
gnuplot> set samples N
gnuplot> plot [xmin:xmax][ymin:ymax] f(x)
```

Per i grafici 2D la variabile indipendente di default è x:

```
gnuplot> plot x**2
```

Per i grafici 3D le variabili indipendenti sono x e y

```
gnuplot> splot (x**2)/y
```

Nel caso in cui non si voglia usare x come variabile indipendente:

```
gnuplot> plot [t=1:10] f(t)
```

Definizione di funzioni

La definizione di funzioni usa una sintassi simile al c. Oltre ai soliti operatori aritmetici (*,+,-,/,**,%,) sono disponibili una serie di funzioni predefinite. L'elenco viene visualizzato da:

```
gnuplot> help functions
```

Mentre il significato delle singole funzioni è spiegato da:

```
gnuplot> help functions nomefunzione
```

E' possibile usare parametri all'interno di funzioni:

```
gnuplot> a=0.24
gnuplot> f(x)=c/((x-a)*(x-a)+b)+d/sqrt(x)
```

Mentre le funzioni stesse posso essere definite in modo parametrico:

```
gnuplot> f(x,a,v)=sqrt(v**2+2*a*x)
gnuplot> plot [0:400] f(x,0.5,5),f(x,0.5,10),f(x,0.5,20)
```

E' inoltre possibile definire delle funzioni discontinue con una sintassi simile al c: (condizione) ? condizione vera : condizione falsa

```
gnuplot> f(x)=(x<=3) ? x : (x-3)**2+3
gnuplot> f(x)=(x<3 || x>5) ? x : (x-3)**2+3
```

Risoluzione di equazioni in due variabili

La soluzione di una equazione in due variabili

```
f(x,y)
```

può essere tracciata come luogo dei punti in cui il grafico 3D

```
z=f(x,y)
```

incontra il piano xy (cioè dove z=0).

Ecco la sequenza di istruzioni necessaria a gnuplot per tracciare un simile grafico:

```
f(x,y)=...              # definisce la funzione
set view 0,0            # imposta la visuale dall'alto
set isosample 100,100   # aumenta la densità di punti
set cntrparam levels 0  # imposta una linea di contorno al valore z=0
set contour base        # traccia le linee di contorno sulla base del grafico
unset surface           # rende invisibile la superficie 3D
splot f(x,y)            # disegna
```

L'uso come calcolatore

Tramite il comando **print** gnuplot può essere utilizzato come un potente calcolatore scientifico:

```
gnuplot> print sqrt(2)/2
0.707106781186548
```

Naturalmente il comando può essere utlizzato in combinazione con la definizione di funzioni:

```
gnuplot> f1(x)=sin(x)/x
gnuplot> print f1(0.25)
0.989615837018092
```

Utilizzando numeri interi è necessario fare attenzione agli overflow ed agli arrotondamenti. Gnuplot infatti si comporta come il linguaggio C, dove una operazione tra interi restituisce un intero:

```
gnuplot> print (2**31)
-2147483648                  # Errore: intero 32bit in overflow
gnuplot> print (2.0**31)
2147483648.0                 # risultato corretto
gnuplot> print 2/3
0                            # Errore: troncamento di decimale
gnuplot> print 2/3.0
0.666666666666667            # risultato corretto
```

Si noti come, avendo a che fare con numeri interi, sia in generale conveniente inserire almeno un ".0".

Visualizzazione di dati sperimentali

Per visualizzare dei dati sperimentali contenuti in un file si utilizzano sempre le funzioni plot/splot seguite dal nome del file (eventualmente completo di path):

```
gnuplot> plot "nomefile"
gnuplot> plot "nomefile1", "nomefile2", ...
```

Il file normalmente contiene i dati in colonne separate da blank (spazi, tab, ecc.):

```
v(0,0) v(0,1) v(0,2)...
v(1,0) v(1,1) v(1,2)...
.
.
v(n,0) v(n,1) v(n,2)...
```

Per specificare quali colonne usare (la prima è 1):

```
gnuplot> plot "file" using 1:3
```

blocchi di dati differenti possono inoltre essere contenuti nell stesso file separati da linee bianche o da commenti. Per specificare il blocco (il primo è 0):

```
gnuplot> plot "plotexp.dat" index 0
```

Modifica e salvataggio dell'output

L'output di default delle istruzioni di plot è la generazione del grafico in una finestra. Esistono però molte altre possibilità. Il comando

```
gnuplot> set terminal
```

mostra un elenco delle varie possibilità.

Per modificare i tipo di output si utilizza quindi il comando

```
gnuplot> set terminal tipo
```

mentre per indirizzare l'output in un file (altrimenti verrebbe comunque mandato a terminale)

```
gnuplot> set output "filename.ext"
```

Attenzione: il file viene trattato in append quindi ogni successivo plottaggio aggiungerà dati al file. Per interrompere l'uso del file:

```
gnuplot> unset output
```

Per generare un immagine immagine png quindi:

```
gnuplot> set terminal png size w,h
gnuplot> set output "filename.png"
gnuplot> plot [-pi:pi] sin(x)
```

Mentre per generare un postscript (file stampabile):

```
gnuplot> set terminal postscript
gnuplot> set output "filename.ps"
gnuplot> plot cos(x)*x
```

Il tipo table inoltre genera dati in formato tabellare

```
gnuplot> set term table
gnuplot> plot f(x)
#Curve 0, 100 points
#x y type
0 0 u
0.010101 1.63972 i
0.020202 1.39031 i
0.030303 1.30688 i
   ....
```

Al termine per tornare a lavorare a video:

```
gnuplot> set terminal X11
gnuplot> set output
```

Interpolazione di dati sperimentali

Tramite il metodo dei mini quadrati è possibile ottimizzare una qualsiasi funzione parametrica in modo da cercare di adattarla ai dati sperimentali.

Per eseguire una interpolazione è quindi necessario definire una funzione parametrica adatta ai dati:

```
gnuplot> f(x)=a*(x+b)+c
```

e poi eseguire il comando di interpolazione informando gnuplot su quali siano i parametri variabili:

```
gnuplot> fit f(x) "file.dati" via a,b,c
```

Viene quindi generato un output relativo al calcolo di best fitting al termine del quale vengono assegnati i valori ottimali ai parametri.

A questo punto è possibile, ad esempio, visualizzare contemporaneamente i dati sperimentali e la funzione ottimizzata:

```
gnuplot> plot "file.dati",f(x)
```
